Data Base structure for Borexino

C. Salvo, M. Pallavicini

Genova - November 4, 1998

1 INTRODUCTION

This document is a proposal for the Data Base (DB) structure for experiment Borexino.

It is intended to explain the guidelines that have driven us to the proposed solution and to suggest a data structure suitable to the collaboration needs.

It should be considered a starting point for a collaboration-wide discussion on this issue.

The document is structured as follows: section 2 describes the software package we propose and the reasons why we chose it; section 3 describes how we suggest to implement this software within existing Borexino software and, finally, section 4 is devoted to the data structure.

2 DATA BASE PACKAGE

We propose to use PostgreSQL, a client-server data base package, SQL compatible, which has been developed by the Computer Science Department of the University of California at Berkeley.

Its main features are:

- It is free and available for almost any platform, including Unix Dec OSF1, Lynux, VMS, Sun Solaris and OS, HP-UX, AIX and others.
- While being a light-weight and simple data base, it is SQL92 and SQL3 compatible and it is capable of many features typical of expensive commercial software, like type extensibility that can be particularly useful for a project that is supposed to accumulate data for many years.
- It has a client-server structure that allows all Borexino users to share the same DB, kept and maintained in a single place.

- It has security features.
- It is written in C with all source code and API[1] available.
- It is distributed with an interactive program to make simple operation with the DB.

This package has been already installed in Genova and all preliminary tests have given encouraging results.

3 IMPLEMENTATION FOR BOREXINO

PostgreSQL is a client-server DB, so many different software products can access it from any remote host machine to read data or, if they are allowed to, write data into the DB.

The DB can be accessed both through an interactive interface (a simple program is already available; a WEB interface will be provided) and from any software program through an Application Program Interface.

This API library is written in C and is basically a set of functions that allow the program to open a connection to the DB, submit queries to it and recover data.

A FORTRAN wrapping of these function will be provided.

The program that will certainly need to interact with the DB are:

- Data Acquisition (C,C++): many parameters have to be written by DAQ during data taking. See section 4 for a preliminary list.
- Simulation (Fortran / C++) : Geometry, electronics parameters, scintillator parameters. The simulation program should write into the DB the physical parameters used for the simulation and assign to them a negative run number.
- Reconstruction (Fortran) : Geometry, calibration constants, cables mappings, run paramters. When it runs on simulation data, it reads the physical parameters used by the tracking code and/or Geneb. We assume here that any simulation program will provide an output in RAW DATA format.

[1] API stands for Application Program Interface and it is a library that a program must use to connect to the DB.

It is to be decided whether we want to keep a single server for the whole collaboration or mirror it in a few places. In section 4 we show that the amount of data for Borexino is small enough to allow to use a single DB server.

4 THE DATA STRUCTURE

We have gathered in this section the data that will have to be stored in the DB, dividing them in 4 main groups which are eligible to define 4 different PostgreSQL databases: geometry, cable mappings, scintillator parameters and run parameters. For each group we provide an estimate of data type[2] and size required.

Every DB record is accessed through the run number key and possibly with a time key. This means that run number information and time information is assumed to be associated to every record.

Geometry

The geometrical parameters are basically static, which means that they are stored once at the beginning of data taking and never changed. We may espect to have a single set of numbers that are valid for a whole run, until the detector is physically modified.

- Number of phototubes for each kind: neutrino, inner μ, outer μ (3 I)
- Position θ and Φ and type of each tube (2424*2 F + 2424 I)
- Light concentrator geometry (50 F)
- Photocathode geometry: area and curvature (2 F)
- Spheres geometry (3/4 F) [3]

These data need about 30 kbyte that are written probably one or a few times during the whole Borexino lifetime.

[2] In the following I stands for INTEGER and F stands for (FLOAT)

[3] This is evaluated in the simple hypothesis that the scintillator has the same refraction index of the buffer. A more accurate geometry description may be required if the optical behavior of the nylon buffer surface cannot be neglected.

Cable mapping and electronics

These data can be more variable than geometrical ones but definitely will not change every few runs. They describe electronic setup and hard-wired parameters.

- Number of crates, boards, channels (3 I)
- Bar codes for tubes, cables, stainless steal holes, front-end channels (2424*4 F)
- Phototube to Distribution panel map (2424*2 I)
- Distribution panel to Front-end cardss map (2424*3 I)
- Front end card to Digitizer card map (2424*3 I)
- HV to Digitizer card map (2424 I)
- Crate addresses (13 I)
- Calibration signal to Front end Cards mapping (to be defined)
- Digitizer to trigger card maps (13 I)
- Phototubes quantum efficiency as a function of λ
- Front end calibration parameters (2424*10 F)
- Digital board parameters: thresholds, TDC gain and offset, channel mask, serial numbers.
- Analog sum parameters

Size estimate is around 250 kbyte that are written every time electronic setup is changed or modified.

Scintillator parameters and fluid handling system parameters

Scintillator and buffer parameters are measured at the beginning and maybe checked a few times a year. Some physical parameters are stored only once.

- PC emission spectrum (500 F)
- PPO emission spectrum (500 F)
- Pure PC, PC+DMP, PC+PPO time response (with α and / or β)
- Quenching parameters PC / PPO
- Birks parameter PC+PPO (F)
- Scintillator absorbtion spectrum (500 F)
- Buffer absorbtion spectrum (500 F)
- Scintillator emission spectrum (500 F)
- Scintillator decay function (50 F)
- Scintillator yield and lifetime (2 F)
- Liquid handling parameters (to be defined)
- Radiopurity of detector materials (to be defined)

Disk space required for these parameters is negligible.

Run parameters

These are the numbers that may vary for each run.

- Run number (1 I)
- Starting date and time (4 I)
- Digitizer board thresholds (2424 I)
- Phototubes HV (2424 I)
- Phototubes currents (2424 I)
- Trigger setup (2 I)
- DAQ parameters
- Alarms and anomalies

- Calibration histogram parameters from DSP (2424*12 F)[4]
- File name with Calibration histograms.

Data size is around 150 kbytes, written every run. Assuming to write 2 runs per day we make a very crude estimate of $2 \times 365 \times 10 \times 150$ kbyte = 880 Mbyte disk space for the whole Borexino lifetime.

This means that a small disk can store the whole Borexino lifetime[5]. With a good network link it is probably possible to keep a single data base server usable even from the U.S. (with 15 Kbyte/s link, the program may need 10 to 15 seconds per run to retrieve data base information).

[4]These are parameters obtained by fitting the single photoelectron histogram and the time histogram for laser events. The real histograms need far more space than any other data set and should be stored in separate files.

[5]Excluding calibration histograms.