GES 2007 German e-Science

Available online at http://www.ges2007.de

Inspector Computer

M. Sutter[1], T. Müller[2], R. Stotzka[1], T. Jejkal[1], M. Holzapfel[1] and H. Gemmeke[1]

[1] Institute for Data Processing and Electronics, Forschungszentrum Karlsruhe, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany
[2] Institute for Nuclear and Energy Technologies, Forschungszentrum Karlsruhe, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany
email: michael.sutter@ipe.fzk.de
phone: (+49 7247) 82 5676

Abstract

Inspector Computer is a real world application based on Grid technologies helping users to find stolen goods in the internet. All the user has to provide is a reference image and a short, textual description of the items he is searching for. After initialization Inspector Computer queries several existing search engines in the Internet to get the necessary data. From the results of the query the images are extracted and compared to the reference image. The sophisticated comparison methods score images by their content, i.e. if two images contain the same object. Even comparison of two images needs a lot of computing power and in general thousands of comparisons have to be done. So it is obvious that only a Grid provides the necessary resources for Inspector Computer. Finally a sorted list of the most likely hits is delivered.

1 Introduction

In Germany approximately 2.8 million thefts occurs every year. Even if unique copies are stolen the detection rate is less than 30 % [1]. To retrieve his stolen goods the owner might observe the daily offers in internet auctions. But this is a very time consuming and frustrating process, especially if the search has to be done repeatingly for a couple of month until the item is found.

An automatic search based on the content of the images in auctions will reduce the effort to monitor the internet auctions, but at the moment the existing search engines support only retrievals based on meta data or keywords. Looking for a stolen vase might reveal thousands of hits, which have to be evaluated manually.

Another reason against the manual monitoring is the expanding spread of auctions and the corresponding probability, that vender of stolen goods use diverse auction houses to resell the items.

These are some reasons why the vender's in auction houses feel save and anonym. Another reason is the elementary usage of the different auction houses, making it very easy to sell items. So Inspector Computer was developed to automate the search process and to assist the user monitoring several auction houses by querying them automatically and doing an image comparison using

a reference image. The user only has to take a look in the presented results to find out, if his goods are found. So the manually effort is marginal and he could spent his time for more important things.

2 Fundamentals

Inspector Computer contains several processing steps. These are:

- Submission of the reference image and the description for searching in the auction houses
- Querying the search engines in the auction houses
- Storing the found images and meta data in a database
- Image comparison of the found images with the reference image
- Presentation of the results ordered by similarity of the images

The searching for the auctions e.g. is a processing step already integrated in every existing auction house. Otherwise it would be nearly impossible for the users to find interesting items in the whole content of them.

The really new processing step is the image comparison and the way to find images by their content. In no existing search engine such a thing is integrated. They are all indexed by meta data of the images, e.g. the description.

2.1 Image comparison procedures

One common method for image comparison is a model based approach. Thereby a model of the searched object is created to describe it best as possible. After that the model is used for searching the object in images. The disadvantage of this method is that it is only usable when searching always for the same type of objects. Otherwise a model for every used object must be created. A normal use case for this method is face recognition. Usually every face has some distinctive points like the eyes or the ears. So it is profitable to make a model of the face of a person once and use it for the recognition afterwards.

If it is not possible to declare a model the only way is to use the content of the images, possible in two ways – with segmentation and without. The goal of segmentation is to separate the objects of an image from the background. For segmentation several different algorithms exist, e.g. Region–Growing and Histogram based methods [2] and [3].

If segmentation is not used a registration of the images is the other possible solution. Registration is a process of transforming the image content of two images into a common coordinate system. This means that the objects of one image lay on top of the objects of the other image congruently. So it is very easy to make a decision if both images show the same object.

2.2 Objectives

The objectives of Inspector Computer sound very simple. All it has to do is finding auctions by comparing the images of them with a provided reference

image. You could simply say: Putting two images in and getting a measurement for the similarity out. Therefor several boundary conditions exist:

- Only one object per image
- The object should be central and dominant in the image
- The object should be a unique copy (searching for a standard item is no good idea, because they are sold everywhere)
- Different illumination in the images is solved by usage of only grey value images

But the objectives concurrently also the biggest problem of Inspector Computer, because the content of the images should be compared and not the description. Therefore special algorithms must be used, customized or developed because the images could not be compared pixel by pixel. It is obvious, that they are acquired under completely different conditions as you could see in figure 1.

Figure 1: The images illustrate the difficulties of the image comparison and the need for comparing the objects in the images. All images show the same car, but from a different viewpoint. For a human it is very easy to state that all images contain the same car, but for an algorithm this is a very complex problem. For the car on the left image it is nearly impossible to be detected as the same car on the other images by algorithms. We focus only on variations shown in the second and third image to detect the correspondence between both images.

A requirement for the image comparison is that it should be generic regarding the object types. So that it is possible to detect nearly every imaginable object. This makes the usage of a model based comparison method impossible – nobody wants to generate a model for every object.

Because of the expected enormous hit rate (up to several thousand images per search) when querying auction houses and comparing the found images Inspector Computer should use different Grid technologies and store the found images with meta data in a MySQL database [4]. So it is possible to provide on the one hand the necessary computing power and storage capacities and on the other hand a demonstrator for testing different Grid technologies and their cooperation in a real world application is implemented.

3 Workflow

After the necessary input is provided and the application is started Inspector Computer queries the search engines and stores the found images with additional meta data in a database. When the search is completed the images are compared to the provided reference image and a list ordered by similarity of the images is presented. The first entries in the list show the images which correspond best to the reference image. The user inspects the list manually in the order of most likely success to find the searched article if applicable. Figure 2 shows these processing steps of Inspector Computer in principal schema.

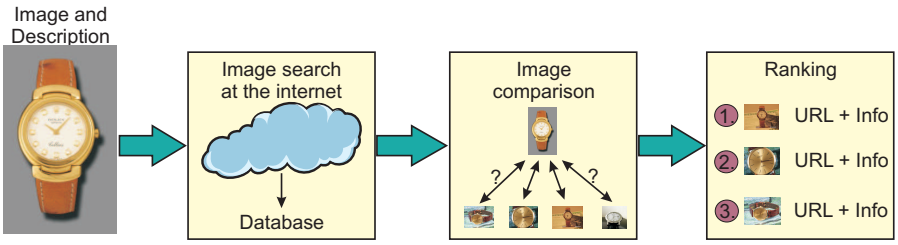

Figure 2: The principal processing steps of Inspector Computer. You can see the providing of the reference image, searching the internet, comparison of the images and getting the list of results.

This approach is similar to the manual search. But Inspector Computer facilitates the manual monitoring drastically, by reason that most of it is automated. Only the reference image and a short and accurate key word as description are necessary to start the application. During execution the user only has to wait until the results are presented.

4 Architecture and Implementation

Inspector Computer is a system using several Grid technologies as well as common software. It is implemented in the Java [5] programming language as a Servlet, controlled using a web interface. The web interface asks for the needed input, start the application and retrieve the results. During runtime the processing steps of Inspector Computer (figure 2) are executed and status messages are presented to the user.

For the Grid middleware the Globus Toolkit 4 (GT 4) [6] is chosen. It is a widely spread and easy to extend software package. GT 4 is based on the OGSA [7] and WSRF [8] standards and provides a service oriented architecture, structured in modular components. As middleware the focus of GT 4 lies only on low–level services for administration and control. The integrated API (Application Programming Interface) provides the functionality to extend a GT 4 installation with own services.

4.1 Webcrawler

One important part of Inspector Computer is the discovery of the images and meta information from the search engines in the internet and storing of the data in a MySQL database. For this purpose a webcrawler is used. During its execution it normally starts with a given URL, follows the links on the starting page and stores the data in a folder on the local file system. Normally it is possible to configure the webcrawler how deep it should scan the web sites.

Before the implementation of Inspector Computer was started several existing webcrawlers were evaluated. These are Heritrix, WebSPHINX, JSpider, WebEater, Java Web Crawler, WebLech and Arachnid [9]. They are all implemented in Java, making the integration in Inspector Computer very simple. During the evaluation it was realized, that all these webcrawlers are not applicable because they store the data in the local file system and scan only a special URL. The query of a search engine and storage of the data in a database needs to be integrated in all of them. As a result of this it was simpler to develop a new webcrawler.

The implemented webcrawler simply performs an HTTP (Hypertext Transfer Protocol) request. The request with the given search term is sent to a search engine and the content of the search engine is queried. From the retrieved response the necessary images and meta information are extracted and written to the database. The used interface to the database is described in the section below. A closer look on the evaluation and implementation of the webcrawler is described in [10].

4.2 Database Access

The common way for accessing databases with Java is the usage of JDBC (Java Database Connectivity) [11]. JDBC is a Java API for interaction with databases. This technology is thoroughly tested and provides full access to most database systems. The manufacturer of a database has to provide a device driver for the communication with the database. After the driver is successfully initialized it is possible to send SQL (Structured Query Language) commands for communication with the database over the JDBC connection.

For the communication with databases in Grid environments the OGSA–DAI (Open Grid Services Architecture – Data Access and Integration) [12] package has been developed. OGSA–DAI is a toolkit providing GT 4 Grid Services virtualizing databases and is the de facto standard for accessing databases with Grid technologies at the moment. At server side the databases are made available by deploying a data service and exposing a resource (database) to it. At client side available databases can be identified and accessed using Grid Services over OGSA–DAI Activities. For the usage of common features of a database, e.g. storing, update and query of data the Activities are already delivered within the standard software package. To query a dataset it is only necessary to give the SQL command to the appropriate Activity and execute it. OGSA–DAI provides also a flexible client and server API, so it is possible to extend the functionality

at any time.

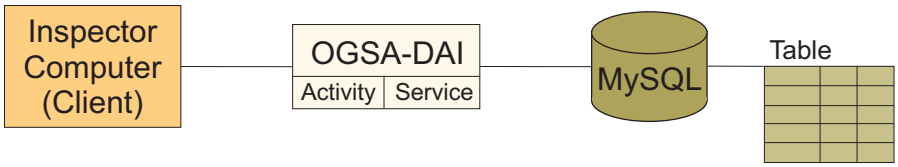

Figure 3: The schema for accessing databases over OGSA–DAI. The client of Inspector Computer has to execute an Activity of OGSA-DAI. The Activity calls the appropriate OGSA-DAI service encapsulating the database and executes the SQL command of the Activity.

After the webcrawler has extracted the necessary data from every visited auction a SQL command for storing the data is created. The SQL command is given to the OGSA–DAI Activity responsible for storing data. During execution of the Activity the dataset is written in the database using the OGSA–DAI services.

Since OGSA-DAI in the early versions had problems to store big datasets over its services the images are stored using a JDBC connection. The JDBC connection is responsible for the storage and reading of the images. All the other data is only accessible over the OGSA–DAI service by Inspector Computer.

4.3 Parallelization of Image Comparison

As already mentioned the search finds thousands of images and so just as many comparisons have to be done. This exceeds the resources provided by a single workstation – the comparisons have to be parallelized. As Inspector Computer is using GT 4 the WS–GRAM (Web Service Grid Resource Allocation and Management) [13] service is integrated in it for the utilization of computers accessible by the GT 4 installation. The WS–GRAM service encapsulates the access to local schedulers e.g. like PBS (Portable Batch System) [14] or Condor [15]. Running multiple jobs in parallel decreases the calculation time drastically.

For the usage of WS–GRAM a job description file in XML (Extensible Markup Language) [16] syntax is needed. In the job description every job with parameters, the scheduler to use and the URL of the WS–GRAM service is declared. The job description is sent over a client to the service. The service creates the necessary input for usage of the specified scheduler. Afterwards the input is sent to the scheduler, the jobs are executed and the output or maybe the error messages is sent back over the WS–GRAM service to the client.

For testing purposes two schedulers are used in Inspector Computer. One version runs with Condor in our workstations. So the speed of the comparison depends on the usage of the workstations by their original user. For a fast and highly reliable testbed the CampusGrid [17] is used. CampusGrid consists of 64

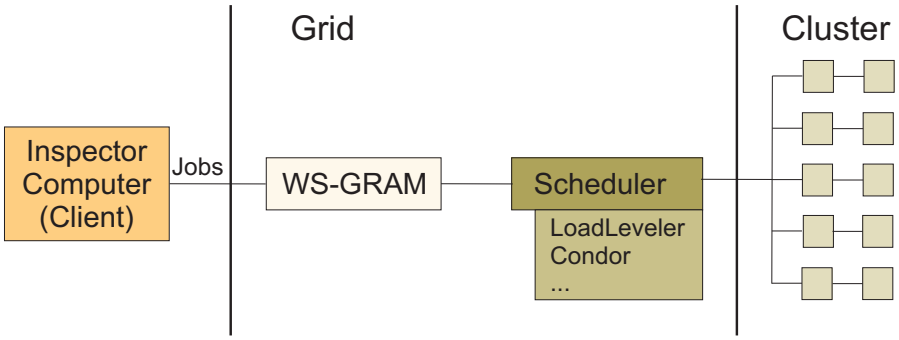

Figure 4: The schema for execution of jobs over the WS–GRAM service. First the job description is send to the service encapsulating schedulers. The service creates the input for the scheduler, executes the jobs over the scheduler and sends the output back to the client.

AMD Opteron nodes running under Linux and as well of 30 nodes running AIX accessible over an IBM LoadLeveler [18] as scheduler.

After the search is finished and all data is written to the database Inspector Computer queries the necessary datasets, creates the job description file for the chosen scheduler and sends it to the WS–GRAM service. For a faster comparison and decreasing of communication overhead several comparisons are clustered in one job. They are executed successive on the computation node of the chosen scheduler. After comparison the results are sorted and displayed in the web interface.

4.4 Image Comparison

The heart of Inspector Computer is the image comparison. Therefor Autopano–Sift [19] is adopted to the use case and used for the comparisons. Primarily the Autopano–Sift algorithm is developed to stitch panorama images. For a panorama image two images are needed, captured from a similar viewpoint. It is necessary that these images share a part of their view to make one large image – this is called stitching of the images.

To stitch the images it is important to have control points indicating where they overlap. The creation of these control points are normally done manually by the user, which is a very time–consuming process for him. Autopano–Sift is a program for the automatic creation of the control points, so that the images could be stitched very easy.

Autopano–Sift is implemented in C# and running under Mono [20] providing the .NET Framework [21] as runtime environment for several operating systems. Using the WS–GRAM service it is only possible to run jobs on Linux computers. Therefore Mono must be installed on every node the image comparison algorithm should be executed. The usage of Mono is no problem at all since Autopano–Sift

is completely developed under Mono.

The image comparison in Inspector Computer uses a modified version of Autopano–Sift. The creation of the control points is similar, the really big difference is, that it is counted how many control points in two images have the same attributes to detect the similarity. The customization of the Autopano–Sift algorithm was done and is explained in [22].

5 Results

To test Inspector Computer with the whole functionality several image comparisons to measure the quality and speed of them were made. These are twelve image comparisons with different reference images and search terms. From every of the twelve objects an image were captured and the adequate description specified. With every description Inspector Computer searched in the internet and made the image comparison with the corresponding reference image. As scheduler Condor with seven of our workstations was used.

In the mean 320 images for comparing with every of the twelve reference images were used. One image comparison takes about two minutes in average. This means about 10 hours computing time on a single workstation. The parallelization and execution through the Grid takes about one hour and a half, speeding up the calculation time drastically.

For every of the twelve image comparisons the position of the first correspondence between reference image and sorted images in the list of results was measured. From this the mean of the first matching entry in the twelve lists of results is calculated. In average the user has to look only on the first eight images to find a correspondence. A more precise explanation, on how the results are created can be found in [10].

6 Discussion and Future

The biggest disadvantage of Inspector Computer is the usage of the Autopano–Sift algorithm for the image comparison. The results of the algorithm are very good, but the implementation in C# and the necessary usage of Mono as runtime environment includes a high administration effort. Mono must be installed on every computing node in the Grid which is no problem in our computing environment, but may cause problems in other computing sites.

For the future more of the available image comparison algorithms will be tested to find out if there exists a better one. On the other hand more efficient algorithms are developed and integrated to speed the image comparison up and make it more flexible and feasible.

Therefor one example is the development of an image comparison algorithm based on registration (section 2.1). For the registration we are using the TurboReg algorithm which is optimized for a very fast execution [23]. During the registration the images are transformed with scaling, rotation and transformation in several steps done by the registration algorithm to get the best possible

result. After the registration the quality of the overlap between the target image and the image created by the registration algorithm is calculated with a quality function. For the calculation of the quality function three different methods are implemented at the moment. These are Mean–Square Difference, Squared Correlation Coefficient, Normalized Mutual Information and explained in detail in [24].

The registration algorithm is implemented as plugin in ImageJ [25]. ImageJ is a scientific image manipulation software implemented in Java. The quality functions are implemented by our group and also in Java. So it is very easy to integrate the algorithm in Inspector Computer, which will be done in near future.

Another new approach is to use Inspector Computer as image search engine in the internet. Who does not know the problem: If you have an image and want to have an image with nearly the same content. At the moment it is only possible to use the existing search engines in the internet, based on the description of the images. With this approach it could be that you are searching for an image of a cat and get an image of a dog. The planed search engine is based on the content of the images and so only getting images of a cat when searching for it.

7 Conclusion

Inspector Computer is an application used to demonstrate different Grid technologies and their cooperation in a real world application. The used technologies are the GT 4 as Grid middleware, OGSA–DAI for communication with the database and the WS–GRAM for encapsulation of the different schedulers LoadLeveler and Condor.

The biggest advantage of the described method is the automation of image comparison. Instead comparing thousands of images manually the user only has to take a look on the first eight images in the list of results in average to find a similar object if available. This is a significant advantage compared to the manual searching for stolen articles. With Inspector Computer it is also possible to make the search for one article repeatedly until it is found – even if the offer changes very often within the auction houses and must be restarted e.g. every day.

References

1. Bundeskriminalamt Wiesbaden. (2007) Polizeiliche Kriminalstatistik 2005. [Online]. Available: http://www.bka.de/pks/pks2005/index2.html
2. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall International, 2002.
3. B. Jähne, *Digital Image Processing*. Springer, 2002.
4. MySQL Project. MySQL database engine. [Online]. Available: http://www.mysql.com/
5. Sun microsystems. Java. [Online]. Available: http://www.java.sun.com/

6. The Globus Alliance. The Globus Toolkit Homepage. [Online]. Available: http://globus.org/toolkit
7. A. Savva. Open Grid Services Architecture WG (OGSA–WG). [Online]. Available: http://forge.gridforum.org/sf/sfmain/do/viewProject/projects.ogsa-wg
8. T. Banks, "Web Services Resource Framework (WSRF) – primer v1.2," OASIS Open, Tech. Rep., 2006. [Online]. Available: http://www.oasis-open.org
9. Java–Source.net. Open Source Crawlers in Java. [Online]. Available: http://java-source.net/open-source/crawlers
10. M. Sutter, "Anbindung einer webbasierten Bildsuche an eine Grid–Datenbank für die digitale Forensik," Master's thesis, Forschungszentrum Karlsruhe, 2005.
11. Sun Developer Network. JDBC technology. [Online]. Available: http://java.sun.com/products/jdbc/
12. OGSA–DAI project. Open Grid Services Architecture – Data Access and Integration. [Online]. Available: http://www.ogsadai.org.uk/index.php
13. Globus Toolkit Team. GT 4.0 WS–GRAM. [Online]. Available: http://www.globus.org/toolkit/docs/4.0/execution/wsgram/
14. Altair Grid Technologies. Portable Batch System. [Online]. Available: http://www.openpbs.org/main.html
15. The Condor Project. Condor High Throughput Computing. [Online]. Available: http://www.cs.wisc.edu/condor
16. W3C. (2007) Extensible Markup Language (XML) 1.0 (Fourth Edition). [Online]. Available: http://www.w3.org/TR/REC-xml/
17. F. Schmitz. CampusGrid. [Online]. Available: http://www.campusgrid.de
18. IBM. Tivoli Workload Scheduler LoadLeveler. [Online]. Available: http://www-03.ibm.com/systems/clusters/software/loadleveler.html
19. S. Nowozin. Autopano–Sift, making panoramas fun. [Online]. Available: http://user.cs.tu-berlin.de/~nowozin/autopano-sift/
20. (2007) Mono Project Team. Mono Project Homepage. [Online]. Available: http://www.mono-project.com/Main_Page
21. Microsoft. (2007) .NET Framework. [Online]. Available: http://www.microsoft.com/net/
22. B. Körtner, "Der "Stolen Goods Internet Detector" und sein Bildvergleich für die digitale Forensik," Studienarbeit, Forschungszentrum Karlsruhe, 2005.
23. P. Thévenaz, U. Ruttimann, and M. Unser, "A pyramid approach to subpixel registration based on intensity," *IEEE Transactions on Image Processing*, vol. 7, no. 1, pp. 27–41, 1998.
24. F. Saad, "Software Implementation and Its Evaluation of Different Object Recognition Algorithms for Digital Forensics," Master's thesis, Forschungszentrum Karlsruhe, 2007.
25. W. Rasband. (1997–2006) ImageJ, U. S. National Institutes of Health, Bethesda, Maryland, USA. [Online]. Available: http://rsb.info.nih.gov/ij/