Proceedings of the First EELA-2 Conference
B. Marechal et al. (Eds.)
CIEMAT 2008

Enabling the Grid for Experiments in Distributed Information Retrieval

Raúl Ramos Pollán[1,2], Álvaro Barreiro[2]

[1]*Centro Extremeño de Tecnologías Avanzadas, Trujillo, Spain*
raul.ramos@ciemat.es
[2] *Information Retrieval Lab, Dpt. Computación,University of A Coruña, Spain*
barreiro@udc.es

Abstract

Information Retrieval (IR) deals with the representation, storage, organization of and access to information items. It represents the core of the search engines of today and it is behind their popularity and usefulness. Behind this success, lies a well established experimental methodology against large corpuses of data (documents, queries and relevance judgements) through which new IR models and software implementations are accurately tested and evaluated. Recently, distributed IR models and technologies are becoming increasingly important as there is an emerging need to search throughout federated collections of documents, such as the ones that might exist in Grid environments, where different resource centers might possess different collections of documents, needed to be searched in a distributed manner upon an information request from a user.

Testing distributed IR models and technologies in a systematic manner requires significant amounts of resources as each time we want to test a new model or software we need to deploy many collections of documents and run thousands of queries against them measuring effectiveness and efficiency. Our goal is to use the Grid itself for such purpose.

In this work we present the set of technologies we developed in order to be able to run large scale distributed IR experiments on Grid infrastructures. These technologies allow us to easily design, setup and run distributed IR experiments using standard Grid job submission mechanisms. We accomplish this by tightly integrating virtualization and cloud computing techniques within a gLite environment in a model that can be easily generalized to be used by other scientific disciplines. This, of course, also constitutes a significant step forward in making Grid infrastructures easily exploitable by the IR community.

1. Introduction

Information Retrieval (IR) constitutes nowadays a broad field that might be defined as follows [1]:

> "*Information Retrieval is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within very large collections (usually stored in computers)*"

Although it is a field with an established tradition, until the development and popularization of Web it remained a narrow area of interest mainly to librarians and information experts [2]. The arrival of the Web constituted a new and huge field of application for IR, being largely responsible for the usefulness and efficiency of today's search engines and becoming the most frequent form of information access [1]. This success is founded on a solidly structured community, experimental methodology and tools. Recently, research in distributed IR [3,4] is allowing users to search efficiently in federated collections of documents scattered throughout independent document sources. This extends IR experimental methodology and tools to the distributed case providing a well established framework for testing and evaluating new tools, models and technologies.

On the other side, Grid (and lately Cloud Computing) technologies [7,8] allow ubiquitous exploitation of computational and storage resources distributed throughout resource centres of dynamic nature. These technologies provide the ground for users to seamlessly run their applications over processors and data owned by different resource centres available at any given time. In this line, virtualization [9] is becoming the means to decouple the management of the physical infrastructures from the services offered to users. This way, users are starting to encapsulate their applications within virtual machines which then job schedulers dynamically deploy over the physical infrastructure available for exploitation at any moment.

Our aim is to use Grid infrastructures for large scale experiments in distributed IR by encapsulating IR tools and experiment setups within virtual machines which are then deployed to arbitrary physical infrastructures by job schedulers.

This paper is structured as follows. Section 2, describes briefly IR experimental methodology and tools and the goals of our work. Section 3 describes the virtualization and Grid technologies we use. Section 4 explains the architecture we developed and the issues we had to sort out to finally enable distributed IR experiments on the Grid. Section 5 describes our future work and explores the possibilities for applying the technologies developed for IR and other fields.

2. Experiments in Information Retrieval

2.1 Information Retrieval

Information Retrieval is concerned with locating documents within collections that are relevant to a user's information need. In a typical case, document collections are indexed, users express their information need as a query to some front-end, indexes are searched for relevant documents and, finally, users retrieve the located documents by browsing throughout the search results. IR research deals with all aspects of this overall process, including tools and methods for indexing, query processing, searching indexes, document representation languages and models, crawling document collections (such as the Web), etc.

When evaluating IR systems the first concern is measuring their **effectiveness**. Given a document collection and a user's information need, each document is either *relevant* or *non-relevant* with respect to that information need. In general, evaluating the effectiveness of a given IR tool is about measuring how well it retrieves the relevant documents for each information need. **Test collections** are used to obtain objective and comparable measures of effectiveness of different IR systems, and building useful test collections is a fundamental activity of the IR community. A test collection consists of three things:

- A document collection
- A test suite of information needs, typically expressed as queries
- A set of relevance judgements, which are binary assessments of whether each document is relevant or not to each information need. Relevance judgements are made by "assessors" that manually judge the content of documents against queries.

There exist several test collections for such purpose, being **TREC** (Text Retrieval Conference) [5] the most widely used one. TREC is held annually by the US NIST (National Institute of Standards and Technology). It is organized in tracks that evolve throughout different conferences and each track aims at a specific problem to be solved or context within which IR is to be applied. For instance there is a "Blog" track [10] that aims at dealing with the particular characteristics of blogs (frequent update, posting, content, etc), or a "Filtering" track, aimed at filtering incoming information feeds (such as RSS channels) based on a user's information need. The most general track is the "Ad hoc" one, that aims at satisfying an arbitrary user need and provide the basic test collections upon which other tracks are often built. At each conference, research groups submit their systems and results which are evaluated against the test collections. Also, TREC sets forth the mechanisms for gathering relevance judgements and comparing different IR systems.

As an example, the .GOV2 collection introduced in TREC 2004 contains more that 25 million documents, more than 15 million terms, 17.7 KB of average document size and a total of 426 GB.

IR systems are evaluated through the runs they made by submitting the test queries to the test collections and then measuring their results against the relevance judgements. Two main measures are used for this purpose. **Recall** is the fraction of the relevant documents that are retrieved for a given query and document collection. **Precision** is the fraction of retrieved documents that are relevant.

Secondly, IR is concerned with **efficiency**, this is, how different IR systems perform. In general, IR seeks practical implementation of IR models, requiring to perform well with large document collections (such as the web), many queries, etc. There exist many measures in this respect, but IR pays special attention on avoiding perceivable performance degradation with respect to the size of indexes and query complexity.

Throughout this process many **IR toolkits** have emerged, covering different issues within IR (preprocessing, indexing and searching under different retrieval models)

2.2 Distributed Information Retrieval

Distributed Information Retrieval is concerned on retrieving documents scattered throughout different collections, rather than in a single collection. As described in [3] distributed IR aims at accounting for the distributed location and access to information sources through computer networks. As such, it requires additional problems to be addressed with respect to centralised IR (as described in the previous section):

- **Resource description**: The contents of each collection must be described
- **Resource selection**: Given an information need and a set of resource descriptions, a decision must be made regarding which collection(s) to search
- **Results merging**: Integrating the ranked lists returned by each one of the selected collections into a single coherent ranked list.

Therefore, in a typical use case, the user expresses its information need as a query, a subset of collections are selected for searching based on that query, the query is sent to the selected collections and, finally, the results are merged before returning them to the user.

The experimental methodology for distributed IR follows the same spirit of centralized IR as described above, with the difference that test collections are split into several subcollections to simulate a distributed IR environment [3]. This way, we have complete sets to evaluate systems providing implementations for the different parts of the process (descriptions of the collections, merging results, selection of collections, etc.).

Figure 1 represents the different components that participate in a prototype use case in distributed IR. Notably, we need (1) a set of **collection servers** holding each collection and (2) a **query broker** that receives the user query, selects which collections to query, effectively queries them and merges the received results to pass them back to the user.

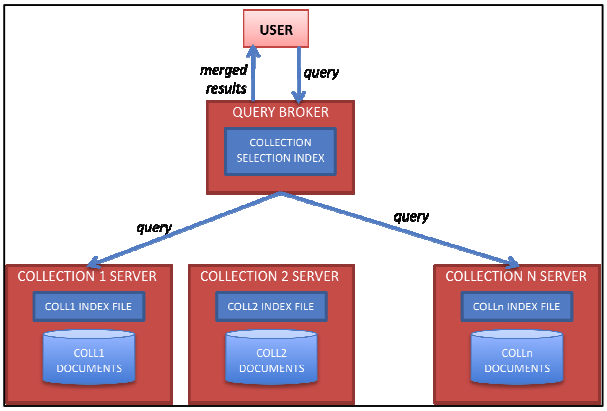

Figure 1 Distributed IR use case

As with classical IR, we want to evaluate both the **effectiveness** and **efficiency** of distributed IR systems. However, query brokers and the collection servers are connected through communication networks, meaning that we need to add software layers and protocols in order to effectively integrate distributed IR systems into production systems. The complexity of those layers will depend on the features we wish to introduce (authentication, encryption, integrity, etc). Although effectiveness should not be altered (providing no data is corrupted by the communications protocols, etc.), efficiency is a real concern as additional software layers introduce performance penalties affecting the usability of any toolkit.

2.3. Goals for Experiments in Distributed Information Retrieval

Available toolkits for distributed IR focus on the effectiveness of the components introduced (collection selection, result merging, etc.) and there exist toolkits implementing models that achieved satisfactory results for test collections that have been split. In order to exploit this effectiveness we must make sure that the software layers needed to enable IR on a networked distributed environment introduce reasonable performance penalties, keeping the system usable.

Therefore, in our work we focus on the efficiency of distributed IR systems. We start from a well known IR toolkit (the LEMUR toolkit) providing proven effective functionality for distributed IR (referred to as **DistLemur**) including collection representation, selection and result merging. As DistLemur focus on proving the effectiveness of its functions, it performs distributed IR experiments sequentially on a single system, this is, without doing the actual distribution of the components. In this context our goals are:

Goal 1. **To enable DistLemur for distributed IR experiments on physically distributed environments.** We want to separate each one of DistLemur components (query broker and collection servers) wrapping them into Java applications, based on RMI (Remote Method Invocation) for communications, that are then distributed across the network. In this process we also want to architecture the system so that selected collections for each query are all queried in parallel. We name this **RMIDistLemur**. Then, taking standard test collections for distributed IR, we run queries against the distributed system measuring:

- **That effectiveness is preserved,** by testing that the results obtained from RMIDistLemur are identical to DistLemur.
- **The overall time taken answering each query**

By doing this, we are now in a position to evaluate the impact of the introduced software layer (Java RMI) in the efficiency of the system. In particular we are interested in its scalability and thus we want to ensure that any performance penalty introduced is *reasonably independent* from the number of distributed collections and the number of queries issued. This is, that times required to answer queries do not grow as we add new collection servers beyond what is strictly required due to communications.

Goal 2. **To enable deployment of arbitrary collection servers over a physical infrastructure.** We want to devise different strategies for experiments, notably with many collection servers deployed over different physical machines. Current test collections for distributed IR are split into around 100 collections which means managing deployment of collection servers into an infrastructure of several physical machines (as many as collection servers). In order to gain thorough knowledge of the behaviour of such distributed systems we want to devise many different experiment setups (with different numbers of queries, collections, etc.) and, therefore, we need to develop mechanisms to be able to easily deploy any number of collection servers over any number of physical machines, run arbitrary queries against them, gather results, measure times, etc.

2. Technologies for Experiments in Distributed IR

An experiment in distributed IR is made of an arbitrary number of collections against which we run an arbitrary number of queries, testing for the integrity of the results and the performance of the system. As mentioned, we seek to have flexibility and agility in setting up and running any experiment. As a single experiment combines collections, queries and machines we need a technological framework allowing us to easily deploy any number of collection servers on a physical infrastructure against which to run any number of queries.

To accomplish this we face two main problems when preparing an experiment:

- **the packaging problem**: setting up the collection servers that make up the experiment
- **the deployment problem**: deploying the collection servers over a physical infrastructure.

We use virtualization to sort out the packaging problem, and job scheduling over a Grid infrastructure to sort out the deployment problem, as described in the following two subsections.

2.1 Encapsulating collection servers within virtual machines

In order to prepare a physical infrastructure for distributed IR experiments we need to deploy the appropriate software packages and configurations into its machines. In our case we want to deploy RMIDistLemur in each collection server along with the collection indexes it serves. In this, we face the problem of ensuring the software install and runs healthy on the machines where it is deployed. This is in general difficult to achieve in arbitrary physical infrastructures since there are often incompatibilities between user applications (such as RMIDistLemur in our case) and the base operating systems of the infrastructure. Such incompatibilities range from libraries availability and kernel versioning to processor architectures, drivers, etc.

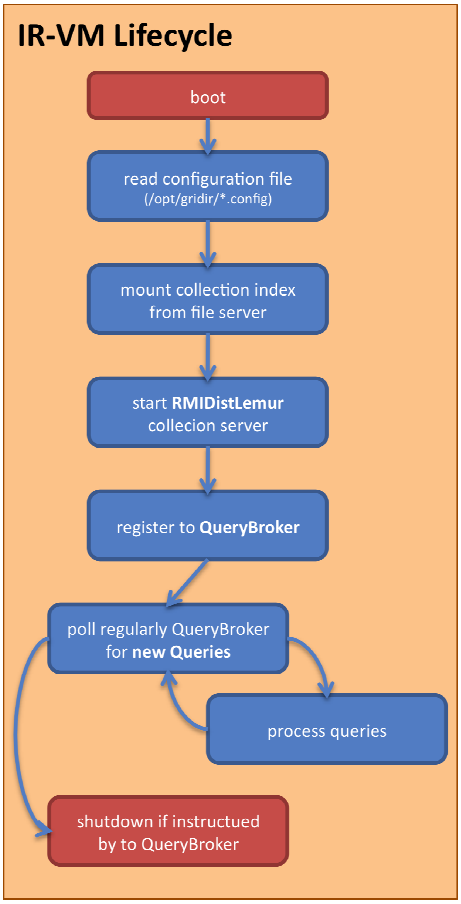

Figure 2 IR-VM lifecycle

Recently, this problem is starting to be addressed by enabling the physical infrastructure to run virtual machines (VM) that encapsulate user applications, through paravirtualization such as Xen [13] or full virtualization such as VMWare [14] or others. This leads to the notion of **software appliances**, decoupling completely the physical infrastructure from application software and providing great flexibility in the deployment, management and provisioning of resources to users and applications. In this context, nowadays, performance looses incurred by virtualization (around 5% to 10% of CPU power using standard benchmarks [9]) are largely overcome by the gained flexibility and decoupling.

Therefore, we set forth to encapsulating collection servers within virtual machines. For this, we build a **model collection server VM** (referred to as **IR-VM**), ready to be cloned and to serve any collection specified in its startup parameters. In our experiments, we have a NFS file server containing the indexes of all subcollections of the test collections we use. New collection servers are created by cloning the IR-VM and changing its startup parameters to mount the corresponding collection index volume from the NFS file server. Figure 2 shows the IR-VM lifecycle.

As it can be seen, an IR-VM is prepared so that at boot it mounts a collection index volume, starts RMIDistLemur and registers with a query broker. This way as the query broker receives queries to process, selects only among the collections servers previously registered and sends the queries to them. This gives us also great flexibility in the management of the query broker as its dynamic registry copes with different collection servers configurations as they change over time. Figure 3 shows an example startup parameters file with which a newly cloned IR-VM is started.

CONFIG FILE for one IR-VM (ap88_1.config)

```
collection.to.serve          = ap88_1
collection.idx.server (*)    = nfs://colls-03.ceta-ciemat.es/data
num.results                  = 500
querybroker.addr             = rmi://qbroker.ceta-ciemat.es/qbroker
querybroker.poll.interval    = 1000
querybroker.max.retries      = 20
querybroker.retry.timeout    = 500
autoshutdown                 = yes
```

Figure 3 IR-VM startup file

2.2 Using job submission mechanisms to deploy virtualized collection servers

Once collection servers are packaged within virtual machines as explained above we need, for each experiment, to deploy them into the available physical infrastructure. Being aware of the large number of collection servers required for the experiments (in the order of 100) we cannot generally count on a dedicated infrastructure. Therefore for each experiment run, we want to use existing computer infrastructures onto which we deploy our set of VM encapsulated collection servers using their standard job submission mechanisms.

To do this, first we create a **vm repository**, which is simply a file server containing the model IR-VM. In the future it will probably contain other machines, new versions of the IR-VM, etc. Then, for each experiment run consisting of n collection servers we create n jobs with a collection server configuration file and a script controlling the job execution. We send the n jobs to the job scheduler which distributes them on the computing nodes. Then, when the job arrives to a computing node it (1) clones the IR-VM from the vm repository, (2) places the collection server configuration file within the cloned IR-VM, (3) boots the cloned IR-VM and (4) waits for it to shutdown. Figure 4 shows this process.

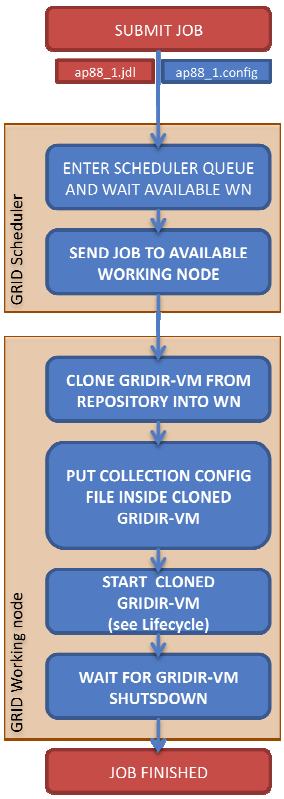

Figure 4 IR-VM deployment

3. An Architecture for Distributed IR Experiments on Grid infrastructures

With all this we obtain the architecture as depicted in Figure 5 for our distributed IR experiments on Grid infrastructures using virtualization. Its components are:

- **The IR-VM virtual machine model**, that encapsulates the RMIDistLemur software ready to serve any collection server
- **The vm-repository**, that contains the IR-VM and is accessed by computing nodes as they receive jobs defining collection servers
- **The collection file servers**, containing the indexes of the subcollections to be used by the different collection servers as they arrive to the computing nodes.
- **The query broker**, that distributes queries to selected collection servers among the ones available.
- **The job scheduler**, that distributed jobs containing collection servers to the computing nodes.
- **The job definitions** for a given experiment, describing the collections that will serve a particular experiment setup.
- **The computing nodes**, the host collection servers within IR-VMs as delivered by the job scheduler.

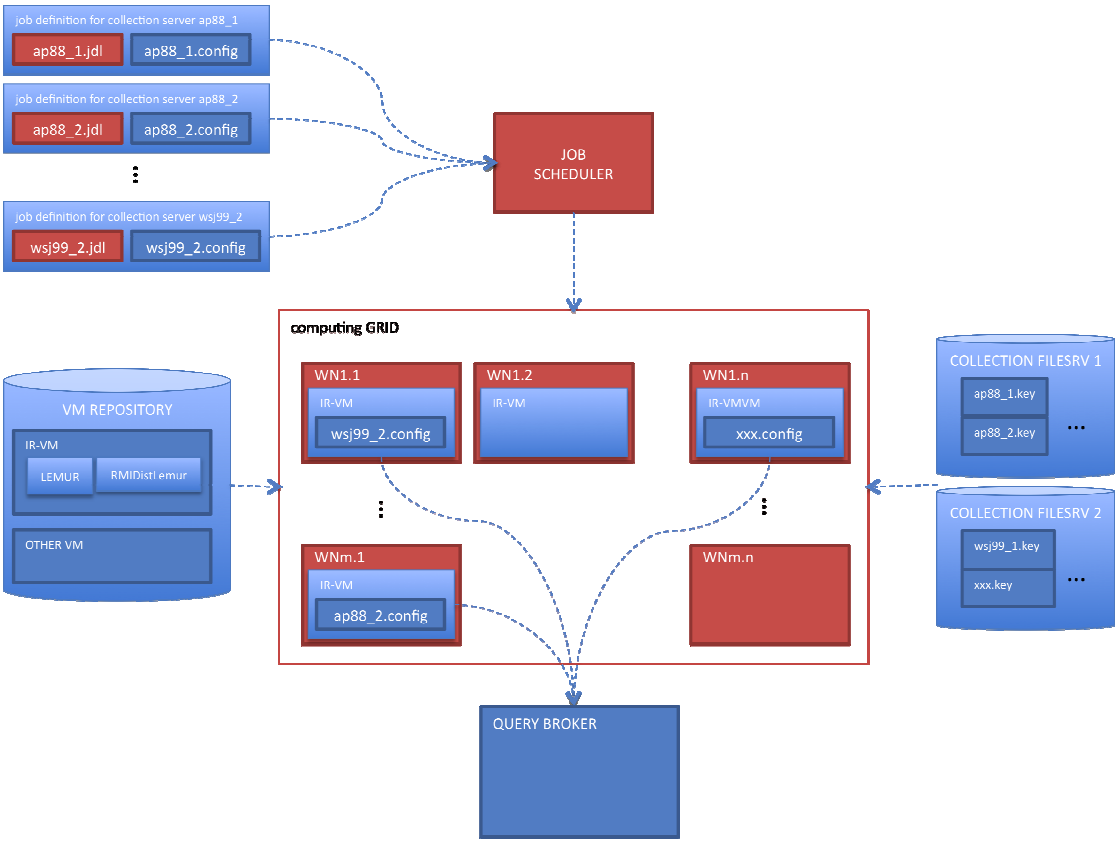

Figure 5 Architecture for distributed IR experiments on a Grid

With this architecture, we are now able to run distributed IR experiments on specific test collections over a Grid infrastructure by (1) Making available in the vm-repository an updated version of our IR-VM, (2) Making available in the collection fileservers the test collection split into subcollections, (3) Deploying the query broker, (4) Creating a job per collection server we want to deploy, (5) Submitting the jobs to the job scheduler and (6) Sending queries to the query broker

In fact, job schedulers were originally designed to deploy batch jobs on clusters with no or little interaction with the outside (other than interchanging occasionally input and output files). We are somehow abusing this mechanism to deploy applications that respond to an external controlling agent (the query broker). This is a technique know as **Pilot Jobs** [6] and it is widely used in order to better exploit Grid infrastructures. In our case, the cloned IR-VMs that arrive to the computing nodes would be the pilot jobs. A good example of using these techniques is the ALICE experiment at CERN [15]

3.1. Issues arising

By adopting this architecture we encountered two problems, one caused by the other. First we encountered the **pilot jobs communication problem**. It arises from the fact that, typically, computing nodes are only visible to the job scheduler and not accessible in general from outside that scope, although they are able to initiate contact to outside servers. In particular, this affects the way the query broker distributes queries to collection servers in what, initially constituted a push model. This is, upon new queries, the query broker selects what collections to query and then initiates contact with them to transfer the query and get back the results. As

collection servers are in computing nodes they cannot be seen by the query broker and therefore it cannot initiate any conversation with them.

To sort this out we adopt a **pull model** where, once deployed, every collection server registers with the query broker and then polls regularly (say every half a second) for new queries. Whenever the query broker has a new query to distribute waits for the selected collection servers to poll and then distributes the query. Of course, this is done by adding the appropriate connection control mechanisms to the query broker (verifying selected collections have been previously registered, ignoring duplicate answers, etc.)

As a consequence of adopting a pull model a new problem arises. This is the **measurement synchronization problem**. It is due to the fact that, in a push model, the query broker measures query answering times starting off from the time it decides to distribute a query to a set of selected collections, until the time it receives all the answers and merges the results. As we are using now a pull model, when the query broker decides to distribute a query it **must wait** for all selected collections to poll for a new query. This period introduces an inherent delay in the measurements that can bias any of the results. We have two options to sort this out. First, we can set the polling interval of all collection servers to the same value, say half a second. Then we know that all measurements will have a delay of a maximum of half a second and therefore we could compensate it statistically providing we make enough experiments to average it. In a more complex alternative, we can program the query broker so that, whenever it decides to distribute a new query to a set of collections, holds the connections opened by each collection server as they poll until all collection servers have made their poll. Then, releases the query to all collection servers at the same time starting the measurement at that instant. This way we are sure we start measuring query processing time for all collection servers at the same time. RMIDistLemur implements this last alternative.

4. Conclusions and future work

We have shown a set of technologies and architectures to allow distributed IR experiments over Grid infrastructures and have tested their feasibility through small experiment sets. Through the use of virtualization we are able to encapsulate collection servers in self-contained virtual machines and decouple completely our experiments from the underlying physical infrastructure. Through the use of job schedulers we are able to deploy any experimental setup in any virtualization ready Grid infrastructure.

This, in fact, lies at the heart of emerging Cloud Computing technologies that are taking control of heterogeneous computing resources which are becoming more and more common. This can be seen in the evolution of the TOP500 Supercomputing sites listings [16], where there is a consolidated tendency for computer centres moving towards gigabit ethernet interconnected cluster architectures based on intel and AMD processors running Linux OS. This is, “super”computing centres are becoming a commodity rather than specialized machines.

Our work opens the path for IR communities to massively exploit computing infrastructures for their experiments and not only evaluating performance of searching in distributed IR, but also for all other processes (indexing, query sampling, merging, etc) of both the distributed and centralized cases. Moreover, this application model (virtualization + scheduling) is applicable in a straight forward manner to other fields with similar needs. We expect to continue our work, using the technological framework presented here to evaluate our RMIDistLemur through large scale distributed IR experiments.

ACKNOWLEDGEMENTS*: This research was co-funded by Ministerio de Ciencia e Innovación, FEDER and Xunta de Galicia under projects TIN2008-06566-C04-04/TIN and 07SIN005206PR.*

References

[1] C.D. Manning (2008), P. Raghavan, H. Schutze, *Introduction to Information Retrieval*, Cambridge University Press

[2] R. Baeza-Yates, B. Ribeiro-Neto (1999), *Modern Information Retrieval*, ACM Press New Cork,

[3] J. Callan (2000), *Distributed Information Retrieval* in W.B. Croft, editor, *Advances in Information Retrieval*, Kluwer Academic Publishers pp 127-150

[4] Avrahami, Thi Truong; Lawrence Yau; Luo Si; J. Callan (2006), *The FedLemur project: Federated search in the real world*, in *Journal of the America Society for Information Science and Technology*, vol 57, nb 3. Wiley Periodicals 2006

[5] Ellen M. Voorhees and Donna K. Harman (2005), *TREC, Experiment and Evaluation in Information Retrieval* MIT Press

[6] K Chadwick et al. (2008), *FermiGrid – Experience and Future Plans*, J. Phys.: Conf. Ser. 119 052010

[7] Foster, Ian and Kesselman, Carl, (2004) *The Grid: Blueprint for a New Computing Infrastructure* 2nd edition, Morgan Kauffman Publishers.

[8] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal, (2008) "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", Keynote Paper, *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications* (HPCC 2008, IEEE CS Press, Los Alamitos, CA, USA), Sept. 25-27, 2008, Dalian, China.

[9] P. Barhan et al., (2009) *Xen and the Art of Virtualization*, in Proceedings of the nineteenth ACM symposium on Operating systems principles, (pp 164-177) Bolton Landing, NY, USA

[10] Ounis, I., de Rijke, M., Macdonald, C., Mishne, G. & Soboroff, I. (2006), *Overview of the trec-2006 blog track*, in E. M. Voorhees & L. P. Buckland, eds, 'The Fifteenth Text REtrieval Conference Proceedings (TREC 2006)', Gaithersburg, Maryland.

[11] P. Ogilvie and J. Callan. (2002.) *Experiments using the Lemur toolkit* In Proceedings of the 2001 Text REtrieval Conference (TREC 2001) (pp. 103-108). National Institute of Standards and Technology, special publication 500-250.

[12] Gospodnetic, Otis; Erik Hatcher (2004). Lucene in Action. Manning Publications.

[13] The Xen hypervisor, http://www.xen.org

[14] VMWare, http://www.vmware.com

[15] The ALICE Experiment, http://aliceinfo.cern.ch/

[16] TOP 500, http://www.top500.com