SUMMARY OF A DISCUSSION ON IDEAS FOR A FIRST POST MORTEM PROTOTYPE

Date: 29th March 2004
Present: Steven Page, Ronny, Billen, Jorg Wenninger, Mike Lamont, Hervé Milcent, Bruno Dupuy, Franck Di Maio, Robin Lauckner

Purpose

This meeting was to try and firm up a proposal from Mike to build a prototype Post Mortem system.

Scope

Robin was concerned about meeting the deadline to specify the PM client API at the end of June. Mike wanted to check out the SDDS protocol and tools in use at Argonne, DESY and RHIC. Hervé wanted to clarify details of the PM system in order to complete QPS system tests in the next months.

Hardware Commissioning

Robin reviewed the basic requirements for Hardware Commissioning. There will be 3 major systems supplying PM information after a Powering Abort: QPS, Power Converters and Powering Interlocks. The basic nature and structure of the data they will provide has been described. First ideas on the analysis of the data have also been collected.
Jorg commented that the 100 Hz data capture in the PO FGCs will not meet the requirements for capturing transients from Warm Magnets.

Data Gathering

Mike proposed that the first PM prototype should explore the usage of SDDS as the data representation and build the Post Mortem Event into a generic file directory. The data collection chain could be a standard network file system, avoiding special developments.
Hervé pointed out that it is essential to ensure that data are secured onto disk. Not only must the information be written but a check that the data is correct must also be applied. He urged that this responsibility should not be left with the clients.
Franck commented that NFS implementation under Lynx OS has been problematic. It is necessary to look into the reliability issues.
Mike suggested that after the data is collected some post-processing may be required to pull in complementary data such as alarms, to build a description of what data is in the event and perhaps to migrate the information to a PM database. While autonomous data scanning would perhaps act on the SDDS data more generic navigation and browsing may be implemented on the RDBMS.

SDDS

Mike ran through a presentation on SDDS that had been given by Michael Borland from the APS. He pointed out that apart from defining the data format SDDS is supported by data analysis tools. These are a set of simple filters designed to be used as pipes between files.
Hervé insisted that engineers will want to use the analysis tools from their offices. Steven and Mike said that there was support in various languages and across different platforms and referred to a paper from R. Soliday of APL.
Mike pointed out that the SDDS tools seem appropriate for our domain. Further evidence of this is that they are used at RHIC for PM.
Franck observed that from his experience such tools are used as a justification for adopting data formats but later on they are not used and you end up with just the data representation.
Steven said that the advantage of SDDS was the associated tools. XML is an alternative but doesn't give you much. SDDS appears to offer PM and Logging support "for free".
Ronny emphasized that it is the end use of the data that should define its format.

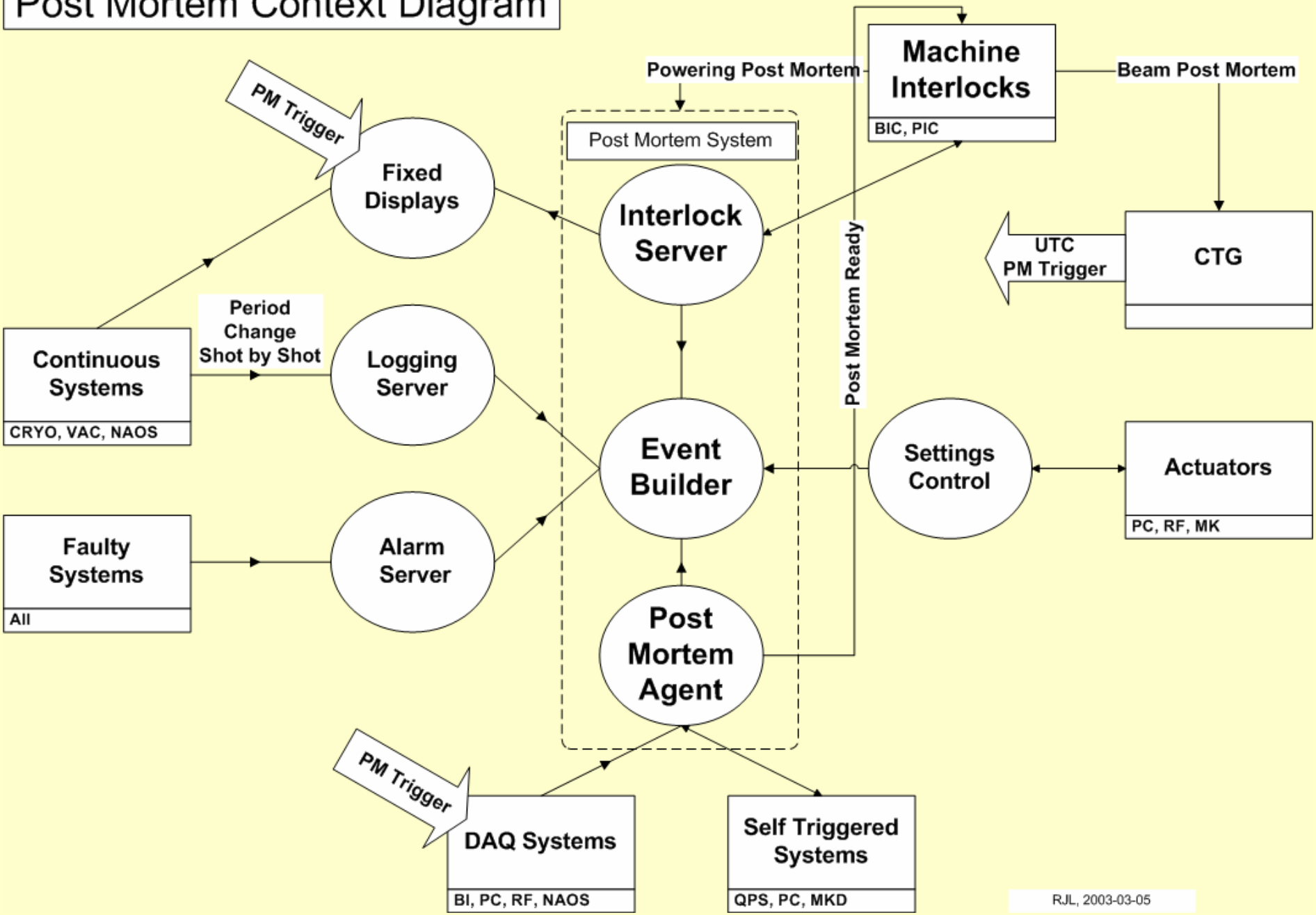
Goals of a Prototype

Robin enumerated the possible goals we could set for a prototype. These went through data gathering, archival, analysis and correlation with alarms and logging. He emphasized that his main concerns are the definition of the client API and the management of data collection in the concurrent environment of the LHC Hardware commissioning.
Hervé pointed out that the QPS has a Test Mode which supplies realistic PM data.

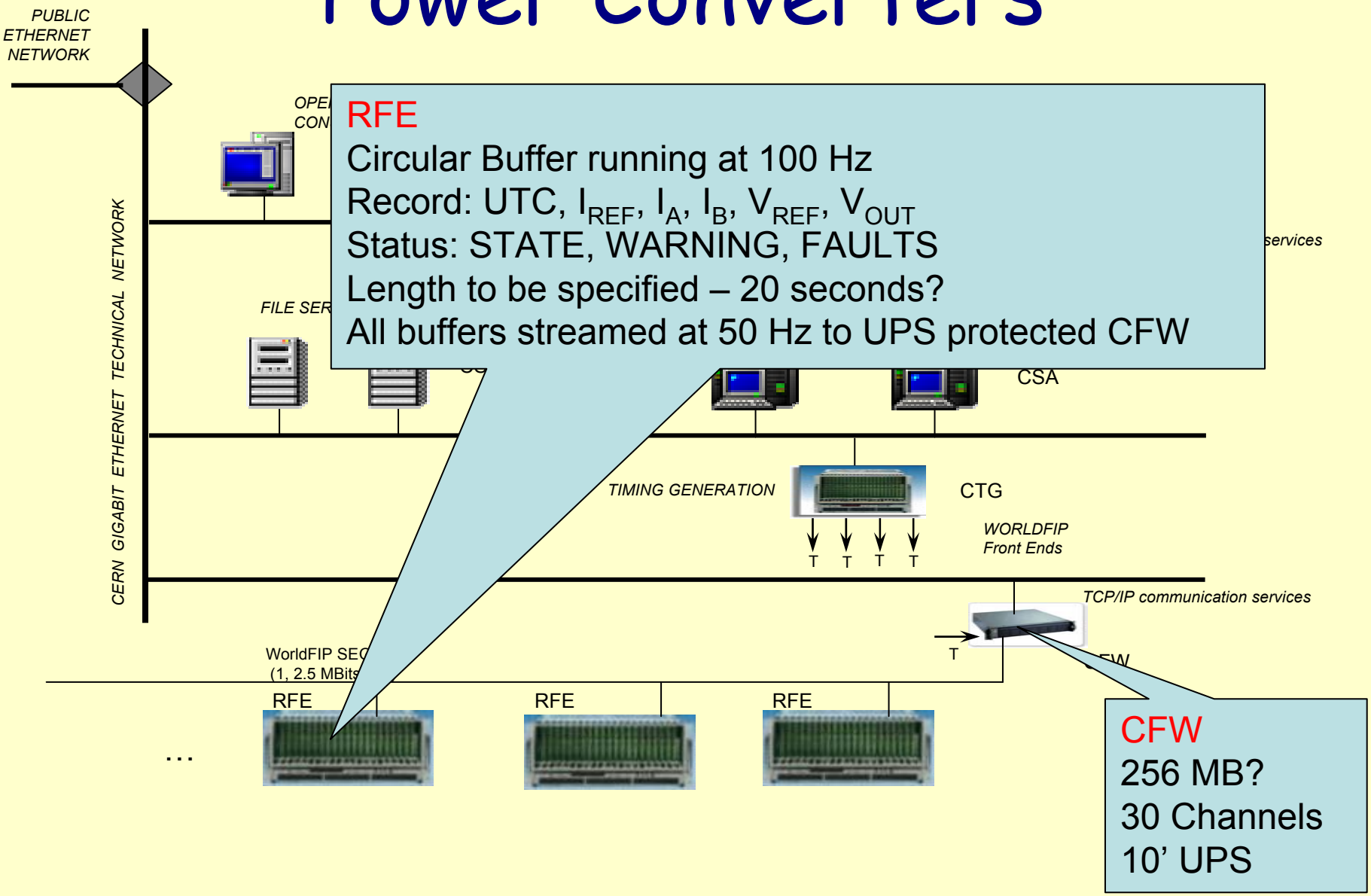
Conclusion

Mike and Robin should put together a more precise proposal.

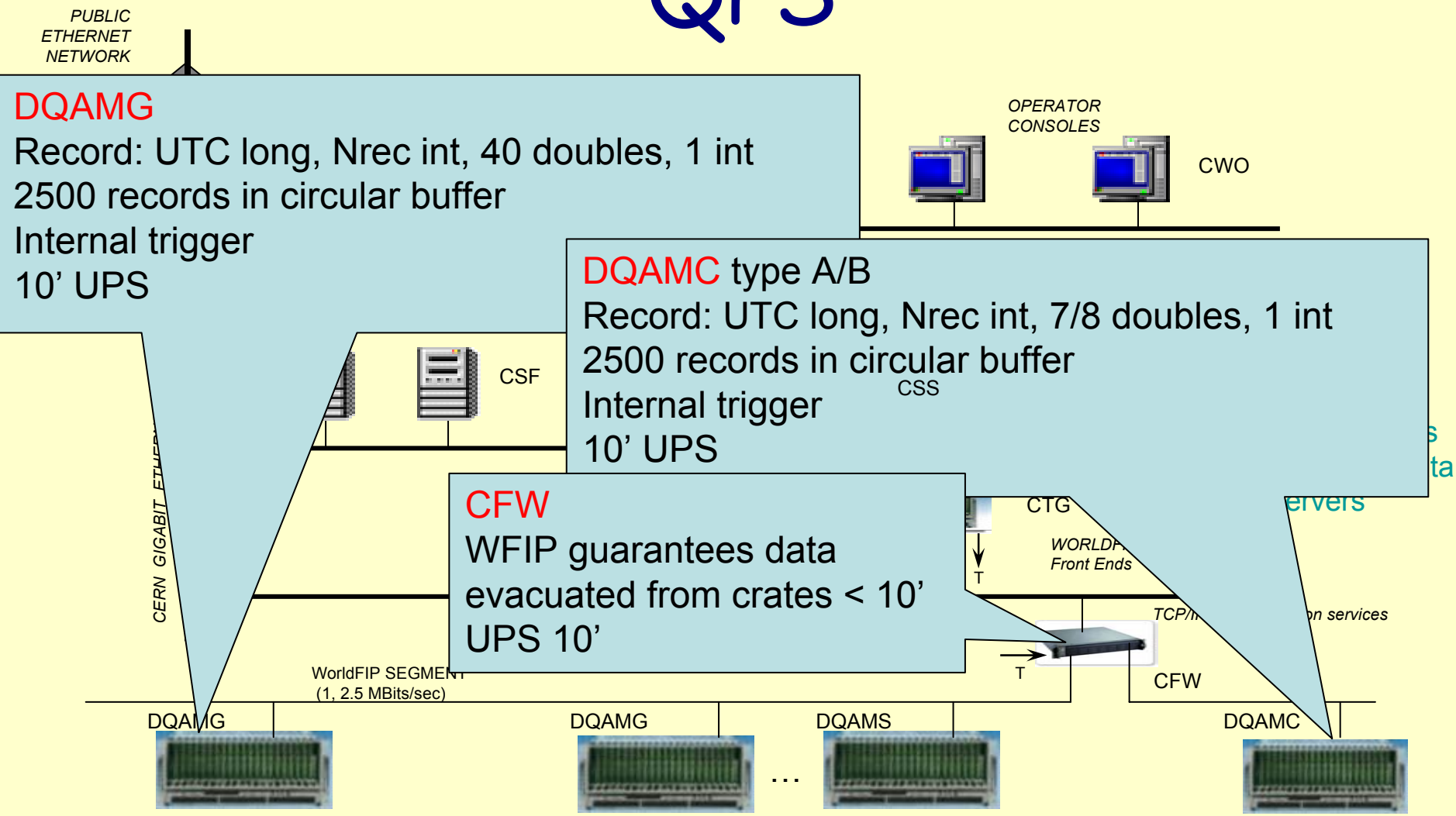
Post Mortem Context Diagram



Power Converters



QPS



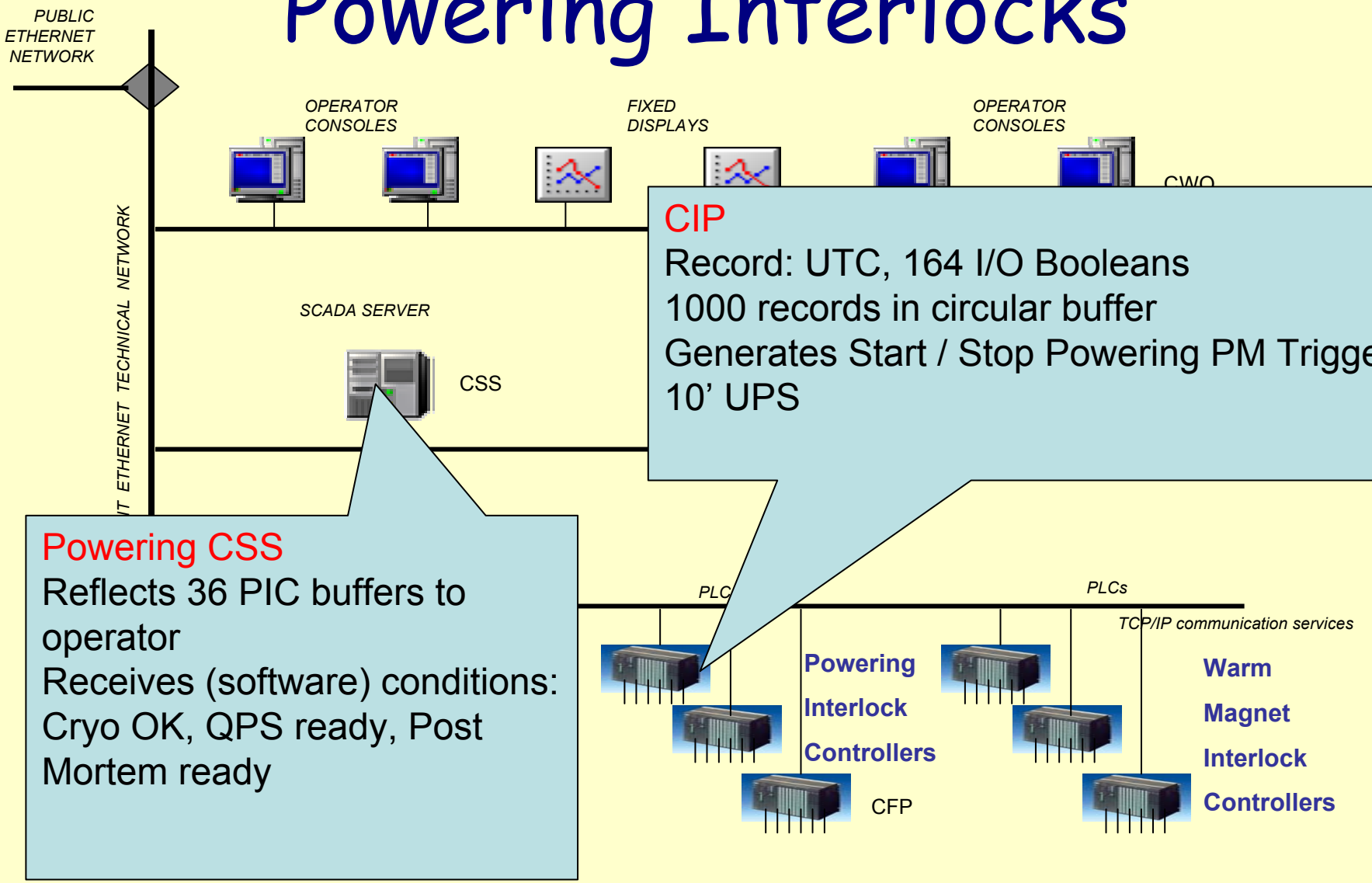
DQAMG
Record: UTC long, Nrec int, 40 doubles, 1 int
2500 records in circular buffer
Internal trigger
10’ UPS

DQAMC type A/B
Record: UTC long, Nrec int, 7/8 doubles, 1 int
2500 records in circular buffer
Internal trigger
10’ UPS

CFW
WFIP guarantees data evacuated from crates < 10’
UPS 10’

Global Protection Energy Extraction Main Magnets

Powering Interlocks



Analysis

1. Verify that the Machine Protection Systems (MPS) carried out all required actions
2. Identify the magnet that quenched
3. Identify the cause of the quench
4. Check that the QPS and Energy Extraction Systems performed correctly and that the circuit behaviour was within the expected range.
   - Inspect characteristics of PM transients
   - Verify that there was no loss of redundancy
5. Check that other actions were performed correctly and with normal behaviour

Goals of Prototype

- Develop PM Client Interface for June
- Define data representation
- Investigate concurrency requirements
  - Triggering
  - Tagging data
- Develop archiving strategy
- Link to reference database
- Link to logging data
- Link to LASER

Using SDDS for Accelerator Commissioning and Operation

Michael Borland
Operations Analysis Group
Advanced Photon Source
www.aps.anl.gov/asd/oag/oaghome.shtml

Introduction

- High-level applications at APS are based on
  - A common self-describing file protocol.
  - A toolkit of commandline programs that manipulate such files.
  - Tcl/Tk scripts to manage these programs and create GUIs.
- The protocol and programs are called "SDDS", for *Self-Describing Data Sets*

Outline of Presentation

- Concept and implementation
- What is self-describing data?
- SDDS file protocol and applications
- SDDS toolkit programs
- Advantages and problems
- Who uses SDDS?
- Applications
- Demos

Concept

- A generic data processing algorithm:
  Output = O_n ... O_2 O_1 *Input*
- Write programs that act as operators.
- Define a generic data-containing object for the operand.
- Applying sequences of programs creates arbitrarily complex transformations.
- Programs are re-used in many unrelated applications.

Examples of the Concept

- Simple lifetime measurement:
  acquireData | compute(Log) | fitPolynomial | display
- Robust lifetime measurement:
  acquireData | compute(Log) | fitPolynomial
  | removeOutliers | fitPolynomial | display
- Beam history analysis:
  acquireData | FFT | smooth | peakfind | collect(ByBPM)
  | display
- Find the noisiest power supply:
  acquireData | compute(Stats) | collect(BySupply) | sort
  | display

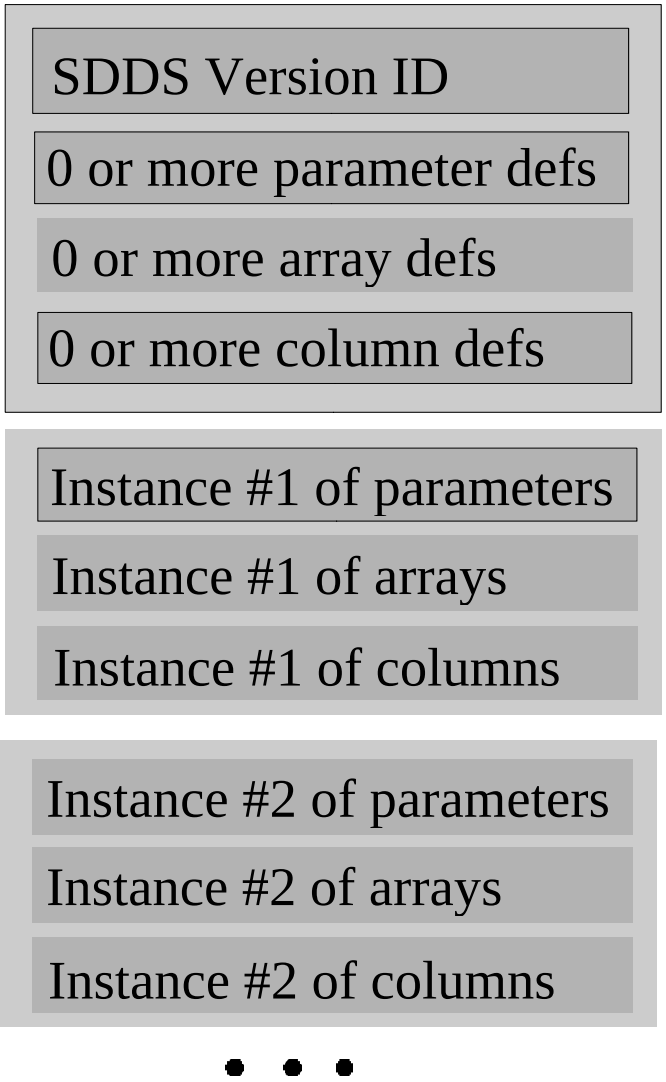
Implementation

- Consistently used a simple, common self-describing file protocol for data.
- Wrote generic, commandline programs using these files
  - Data collection
  - Data analysis
  - Graphics
  - Process control
- Used Tcl/Tk script language to
  - Record/create sequences
  - Create GUIs

What is Self-Describing Data?

- Identified and accessed by name only
- Units, data type, and other meta-data are included.
- Advantages:
  - Truly generic programs possible
  - Programs can verify and adapt to file contents
  - Augment file contents without breaking applications
  - Self-documenting
  - Integrates simulation, control system, and other data sources

SDDS File Protocol



Header: defines a data structure

Page 1: an instance of the structure

Page 2: an instance of the structure

Examples of Uses of SDDS Files

- Back-up and restore files (BURT)
- Archival data from machine monitoring
- Alarm history data
- Magnet conditioning instructions
- Waveforms from scopes and network analyzers
- Beam profile and images
- Feedback matrices
- Orbit correction configuration data
- Beam position monitor status database

SDDS Toolkit Programs

- SDDS is used by a group of about 70 generic programs
- Most of these "SDDS Toolkit" programs both read and write SDDS files, so
  - They can be used sequentially
  - Even simple tools become useful and productive
- About 20 EPICS-specific programs use SDDS
- Programs are commandline driven and hence scriptable

SDDS Toolkit Capabilities

- Device-independent graphics
- Equation evaluation
- Data filtering, sorting, collection, and cross-referencing
- Statistics, correlation analysis, and histograms
- Polynomial, exponential, and gaussian fitting
- Outlier analysis and removal
- Matrix operations
- FFT and digital filtering
- Derivatives and integrals
- Conversion to/from text and other formats

SDDS/EPICS Toolkit Capabilities

- Time-series data collection and statistics collection*
- Glitch/trigger initiated data collection*
- Synchronized data collection
- Alarm data collection*
- Experiment execution**
- Snapshot save, restore, and ramp**
- Feedforward, feedback**, and optimization**

*Used at APS for continuous archiving.
**Used at APS for routine operations.

Advantages of SDDS

- Tools for on-the-fly experiments, data analysis, etc.
- Permits very rapid testing, implementation of ideas
- Gives "muscle" to Tcl/Tk scripts
- Simplifies the development of new applications
- New programs have an amplified and often unexpected payoff
- Analysis capabilities comparable to MATLAB or IDL, but SDDS is free
- Open source

Problems/Complaints

- SDDS commandline tools are hard to use for newcomers and occassional users.
- SDDS files are not random-access files. A page is read into memory, following which the application requests copies of needed data.
- Does not provide cross-platform reading of binary files. (Solved in next release.)
- Slower execution than custom code.

Who Uses SDDS?

- APS depends on SDDS for accelerator operation, archiving, data analysis, and simulation.
- IPNS uses SDDS for archiving, analysis, and display.
- RHIC uses SDDS files throughout the control system but doesn't use SDDS tools at present.
- BESSY II uses SDDS files and tools for data archiving, automated processing, and some applications.
- DESY is adopting SDDS files for their data archives.
- SNS has some limited experimental use of SDDS.
- Accelerator simulators (ANL, DESY, LBL, SLAC, ...)

Selected Accelerator Physics Activities Performed with SDDS

- Magnetic measurement data analysis
- Magnet conditioning and configuration*

- Model-independent steering*
- Obit/trajectory response matrix measurement*
- Orbit correction*
- Insertion device beamline steering*

- Tune and chromaticity measurement*
- Beta-function and dispersion measurement and correction*

*GUI application

Selected Accelerator Physics Activities Performed with SDDS

- Dynamic aperture measurement*
- Energy aperture measurement*
- Physical aperture search

- BPM-to-quadrupole offset measurements with beam*
- BPM intensity dependence measurement and compensation*
- Automated BPM timing scans and timing setup.*

Selected Accelerator Physics Activities Performed with SDDS

- Transport line emittance measurement and beta-function matching*
- Bunch length measurement using rf zero-phasing*
- Automated processing of beam spot images from SASE FEL experiments*

PROLIFERATION OF SDDS SUPPORT FOR VARIOUS PLATFORMS AND LANGUAGES

R. Soliday, APS/ANL, Argonne, IL 60439, USA

Abstract

Since Self-Describing Data Sets (SDDS) were first introduced, the source code has been ported to many different operating systems and various languages. SDDS is now available in C, Tcl, Java, Fortran, and Python. All of these versions are supported on Solaris, Linux, and Windows. The C version of SDDS is also supported on VxWorks. With the recent addition of the Java port, SDDS can now be deployed on virtually any operating system. Due to this proliferation, SDDS files serve to link not only a collection of C programs, but programs and scripts in many languages on different operating systems. The platform-independent binary feature of SDDS also facilitates portability among operating systems. This paper presents an overview of various benefits of SDDS platform interoperability.

1 VARIOUS PORTS

1.1 Versions of C

Originally SDDS programs [1] were written to store and manipulate accelerator data at the Advanced Photon Source (APS). Since that time SDDS usage has spread to many additional facilities with different needs and different hardware. This was accomplished by undertaking the effort to port SDDS programs to additional hardware platforms and to additional programming languages. With the addition of these various ports we have been able to keep a generic source code base that compiles on all the supported operating systems. This has helped to facilitate upgrades because changes in the source code affect all of the SDDS versions.

The SDDS Toolkit, which consists of over 100 applications, was originally written in C on Solaris. A Tcl/Tk extension was also written for Solaris. In order to accommodate other users, the SDDS Toolkit was first ported to the GNU C compiler on Red Hat Linux because it was deemed to be the easiest place to start. This version of the SDDS Toolkit is now available in the form of a binary Red Hat package. It was then ported to Visual C++ and the free Borland C++ Builder on Windows. These versions are now available as self-installing executables. The APS is now also using a VxWorks port of SDDS so that the Experimental Physics and Industrial Control System (EPICS) input/output controllers (IOCs) can read and write SDDS files.

1.2 Wrappers and Extensions

Using a FORTRAN wrapper for the SDDS C libraries, it has been possible to incorporate SDDS function calls in FORTRAN programs. A similar technique was used with the Python programming language. Both of these ports require that the SDDS C libraries also be built. This was done because there was a desire to avoid having to maintain multiple source code bases.

1.3 Java

Most recently SDDS has been ported to Java. In order to take full advantage of Java's cross-platform capabilities it was decided to avoid using the existing SDDS C libraries. This required writing the Java SDDS port from scratch. This code has evolved over time and is now totally SDDS compatible with the exception of SDDS array types. Benefits of this port include the fact that any Java SDDS application is automatically cross platform and the ability to extend existing Java programs so that they are SDDS compatible.

2 APPLICATIONS

2.1 SDDS Toolkit and OAG Tcl/Tk Software

Various SDDS applications are distributed through the Operation Analysis Group (OAG) web site[1]. According to our statistics 40% of the downloads are for precompiled Windows software, 20% for precompiled Linux software, and 40% for source code which can be compiled on any of our supported platforms. These applications include the SDDS Toolkit, which is used for postprocessing of the databases [2]. Also included are SDDS EPICS software that can be used to monitor and manipulate process variables (PVs) in the IOCs. There are also various accelerator modeling and simulation programs using SDDS databases.

The original versions of most of the SDDS programs have been refined at the APS on Solaris operating systems. The operators at the APS use C and Tcl/Tk

[1] http://www.aps.anl.gov/asd/oag/oaghome.shtml

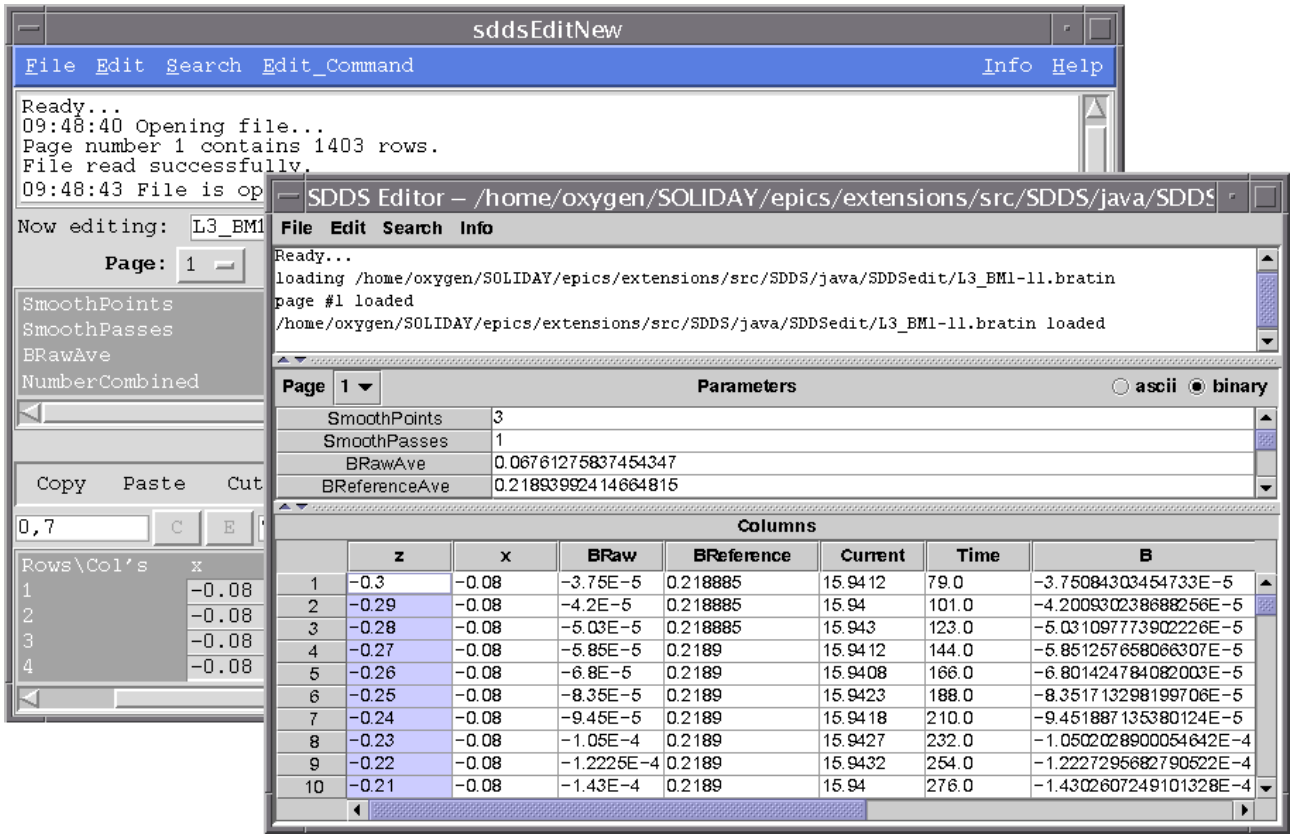


Figure 1: Example of an SDDS file editor written in Java and Tcl/Tk.

SDDS applications extensively. Over time enhancements have been added to existing programs and new programs were written to add further functionality. The usual setup of a program used in the APS control room consists of a Tcl/Tk graphical user interface (GUI) that directly interfaces with SDDS files and PVs or a Tcl/Tk GUI that executes various SDDS C programs. These programs include the Procedure Execution Manager (PEM), which has been used to automate accelerator operations [3,4].

2.2 VxWorks Applications

The core SDDS library routines that have also been ported to VxWorks are now being used at the APS to load and store configuration data in some of the IOCs, as well as being used to write video images to disk. Another more limited use of SDDS in the IOCs has been to run some of the SDDS Toolkit applications. If demand for this increases further, SDDS Toolkit applications will be ported to VxWorks.

2.3 Accelerator Simulators

As a result of requests from other institutions and the prospect of using faster and less expensive hardware, these programs have been ported to both Windows and Linux. Beyond simply porting these programs, they are also now available in precompiled binaries. This makes deployment of these programs much easier for most users. Currently these ported versions are being used at the APS to run accelerator simulations on very fast computers that are relatively inexpensive. Future plans include investing in more x86 processors to run large-scale simulations.

In addition to porting to other platforms, there has been work done to port to additional languages. A FORTRAN wrapper is used to add SDDS functionality to GENESIS [5], a time-dependent free-electron laser (FEL) simulation code written by Sven Reiche. The code was modified so that the output files can be plotted with existing SDDS plotting programs. Also the output from elegant, an accelerator simulation program, can now be used as input to GENESIS. This work has helped to achieve start-to-end accelerator simulations [6].

2.4 Java

Most recently SDDS has been ported to Java. Using this new language, a few new programs have already been created. A cross-platform SDDS file editor (see Figure 1) was written loosely based on a Tcl/Tk SDDS

editor. Another program written in Java is a three-dimensional plotting program (see Figure 2). This program can be used to plot SDDS data as a surface plot or a scatter plot. New SDDS applications can be relatively easy to create in Java from scratch or by integrating SDDS functionality with a preexisting program. One obvious advantage to writing programs in Java is that the programs will run on operating systems that are not supported by previous SDDS ports, such as Macintosh.

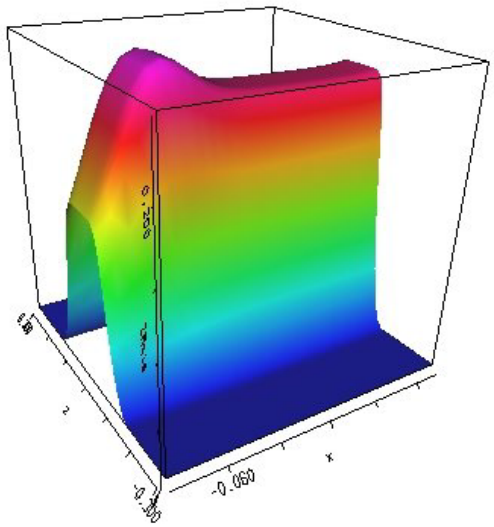


Figure 2: Three-dimensional plot using Java SDDS 3D plotter.

3 BENEFITS

3.1 Cross Platform

One benefit to having cross-platform database software is that there is no need to change the existing computer infrastructure to use SDDS software. It is not even required that all the computers at one site be of the same type. The SDDS software has binary interoperability, which allows the programs the ability to read SDDS files that were created on different operating systems.

3.2 Easy Deployment

Now that there are binary SDDS programs available on Windows and Linux, SDDS is much easier to deploy since the source code does not have to be recompiled. This has brought SDDS software up to the standards of many commercial packages that people have become accustomed to using. The programs are packaged using InstallShield on Windows and RedHat Package Manager on Linux. For the Solaris operating system it is still necessary to compile the source code because this is the standard distribution method for Solaris.

3.3 Stable Software

One unexpected advantage of porting SDDS to various operating systems and different C compilers was the ability to detect and remove problems in the source code. Each C compiler has a different set of compiler warnings that we used to refine the SDDS software. With these changes many possible problems have been averted before they became apparent. Using this technique we can check changes to the source code prior to public release.

Automated test scripts written for Solaris, Linux and Windows also test these programs. These scripts run each program through a series of tests using various options. More detailed debugging is accomplished on Solaris by using Purify and Quantify from Rational Software.

ACKNOWLEDGMENT

Work is supported by U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

REFERENCES

[1] M. Borland and L. Emery, “The Self-Describing Data Sets File Protocol and Program Toolkit,” Proc. 1995 ICALEPCS, May 1-5, 1995, Dallas, Texas, pp. 653-662 (1996).

[2] M. Borland, “User’s Guide for SDDS Toolkit,” http://www.aps.anl.gov/asd/oag/manuals/SDDStoolkit/SDDStoolkit.html.

[3] R. Soliday et al., “Automated Operation of the APS Linac using the Procedure Execution Manager,” Proc. International Linac Conference (LINAC2000), August 21-25, 2000, Monterey, California, pp. 524-526 (2000).

[4] S. Pasky et al., “Efficient and Effective Operation of the APS Linac,” Proc. Workshop on Accelerator Operation (WAO2001), January 28 – February 2, 2001, Villars sur Ollon, Switzerland, to be published.

[5] S. Reiche, *Nucl. Instr. Meth.* A **429**, p. 243 (1999).

[6] M. Borland et al., “Start-to-End Jitter Simulations of the Linac Coherent Light Source,” Proc. 2001 Particle Accelerator Conference (PAC2001), June 18-22, 2001, Chicago, Illinois, to be published.

Hi,
Possible idea for prototyping the data collection aspect of the Post Mortem system:

- equipment groups are collecting their front-end PM buffers on the gateways following PM event
- from the the gateway write buffers to file system (would imagine a symbolic link to "current PM" directory structure - would clock this at the top level)
- Format files in Self Describing Data Sets (SDDS) format (see links below) (XML as another option)
- post-processing of data sets could then take place as necessary (including the possibility of writing some stuff to Oracle)

SDDS was developed at Argonne for the APS, has come through several iterations. Forms the basis of their control system (settings, measurements etc. - not suggesting that). Lots of supporting tools. Used by RHIC in their post mortem system (see below).

Stephen Page has had a very quick look - his comments attached. (Will discuss more with him over a beer or two)

Cheers
Mike

http://www.aps.anl.gov/asd/oag/manuals/SDDStoolkit/SDDStoolkit.html

For an example of the format see: looks very intuative.

http://www.aps.anl.gov/asd/oag/manuals/SDDStoolkit/node3.html

RHIC...
http://arxiv.org/ftp/physics/papers/0111/0111159.pdf

Stephen wrote:

I had a very quick look at the SDDS stuff. It looks interesting and seems a potential option for PM logging.

One could envision doing a similar thing with XML, so while I was reading
the SDDS docs I was trying to compare it to what could be done with XML.

Advantages of SDDS seem to be (at first glance):

1. Someone has used it before for a similar application and thought about which fields are needed. If we used XML, we would have to define a set of standard fields (e.g. name, type, units, description) and the structure of the files.

2. Supporting library exists to write the files(?). Not quite sure about this one, but I think I saw a few references to a library. XML libraries do exist (such as expat), but they are generic and further code would need to be written to provide for our specific needs.

3. Analysis tools are provided. SDDS supplies tools to perform some analysis which may or may not be useful for us. Hopefully though, this would imply that tools / libraries to parse generated SDDS files are provided as part of the toolkit.