KID - KLOE Integrated Dataflow

The KLOE collaboration*
presented by
I. Sfiligoi

INFN LNF, via E. Fermi 40, 00044 Frascati, Italy

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 files, a URI-based mechanism has been put in place. The KID package is an implementation of that mechanism and is presented in this paper.

INTRODUCTION

The KLOE experiment

KLOE[1] is the experiment for which the INFN *DAΦNE* ϕ -factory in Frascati, Italy, was built. Its main goal is the measurement of CP violation at sensitivities of $O(10^{-4})$, but is also capable of investigating a whole range of other physics. The most interesting studies include kaon form factors, kaon rare decay and radiative ϕ decays measurements.

To achieve its goal, KLOE needs to acquire $O(10^{11})$ events, which corresponds to about 1 petabyte of data. All of them must be reconstructed and analyzed to minimize the systematic errors.

Data storage

In KLOE, events are stored as YBOS[2] files of reasonable size (up to 1 Gb at the time of writing). These files are stored first on disk and then moved to tape for long term storage and 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[1] is used[3].

Three types of event files are present in the system:

- raw files, containing only electronic readout of the events
- datarec files, containing raw data plus some additional information gathered during data reconstruction and classification stage, such as drift chamber tracks and calorimeter energy deposits
- Monte Carlo files, containing simulated events

All the events acquired during the data acquisition will be stored as raw files and maintained for the life of the experiment. The events that will be classified as of physics interest during event classification, will be stored a second time as datarec files. Since the data reconstruction and classification process can change during the life of the experiment, several versions of the same event can reside in different datarec files, although in the long term, only the most recent version is planned to be kept.

* 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, G. Cataldi, F. Ceradini, F. Cervelli, F. Cevenini, G. Chiefari, P. Ciambrone, S. Conetti, S. Conticelli, E. De Lucia, G. De Robertis, R. De Sangro, P. De Simone, G. De Zorzi, S. Dell'Agnello, A. Denig, A. Di Domenico, 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, 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, V. Kulikov, F. Lacava, G. Lanfranchi, J. Lee-Franzini, T. Lomtadze, C. Luisi, C. S. Mao, M. Martemianov, A. Martini, M. Matsyuk, W. Mei, 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, C. Pistillo, 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, T. Spadaro, S. Spagnolo, E. Spiriti, C. Stanescu, G. L. Tong, L. Tortora, E. Valente, P. Valente, B. Valeriani, G. Venanzoni, S. Veneziano, Y. Wu, Y. G. Xie, P. P. Zhao, Y. Zhou

[1] Relational DataBase Management System

System overview

The KLOE computing environment is made of a set of UNIX servers, connected via a switching network. Logically, the servers can be divided into:

- DAQ[2] nodes
- CPU suppliers
- disk servers
- tape servers

Typically, an application runs on a CPU supplier and reads the necessary data from a remote node via a KID daemon(see next section).

THE KID PACKAGE

The KID package is made of a programming library and a not-privileged daemon, called the KID daemon. In this section, both of them are presented.

User interface

As seen in the introduction, the number of available events is very large. To select only the most interesting ones an easy to use user interface is very important; the KID approach is based on Uniform Resource Identifiers (URIs)[4].

<scheme>:<scheme-specific-part>[?<options>]

Several schemes (or protocols) are available in the main package, but user-specific ones can be added as needed.

Library

The KID programming library is written using ANSI C for compatibility reasons. Nevertheless, the object-oriented approach is followed as much as possible; the access to resources is object (or identifier) based and protocol selection is achieved through function pointers.

The external interface is very simple, having essentially only four classes of functions:

- open - get a new identifier, the only parameter is an URI
- get - read a new event from the data source (identifier-based)
- skip - discard an event from the data source (identifier-based)
- close - destroy the identifier and release any associated resource

Internally, the library is modular; the functions of the core module parses out the schema and then calls the appropriate functions of the schema-specific module. This allows a user-supplied module with its specific schema support to be put in place even at run-time.

The library is also POSIX thread safe, having mutual exclusive locks on identifiers implemented in the core module.

KID daemon

The KID daemon is a key element of the system; it allows remote applications to access local resources, such as memory buffers and local files.

It is implemented as a multi-threaded not-privileged daemon. The main thread listens for TCP/IP requests and creates a new thread for any accepted connection.

Connection specific threads read a client-supplied URI and use it to get an identifier from the KID library that will be used to read the events to be sent to the client (see Fig. 1).

To speed-up the communication, several events (typically 100 at the time of writing) are packed together and the packets are sent before the client actually asks for a new packet. Since this can be a problem for a small fraction of applications (for example event displays), the packet size can be set and the push mode reverted to pop mode at connection time.

Interface to the data handling system

Being able to access a remote resource is surely an essential facility, but it is not enough. The user still needs to know where the resource is located and how it is called. For this reason an interface to the data handling system (DH) has been created.

The interface is implemented as a KID library module. It processes the schema specific part of the URI as an SQL query, passes the resulting list of files to the cache manager (see [3] for more details) and then uses the information returned by the cache manager to open those files (that are now residing on one or more disk areas) (see Fig. 2).

[2] Data AcQuisition

KID standard modules

Several ready-to-use modules are present in the KID package. In this subsection, the most interesting ones are presented.

Local access modules

These modules allow the access to local resources; two of them are supplied:

- spy - allowing access to a memory buffer
- ybos - allowing access to a local file

<schema>:<resource-name>[?options]

Remote access module

This module creates a connection with a KID daemon and passes back the events to the application.

remote:<URI>@<IP-node>[?options]

DH enabled modules

The main module expects an SQL query as explained in the previous subsection.

db:<SQL-query>[?options]

However, writing general SQL queries can be quite annoying, so modules specialized in different types of queries have been created:

- dbraw - select only raw files
- dbdatarec - select only datarec files
- dbmc - select only Monte Carlo files
- dbfile - select a file based on its filename

<schema>:<where-part-of-a-SQL-query>[?options]

Support modules

Some support modules are also provided; the most used are:

- merge - get events from multiple sources
- try - find an event source from a list

<schema>:(<URI>,...,<URI>)[?options]

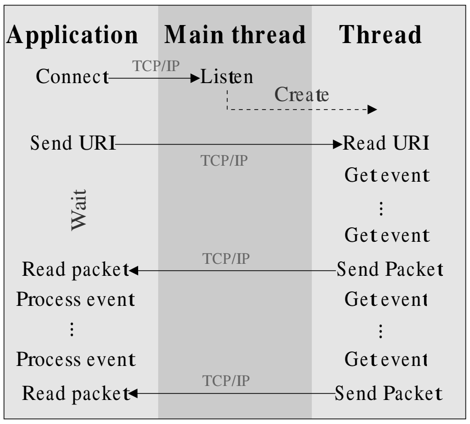

FIGURE 1. KID daemon internals

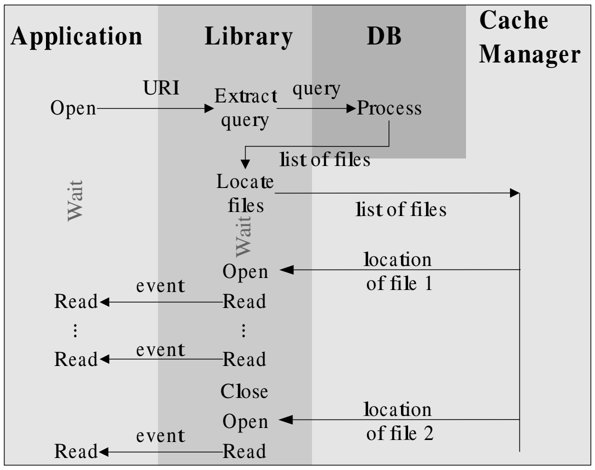

FIGURE 2. KID-DH interface

CONCLUSIONS

KID is now in use in KLOE for about 6 months. Most of the user applications and a considerable fraction of the production software is now using it.

The use of KID has simplified most of the analysis tasks and made even new types of analysis possible.

REFERENCES

1. The KLOE Collaboration, *KLOE, A General Purpose Detector for DAFNE, LNF-92/019 (IR), Frascati, 1992*
2. *CDF Computing Group, YBOS, Programmers Reference Manual, CDF Note No. 156, Batavia, 1992*
3. *I.Sfiligoi, "Data Handling in KLOE", CHEP2000 conference proceedings*
4. *T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Locators (URI): Generic Syntax", W3 RFC 2396, 1998*