USING GRIDS TO SUPPORT INFORMATION FILTERING SYSTEMS

A Case Study of Running Collaborative Filtering Recommendations on gLite

Leandro N. Ciuffo and Elisa Ingrà
Istituto Nazionale di Fisica Nucleare, Sez. di Catania, Via S. Sofia 64 – 95123, Catania, Italy

Keywords: Recommender systems, Personalization, Collaborative filtering, Grid computing, Distributed computing, gLite middleware.

Abstract: Today's business becomes increasingly computational-intense. Grid computing is a powerful paradigm for running ever-larger workloads and services. Commercial users have been attracted by this technology, which can potentially be exploited by industries and SMEs to offer new services with reduced costs and higher performance. This work aims at presenting a "gridified" implementation of a Recommender System based on the classic Collaborative Filtering algorithm. It also introduces the core services of the gLite middleware and discusses the potential benefits of using Grids to support the development of such systems.

1 INTRODUCTION

Recommender Systems (RS) are best known for their use in e-commerce Websites. Such systems are a type of Information Filtering system that use inputs about customers' interests to generate a list of tailored items for them. In general, the more accurate the recommendations are, the bigger will be the sales.

In recent years, Collaborative Filtering (CF) (Resnick,1994)(Cöster,2002) has proven to be one of the most popular algorithms used in Recommender Systems. Its technique essentially automates the process of "word-of-mouth" recommendations. Items are recommended to an active user based on the evaluations of users who have similar preferences.

However, CF requires computations that are very expensive and grow polynomially with the number of users and items in a system. For a large retailer like Amazon.com, with huge amount of data, tens of millions of customers and millions of distinct catalog items, generate accurate recommendations in real-time is impractical (Linden,2003). Such limitation has driven the research of a vast amount of alternative solutions. Hence, several variations of the classic CF algorithm are present in the literature, including the adoption of Bayesian network, clustering techniques, dimensionality reduction, sampling and many others – all of which reduce, in a certain extent, recommendation's quality.

In this scenario where traditional CF systems have suffered from scalability issues, Grid computing appears as an innovative approach that can be used for running ever-larger workloads and services.

Although developed over a relatively short timeframe, Grid computing is a fairly mature area which is rapidly gaining momentum in many diverse industries and among the biggest IT players like IBM, HP, Microsoft, Sun and Oracle. In addition, several international Grid projects such as EGEE (2008), BEinGRID (2006) and Biz2Grid (2008) have pushing industries and SMEs (Small and Medium-sized Enterprises) to have their business applications running in their Grid infrastructure.

Also, it is commonly recognized that computing resources can be treated as a commodity that can be sold. The provision of on-line business services can ease ICT companies from over-provisioning and from dealing with low level resources. In fact, many users are gaining confidence of outsourcing production services and part of their IT infrastructure to cloud providers such as Amazon (Bégin,2008). For this reason, the Grid community is working on the implementation of new services and accounting mechanisms that will soon allow the commercial use of Grids (Ragusa,2007).

N. Ciuffo L. and Ingrà E. (2009).
USING GRIDS TO SUPPORT INFORMATION FILTERING SYSTEMS - A Case Study of Running Collaborative Filtering Recommendations on gLite.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Software Agents and Internet Computing*, pages 12-18
DOI: 10.5220/0001845300120018

This work aims at presenting a "gridified" implementation of the classic CF algorithm as well as discussing the potential benefits of using Grids to support the development of information systems. In order to test empirically our proposed approach, we developed an application that runs on a Grid test-bed based on gLite middleware (2008) and uses CF to generate recommendations to the users of an on-line Movie Recommender System (Ciuffo,2001). To the best of our knowledge, no other group has deployed a recommender system over a gLite-based Grid.

2 RECOMMENDER SYSTEMS

Recommender Systems (RS) have been used for suggesting items (books, movies, songs, restaurants and even jokes) (Herlocker,2000) that users might like. In e-commerce environments, Recommender Systems are software applications that aim at supporting users/customers in the decision-making and buying process. The main tasks of such systems typically include the elicitation of user preferences, the construction (or update) of the user's profile and the generation of personalized recommendations. Figure 1 depicts the operation of a simple basic RS.

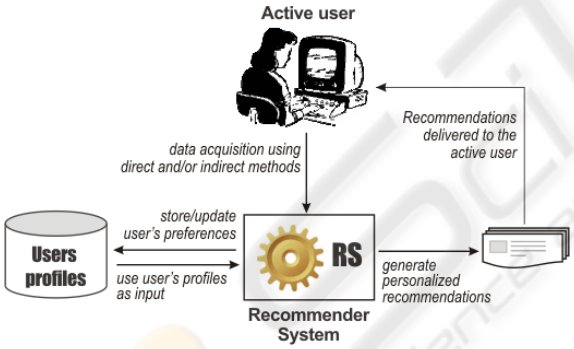

Figure 1: Operation schema of a simple RS.

2.1 Collaborative Filtering

Roughly, Collaborative Filtering (CF) considers the following hypothesis: users who agreed in the past will probably agree again in the future. For instance, imagine an active user U_a who wants to receive a movie recommendation. First, the RS must look for users who have similar profiles, i.e., users who have watched the same movies in the past and evaluated them in a similar way (the neighbors of U_a). Then, the RS will either recommend the movies best evaluated by the similar users or predict the U_a ratings based on its neighbors' profiles.

A traditional CF algorithm uses as input a **m**X**n** matrix of ratings (Table 1). Each line represents an user and each column a distinct catalog item. The cells contain normalized ratings provided by user **m** to item **n**. The empty cells indicate that the item has not been rated by the respective user.

Table 1: A simple matrix of ratings.

	item 1	item 2	item 3	item 4	item 5
John		2	2	5	
Rick	6	4	4	2	
Jully	5	4	3	2	5
Maria			5	4	4
Paul	4	5	6	3	5

In order to compute the similarities between users, a variety of similarity measures has been proposed, such as Pearson correlation, cosine vector similarity, Spearman correlation and mean square difference. The proposed application adopts Pearson correlation, since it is the most commonly technique used in the literature. Thus, the correlation between an user **a** and another user **b** is calculated by:

$$r(a,b) = \frac{\sum (N_{a,i} - \overline{N}_a)(N_{b,i} - \overline{N}_b)}{\sqrt{\sum (N_{a,i} - \overline{N}_a)^2 \sum (N_{b,i} - \overline{N}_b)^2}} \quad (1)$$

where:

$\boldsymbol{i} \in$ items evaluated by both users: ***a*** and ***b***

$N_{u,i}$ the rating given by user ***u*** to the item ***i***

$\overline{N}_u$ the average rating of user ***u*** (arithmetic mean)

Applying the correlation calculus for each pair of users in Table 1, the following matrix is generated.

Table 2: Similarity matrix.

	John	Rick	Jully	Maria	Paul
John	**1**	- 0.816	- 0.618	- 0,8	- 0.943
Rick	- 0.816	**1**	+ 0.916	+ 0.447	+ 0.314
Jully	- 0.618	+ 0.916	**1**	- 0.176	+ 0.269
Maria	- 0.8	+ 0.447	- 0.176	**1**	+ 0.754
Paul	- 0.943	+ 0.314	+ 0.269	+ 0.754	**1**

The result varies from +1 (identical profiles - perfect linear relationship) to -1 (opposed profiles). A correlation degree close to zero can't be used to infer accurate predictions.

Analyzing the Table 2, one can note that John and Paul have opposite tastes, i.e., John like the items that Paul doesn't like.

Once calculated the similarity among the active user and all other users in the matrix, the next step consists of using this information to select the neighborhoods. Basically two different approaches can be used: Correlation Weight Threshold and Maximum Number of Neighbors (Herlocker,2000). The first technique is to set a default correlation degree, where all neighbors with absolute correlations greater than the default threshold are selected. The second approach is to pick the best **N** neighbors for a given **N**. This technique may reduce the accuracy of the recommendations, but on the other hand it always manage to set neighborhoods, even for small datasets.

The third and last step of the algorithm is to infer, for each unrated item of a given user, his/her most likely rating based on his/her neighbors' ratings for that item. The rating prediction $p_{a,i}$ for the active user a for an item i can be calculated by means of the weighted average presented below:

$$p_{a,i} = \overline{N}_a + \frac{\sum_{u=1}^{n} N_{u,i} - \overline{N}_u * r_{a,u}}{\sum_{u=1}^{n} |r_{a,u}|} \quad (2)$$

where n is the number of neighbors who have rated i and $r_{a,u}$ is the similarity weight between the active user a and his neighbor u, as defined by the Pearson correlation.

One of the advantages of the CF is that it can be applied in every domain, with no additional efforts required for humans editors to classify items or tag contents. However, this technique depends on the participation of a great number of users to tune the RS. In general, the larger the number of evaluations provided, the more accurate the recommendations will be. For this reason the CF algorithm is best suitable to operate with large databases and therefore suffers from complexity issues.

2.1.1 Worst Case Complexity

The most expensive computation of the classic CF is the calculation of the user-to-user similarities. In order to deal with this, major e-commerce systems use to carry out such computations offline and feed the database with updated information periodically (Linden,2003). This approach may allow them to provide quick recommendations based on pre-computed similarities, but it fails on providing recommendations with the highest degree of confidence, since ratings submitted between two offline computations are not considered. The worst case complexity of maintaining a similarity matrix with the Pearson correlation between every pair of users is $O(m^2n)$ (Papagelis,2005), where m is the number of users and n is the number of catalogue items. Alternatively, if the user similarities are not pre-computed offline, they need to be calculated at the time a recommendation is requested. In this case, there is no need for computing the whole user similarities matrix, but only the similarities between the active user and all the others. This computation has a computational cost of O(mn).

3 GRID COMPUTING

A computational Grid is a geographically distributed system aimed at putting together large sets of heterogeneous resources (computing power, storage space, software applications, data etc.) among communities of users federated in the so-called Virtual Organizations (VOs). It derives its name from the fact that its idealistic goal is to grant access to computing resources in a transparent and easy way as the Power Grid does with the electrical power. VOs are sets of individuals and institutions that agree upon common policies for sharing and accessing resources.

The interaction between users and the resources is made possible by a software layer known as *middleware*. It is a set of components that provides the user with high-level services for scheduling and running computational jobs, accessing and transferring data, obtaining information on the available resources and so on, covering the most relevant operations that can be done in a Grid environment.

Currently, there is not a single Grid (as there is one single "Internet"). Instead, there are many Grid initiatives around the world created by group of organizations wanting to share their local resources to increase their individual access to the overall Grid resources.

The work presented in this paper makes use of the GILDA Grid (Babera,2008), a fully fledged Grid test-bed devoted to training and outreach activities which has been adopted as the official training infrastructure by many Grid projects all around the world.

GILDA adopts the gLite middleware (2008), an European solution developed by the CERN jointly with the EGEE project. Next section briefly presents the main characteristics of the gLite architecture.

3.1 The gLite Middleware

Distributed under a business friendly open source license, gLite integrates components from the best of current middleware projects, such as Condor (Thain,2003) and the Globus Toolkit (2007), as well as components developed for the LCG project (2008).

The gLite middleware is based on the concept of "Job Submission". A job is an entity which contains information about all the stuff needed for a remote execution of an application, such as its executable program, the environment settings, the input data and the expected output files to be retrieved. All these parameters are defined in a text-file written in the Job Description Language (JDL) syntax, that is a high-level specification language based on the Classified Advertisement language (2004).

End users can access the Grid through a Grid component called User Interface (UI), which is a client properly configured to authenticate and authorize the user to use the Grid resources. When a user wants to execute a computational job, he/she composes a JDL file and submits it to the Workload Management System (WMS), which is the service in charge of distributing and managing tasks across the computing and storage resources. The WMS basically receives requests of job execution from a client, finds the required appropriate resources and then dispatches and follows the jobs until completion, handling failure whenever possible.

The jobs are sent to Computing Elements (CEs), which manage the execution queue of the Worker Nodes (WNs), the computers where a job actually run. In other words, a CE acts as an interface to the computing farm installed in a Grid site.

Once the task has been accomplished, all the output files listed in the JDL are packed in the Output Sandbox and sent back to the WMS. Finally, the user can retrieve the Output Sandbox onto his/her UI. Applications that need to manage large data files (either as input or output) can store/retrieve the data by accessing directly the Grid Storage Elements (SEs) using the data management API provided by the middleware.

It is important to mention that although data files are held in SEs, they are made available through File Catalogues, which record information for each file including the locations of its replicas (if any). The File Catalogue adopted by gLite is the LCG File Catalogue (LFC). This tool allows users to view the entire Grid as a single logical storage device.

Figure 2 roughly presents the interaction among the aforementioned Grid elements. Please notice that although only one SE and CE are represented in the picture, a Grid infrastructure might have hundreds of these elements geographically distributed.

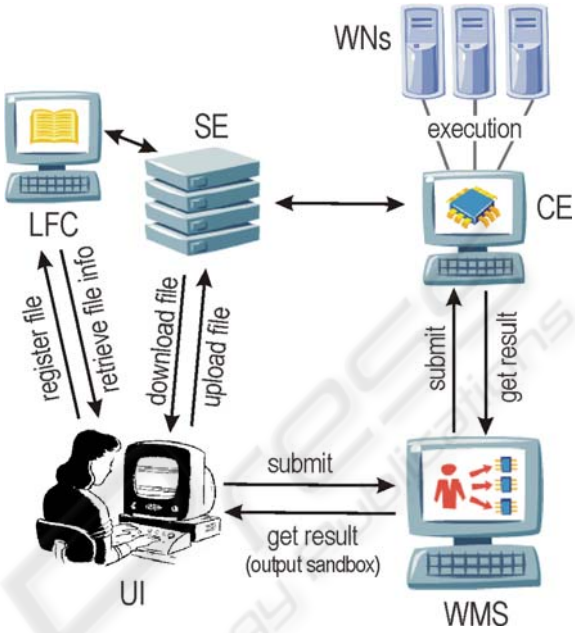

Figure 2: Schematic diagram of some Grid elements.

Another Grid service relevant to this work is the Metadata Catalogue, which permit users to attach metadata to the files stored in the Grid. Typically, each file being described has a respective entry in the catalogue. The entries are described by user-definable attributes, which are key/value pairs with type information. The Metadata Catalogue adopted by gLite is the Arda Metadata Catalogue Project (AMGA,2006).

For further definitions about the gLite architecture, the reader is requested to consult (Laure,2006).

4 GRIDIFICATION APPROACH

Just like "webifying" applications to run on a web browser, Grid users need to "gridify" their applications to run on a Grid. This process may comprises the creation of additional bash scripts as well as changes in the original source codes in order to include APIs to directly interact with Grid components. In some cases, it is also necessary to stop using services and libraries that are not supported by the standard distribution of the adopted Grid middleware.

As aforementioned, we developed a RS to recommend movies at Cinefilia website (Ciuffo,2001). To be able to get movie recommendations, Cinefilia visitors need to sign in and to rate at least 20 movies. The rating scale varies from 0 to 6 for each watched movie. There is also an option that enable users to indicate the movies that he/she hasn't watched yet.

Altogether, more than 800 movies are available to be rated, from classics to the latest releases. Our current dataset consist of 32,817 ratings (ranging from 0 to 6) provided by 327 unique users. It is important to remark that this is considered a small dataset for what concern the measurement of either the accuracy of the generated recommendations or the performance of the proposed workflow. Both analyses are out of scope of this work.

The implementation strategy for porting our Recommender System on the GILDA test-bed is depicted in Figure 3 and explained as follows: users can freely interact with the Cinefilia website to rate as much movies as they can. Cinefilia makes use of a standard MySQL database to store and retrieve information about its users (1). In order to generate recommendations, a pre-processing script must be executed in the Grid User Interface. This is a bash script that can be launched either on-demand or by using cron to run it many times a day on a schedule. The pre-processing script is in change of downloading all ratings from the on-line database (2); chopping the users' ratings into several text-file strings - one per user (3); registering the files in the LFC (4) and storing them on a Grid SE (5). The pre-processing script should also dynamically create the JDL file (6). In our approach we are using parametric Jobs (Giorgio,2008) where each user ID is a parameter of the actual executable file called "*recommender*". This is a compiled code originally written in C that implements the classic CF algorithm and calculates recommendations for a single user – specified as a parameter. The "*recommender*" code uses as input the text files obtained from the database and generates as output a .SQL file containing recommendations to the specified user. For the reader's convenience, the JDL file is shown in Figure 4.

The last action of the pre-processing script is to submit the parametric job to the WMS (7), which dispatches multiple jobs (one job per user, as defined in the parameters list) to the available Computing Elements (8). The usage of parametric jobs ensures the distributed characteristic of our approach, where each user can have his/her recommendations calculated simultaneously by different Worker Nodes (in different Computing Elements queues).

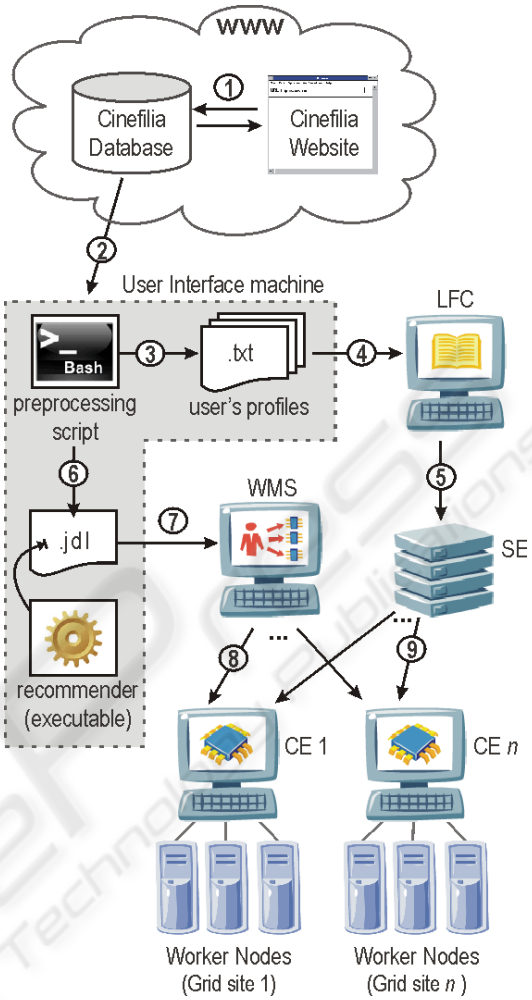

Figure 3: Implementation workflow.

```
[
Type = "Job";
JobType = "Parametric";
Executable = "start-recommender.sh";
Arguments = "_PARAM_";
StdOutput = "output_PARAM_.out";
StdError = "error_PARAM_.err";
InputSandbox = {"recommender",
"start-recommender.sh"};
OutputSandbox = {"recommendations.sql",
"output_PARAM_.out","error_PARAM_.err"};
Parameters =
{userID1,userID2,userID3,...,userID320};
]
```

Figure 4: JDL file generated by the pre-processing script.

One can note that the "*recommender*" code is included in the Input Sandbox and therefore transferred to be executed in the Worker Nodes.

Also, the text-files previously stored in the SE are downloaded to be used as input files by each job execution (9).

At the end of the execution, it is possible to retrieve multiple output files (one .SQL file per user). Therefore, another script must be launched in order to assemble all these files into a single one containing recommendations for all users. The ultimate step of this workflow is to update the on-line database using the generated .SQL file.

To keep track of all executions of our Recommender System, the post-processing script is also in charge of storing some relevant statistics into the AMGA Metadata Catalogue, such as the date and time when the .SQL file was generated, the total of ratings, the amount of users entitled to get recommendations, the number of generated recommendations and so forth.

5 CONCLUSIONS AND FUTURE WORKS

Grids first emerge within scientific communities, like High Energy Physics (HEP) experiments. However, the enormous research activity in recent years has contributed to the development of new areas of interest. Commercial users have been attracted by this technology, which can potentially be exploited by industries and SMEs to offer new services with reduced costs and higher performance.

This expansion from science to business is nearing Grids to "utility computing", where computing power is viewed as a utility, available on a pay-as-you-use basis, like gas or electricity. This is not yet the case, but there are several ongoing initiatives to develop new tools and Grid services that should allow the outsourcing of computing resources in the short term.

Although the case study presented in this paper did not focus on a comparative analyses between our approach and other implementations of CF algorithm, it is interesting to note that the resources required to run our Recommender System are owned by other entities, letting our own resources (computers and storage) free to perform other tasks. This is an immediate benefit of using Grids, specially for SMEs that cannot afford to have their own computer farm to run their simulations/algorithms.

The distributed approach used in this case study has helped to reduce the time consumption of the original algorithm, since each job launched to the Grid is in charge of calculating the recommendations for one single user - O(mn). The overall execution time to generate recommendations to all users will depend on the number of Computing Elements and Worker Nodes available in the Grid. The best scenario would be to have all jobs running simultaneously in different Worker Nodes. The bigger the number of free CPUS in a Grid, the better is the chance of this scenario occurs.

The strategy of fetching data from the database and splitting it into several text-files is due to a lack of a Grid enabled DataBase Management System (DBMS) supported by the middleware and available to the users. However, since many applications need to access, manage and process huge amount of data, there are several initiatives trying to face this important challenge. One of them is the Grid Relational Catalog Project (GRelC,2007), which provides a data access interface to access standards DMBS (MySQL, PostgreSQl, Oracle etc.) in a Grid environment.

As a future work, we intend to create a new version of our Recommender System exploring the GRelC service. We also intend to deploy our system in the EELA-2 production Grid infrastructure, since GILDA is devoted to learning purposes only. The results and discussions of both experiments will be presented in future papers.

REFERENCES

Amga (2006). *The gLite Grid Metadata Catalogue*. Retrieved February 8, 2009, from: http://amga.web.cern.ch/amga/

Barbera, R., Ardizzone, V., Ciuffo, L.N. et al. (2008) *Grid INFN virtual Laboratory for Dissemination Activities - GILDA* (2008). 6th International Conference on Open Access, Lilongwe, Malawi. GILDA portal available at: https://gilda.ct.infn.it/

Bégin, M.E. (2008) *An EGEE comparative study: Grids and Clouds - Evolution or Revolution?* Retrieved February 8, 2009, from: https://edms.cern.ch/file/925013/4/EGEE-Grid-Cloud-v1_2.pdf

Beingrid. (2006). *Business Experiments in Grid.* Retrieved February 8, 2009, from: http://www.beingrid.eu/

Biz2Grid. (2008). *Moving Business to the Grid*. Retrieved February 12, 2009, from: http://www.biz2grid.de

Ciuffo, L.N. (2001). *Cinefilia website*. Retrieved February 12, 2009, from: http://canalcinefilia.com.br

ClassAd. (2004). *Condor Classified Advertisements.* Retrieved February 12, 2009, from: http://www.cs.wisc.edu/condor/classad

Cöster, R. (2002) *The architecture and implementation of a system for collaborative and content-based filtering*, Technical Report, Stockholm University.

Eela-2. (2008). *E-science grid facility for Europe and Latin America*. Retrieved February 7, 2009, from: http://www.eu-eela.eu

Egee. (2008). *Enabling Grids for E-sciencE*. Retrieved February 8, 2009, from: http://www.eu-egee.org/

Egee and Business (2008). Retrieved February 8, 2009, from: http://www.eu-egee.org/index.php?id=120

Garcia, A. C. B. & Ciuffo, L.N. (2005) *Applying the HYRIWYG incentive mechanism in a Recommender System*. In: IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), Compiègne. Proceedings of the WI'05. IEEE Computer Society, 2005. pp.770-773.

Giorgio, E. (2008). *Gilda Wiki - Quickstart for complex jobs*. Retrieved February 7, 2009, from: https://grid.ct.infn.it/twiki/bin/view/GILDA/WmProxyUse

gLite (2008). *Documentation*. Retrieved February 7, 2009, from: http://glite.web.cern.ch/glite/documentation/

Globus Toolkit. (2007). *Homepage*. Retrieved February 7, 2009, from: http://www.globus.org/toolkit/

Grelc. (2007). *Grid Relational Catalog Project*. Retrieved February 7, 2009, from: http://grelc.unile.it/home.php

Herlocker, J.L. (2000). *Understanding and Improving Automated Collaborative Filtering Systems*, Ph.D. Thesis, University of Minnesota.

LCG (2008). *Large Hadron Collider (LHC) Computing Grid*. Retrieved February 7, 2009, from: http://lcg.web.cern.ch/LCG/

Laure, E., Fisher S.M., Frohner, A., Grandi, C. et al (2006) *Programming the Grid with gLite*. Computational Methods in Science and Technology. pp.33-45.

Linden G., Smith, B. & York J. (2003). *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*. IEEE Internet Computing, ISSN:1089-7801, v.7, Issue 1. pp.76-80.

Papagelis, M., Rousidis, I., Plexousakis, D. & Theoharopoulos, E. (2005). *Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms*. Lecture Notes in Computer Science. Heidelberg, Germany: Springer-Verlag, ISSN 0302-9743, Issue 3488, 2005. pp.553-561.

Ragusa, C., Arinisi, S., Longo F. & Puliafito, A. (2007). *A Grid-based Infrastructure for Business Applications*. Proceedings of the Symposium "Grid Open Days at the University of Palermo", Italy, ISBN: 978-88-95892-00-9. pp.205-211.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. & Riedl, J. (1994). *GroupLens: an open architecture for collaborative filtering of netnews*. Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'94), Chapel Hill, NC, EUA. ACM Press, pp. 175-186.

Shen, R.-m., Yang, F., Han, P. & Xie, B. (2005). *PipeCF: a DHT-based Collaborative Filtering recommendation system*. Journal of Zhejiang University SCIENCE, China, ISSN 1009-3095, v.6A; pp.118-125.

Thain, D. Tannenbaum, T. & Livny M. (2003) *Condor and the grid. Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and T. Hey, Eds. John Wiley & Sons Inc.