

## Chapter 13

# Computing and Software

There are a large number of computing challenges that CDF must face in Run II. The data rates will be much higher than those we have dealt with in the past. Consequently we will have much larger datasets which must be stored and made accessible in a timely fashion to many physicists at many collaborating institutions. In addition we would like to evolve our software environment to use more modern programming techniques and languages such as Object Oriented programming and C++.

In the next section we review the present Run I system followed by a section describing the expected data volume in Run II. The remaining sections describe the computing model, the software environment, the event reconstruction, the calibration database, and finally the simulation requirements.

### 13.1 The Run I System

In Run I CDF recorded 64 million events in the main data stream (Stream B) which were processed in real-time and made available to the physicists within about 2 weeks of the data being taken. In addition, 3.7 million events out of the total were also recorded in a separate “express line” stream (Stream A) and processed within hours. This stream contained the high-transverse momentum leptons necessary for the top quark search. It was also used to monitor data quality during the run. An additional 28 million events were recorded for processing after the data taking ended (referred to as Stream C). These events were mainly low-transverse momentum leptons intended for studies of b-quarks.

The analysis computing needs for Run I were provided by a mix of central and desktop systems. These systems were a combination of VAX/ALPHA machines running VMS plus Silicon Graphics and IBM machines running UNIX. The central VMS system

consisted of about 1200 MIPS of CPU (FNALD) and the central UNIX system (an SGI Challenge XL with 28 processors - CDFSGA) consisted of 2400 MIPS. In addition, the desktop systems add 3000 MIPS of VMS computing and 1600 MIPS of UNIX computing.

The production computing needs were met by the centrally supported UNIX farms. CDF had an allocation of 3000 MIPS of farm computing in the form of 63 SGI 4D/35 nodes and 32 IBM 320H nodes. The data were processed on the farm nodes and then split up into physics datasets on the farm I/O nodes and then staged to 8mm tape.

The data sizes and volumes are summarized in Table 13.1. There were 27 DST datasets and 39 PAD (PAD - Physics Analysis DST) datasets. The DSTs from the main data stream are not heavily used in analysis. The total data volume for Run I was 41 Terabytes, of which 16 Terabytes was due to inclusion of raw data on the inclusive and split DSTs. The information about where a file is located is stored in an experiment specific database which uses FORTRAN indexed files and is kept on FNALD.

The primary means of data storage for Run I was 8mm tape (double density tapes with 5 Gbytes of storage and a maximum I/O rate of 450 Kbytes/second). Selected datasets were further reduced and stored in an STK tape robot which had an initial capacity of 1 Terabyte and has recently been upgraded to 3 Terabytes. This system is accessed via the FATMEN catalog system (a CERN product) and an automatic staging system - users do not need to know which tape the data is stored on, only the filename. The central UNIX and VMS systems also have about 500 Gbytes of disk devoted to physics datasets. There is over 200 Gbytes of disk in the staging pool for the STK robot. There are also significant disk resources (about 600 Gbytes) attached to desktop systems.

Data Type	Size (Kbytes)	Total Volume (Terabytes)	Comments
RAW	130	8.3	
Inclusive DST	190	12	The DST includes the RAW data
Split DST	190	16	There is a 30% overlap of events on the split DST
Inclusive PAD	32	2	
Split PAD	32	2.7	There is a 30% overlap of events on the split PAD

Table 13.1: Summary of the Run Ib data volume.

Copies of selected datasets were made for distribution to remote institutions. In Run Ib about 17000 8mm tape copies were made. About 3600 of these were kept on site at Fermilab for use by physicists in the CDF Portakamp complex and the rest were sent offsite.

The software environment uses FORTRAN as the primary programming language with limited use of C in system level applications. A mix of experiment developed applications, Fermilab Computing Division products and CERN products are used in the software development and analysis. The code is presently supported on VAX/ALPHA VMS, SGI UNIX and IBM AIX platforms.

## 13.2 Expected Data Volume and processing requirements

The upgraded DAQ system is expected to be able to handle a Level 3 input rate of about 300 Hz with an peak output rate of 30-40 Hz assuming a 90% rejection factor. We expect the raw event size to be about 250 Kbytes per event, hence the Level 3 output rate is expected to be 7.5-10 Mbytes/second to mass storage. Past experience has shown us that the 40 Hz peak translates to a 20 Hz average rate while there is beam in the accelerator and the experiment is recording data. This implies that we will record about 300 million events per year, yielding 80 Terabytes of raw data. This is about 10 times the total Run I volume. In a two year run which is assumed to be about  $2 \text{ fb}^{-1}$  the dataset will be 600 million events

and 160 Terabytes in size. This corresponds to the main dataset, we have not included any low priority stream such as the Run I Stream C data.

In order to estimate the sizes for the reconstructed data we have assumed that the reconstructed data size remains at about 50% of the raw data size and that the size of a PAD event will be about 15% of the raw data size. We also assume that there will be an “express-line” stream which will be about 10% of the total dataset, and that these events will be written out separately so they can be processed immediately. As noted above, we have duplicated the raw data on both the inclusive DSTs and the split DSTs. We will assume that this is no longer practical due to the larger data volumes expected and so the DST format will no longer include the raw data. We will also assume that for most data streams there will not be a split DST. Under these assumptions the DST+PAD information will add 75 Terabytes of information. For ease of access we will probably need to store the split PAD events separately which will add another 24 Terabytes of data. The “express-line” raw data will be 16 Terabytes and the DST+PAD information will add 8 Terabytes. The split “express-line” PADs will be another 2 Terabytes. So the total volume will be about 300 Terabytes for a  $2 \text{ fb}^{-1}$  run (allowing for a 30streams). Note that if the DAQ system delivers more Level 3 output rate then these numbers will increase. We take 300 Terabytes for  $2 \text{ fb}^{-1}$  as our baseline.

At present we do not have an estimate of the CPU time per event for reconstruction in Run II. Much of the code will be new and does not yet exist. We

can make an estimate of the production requirements based on the execution time per event in Run I. In Run Ia the execution time was 400 MIPS seconds/event for an average instantaneous luminosity of  $3 \times 10^{30} \text{ cm}^{-2} \text{ s}^{-1}$ . In Run Ib the execution time was 700 MIPS seconds/event at an average luminosity of  $9 \times 10^{30} \text{ cm}^{-2} \text{ s}^{-1}$ . This increase was due to the extra interactions per crossing at the higher luminosity. Although the luminosity will increase further in Run II, the number of bunches in the accelerator is increasing so we estimate only an additional 20% increase in processing time from this source. Allowing for some increase in processing time we will use 1200 MIPS second/event for our baseline execution time for Run II. We need to process 300 million events per calendar year, i.e. 6 million events per week. The input rate required would be 2-3 Mbytes/second and the CPU requirements for processing would be between 15,000 MIPS to 25,000 MIPS depending on the processing efficiency.

### 13.3 Computing Model

In this section we describe the proposed computing model for Run II and its various components. We will first describe the Run I computing model. The Run II model will be similar but with some important changes to provide faster and more transparent access to large datasets.

#### 13.3.1 Run I Computing Model

In Run I we have followed a model where the data and the CPU are tightly coupled. We have not provided high-bandwidth access from desktop systems to the central data stores. In addition, we have no dedicated batch systems, all the central systems are used for both batch and interactive computing.

The current CDF data persistency package is YBOS<sup>[1]</sup>. The data is stored in YBOS banks. The raw data is stored on 8mm tape. The event reconstruction is then performed on farms of UNIX workstations. A subset of the YBOS banks containing physics information from the reconstruction are also compressed to create PAD events. The events are also split into production datasets. These datasets are taken by the physics groups and further split or compressed. This can be a lengthy step and difficult to recreate if a problem is discovered later. This is an I/O intensive job and is usually performed on the

central systems (usually by a small number of people) where there is efficient access to large production datasets on 8mm tape. The resulting datasets are stored on disk or if they are too large for disk they are placed in the STK tape robot. Up to this point in the process, the data is still in the form of YBOS banks and access to the events is file-based.

The users doing physics analyses perform further selections of these datasets, normally on the central systems. The result of these event selections is typically an ntuple<sup>[2]</sup> which stores the data in ZEBRA RZ<sup>[3]</sup> format. These ntuples are usually small enough to be transferred to the desktop system for further analysis. Most physics analyses start from the PAD data. However a small number of analyses such as the W-mass measurement require the DST data. All analyses may require the full DST for a small number of events. The amount of data used from the 32 Kbyte PAD event ranges from 1 Kbyte for exotic searches to about 10 Kbyte for a typical b-quark analysis.

#### 13.3.2 The Run II Computing Model

The primary operating system in Run II will be UNIX. The VMS operating system will not longer be a supported operating system at Fermilab and will not be part of the Run II computing environment. We do not yet know what role personal computers (PCs) might play. The goal is to design a system that is as operating system independent as possible to allow us to respond to future trends in computing with minimal upheaval.

**Production** UNIX farms are a cost effective way to obtain the necessary CPU cycles to perform the event reconstruction and we expect that this will be the primary method of doing bulk event reconstruction in Run II. However, we do not ignore the possibility that farms of PCs running LINUX, for example, could play a role.

The production farm I/O servers will read raw data directly from the tape robot and write the processed data back to the same robot. The processed information for an event will not be stored in the same file or tape as the raw information. As stated above, the DST format will no longer include the RAW data as this will significantly increase the storage requirements. The data will also be split into physics streams at this stage. In most cases the splitting

will be done at the PAD level. Only for a few selected datasets would split DSTs be produced. The information about dataset location and which events belong to a particular dataset would be recorded in a central database.

**Mass Storage and Data Access** The baseline data volume expected in Run II is 250 Terabytes. It is not practical to imagine storing this quantity of data on 8mm tape. We plan to write the data to staging disk at the experiment and transmit it directly to staging disk attached to a robotic tape store in Feynman Computing Center (FCC). Tape robots already exist that can handle the data volume we will have in Run II with the required data transfer speeds to and from the magnetic media. Clearly the exact choice of technology will be made closer to the start of Run II. In addition to the robot, one requires Hierarchical Storage Management (HSM) software to control the robot.

It may be desirable to store all the information about a particular type of physics object in one location in the robot to improve the access time for event selections, which typically look at a small amount of information.

A database will be necessary to record the information about which events belong to a dataset and on which physical tape they reside. The database must interface to the HSM software to allow the information to be retrieved from the tape. Retrieval of data should be transparent to the user, i.e., the user need only know the name of the dataset they are interested in, not the list of files that make up that dataset. Also the user should not need to know the physical organization of the dataset. Users would submit event selection requests to the central database by specifying which dataset they want to access and providing a set of selection criteria. The physics datasets could be created from the production dataset in the same way. If these datasets were large it would probably not be practical to physically duplicate the events. Instead the list of events that make up the dataset would be recorded in the central database.

The robot will be attached to the central analysis facility which will consist of some number of multiprocessor UNIX machines coupled together by a high-speed network. These machines will support both batch and interactive use. Users could also submit event selection requests to the central database from their desktop workstation and have the selected data

returned to the workstation. The event selection will run on the central system, thus ensuring efficient access to the central data store, and only the selected data will be returned, thus minimizing the network traffic. A goal would be to read a 2 Terabyte dataset in a few days, this requires tape speeds of the order of 10-20 Mbytes/second. If we are able to read pieces of events then this time could be reduced even further. With this kind of access time, remaking a dataset would not incur a severe time penalty.

Clearly, there will still be a need to export datasets to remote institutions. The wholesale duplication of some of the larger production datasets may not be practical. It may be more useful to provide copies of the physics datasets which will be smaller. Given the goals stated above for reading datasets, it will also be possible for users to create specialized datasets on the central system and copy them to tape for export.

Most frequently accessed datasets would be kept on disk. It might be necessary, in order to maximize the use of the disk space, to keep only the most frequently accessed components of the events on disk, .e.g. for a QCD dataset we may decide that it is not necessary to keep the tracking data on disk. If a component of an event is requested and the component is not on disk then a staging operation would be performed to retrieve the necessary information. This would be transparent to the user.

Another idea that is being investigated is the concept of "data mining". A subset of the data is stored on disk in a format which allows for fast event selection queries to be run over large amounts of data. The selected data can be output in a user-selected format, e.g. ntuples. The event selection only uses the disk resident scannable data but can then trigger retrieval of additional data from tape for any given event. This approach is under investigation in the High Performance and Parallel Computing group in the Fermilab Computing Division.

Some of these goals imply changes to the CDF persistency package YBOS. This package treats events as a sequential list of banks which is too restrictive for Run II. We are presently evaluating alternative persistency mechanisms. One option is to rewrite the YBOS package in C++ to include the necessary features. Some work has already been done on this option and it looks feasible. Another interesting option is the use of an Object Oriented database to store the data. This approach is being seriously studied by the RD45 project at CERN <sup>[4]</sup>. The potential cost of

such an approach is an issue as is the maturity of the technology. Even if this is not a viable option for the beginning of Run II, we should not design a system that prevents the use of such approaches later on.

**Networking** We are at present planning an upgrade to the networking infrastructure at CDF which will bring fiber to each desktop, thus opening up the possibility of FDDI, ATM etc. to the user's workstation. This upgrade should be sufficient to meet the desktop networking needs for Run II.

In the central systems we may require higher bandwidth connections. We will have to connect together multiple multiprocessor UNIX machines and provide each with a high-bandwidth connection to the robot. Possible candidates include Fiber Channel or HIPPI.

A dedicated 10-20 Mbyte/second connection between B0 and FCC may also be necessary for the transfer of the raw data from the staging disks in B0 to the robot in FCC.

**Analysis Computing** The analysis computing will be provided by a mix of central systems plus desktop workstations. As stated above, the machines will be running the UNIX operating system. The role of PCs in the system has not been considered at present.

We can make a rough estimate of the CPU needs by scaling from Run I. This suggests that we will require roughly 30,000-40,000 MIPS of computing to meet the Run II analysis needs. This can be spread between central systems and desktops.

## 13.4 Software environment

In this section we describe the various components of the software environment. This is the part of the computing environment that most directly impacts the user and includes such things as the choice of programming languages and the framework used for analysis.

### 13.4.1 Software Methodology and Languages

The computing problem faced by a large experiment in high-energy physics involves the manipulation of large volumes of complex, structured data which describe the observed physical interactions, components of the detector, and the intellectual framework used

to interpret the data. Low-level data structures are combined to form lists which are utilized by algorithms to generate structures at a higher level of abstraction. A typical example of this process is the assembling of hit data from a tracking chamber into found track segments, which are then assembled into track helix parameters.

There is a natural relationship of this process to the concepts of the object-oriented programming methodology. Indeed, the current CDF software may be described as an object-oriented data design operating within a Fortran framework. To accomplish this goal, very significant extensions of the native Fortran environment have been provided in the past, using customized products with a very large overhead in maintenance and low portability.

Mapping of software objects into code may be naturally accomplished by an object-oriented programming language such as C++, in which object description is provided at a low, native level in the compiler. In addition, other potentially useful features of the OO model, such as inheritance and overloading, are provided in such a language.

The use of the C++ programming language will be a feature of the software development environment for the CDF Run II upgrade. We expect to devote a large effort to providing the necessary design tools and support required by this initiative. The choice of C++ is mandated by its wide use in scientific and technical fields, including most new projects in high energy physics (BaBar, CLEO III, and LHC experiments).

Because of the extensive existing software base of the CDF experiment, we expect to need to continue support for the Fortran 77 programming language. The exact level at which the two languages must interact is still under discussion. One possible model is to allow the use of relatively large scale software units ("modules") in either Fortran or C++, but to require language consistency within a single module.

### 13.4.2 Code Management and Distribution

For a software project of the scope of the CDF upgrade, it is necessary to provide an adequate structure to manage and track the development of the system. Developers must be able to easily access the code they are working on and transparently benefit from the work of others. In some instances, a reservation mechanism can prevent conflicting changes in

programs; at the least a mechanism for detecting such changes is essential. It is also desirable to provide access control to protect the software and avoid unauthorized modifications.

The primary code repository for the CDF Run II upgrade software will be the central UNIX computing facility located at Fermilab. CDF is currently carrying on active discussions on implementation of this repository in the context of the Configuration Management Working Group, a joint working group of CDF, D0 and the Fermilab Computing Division. It is very likely that the recommendations of this group will focus on the Concurrent Versioning System (CVS) utility, which is widespread on UNIX systems and in the high energy physics community. A system based on CVS allows any part of a directory tree to be accessed, modified, and tracked as a unit, from individual files to the entire repository.

We expect to provide a mechanism for accessing code in the CVS repository using a client-server model. This technique, for which there is a working large system to serve as a prototype (the Sloan Digital Sky Survey), will allow developers to work on local UNIX systems with the code they require, then merge it back into the primary code repository. It also provides the possibility of integration of user-defined procedures for such functions as detailed access control and required testing at check-in time. CVS permits concurrent changes by multiple developers and blocks assembly of conflicting changes. If reservations are deemed necessary, a user-defined procedure can be used to provide this feature.

The software package must be distributed to collaborating institutions and other supported computing installations (e.g. the distributed desktop systems used by CDF physicists). We expect to use the functionality provided by the code management system described above to allow on-demand updates of the code while it is under development (this is an ongoing process). Tagged releases of packages will be distributed via batch updates in compressed format, as we have successfully used in Collider Run I.

### 13.4.3 Reconstruction and Analysis Framework

A framework for experimental application programs should provide adequate flexibility to address the diverse computing needs of the CDF collaboration. These include Monte Carlo Simulation, bulk produc-

tion of analysed datasets, user analysis, and the Level 3 trigger system of software filtering. The framework should support both interactive and batch operation, and transparently address all supported media.

CDF has a framework which has been used with great success in Run I and previously. It enables the user to define a set of software modules which are then incorporated at run time into sets which correspond to particular analysis pathways. Standard modules provide, for example, data i/o and reporting of data contents. The intrinsic modularity of the structure encourages a great deal of software reuse. Application building tools have been provided to assist the user.

The current CDF framework is heavily dependent on custom utilities to handle internal communications and generate the user interface. We plan to reduce reliance on these products as they become increasingly unmaintainable. This will require modifications to the framework software to maintain compatibility. The design and implementation of these changes, including possible new products, will benefit heavily from our extensive experience in this area. Other experiments are also active in this area, and we will examine the possibility of joint solutions.

The user interface for the Run II framework will take advantage of developments in third-party graphical user interfaces (GUI). One possible candidate for such an interface is the TCL/TK scripting language and toolset. The BaBar experiment has a prototype framework based on this product, which we are testing. Other public domain and commercial products will also be evaluated where appropriate.

## 13.5 Event Reconstruction

In Run II we will have many new detector subsystems for which we must write new event reconstruction code. As stated above, we expect use of the C++ programming language to feature in the Run II reconstruction package. A number of detector groups in CDF (SVXII, IFT) are experimenting with the use of C++ in their reconstruction code. The results look promising.

The event reconstruction package is constructed of a number of independent “modules” that each perform a specific task. Each module communicates its data to another by use of a well defined interface (presently this interface is YBOS data banks). Modules execute sequentially and in a given order, i.e., the

## 13.6 Calibration Database

In order to maintain an adequate record of detector performance, detailed information on calibrations, detector status, and beam conditions is recorded during checkout and running. Since a great deal of calibration is performed locally, the calibration information can be divided into data required by offline processing (which must be served to the processing clients) and strictly local information, which is maintained at the experiment for diagnostic purposes. In addition, the information may be generated either by hardware calibrations in advance of datataking, or by monitoring software during datataking. (Examples of the latter are dead channel counts and information on beam conditions.)

In previous runs, calibration data has been maintained everywhere in a custom database using YBOS as its primary keying mechanism and data retention mechanism. A run-number based index provided efficient access to the data. Changes in YBOS will have implications for calibration data.

Design decisions for the calibration database will be based on an analysis of the amount of calibration information to be generated by the experiment and where it will be needed during reconstruction. The results of this analysis will likely constrain our choices of implementation. An example is the use of commercial databases, which may only be useful in limited locations due to licensing costs.

In light of past experience, and considering the large amount of new hardware being commissioned for Run II, we expect to need to be able to modify the detailed implementation of our calibration storage. A well-designed interface to the calibration data will make this evolution possible. We plan to test such an interface during FNAL testbeam running, even if the final low-level implementation is still changing.

## 13.7 Simulation

The CDF simulation will require major modifications before the next Tevatron run. These modifications are necessary both because the detector itself is changing (new tracking detectors, new calorimetry in region  $\eta > 1.0$  and a change in the  $z$  position of the forward muon toroid) and because the CDF software environment is changing. In particular, if CDF decides to move towards the use of the C++ programming language, the simulation will move in

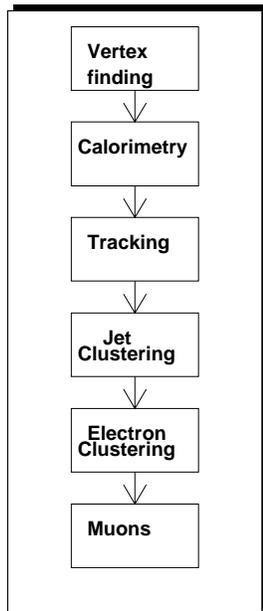


Figure 13.1: The logical flow of CDF event reconstruction in Run I

tracking module must run before the electron finding module because electron identification requires tracks. This will be true even in an object-oriented model. Parameters can be set at run-time that govern the behaviour of a particular module. In Fig. 13.1 we show the logical flow of the existing CDF event reconstruction.

As described previously, the event reconstruction is carried out on farms of UNIX workstations (worker nodes) connected to a central I/O server. Each event is read from tape or disk and sent to a worker node for processing. When processing is complete the event is sent back to the I/O server for output. The CPS (Cooperative Process Software) product developed by the Fermilab Computing Division has been used to carry out the event distribution. We anticipate continuing to use a similar product for Run II.

that direction as well.

For Run I, CDF has two independent simulation packages, a fast simulation (QFL) that produces as its output high-level reconstruction banks and a slow simulation (CDFSIM) that produces raw data banks. Both simulations rely completely on CDF-specific code (i.e. neither one uses the GEANT simulation package). Both run within the CDF analysis framework and use the CDF persistency mechanism (YBOS).

The purpose of the CDF fast simulation (QFL) is to provide an appropriate tool for generating large statistics Monte Carlo samples to determine the overall acceptance for various physics analyses and to study the dependence of this acceptance on systematic variations in either the detector response or the physics model. The philosophy of the package has been to make the highest level data structures possible rather than simulate raw data and to use parameterized responses rather than a first principle approach. Significant effort has been put into parameterizing the response of the calorimeter (including calibration, non-linearities, cracks and variations in gain across the face of the detector), the muon system (including non-uniform magnetic field effects and multiple scattering), the pre-radiator and the strip detectors at shower max. Because the performance of the silicon vertex detector plays such an important role in the top analysis, this detector has received special attention. In this case, raw hits are simulated (including the effects of multiple scattering and Landau fluctuations) and the CDF pattern recognition and fitting code is used to associate these hits with tracks. In the case of the central tracking chamber, however, the simulation does not produce raw hits. Instead the tracking resolution is parameterized by generating a 5x5 covariance matrix for each track and then using this covariance matrix to create the smeared track parameters. Thus, the fast simulation does NOT allow detailed studies of pattern recognition efficiencies. Instead, pattern recognition studies are performed using the slow Monte Carlo (CDFSIM). An alternative technique which is commonly used in CDF is to measure track reconstruction efficiencies is to imbed single Monte Carlo tracks in real data events.

The current version of QFL can simulate a  $t\bar{t}$  event in approximately 0.7 MIP-sec. By comparison, full reconstruction of such events from raw data takes about 700 Mip-sec. The speed of the package re-

sults from the fact that the number of volumes in the simulation is small ( $\sim 30$ ) and the step size is large, from the fact that hadronic and electromagnetic showers are parameterized and from the fact that pattern recognition in the central tracker is not performed. In QFL, the step size is set by the path length to the volume boundary unless a discrete process (bremsstrahlung, pair production, decay-in-flight) occurs within the volume, in which case the particle is stepped to the point where that process occurs. The trajectory is corrected for continuous processes (energy loss and multiple scattering) at the exit point of the volume. At each step, the track trajectory (momentum and position) is stored in a data structure. Genealogical information relating generated tracks and vertices to the simulated tracks is stored, including information necessary to trace a generated particle's fate through discrete processes such as decays. Structures that store the genealogical relationship between generated tracks and hits in the detector are available as an option. All such structures can be output as part of the standard CDF event record.

Our goal for Run II is to maintain the speed and functionality of the current fast simulation, while improving its flexibility. In particular, we plan to incorporate current CDFSIM functionality (the creation of raw hits in the tracking data) as an option within the fast simulation framework.

We are currently exploring options for restructuring the simulation. One possibility would be to use a GEANT4 framework. In this case, GEANT4 routines would be used to trace particles through the detector. To maintain the speed of the simulation, we would continue to use CDF specific parameterizations of calorimeter showers and we would keep the geometry description extremely simple. A second option for Run II would be to use as a framework the MCFAST simulation package currently being developed by the Fermilab Computing division. The third option would be to continue the current practice of using CDF-specific code. Prototyping efforts on all 3 approaches are currently underway. A final decision on the framework will be made in Spring 1997, several months after the first beta-test release of GEANT4.

# Bibliography

- [1] YBOS is an extension of the BOS package from DESY. It is written in FORTRAN and combines the functions of data persistency and memory management.
  
- [2] An ntuple is like a table where each event is a row of the table and each variable in the event is a column. It is part of the CERN HBOOK package and is widely used in HEP experiments. HBOOK, CERN Program Library Long Writeup Y250, Version 4.2, Application Software Group, Computing and Networks Division, CERN, 1993.
  
- [3] ZEBRA is a CERN package that performs the same functions as YBOS. It is written in Fortran and combines the functions of data persistency and memory management. ZEBRA, CERN Program Library Long Writeups Q100/Q101, Application Software Group, Computing and Networks Division, CERN, 1993.
  
- [4] RD45 is a CERN project to study persistent object managers for HEP. Their latest status report can be found at <http://wwwcn.cern.ch/pl/cernlib/rd45>.