SDA TIME INTERVALS*

Timofei B. Bolshakov, Jerry Cai, Elliott McCrory, Dennis J. Nicklaus
FNAL, Batavia, IL 60510, U.S.A.

Abstract

SDA (Sequenced Data Acquisition) Time Intervals is a hierarchical logging system for describing complex large-scale repeated processes. SDA has been used extensively at Fermilab for fine tuning during Tevatron Collider Run II. SDA Time Intervals is a new system born during discussions between CERN and FNAL about routinely recording relevant data for the LHC. Its main advantages are extremely low maintenance and good integration with traditional "flat" dataloggers. The Time Intervals (TI) system records the time of key events during a process and relates these events to the data that a traditional datalogger archives.

OBJECTIVES

Traditional SDA, Terms and Architecture

SDA (Sequenced Data Acquisition) is hierarchical datalogging system based on rules. The significant terms in these rules are event, device, collection and shot. A **collection** is a set of devices collected on specified events. There are several types of *collections*. The type is determined by a set of rules for different devices. A **shot** contains certain types of *collections* and event-based rules for starting and stopping. The source of many of those events is the Sequencer [1]. The data collected during the *shot* is stored in a relational database. Every *collection* has a type and a name associated with it, for example *collection* type 6 of Collider Shot has the name “Inject Pbars”. Several *collections* of the same type in one particular shot are called **case**. If *collection* is repeated several times then the *case* may have **sets** - several instances of the same *collection*. *Shots, cases* and *sets* are the main terms in SDA.

SDA data are processed by the numerous tools and applications of several layers. One of the first level application is SDA Viewer. It allows for viewing data and timestamps collected for shots, cases, and sets. Numerous daemons compute different summary tables during the Collider Shot and Pbar Transfer Shots. Other tables provide a “shot by shot” analysis, showing progress achieved weekly, monthly and yearly [2].

SDA also provides common language and performance numbers for different groups at Fermilab. Physicists found the role of this system critical for debugging and fine tuning the accelerator complex during Collider Run II.

Responsibility of maintaining SDA system is distributed between operators, software professionals, and special a SDA integration group. So, SDA is a costly, but necessary system.

*Fermilab is operated by Fermi Research Alliance, LLC., under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy.

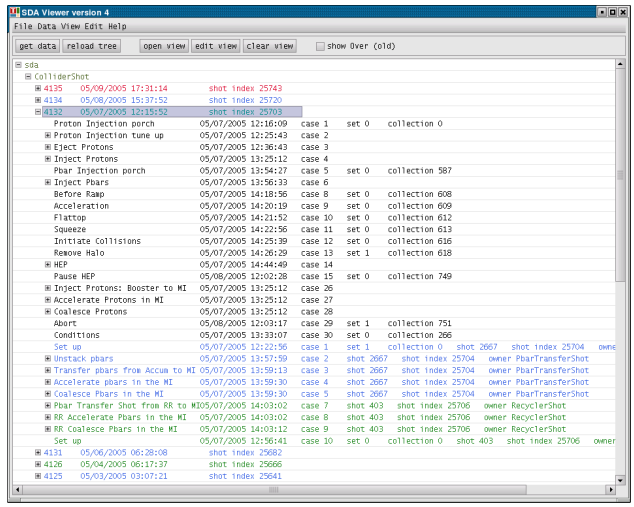

Figure 1: Shots cases and sets in SDA Viewer.

Reasons for New Implementations

LAFS (LHC At Fermilab Software) group was formed at Fermilab in autumn of 2006. The goal of this group is to share experience, ideas and software with LHC operators and control professionals. A portable version of SDA became one of initial LAFS projects, because the thought was that achieving LHC goals hardly possible without an SDA system or its analogue.

First suggestion was to create a “plugin based” version of the SDA system with an XML database at its heart. A “prove of concept” version of such a system was created and tested.

Drawbacks

Our CERN collaborators mentioned that such a system would be hard to maintain due to

- non-standard approach (namely by the use of XML database)
- significant time would be required to bring it up to speed, because all the devices have to be defined in a new database for each collection type. Yet, all devices are already logged in a standard timestamp-based datalogger.

Solution

SDA Time Intervals (SDA TI) solves the problem of high maintenance and initial time investment because the work to define device rules is not needed. It is implemented on relational DB. And it complements timestamp-based relational dataloggers. It provides ~90% of functionality we use in Fermilab. And additional ~10% of functionality can be added later, when the need becomes inevitable. SDA TI implementations is operational in Fermilab from spring 2007.

PROBLEM SETTING

SDA Basics – Reiterated

Let's look at a complex process (Pbar Transfer Shot from Accumulator to Recycler) and its SDA representation from an operator's point of view. In order to make the transfer Pbars should be unstacked from Accumulator and transferred to Main Injector, somehow

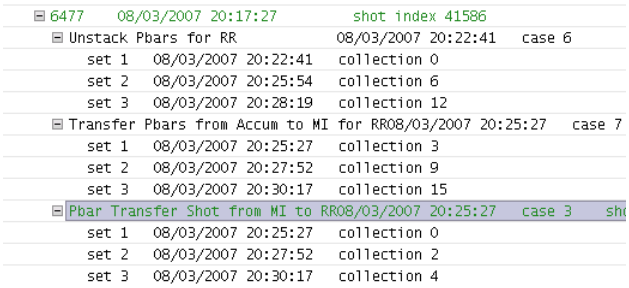

Figure 2: Pbar Transfer Shot in SDA Viewer

conditioned there, and injected into Recycler. Several transfers are made within a small time period, and Accumulator returns into the stacking mode. So, each of such a transfer involves 3 different accelerators, what means 3 groups of people and 3 different hardware systems. To collect the relevant data, each group is given one type of collection. Each type of collection consists of rules : on which event the collection should be started, when it should be stopped, and which device to collect on which event. Couple of rules regulate when to start and stop the whole shot.

Such a system also makes it possible to look how some subsystem was performing during the year. One needs to analyse data from just one type of collections for many shots.

Where Working Hours are Spent and How to Minimize Them

Most of the initial time investment should be spent for defining the rules for particular devices for every collection type. On the new complex, it may be unclear what would be needed to analyse the performance of a particular subsystem.

Rules for the collections – when to start a given collection and when to stop it – are a small fraction of the SDA configuration. And those rules are easier to understand from the beginning.

All the devices saved in SDA are supposed to be saved in standard timestamp–based datalogger. So, now solution seems to be natural – let's define the collection types and the collection rules and let's keep only the collection timestamps (start and stop) in a database. So, each collection will become a time interval. And let's tie each of those time intervals to a standard datalogger.

IMPLEMENTATION

Concept

To reiterate – we have following time interval's hierarchy: shot (of several types – currently Collider Shot or Pbar Transfer Shot), case (series of similar smaller time intervals), and set (a time interval of successful operation). Consecutive increasing numbers will be assigned to shots of the same type, to the cases inside each shot and to the sets inside each case. A name will be assigned to each case (type of time interval) and shot type. Each lowest-level time interval (an analogue of collection) will also have a "success" flag. Those time intervals form a tree-like hierarchical structure. Shots of different type belong to another tree.

Flat, timestamp based datalogger may be represented by another table with following fields: device index, timestamp, data.

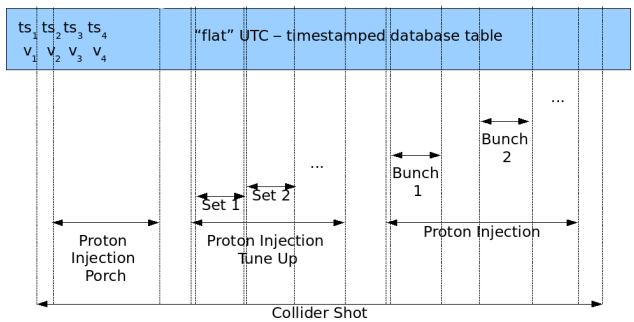

Figure 3: Time intervals in Collider Shot.

SQL Table Structure

The table structure is best described by the plaint SQL statement (Sybase):

```
create table flat_datalogger(
        devIndex      INT not null,
        ts            Timestamp  not null,
        devReading    REAL not null
)
create table sda_ti(
        shot_type     INT not null,
        shot          INT not null,
        theCase       INT not null,
        theSet        INT not null,
        file_idx      INT not null,
        coll_idx      INT not null,
        successful    TINYINT DEFAULT 1,
        tStart        Timestamp not null,
        tStop         Timestamp not null
)
```

file_idx and coll_idx are used here for compatibility reasons.

Data Acquisition

To fill the sda_light table we need to define the start and stop events for the shot, the case and the set time intervals, and write a simple event-driven data acquisition process. This procedure is straightforward.

Flat Datalogger – Filtering

Despite its simplicity, the suggested structure provides deep integration with "flat" dataloggers. SQL query for selecting "all values of device X during third proton bunch injection for the last year" is obvious. We call this functionality *filtering*.

Flat Dataloggers – Tagging

One can also easily tag each reading of flat database with "shot/case/set" or plot time intervals similar to Figure 3 on the standard datalogger plot. It can be done using simple SQL statement.

SDA Tree

The hierarchical SDA tree with "shot/case/set" nodes can be also easily build. The task of finding the correspondence between shots of different types also can be easily implemented.

Temporal Logic

Such a table defines temporal logic. Temporal logic operations ("before", "after", "overlaps", "contains", "is inside") over these time intervals can be easily implemented and used in application development.

Performance

It is worth to mention that all tasks described above can be efficiently implemented, because straightforward SQL statements can be written to implement most of functionality described above.

Summary Tables

Summary recomputed tables, analogues to Fermilab Supertable, Recomputed Emittances, Recomputed Intensities, Transfer Supertable and Bunch-by-bunch tables [2] can be easily implemented over SDA TI. OSDA API can be implemented also.

Missed Feature

The only functionality that cannot be implemented with SDA Time Intervals approach is saving big chunks of high frequency data for relevant events. This functionality is rarely used by most of the Fermilab SDA users, but it was important for several machine specialists.

CONCLUSION

The described SDA Time Intervals System was implemented in spring 2007 and works since then at Fermilab. Java API with implemented temporal logic helps to build "SDA Recomputed Tables" and to extract necessary flat datalogger data relevant to the accelerator state.

Author considers the described system as a good hierarchical model for many complex repeatable process. The Java API for temporal logic over time intervals, application analogues to the SDA Viewer are portable and can be applied to different processes.

REFERENCES

[1] J. Annala, "The Fermilab Sequencer - Use in Collider Operations", proceedings of ICALEPCS, 1995, Chicago.

[2] T.B. Bolshakov, P. Lebrun, S. Panacek, V. Papadimitriou, J. Slaughter, A. Xiao, SDA-based diagnostic and analysis tools for Collider Run II PAC 2005, Knoxville, TN.