Fusion Engineering and Design 89 (2014) 712–716

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

Integrated data acquisition, storage, retrieval and processing using the COMPASS DataBase (CDB)

J. Urban[a,*], J. Pipek[a], M. Hron[a], F. Janky[a,b], R. Papřok[a,b], M. Peterka[a,b], A.S. Duarte[c]

[a] Institute of Plasma Physics AS CR, v.v.i., Za Slovankou 3, 182 00 Praha 8, Czech Republic
[b] Department of Surface and Plasma Science, Faculty of Mathematics and Physics, Charles University in Prague, V Holešovičkách 2, 180 00 Praha 8, Czech Republic
[c] Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade Técnica de Lisboa, 1049-001 Lisboa, Portugal

HIGHLIGHTS

- CDB is used as a new data storage solution for the COMPASS tokamak.
- The software is light weight, open, fast and easily extensible and scalable.
- CDB seamlessly integrates with any data acquisition system.
- Rich metadata are stored for physics signals.
- Data can be processed automatically, based on dependence rules.

ARTICLE INFO

Article history:
Received 21 May 2013
Received in revised form 4 March 2014
Accepted 10 March 2014
Available online 12 April 2014

PACS:
52.55.Fa
07.05.Kf
07.05.Hd

Keywords:
Tokamak
CODAC
Database
Data management
Data acquisition

ABSTRACT

We present a complex data handling system for the COMPASS tokamak, operated by IPP ASCR Prague, Czech Republic [1]. The system, called CDB (COMPASS DataBase), integrates different data sources as an assortment of data acquisition hardware and software from different vendors is used. Based on widely available open source technologies wherever possible, CDB is vendor and platform independent and it can be easily scaled and distributed. The data is directly stored and retrieved using a standard NAS (Network Attached Storage), hence independent of the particular technology; the description of the data (the metadata) is recorded in a relational database. Database structure is general and enables the inclusion of multi-dimensional data signals in multiple revisions (no data is overwritten). This design is inherently distributed as the work is off-loaded to the clients. Both NAS and database can be implemented and optimized for fast local access as well as secure remote access. CDB is implemented in Python language; bindings for Java, C/C++, IDL and Matlab are provided. Independent data acquisitions systems as well as nodes managed by FireSignal [2] are all integrated using CDB. An automated data post-processing server is a part of CDB. Based on dependency rules, the server executes, in parallel if possible, prescribed post-processing tasks.

1. Introduction

With the increasing volume and complexity of diagnostics and synthetic data needed for tokamaks or other pulsed experimental devices, the demands on the data storage system are becoming very challenging. Present technologies are often difficult to scale, either due to the used technologies or due to the internal architecture. Since the COMPASS tokamak (re)started its operation at IPP Prague [1], more and more issues related to data storage emerged. This fact, together with the experience from other tokamaks, motivated the development of the *COMPASS DataBase* (CDB)—a system to be used to store and retrieve any (COMPASS) tokamak related numerical data.

This paper first discusses the fundamental goals and motives of CDB in Section 2. The CDB architecture is described in Section 3. The core functionality is implemented in Python with bindings for many different languages and interfaces, as described in Section 4. The integration of the various data acquisition systems is addressed

* Corresponding author. Tel.: +420 266053564.
E-mail address: urban@ipp.cas.cz (J. Urban).

http://dx.doi.org/10.1016/j.fusengdes.2014.03.032
0920-3796/

in Section 5. A recently implemented automatic post-processing capability is described in Section 6. Finally, Section 8 gives our conclusions.

2. Goals and motivation

As already noted above, it was the re-installation of the COMPASS tokamak at IPP Prague that provided the fundamental motivation for the development of a new data storage system. Within a close collaboration with IPFN Lisbon, a new CODAC (Control, Data Access and Communication) system [3,4] has been built. In the beginning, the system consisted of ATCA data acquisition hardware and FireSignal [2], which controlled the data acquisition nodes. SDAS [5] was used to access the data provided by FireSignal. This original design was strongly focused on data acquisition as the data signals were identified by hardware identifiers while the natural identification is the (measured) physical quantity. Moreover, data are written to and read from a central server, which can thus become overloaded. This is the case of COMPASS as around 2 GB of diagnostic data are presently produced for a single discharge; this number can become several times larger in the near future. Moreover, the API was not very flexible and numerical data were stored in custom binary files.

COMPASS DataBase (CDB) has been designed to enhance the capabilities and performance of COMPASS CODAC, focusing mainly on the final storage system. CDB provides a way to write data from data acquisition nodes in parallel, without a central collection point. As such, the FireSignal central server can be used for nodes control only, while the data are written directly to CDB. CDB is easily scalable, portable and extensible. Although targeted to be used primarily on COMPASS, the aim is to create a universal tool. The CDB architecture is depicted in Fig. 1 and described in detail in the following.

3. The architecture of CDB

3.1. Data and metadata

Scientific data that we need to store are well structured (we know in advance the type, the dimensionality, etc.) and can be decomposed into actual data (numbers) and metadata, which contain the information about the data content (e.g., the physical quantity, units, etc.).

The data model of CDB, depicted in Fig. 2, is based on *generic signals*, which contain the descriptions of possible data types that can be stored, and *data signals*, which are instances (realizations) of generic signals and contain the *metadata* of a particular dataset, including the *data file*, in which the numerical data are stored. Generic signal description also contains the axes, which are generic signals as well.

A single data signal as well as single data file always belong to a particular *record number* and can exist in one or more *revisions*. The record number denotes either a particular experimental discharge or a model (simulation) or a void (e.g. a trigger test) record.

Two types of data signals can be stored in CDB: FILE and LINEAR. FILE signals have the data in data files while LINEAR signals are described by a linear function. Linear transforms of FILE signals can also be stored, particularly for converting from data acquisition levels to physical units or for correcting the data, so that no new data file has to be created.

All metadata, i.e., generic and data signals, records, data file descriptions and more, are stored in a relational database, particularly MySQL, although a different database engine can be used. The numerical data are stored in files on a network attached storage (NAS). The primary file format is HDF5 [6]. There are many good reasons to employ HDF5 files on a NAS as the numerical data storage back end. HDF5 is one of the most enhanced and wide spread file formats, which allows to conveniently read and write almost any kind of numerical datasets, organized hierarchically in a file. Very importantly, HDF5 API is extremely rich in its functionality and is available for all relevant programming languages. Each HDF5 file can contain one or more CDB data signals. CDB also allows to store any other data type, however, without providing any compatibility to read the data. This feature can be convenient, for example, for storing whole custom data files of a simulation code (instead of storing it "somewhere", CDB stores the file and its metadata in a well defined place) or for storing encoded videos or images.

Since CDB uses conventional file access protocol, any NAS can be used to read and write the data files. This is, in most cases, a big advantage as the storage can be tailored and optimized independently of CDB itself. A cluster storage system is actually used for COMPASS [7]. To increase the performance of writing data from FireSignal, we have an additional local cache on the central server.

3.2. Signal identifiers

CDB *generic signals* are uniquely identified either by their numeric id, by a combination of name and data source or by an alias. In order to have a unified API for different languages and for different (and extensible) identification schemas, CDB uses string identifiers. We define `gs_str_id` (generic signal string id) as a string that contains one of the unique generic signal identifiers.

Data signals are uniquely identified by a unique generic signal identifier (see above) in combination with a record number and a revision. For this reason, we define a string identifier `str_id`:

```
str_id:=<CDB:> gs_str_id <:record_number <:revision»<
[units]>| FS | DAQ: channel_id <:record_number <:revi-
sion»<[units]>
where
channel_id:= computer_id/board_id/channel_id
```

Here CDB, FS and DAQ are schemas for signal identification. The default is the native CDB schema while FS and DAQ are used for identification by data acquisition channels using FireSignal or CDB native id's, respectively. The following `str_id` examples refer to the same signal, the first one by its alias while the others by its DAQ and FireSignal id's:

1. `I_plasma:4073:-1[default]`
2. `DAQ:ATCA_1/9/13:-1`
3. `FS:PCIE_ATCA_ADC_01/BOARD_9/CHANNEL_013:4073`

String identifiers can be used in any programming language, hence simplifying and unifying the bindings, and are easy to compose and parse. They can also be straightforwardly extended for, e.g., multiple tokamaks (by adding the tokamak name as a prefix), different databases or even indexing and slicing. An additional schema can be implemented if needed. In particular, an ITM CPO [8] schema is planned to be implemented, which would allow to search signals by CPO field names.

3.3. Data safety and consistency, revisions

When a new version of a signal is stored, e.g., in case of an error, a new revision is stored instead of overwriting the existing data. As such, full history is saved and data consistency and persistence is assured. When a data file is closed, CDB client changes its permissions to read only. Moreover, the cache mechanism changes the