Matchmaking, Datasets and Physics Analysis

Heinz Stockinger[1,2], Flavia Donno[1,2], Giulio Eulisse[3], Mirco Mazzucato[1], Conrad Steenberg[4]
Heinz.Stockinger@cern.ch
1) INFN, National Institute for Nuclear Physics, Italy
2) CERN, European Organization for Nuclear Research, Geneva, Switzerland
3) Northeastern University, Boston, USA
4) California Institute of Technology, Pasadena, USA

Abstract

Grid enabled physics analysis requires a Workload Management System (WMS) that takes care of jinding suitable computing resources to execute data intensive jobs. A typical example is the WMS available in the LCG2 (also referred to as EGEE-0) software system, used by several scientific experiments. Like many other current Grid systems, LCG2 provides a file level granularity for accessing and analysing data. However, application scientists such as High Energy Physicists often require a higher abstraction level for accessing data, i.e. they prefer to use datasets rather than files in their physics analysis.

We have improved the current WMS (in particular the Matchmaker) to allow physicists to express their analysis job requirements in terms of datasets. This required modifications to the WMS and its interface to potential data catalogues. As a result, we propose a simple Data Location Interface that is based on a web service approach and allows for interoperability of the WMS with new dataset and file catalogues. We took a particular High Energy Physics experiment as the source for our study and show that physics analysis can be improved by our modijications to the current Grid system.

1 Introduction

Resource management is one of the fundamental aspects in Grid computing. In particular, a resource can be a computing, storage or networking unit that is required to run a Grid application (also called *job* in this article) in a distributed environment. The actual resource selection, job submission and execution is typically done by a Grid middleware service, referred to as *Workload Management System (WMS)* in this context. An example implementation of such a Workload Management System is the one developed within the EU DataGrid project [10] and its partner project LCG [14].

The Workload Management System is capable of dealing with several different *job types,* depending which resources the end-user application requires. The job type as well as other job characteristics can be described in a Job Description Language (JDL) [16]. The most basic and therefore default job type is a computing intensive job that basically requires computing resources (CPU cycles) for execution. However, Data Grids are known for providing access to large amounts of data that are stored in storage resources called *Storage Elements.* Data intensive applications often require run time access to such data stored in Storage Elements. This data requirement can be expressed as *InputData* in the Job Description Language. Therefore, data intensive jobs need to be treated differently than pure CPU intensive jobs since a computing resource (also referred to as *Computing Element)* should be selected in a way such that data transfer is minimal.

In this article, we focus on a special aspect of the resource management process, namely the selection of Computing Elements given certain data requirements. This process is referred to as *matchmaking* and is sometimes regarded as the "brain" of the Workload Management System.

In many current Data Grids, several file replicas are stored in various Storage Elements. Replica catalogues are used to keep track of file replicas in a universal name space. One of the main tasks of the matchmaking process is to obtain replica locations of required data and then select a Computing Element that is relatively close (in terms of network connectivity).

Among several other data intensive, scientific domains, High Energy Physics (HEP) is a typical example of how Grid middleware can be used in a distributed environment. In the remainder of this article, we take one HEP experiment and discuss a simple physics analysis job that is run on Grid middleware. Although file access is already available in current Data Grid systems, several physicists from the CMS experiment require a higher level abstraction that is based on datasets, that are described by physics meta-data. Therefore, a conventional interface to a replica catalogue was not sufficient and we designed and implemented several enhancements for the Workload Management System that allow for access to experiment specific dataset catalogues.

Moreover, replica and file catalogue systems have been changing rapidly over the last years whereas the basic interface to the Workload Management System has been rather stable. In order to shield the Workload Management System from changes in the underlying replica catalogues, we designed a generic *Data Location Interface (DLI)* that allows the Matchmaker to query the location of any kind of data, be it a file or a dataset.

We identify the main contributions of this article as follows:

- Provide a possible solution to an important problem in Grid based physics analysis.
- Interoperability and compatibility: allow for a general approach that allows to interface an existing and established software system (the WMS and its Matchmaker) to current and future data location services.

The article is organised as follows. We first briefly explain the dataset problem in a High Energy Physics experiment and point out the limitations of the current architecture (Section 2). In Section 3 we give details about the current WMS architecture and the main interaction with end-users. Our main contributions and enhancements to the architecture and the user interface are described in Section 4. We then apply the new architecture and integrate it with the CMS experiment. A detailed case study is given in Section 5. In Section 6 we give experimental results with our new system. Finally, related work is presented in Section 7 before we conclude the article.

2 Physics Analysis and Datasets

The following section briefly introduces the High Energy Physics community and its Grid usage for data analysis.

2.1 Background

The CMS experiment is one of the four High Energy Physics (HEP) experiments at CERN. CMS is part of the LHC (Large Hadron Collider) programme and is currently being built. The main aim of CMS (as well as other HEP experiments) is to study the origin of mass. Physicists of the CMS collaboration are currently constructing a physics apparatus called the CMS detector that is scheduled to be operated in the year 2007. In addition to this hardware construction, several physicists, engineers and computer scientists work together on a software infrastructure that will allow physicists to analyse data produced by the physics detector.

From a computing point of view, physics analysis is a very complex task since it includes several computing challenges. On the one hand, large amounts of data (in the order of Tera- and Petabytes) have to be stored and replicated to several geographically distributed sites [18]. Next, physics analysis code needs to be in place that understands the complexities of the physics process in the detector. Finally, a distributed software system is required to provide the necessary computing power. Data is distributed and a software system must be in place to locate the data and then execute the job.

2.2 Grid Usage

Due to the distributed nature of the High Energy Physics community (physicists participating to the CMS experiment are located in several institutions all around the world), a Data Grid infrastructure is one possible solution to tackle the problem. CMS has already long experience in participating in Grid projects, and has used a recent LCG Grid software system for one of its scheduled data challenges to support physics analysis [2].

Current Grid tools in LCG are file based as regards cataloguing of data. In detail, each single file stored in a Storage Element is identified by a Logical File Name (LFN), a Global Unique IDentifier (GUID) and its physical storage address [5]. The Workload Management System uses the same concept and therefore a physics end user needs to describe her data requirements using LFNs and GUIDs.

However, physicists in CMS have recently expressed a new requirement that the smallest granularity of interaction with the Workload Management System should be based on a higher level concept called a *dataset*. Simply put, a physical dataset is a set of physical files that needs to be available as an atomic unit of data for a physics analysis job. A dataset itself can be fully replicated to several sites. In addition, the CMS experiment has started to provide a "dataset catalogue" that keeps track of physical locations of entire datasets. A more general definition of dataset is given in Section 4.1.

Consequently, this requirement for datasets needs to be addressed in the Workload Management System on both the client as well as on the server side. More details on these changes will be given in Section 4.

3 Current Situation: Workload Management System Overview

The following section gives a short overview of the current WMS not including our enhancements. In the discussion below we outline the user interface, i.e how a physicist needs to describe a data intensive job, and then go into some details of the architecture.

3.1 General Architecture

A general and very simplified view of the WMS and its related Grid services is as follows and depicted in Figure 1:

- The main user interface to the WMS consists of client software tools that are called **User Interface (UI).** For the user, the WMS looks like a single service although the server side consists of several interacting Grid services.
- Simply put, the server side is known as the **Resource Broker (RB)** that takes care of matching the job requirements **(Matchmaker),** selecting an appropriate Computing Element and finally submits the job there. In addition, there is a Logging and Bookkeeping component that keeps track of the job status.
- The Matchmaker needs to contact the **Replica Location Service (RLS) [6]** in order to obtain the detailed physical location of files stored in Storage Elements.
- The **Information System (IS)** provides basic information about which Grid services (CEs, SEs, etc.) are available as well as their status.

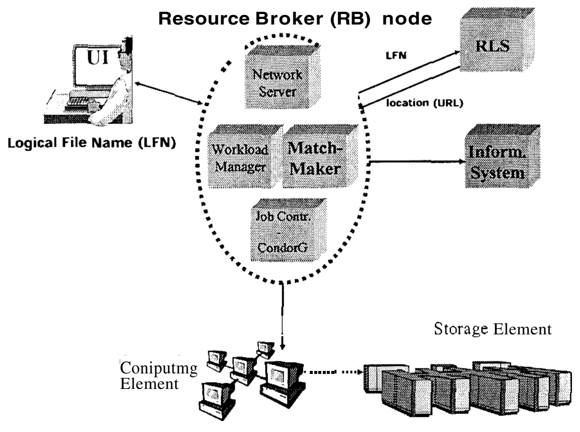


Figure 1. Basic overview of the WMS and its interaction with related Grid services. The main iteractions are user - RB - RLS and Information System.

A detailed discussion of all the server side components is beyond the scope of this articlc, and we refer the reader to [3]. In the following sub-sections, we only concentrate on the components that are necessary in order to understand the overall interaction.

3.2 User Interface

Before a physics application or any general job can be submitted to the workload management system, the job needs to be described using a particular language called **Job Description Language (JDL).** Since the job will be executed at a remote site other than the UI, it is necessary to define the executable name with input arguments, along with information on where potential standard output and standard error should be written. The job description is stored in a file (JDL file). Finally, the JDL file and a potential end-user application is submitted to the WMS which then takes care of the job execution and basic monitoring. Note that another alternative is that certain application programs are already pre-installed on certain Computing Elements and therefore do not need to be sent to the execution site.

For a data intensive job that requires to have runtime access to files stored in a Storage Element, a particular variable called *InputData* needs to be specified in the JDL. InputData contains a list of all LFNs or GUIDs which need to be accessed at runtime. A typical JDL example is as follows:

```
Executable = "analysisProgram";
InputData = "lfn:file1";
DataAccessProtocol = "gsiftp";
Stdoutput = "message.txt";
StdError = "stderror";
OutputSandbox = {"message.txt","stderror"};
```

In the example above, it is assumed that a program called `analysisProgram` is already pre-installed on Computing Elements. Next, the program requires a physical replica of a file identified by its LFN `lfn:file1`. The protocol that should be used to access a physical file identified by the LFN also needs to be specified as *DataAccessProtocol*. Finally, output and error messages are described and how they are returned to the user once the job has finished its execution. For further details on JDL parameters refer to [16].

3.3 The Matchmaker

Once the job has been submitted to the WMS, the Matchmaker needs to find a suitable Computing Element where the job can be executed. The Matchmaker first needs to parse the JDL and find resources that match the job requirements defined there. The available Computing Elements are obtained from the Information System (see Figure 1).

In case of data intensive jobs that require InputData, the Matchmaker needs to contact the Replica Location Service

(RLS) in order to retrieve Storage Elements where physical files are located. In detail, the physical file location is identified by an URL that contains the hostname of a Storage Element (SE). Once the Matchmaker has retrieved the list of all SEs, a Computing Element needs to be selected that is "close" to a given SE. The term "close" refers to closeness in terms of network connectivity, i.e. a CE is close to an SE if it is available in the same local area network. If several CEs satisfy the job requirements, the WMS selects one of them and finally sends the job there.

To sum up, resource selection of data intensive jobs is based on optimising data transfer, i.e. a job is sent to a Computing Element that in turn is close to a Storage Element which contains the necessary data.

In the current implementation of the WMS, the Matchmaker uses a C++ client library to query the Replica Location Service. Although this interface worked rather well for certain application in the LCG project, several limitations were discovered [2] on the server side that reduced the query performance. In addition, several new catalogue implementations are being developed without a standard interface for Matchmaker. This makes it hard for the WMS to be interfaced without including an additional client library per replica catalogue.

4 Reviewed Architecture

Given the requirement of using datasets in the physics analysis process, we have made several changes and enhancements to the WMS presented in Section 3. Basically, the WMS needs to allow for datasets in the JDL. In addition, the WMS (Matchmaker) needs to interact with a dataset catalogue. Design and implementation details are discussed in this section.

4.1 Logical Datasets in the Job Description

A **dataset** is considered to be an atomic unit of data that is defined within a Virtual Organisation (VO) or a physics experiment. Furthermore, a dataset itself can consist of several physical files but the end-user (physicist) mainly only knows the dataset concept.

We define the term **Logical DataSet (LDS)** to refer to an entity of data. In other words, a logical dataset can also be regarded as a "file collection". One logical dataset can have several physical replicas. For simplicity we assume that all physical files that belong to a dataset are stored at the same Storage Element and are accessible via the same protocol. A final assumption is that a dataset catalogue exists which stores the physical locations of all replicas.

Let us now have a look at a typical example of a dataset in a physics experiment (CMS). For example, a logical dataset is named

```
hg_2x1033PU761_TkMu_g133_CMS/eg03_hzz_2e2mu_350
```

In principle, the name of a logical dataset can be any string but it needs to have a meaning in the experiment. As for CMS, the dataset name uniquely identifies a certain physics event that consists of several particles (e.g. muons etc.). For further information on the dataset meaning we refer to [13].

This LDS name can then be used in the JDL as InputData. Since the WMS needs to distinguish between LFNs, GUIDs and datasets, the variable needs to be prefixed with "lds:". Consequently, we introduce a new InputData type that is recognised by both the User Interface as well as the server side of the WMS.

4.2 Data Location Interface for the Matchmaker

Conceptually, LFNs and logical datasets can be treated in the same way by the Matchmaker. For both data types, LFN and LDS, it is important to know where physical replicas are located. For LFNs, the replica location is returned by the RLS. For logical datasets, an equivalent service needs to implement such a query method.

We designed a general **Data Location Interface** that acts as the main interface between the Matchmaker and a catalogue for a specific data type. The following method is defined:

```
URLArray listReplicas(string inputDataType,
                      string inputData)
```

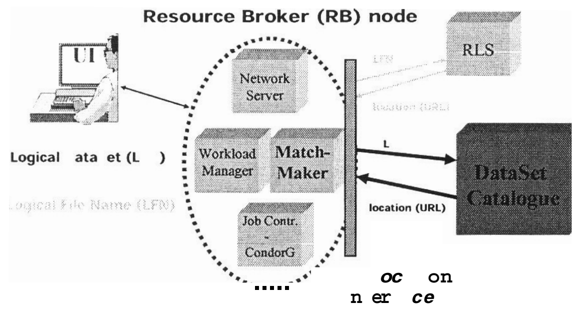
The parameters are as follows:

- `inputDataType` corresponds to the data type of InputData. It can have the following values: `lfn`, `guid`, `lds` or `query`. (`query` refers to a very general query that can be expressed as an SQL statement or in any other query language.)
- `inputData` is the actual value for the data type.
- `URLArray` corresponds to an array of URLs that is returned.

The above minimal interface needs to be implemented by either the replica catalogue or a dataset catalogue.

Web service standards are currently considered to be the most popular and well accepted standards in the Grid community. We therefore designed the Data Location Interface in WSDL (as described in [S]).

Consequently, the Matchmaker can interact with any catalogue (providing data locations) that exposes a web service interface. An updated architecture overview is given in Figure 2.



```
Data Location Interface:
        URL-List = listReplicas("lfn","lfn:my-lfn")
        URL-List = listReplicas("lds","lds:mu03_tt_4mu/mu_Hit245_2_g133)
```

Figure 2. Data Location Interface

4.3 Interoperability to Data Catalogues

By introducing the DLI into the WMS, we achieve two main goals. On the one hand, we satisfy the requirement of supporting the dataset use case. On the other hand, we defined a standard interface for data catalogues and solved the interoperability problem with new file or replica catalogues. The later is of particular importance since a standard interface and an agreed protocol allows the WMS to interoperate with any kind of catalogue.

4.4 Implementation Details

In order to fully include the DLI into the WMS, changes were required on the client as well as on the server side. As regards the client side, the client tools now also allow for the new InputData types `lds` and `query`. The code is fully backward compatible and only adds this new feature.

On the server side, a few more changes were necessary. The main one is the integration of the DLI client into the Matchmaker. Since the WMS is mostly written in C++, we have chosen gSOAP 2.6 in order to implement the DLI client. Therefore, DLI calls follow generally accepted web service standards.

Although we introduced a new feature to the WMS, we still want to keep the previous functionality where the Matchmaker interacts with the RLS (see Figure 1) and uses the specific client libraries. A major design idea was to introduce as much new functionality as possible but still be backward compatible to the previous user interface. In addition, for each job described in the JDL file, one can *either* use LFNs and/or GUIDs: in this case the RLS is queried for replica locations. Alternatively, one can use LDS: in that case the DLI is used to query a dataset catalogue (see Figure 2). In the current implementation it is not possible to mix LFN and LDS in the same JDL file.

In order to achieve this behaviour, we introduced a server side configuration parameter. For each Virtual Organisation one has to specify if either the RLS or the DLI is called by default. Consequently, when a data intensive job request arrives, the Matchmaker checks the InputData type and the server side configuration in order to distinguish between RLS and DLI calls. There are the two basic options:

- RLS is configured: *lfn* or *guid* requests are sent to the RLS. *lds* and *query* requests are sent to dataset catalogue that supports the DLI web service interface.
- If RLS is not configured, requests for all possible InputData types are sent to the DLI web service. A catalogue does not necessarily need to implement each data type and is free to return a "NotImplemented" SOAP Fault. This allows for maximum flexibility on the server side. In addition, in case a new InputData type is defined, a catalogue might also implement that.

By default, the URL of the web service implementing the DLI is stored in the Information System (in our case Globus MDS).

5 Use Case Study: CMS Experiment

In the CMS experiment there is a main dataset metadata catalogue called RefDB [13] that defines logical datasets. Each dataset includes physics specific information (metadata) that is related to collisions in the physics detector. Each logical dataset itself consists of several logical files.

Since datasets are replicated to several sites, the physical location needs to be obtained. This is currently done in two steps. Each site itself maintains a PubDB [2] that publishes the locations of file catalogues that in turn point to the physical files.

The front end service to both RefDB and PubDB is implemented in Clarens as described below. Clarens [17] is a Grid-enabled web services framework developed at California Institute of Technology (Caltech). It is written in Python on top of standard software components like apache2 and mod-python and it is completely open-source.

The Data Location Interface web service is a Clarens web service. It is actually a wrapper web service based around two other web services that can be found in PhySh (Physical Shell) to access the data location services RefDB and PubDB.

The current implementation relies on the fact that all information about dataset publication is back propagated from PubDB to RefDB. Consequently, only accessing the latter is required. A future implementation is foreseen where non-registered PubDBs are queried as well to keep track of datasets produced in private productions.

When a client wants to know all physical locations for a given LDS, a SOAP request is sent to the Data Location Interface web service. The web service issues the proper SQL query to RefDB to retrieve the possible locations where the dataset is available. The results are then mapped internally by the service to have the corresponding Storage Element hostnames. Once this is done, the SOAP answer conforming the Data Location Interface is created using the ZSI python module, and is then returned to the client (i.e. the Workload Management System).

5.1 Distributed Analysis Based on Datasets

After discussing the individual services and components used in CMS, we show how they work together with the Workload Management System. An overview is depicted in Figure 3.

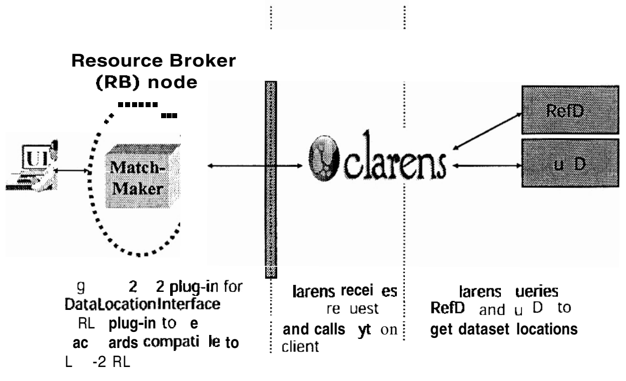


Figure 3. Interaction WMS and Dataset Catalogue in CMS

Given that Clarens implements the Data Location Interface, the interaction with the Workload Management System appears to be rather straight forward. In this way, Clarens shields all the CMS specific details of datasets from the Matchmaker and just passes the dataset query to the CMS specific catalogues, RefDB and PubDB.

A CMS physicist can now use her analysis code that requires a certain dataset and submits it to the WMS which then queries Clarens via the DLI for a specific location. If one or more dataset locations are found, the analysis code is sent to a Computing Element close to the physical location of thc dataset. Details will be given in the next section.

6 Experimental Results

The new architecture as discussed in Section 4 has been implemented and included in LCG release 2.4 and made available to end-users on the LCG production facility [15]. We used a pre-release as a test environment. In the following section we describe a simple pseudo analysis job that accesses datasets. We also measure the performance of individual components as well as the entire job submission system.

6.1 Test Environment

Our test environment consists of a subset of the LCG production facility [15] that has potentially access to more than 80 sites all around the world. Since our changes only influence the User Interface and the server side of Workload Management System (also referred to as RB node), all the available Storage Elements and Computing Elements can be used. They are currently running LCG version 2.2 on RedHat Linux 7.3. We installed a new User Interface and a RB node at CERN to act as the main entry point to the LCG facility. The entire test setup is as follows (for security reasons we do not list the real hostnames but only pseudo names):

- User Interface (UI) at CERN (uil.cern.ch). Hardware: Pentium III, dual processor with 1 GHz each, 512 MB RAM, 100 Mbps network card.
- CMS RB (cmsrb.cern.ch). The Matchmaker is configured to use Data Location Interface for the VO CMS. The Resource Broker node is further configured to use sites that support the CMS VO. Hardware: Pentium III, dual processor with 1 GHz each, 512 MB RAM, 100 Mbps network card.
- Clarens based CMS Dataset Catalogue (cmscat.cern.ch) with interface to RefDB (refdb.cern.ch) and several PubDBs. RefDB and PubDB services are the ones that are used in production mode in the CMS experiment. This machine is running on Scientific Linux CERN version 3. Hardware: Pentium Xeon, dual processer with 2.8 GHz each, 2 GB RAM, 100 Mbps network card.
- Storage Elements and Computing Elements from LCG-2. Datasets are replicated to several sites that are part of the CMS experiment. Not all datasets are directly stored on Storage Elements but on data servers which are in the same local area network as a co-located SE and CE.

6.2 Performance Results

For our performance tests, we take the following approach: we first test a single component (DLI), then the Matchmaker using the DLI and finally the entire job submission. In this way, we can isolate the performance issues and discuss them in the right context.

We first test the query performance of Clarens' DLI lookup using a simple C++ client program (using gSOAP 2.6.2). The DLI request consists of the data set given above plus the InputData type `lds`. The Clarens web service returns 3 SEs indicating sites that hold replicas. We ran the simple DLI C++ client on the same local area network where the Clarens server is located. The actual query response time is in the order of about 50 ms. The total query time for a single request consists of: SOAP request to Clarens which then contacts RefDB (with a MySQL client) for the actual datasets. The lookup performance for sequential queries scales linearly with the number of lookups. For instance, 10 sequential lookups take about 440 ms, 100 lookups take about 4530 ms (4.5s). For parallel queries (2, 4, 8, 16 and 32 parallel clients on the same machine: one client uses one process), the overall query response time is given in Figure 4. We observe that the average query performance for parallel queries (i.e. parallel clients) improves. This also shows that the server can serve more than 60 lookup requests per second.

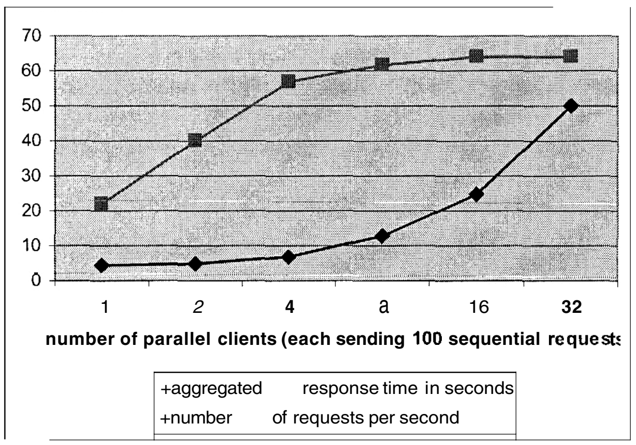


Figure 4. Client side DLI lookup performance for bursts of 100 lookups per 1, 2, 4, 8, 16 and 32 parallel clients.

Next, we measure the overall response time for a matchmaking request that we send to the WMS. In detail, we use the command line tool `edg-job-list-match` which triggers the WMS to find suitable resources based on a given JDL. The overall response time consists of the client call to the WMS, which then invokes the Matchmaker with its request to the dataset catalogue via DLI. We compare several different JDLs which request 1, 2, 4, 8, 16 and 32 different datasets, respectively. The results are depicted in Figure 5. If we do not specify any InputData, i.e. we just send a simple HelloWorld program to the Matchmaker, the response time is about 1.7s.

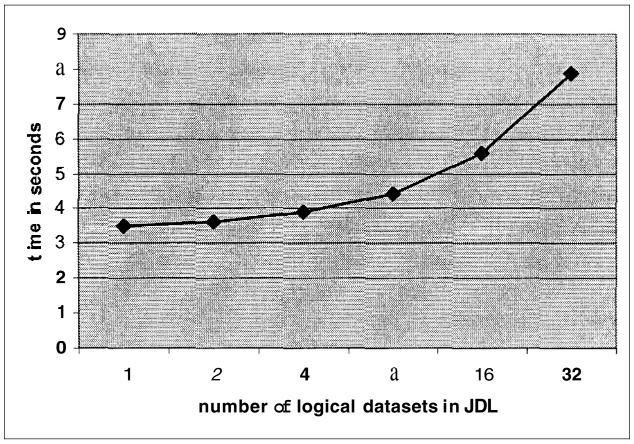


Figure 5. Client side performance of edg-job-list-match for JDLs with 1, 2, 4, 8, 16 and 32 LDSs.

Finally, we submit the entire job requesting datasets to the WMS using the client command `edg-job-submit`. We measure the overall time it takes from job submission, execution on the Computing Element and storing the final job status in the WMS. In our test case, the job is executed at a Element in Bologna (Italy). The job execution itself took less than a second. However, the overall time for job completion was about 6 minutes. This long latency is due to the latency of the Computing Element (Globus GRAM and LSF submission). Note that we do not have an influence on the response time of LSF, and therefore consider the performance of the Computing Element to be outside the scope of the our performance measurements. In general, we observed an overall job latency (even for very small jobs) of at least 5 minutes.

We conclude that the overall performance for a job execution is not significantly reduced by the latency of the dataset lookup. However, since a dataset contains many logical files (in the order of 10to 100), the number of data location lookups is usually decreased which results in a *reduced* latency for dataset queries with respect to LFN queries (reported in detail in [2]).

7 Related Work

The most closely related system is AliEn [4] which partly allows for datasets using the AliEn file catalogue. However, AliEn does not necessarily provide a general interface to easily include new catalogues, i.e. in the CMS case, information from PubDB and RefDB would need to be copied into an Alien file catalogue before Alien can use it. This approach is currently also applied in the early version of the gLite [11] system of the EGEE project (http://www.eu-egee.org).

Nimrod-G [1] provides a similar resource broker that uses an economic model for resource selection. However, Nimrod-G does not yet provide a general dataset interface. An important related aspect is the data placement for Grid schedulers. This is discussed in detail in the Condor project, in particular in Stork [12]. Further work on workflows and brokering in scientific data domains can be found in the Pegasus project [9].

Additional related work is done in HEP related Grid projects such as EGEE, PPDG (http://www.ppdg.net) and OSG (http://www.opensciencegrid.org). All of them work towards data intensive sciences and partly also industry related applications.

8 Conclusion

We have presented a general Data Location Interface (DLI) and an enhanced Workload Management System (WMS) that allow for matchmaking based on datasets. Our integration and performance results with a dataset catalogue from a High Energy Physics experiment show that the overhead introduced by DLI is minimal. In addition, the DLI makes it easier for physicists to due their physics analysis. With the DLI the WMS is on the one hand backward compatible *to* interface the currently used RLS, but it now provides a standard interface for querying any kind of data catalogue. Using a web service approach for the DLI also increases the forward compatibility to future catalogue implementations.

References

[1] D. Abramson, R. Buuya, and J. Giddy. A Computational Economy for Grid Computing and its Implementation in the NimrodG Resource Broker. Future Generation Computer Systems, 18(8), Oct. 2002.

[2] J. Andreeva, et al. Use of Grid Tools to Support CMS Distributed Analysis, IEEE Nuclear Science Symposium, Rome, Italy, Oct. 2004.

[3] C. Anglano, et al. Integrating GRID tools to build a computing resource broker: activities of DataGrid WP1, Computing in High Energy and Nuclear Physics (CHEP), Beijing, China, Sep. 2001.

[4] P. Buncic, A. Peters, P. Saiz. The AliEn system, status and perspectives. Conference for Computing in High-Energy and Nuclear Physics (CHEP), La Jolla, California, Mar. 2003.

[5] D. Cameron, H. Stockinger, F. Donno, et al. Replica Management in the EU DataGrid Project, to appear in International Journal of Grid Computing.

[6] A. Chervenak, E. Deelman, I. Foster, L. Guy, A. Iamnitchi, C. Kesselman, W. Hoschek, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tiemey. Giggle: A Framework for Constructing Scalable Replica Location Services. IEEE/ACM Supercomputing Conference (SC2002), Baltimore, USA, Nov. 2002.

[7) Data Location Interface (DLI) - WSDL Description: http://cmsdoc.cem.chlcms/grididocs/DataLocationInterface.wsdl, Nov. 2004.

[8] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. Su, K. Vahi, M. Livny. Pegasus : Mapping Scientific Workflows onto the Grid, Across Grids Conference 2004, Nicosia, Cyprus, Jan. 2004.

[9] EU DataGrid Project: http://www.eu-datagrid.org, Mar. 2004.

[10] gLite: http://www.gLite.org, Dec. 2004.

[11] T. Kosar, M. Livny. Stork Making Data Placement a First Class Citizen in the Grid 24th IEEE Int. Conference on Distributed Computing Systems (ICDCS2004), Tokyo, Japan, Mar. 2004.

[12] V. Lefebure, J. Andreeva. RefDB: The Reference Database for CMS Monte Carlo Production. Computing in High Energy and Nuclear Physics (CHEP), La Jolla, California, Mar. 2003.

[13] LHC Grid Computing Project (LCG): http://cem.chilcg/, Dec. 2004.

[14] LCG Production Facility: http://goc.grid-support.ac.uk/gridsite/gocmain/monitoring/, Dec. 2004.

[15] F. Pacini. Job Description Language HowTo. http://serverl1.infn.it/workload-grid/docs/DataGrid-01-TEN-0142-0_2.pdf, Oct. 2003.

[16] C. Steenberg and E. Aslakson, J. Bunn, H. Newman, M. Thomas, F. Van Lingen. The Clarens Web service architecture. Computing in High Energy and Nuclear Physics (CHEP), La Jolla, California, Mar. 2003.

[17] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, B. Tierney. File and Object Replication in Data Grids. Journal of Cluster Computing, 5(3):305-314, 2002.