KID - KLOE Integrated Dataflow

The KLOE collaboration*
presented by
I.Sfiligoi
Laboratori Nazionali di Frascati dell'INFN, Frascati, Italy.

July 13, 2001

Abstract

KLOE is acquiring and analyzing hundreds of terabytes of data, stored as tens of millions of files. In order to simplify the access to these data, a URI-based mechanism has been put in place. The KID package is an implementation of that mechanism and is presented in this paper.

Keywords: CHEP, KID, data handling, URI

1 Environment overview

1.1 The KLOE experiment

KLOE[1] is the experiment for which the INFN $DA\Phi NE$ ϕ -factory in Frascati, Italy, has been built. Its main goal is the measurement of CP violation parameters at sensitivities of $O(10^{-4})$, but is also capable of investigating a whole range of other physics. To achieve its goal, KLOE needs to acquire $O(10^{11})$ events, which corresponds to about 1 petabyte of data.

1.2 Computing system overview

The KLOE computing environment is based on a set of distributed UNIX servers. Several UNIX flavors are supported, the most used being Sun Solaris, IBM AIX and Linux.

The computing nodes can be divided into four classes:

- DAQ[1] nodes

*M. Adinolfi, A. Aloisio, F. Ambrosino, A. Andryakov, A. Antonelli, M. Antonelli, F. Anulli, C. Bacci, G. Barbiellini, F. Bellini, G. Bencivenni, S. Bertolucci, C. Bini, C. Bloise, V. Bocci, F. Bossi, P. Branchini, S. A. Bulychjov, G. Cabibbo, A. Calcaterra, R. Caloi, P. Campana, G. Capon, G. Carboni, A. Cardini, M. Casarsa, V. Casavola, G. Cataldi, F. Ceradini, F. Cervelli, F. Cevenini, G. Chiefari, P. Ciambrone, S. Conetti, E. De Lucia, G. De Robertis, R. De Sangro, P. De Simone, G. De Zorzi, S. Dell'Agnello, A. Denig, A. Di Domenico, C. Di Donato, S. Di Falco, A. Doria, E. Drago, O. Erriquez, A. Farilla, G. Felici, A. Ferrari, M. L. Ferrer, G. Finocchiaro, C. Forti, A. Franceschi, P. Franzini, M. L. Gao, C. Gatti, P. Gauzzi, A. Giannasi, S. Giovannella, V. Golovatyuk, E. Gorini, F. Grancagnolo, W. Grandegger, E. Graziani, P. Guarnaccia, H. G. Han, S. W. Han, X. Huang, M. Incagli, L. Ingrosso, Y. Y. Jiang, W. Kim, W. Kluge, C. Kuo, V. Kulikov, F. Lacava, G. Lanfranchi, J. Lee-Franzini, T. Lomtadze, C. Luisi, C. S. Mao, M. Martemianov, A. Martini, M. Matsyuk, W. Mei, A. Menicucci, L. Merola, R. Messi, S. Miscetti, A. Moalem, S. Moccia, M. Moulson, S. Mueller, F. Murtas, M. Napolitano, A. Nedosekin, M. Panareo, L. Pacciani, P. Pagès, M. Palutan, L. Paoluzi, E. Pasqualucci, L. Passalacqua, M. Passaseo, A. Passeri, V. Patera, E. Petrolo, G. Petrucci, D. Picca, G. Pirozzi, M. Pollack, L. Pontecorvo, M. Primavera, F. Ruggieri, P. Santangelo, E. Santovetti, G. Saracino, R. D. Schamberger, C. Schwick, B. Sciascia, A. Sciubba, F. Scuri, I. Sfiligoi, J. Shan, P. Silano, T. Spadaro, S. Spagnolo, E. Spiriti, C. Stanescu, G. L. Tong, L. Tortora, E. Valente, P. Valente, B. Valeriani, G. Venanzoni, S. Veneziano, A. Ventura, Y. Wu, Y. G. Xie, P. F. Zema, P. P. Zhao, Y. Zhou

[1]Data AcQuisition

- reconstruction and Monte Carlo nodes
- disk and tape servers
- analysis nodes, i.e., user machines

The first three classes of nodes, called the production pool in the following, are managed centrally by the KLOE computing group, while the user machines are managed by the owners themselves or by the computing service of their home institution.

No support from the KLOE computing group is provided for the user machines. However, the requested software configuration is minimal; the only request is to run an AFS client or to have a local copy of the KLOE software distribution (root access is **not** needed).

1.3 Event building and processing

A KLOE event is first totally built[2] inside a DAQ node; it contains raw electronic readout data only. First it gets written in a memory buffer and is later on moved into a file. At a later time, the event is read in by a reconstruction process on a reconstruction node that reconstructs raw data into higher level structures and if the event is found to be of physics interest, all data in the event get written to another file.

The raw event is normally used again only if a new version of the reconstruction program is produced; the reconstructed event is instead of physics interest and as such read-in by several analysis programs on several user machines during the lifetime of the experiment.

1.4 Data storage

In KLOE, events are stored as YBOS[3] files of reasonable size (up to 1 Gb at the time of writing). The files are created on local disks first and moved at a later time to tape for long term storage. When needed, some of these files are moved back to a (possibly different) disk area for analysis. To keep track of which files have been produced and where they currently reside, a central RDBMS[2] is used[4].

2 Accessing the data

2.1 Data sources

Data analysis, monitoring and reconstruction software all need to access the data, but the data source is quite different in the three cases:

- monitoring tasks read data from memory buffers in order to have the fastest response times
- reconstruction tasks read data from one file at a time to well distribute the CPU load
- analysis tasks read data from possibly several files concurrently to process all the data of interest

2.2 Local vs. remote data

Not all the data sources are local either:

- some monitoring tasks may use local memory buffers for faster execution while others may need to read data from all the DAQ nodes in order to get the full picture
- reconstruction tasks normally read data from remote DAQ nodes, but for debugging purposes a user specified local file is often a better solution
- analysis tasks execute on user machines which are outside the production pool where most of the KLOE data resides, but some data may have been moved or produced locally to speed up the analysis

[2]Relational DataBase Management System

The distinction between local and remote data sources can be lightened by using AFS or NFS, but only for files. The problem of remote memory buffers does not have a standard solution.

2.3 File bookkeeping

At any given time, the event files reside on the DAQ disk area, on tape and/or on one or more analysis disk areas. Excluding the tape, all the disk based locations are temporary and there is no guarantee that a given file will stay in the same place the next time it is looked for. To prevent a file from being deleted from disk while it is in use, the central RDBMS is used; each time a file is used, the application must inform the RDBMS and it should [3] do the same at file closure.

Another problem is created by files that do not reside on an analysis disk area. They must be moved there before an analysis program can use them. This operation is managed by the central data handing system, so the application has just to ask for the files it needs.

3 The KID package

3.1 What is KID

KID[6] is a package that makes all the operations described in the previous chapter fully transparent to the application; the application does not need to worry if the data source is local or remote, whether it comes from a memory buffer or from file, where the files reside and so on. The same code can be used to access any KLOE data source.

3.2 How does it work

The basic concept on which KID works is the use of URIs[5]. To obtain a data source, the application gives to KID a text string containing a KID-recognized URI[7] and events begin to flow in. It is the responsibility of the KID package to decode the URI and to perform all the necessary communication with the available services.

dbdatarec:run_nr between 19005 and 19011 and stream_code='rad'?report=true

Figure 1: An example of a KID URI - Accessing data via the KLOE data handling system

3.3 KID internals

The KID library is written using ANSI C for compatibility reasons. The object oriented approach is however followed as much as possible; the access to resources is object (or better to say identifier) based, while protocol selection is achieved through function pointers. For compatibility reasons, a FORTRAN interface is available as well.

The library itself is modular; the core functions are just an interface to the application, all the work is performed inside the modules. All the general modules[7] (such as reading from a file or a memory buffer, remote access or the interface to the data handling system) are already part of the KID package and are normally the only modules needed by most applications. Anyhow, new modules can easily be developed and added to an application (using the shared libraries this can be done also at runtime).

Although very few new modules are expected to be developed, the development of new modules is quite a simple task. Most of the URI manipulating functions are included in the KID package, so the developer needs only to concentrate on the module specific code.

[3] else it is done automatically at process exit

```
kid_open(URI)                                    "remote:/tmp/B_1_1@farm1"
{
  schema,other = extract_schema(URI)             "remote" "/tmp/B_1_1@farm1"
  id.ptrs = find_module(schema)
  id.hd = ptrs.open(other)                       remote_open
  return id
}

kid_get(id)
{
  return id.ptrs.get(id.hd)                      remote_get
}

kid_close(id)
{
  id.ptrs.close(id.hd)                           remote_close
}
```

Figure 2: KID library internals with an example

3.4 Conclusions

The KID package has greatly simplified the application development in KLOE; since any event source can be read-in by the same (few) functions, the application developer can forget about this detail. Moreover, it has also simplified the user interfaces; since URIs are easy to write and understand, no additional layer is needed to select the data source; the URI itself can be given by the end user.

The end users, especially the physicists, have also gained a lot with the introduction of KID. Having a unique user interface for selecting the data source was well worth the small amount of overhead involved in getting used to KID URIs. Moreover, due to the flexibility of the supplied URIs, complex event source selections are now at the reach of everybody.

References

[1] The KLOE Collaboration, "KLOE, A General Purpose Detector for DAFNE", LNF-92/019 (IR), Frascati, 1992

[2] P. Branchini for the KLOE collaboration, "Front end daq for the KLOE experiment", CHEP'98, Batavia, 1998.

[3] CDF Computing Group, "YBOS, Programmers Reference Manual", CDF Note No. 156, Batavia, 1992.

[4] I.Sfiligoi, "Data Handling in KLOE", CHEP'2000, Padova, 2000.

[5] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Locators (URI): Generic Syntax", W3 RFC 2396, 1998.

[6] I.Sfiligoi, "KID - KLOE Integrated Dataflow", `file:///afs/kloe.infn.it/export/soft/onl/kid/default/doc/index.html`, Frascati, 2000.

[7] The KLOE collaboration, "KID - KLOE Integrated Dataflow", ACAT 2000, Batavia, 2000.