CDF/DOC/COMP_UPG/PUBLIC/4497
version 1.03 February 26, 1998

ROOT-based data handling system for CDF in Run II.

1. Specifications, Data Flow Analysis, Initial Implementation Plan

R.Brun, P.Murat, F.Rademakers

Abstract

This note describes a **proposal** for CDF Run II data handling and analysis system based on data handling technology developed by ROOT project at CERN [1] . We present specifications for the ROOT-based model and describe its principal components, interfaces and performance requirements. We discuss 2 possible data-taking scenarios: high-luminosity scenario and low-luminosity scenario. In case of high-luminosity scenario (instantaneous luminosity of $2 \cdot 10^{32} cm^{-2} sec^{-1}$) using the L3 cross-section of about 350 nb we estimate the total volume of **compressed with ROOT** Run II CDF data to be of the order of 400 TBytes whihc makes it possible to keep all the data on robotic storage. Assuming the total size of central disk repository of 20 TBytes we propose to keep the mostly often used part of the event data (about 16-18KBytes of compressed data) on disk. This amount of data is equivalent to Run I PAD format and should satisfy significant part of Run II physics analyses.

In case of low-luminosity scenario (instantaneous luminosity of $1 \cdot 10^{32} cm^{-2} sec^{-1}$) CDF could use higher L3 cross-section and collect 2 times more data. We discuss the changes in the organization of the system which have to be done to accomodate it to different operational conditions.

We compare ROOT-based system to other data handling approaches proposed for CDF in Run II. The ROOT-based system could operate in a sequential mode with ROOT files replacing (TR)YBOS files. We note that many features discussed at the conceptual level in the proposal of light-weight database are already implemented by ROOT project. Therefore ROOT-based system could be considered as one of the possible implementations of light-weight database. We present results on benchmarking the ROOT I/O mechanism versus other I/O packages: ZEBRA, YBOS and Objectivity/DB and show that ROOT I/O provides better performance and higher data compression ratios, the advantages, which become especially important in case of huge data volumes maintained in the system.

In case of high-luminosity scenario the evaluated overall cost of the central CDF ROOT-based data handling system for Run II is about \$4.7 Million, which includes hardware and media costs.

Finally, the note includes addendums, containing 2 supporting documents which describe **data flow analysis in the system** and the initial steps of the **implementation plan**.

1 Introduction and Motivation for the ROOT-Based Solution

Huge data volumes expected in Run II make a choice of data handling technique one of a critical choices for CDF collaboration in Run II. Changing the data handling model from the sequential one to the database-based one in the middle of Run II (as proposed by the recommendations for CDF Run II data handling [3]) could have an unpredictable impact on the performance of CDF Run II software and analysis system. For this reason we believe that it is vital for CDF to start Run II with the same data handling model we want to use at the end of the run. The proposed model should allow an evolutionary transition from the sequential technique of data access which we are using right now to the technique based on the direct accessing the data, which as it is believed could provide efficient hadling of data volumes expected in Run II.

The data handling model proposed below is based on the technology developed by ROOT project at CERN [1] - technology, which on one hand is available today and therefore could be extensively tested and debugged at CDF before the start of Run II, and on the other hand seems to be capable of handling the data volumes of the Pentabyte scale.

To our opinion ROOT data handling technology is the most advanced one in technical respect among the technologies available today. ROOT supports both - sequential and direct - data access modes and allows to make a transition from one mode to another in a way transparent for users of the system.

This proposal is essentially based on 2 unique features of ROOT:

- parallel multistream I/O, which allows to read/write the event data from/to several streams
- built-in data compression, which is being done for each stream separately and, because of this, provides very high (of the order of 2 for the reconstructed data) compression ratioes

ROOT technology has already proved its reliability: a ROOT-based system is used for data handling in NA49 experiment which has successfully collected and processed about 70 TBytes of data and plans to take more data.

Considering the data handling choices made by the experiments planned on the future colliders we'd like to mention the choice made by PHOBOS collaboration at RHIC. After performing a series of benchmark tests on comparison of ROOT vs another potentially powerful technology - Objectivity/DB (also considered at CDF, see [4] for more details) PHOBOS collaboration has chosen ROOT technology as their **primary** option for the data handling. Some of the benchmark results obtained by PHOBOS collaboration are presented in this proposal.

We also note that the many ideas of this proposal are very close to ideas behind the proposal of light-weight database, discussed in [13], with ROOT technology providing a concrete way of the implementation of these ideas.

2 Overview of the System

3 Architecture of the Central Hierarchical Storage System and its Major Hardware and Software Components

The schematic architecture of the proposed CDF data storage system with the components (they are discussed in detail below) is presented in fig 1.

The central storage system (located at FNAL FCC) consists of robotic tape storage system with the required capacity of about 300-400 TBytes and of primary disk repository of about 30 TBytes. About 20 TBytes of disk space are used to keep permanent datasets of physics groups, other 10 TBytes are used as a temporary cache for the data permanently residing on tapes. Large size of the cache will provide large "disk lifetime" for the data permanently residing on tape and therefore will greatly reduce required tape-to-disk bandwidth and the number of tape mounts.

Below in this note we will use term SILO for the Run II robotic tape storage system (which not necessarily should be STK SILO robotic storage).

One could easily see that this architecture is very close to the architecture of Run I storage system. Different physics groups and universities may have their local data servers, discussion of their architecture, however, goes beyond the scope of this proposal.

3.1 Organization of the Event Data

The following categories of the data are used in the system:

- logical unit of CDF data is a data bank;
- RAW data banks: banks containing the raw data read by DAQ;
- trigger banks: banks containing trigger information
- reconstruction banks: banks produced by the reconstruction code. In Run I CDF used 2 major reconstruction formats: DST and PAD. Overall size of the DST reconstruction banks was about 60 KBytes per event, the size of PAD banks - of the order of 30-40 KBytes (depending on the instantaneous luminosity). Correspondingly we could talk about 2 types of reconstruction banks: PAD and non-PAD ones.

3.2 Estimates of the Data Volumes to be Handled

Whereas in terms of trigger performance change in trigger rates a factor of 1.5 may not require radical reorganization of the trigger architecture, translated into the expected data volumes this change may cost hundreds of thousands if not millions dollars. Therefore to choose the right data handling strategy it is very important to get as accurate expectations for the data volumes to be handled as possible.

The expected integrated luminosity in Run II is 2 fb^{-1}. Given the CDFII Level3 cross-section of 350nb [5] we expect to collect about $7 \cdot 10^8$ events. However in Run I CDF has been successfully using dynamic prescales and presumably we will be using them in Run II. Dynamic prescaling allows to keep the data logging rate (and not the Level3 cross section) at a constant level. Assumption that CDF requests a new beam injection when the instantaneous luminosity drops down by a factor of 3 brings up an additional scale factor of about 1.6, and under this assumption the expected number of events becomes of about $1 \cdot 10^9$. Given an average size of the RAW event data to be 250 KBytes we could expect the total size of the RAW data to be about 250 TBytes. In Run I the size of the reconstruction banks on DST was of the order of 50% of the RAW data, this leads to the estimate of the overall volume of the reconstruction data of the order of 125 TBytes.

In Run I we've been using multiple production streams with the overlap of about 30%. between them. We propose to continue using multiple output streams in Run II production by the cost of having overlap between them. This will significantly simplify handling the data, in particular reprocessing of the data. However we need to study overlaps between the streams using the Run I data and try to minimize the overlaps in Run II.

Assuming the same overlap and accounting for about 10% of the data to be processed in express stream we come to the overall volume of RAW data which has to be stored of about 350 TBytes, the corresponding number for the reconstructed data being about 175 TBytes. The size of the PAD banks is of the order of 50% of the reconstruction banks, i.e. of about 90 TBytes.

The uncertainties in these numbers come primarily from the uncertainty in the expected trigger cross-section and the uncertainty in the expected average event size.

To the moment we assume that there will be no backup copy of RAW data - this is a decision to be made by the collaboration, but in any case this copy is supposed to be shelf-resident and doesn't affect any estimates of automatic tape/disk storage required for Run II.

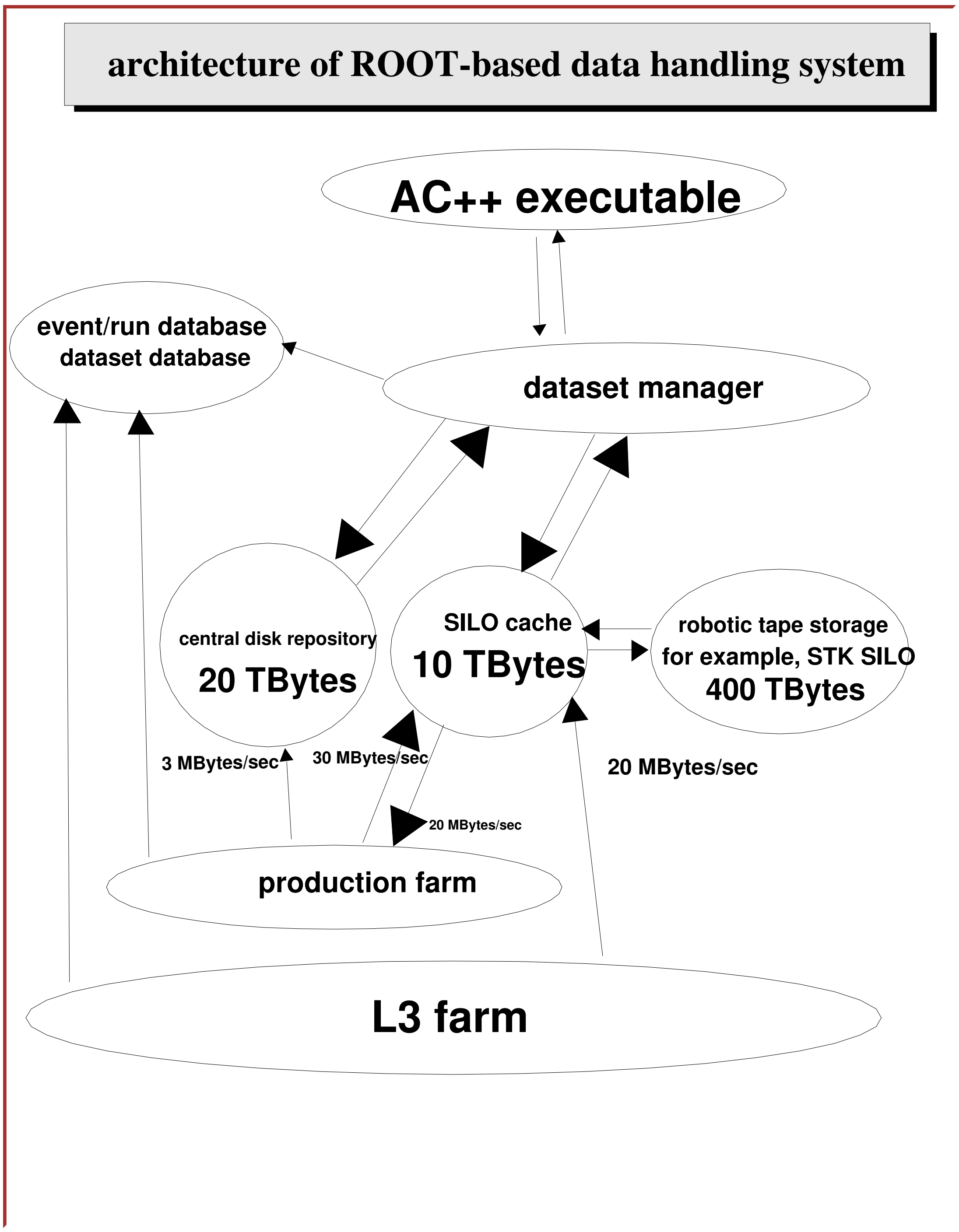


Figure 1: Layout of the ROOT-based data handling system for CDF in Run II

The estimates presented above are about 2 times lower than the corresponding estimates done in [9]. It is therefore important to understand the sources of this difference. The estimates done in [9] are based on the following assumptions:

- data logging rate is constant and it is 75 Hz independent on luminosity;
- data taking period lasts 2 years (about $2 \cdot 3 \cdot 10^7$ sec)
- overall Tevatron and CDF on-time efficiency is 50%

If we assume that the luminosity integrated over this period is $2fb^{-1}$, the assumptions above lead to the typical peak luminosity (instantaneous luminosity immediately after the beam injection) being

$$L_{peak} \simeq \frac{2fb^{-1} \cdot 1.6}{2 \cdot 3 \cdot 10^7 \cdot 0.5} \simeq 1 \cdot 10^{32} cm^{-2} sec^{-1}$$

all over the Run II (as in the previous section we are using a scale factor of 1.6 to account for dynamic prescales). This is 2 times lower than the Tevatron Run II design luminosity of $2 \cdot 10^{32} cm^{-2} sec^{-1}$. Under this assumption it possible to increase the L3 trigger cross section so it becomes about 750 nb. This is about 2 times higher than the L3 cross-section planned for CDF Run II trigger system, estimated in [14] at about 350 nb. Our estimates are based on hte assumption that the data taking starts from the peak instantaneous luminosity of $2 \cdot 10^{32} cm^{-2} sec^{-1}$ and at about $6 \cdot 10^{31} cm^{-2} sec^{-1}$ CDF a new beam injection is requested.

As one could see, the difference in estimated data volumes comes mostly from different assumptions made about the Run II instantaneous luminosity. We, therefore, could consider 2 possible data-taking scenarios corresponding to the integrated Run II luminosity of 2 fm^{-1}:

- hign-luminosity scenario, in which case CDF collects $1 \cdot 10^9$ events over the course of Run II (for about 1 year of data taking);
- low-luminosity scenario in which case CDF uses higher L3 cross-section and collects $2 \cdot 10^9$ events (for about 2 years of data taking);

We present detailed description of the system and its optimal configuration for high-luminosity scenario, then we discuss the changes necessary to optimize the performance in case of low-luminosity scenario.

4 Organization of the Data

4.1 ROOT Data File Format

Output of production is written out as a ROOT tree. We propose to write the data out in split mode, with different CDF data units - banks - being written into separate branches. For the detailed discussion of ROOT file format we refer reader to the primary source of information [1] (see also [2]). Here we just briefly recall the most important features of the file model supported by ROOT.

- ROOT allows to split the event data into parts - branches - and to write branches out via different I/O buffers. Thus ROOT introduces a principally new in HEP concept of parallel multibuffer I/O.
- branches could be written out to one or to several different files, which is completely transparent for user;
- ROOT supports built-in data compression which is being done separately on different branches. For the RAW data the compression ratioes observed are typical for GZIP-type algorithms (1.2-1.3). However for the reconstructed data (which are more "regular") the corresponding compression factors are much higher with the typical value of about 2. This unique feature will play an important role in our considerations;

- As buffer gets filled its contents goes onto disk. Each block of information written via this I/O buffer contains the location of tne next block, belonging to the same branch. Thus ROOT file is a direct assess file where the contents of each branch could be read back independently without the reading the total file.
- ROOT doesn't have typical for commercial object-oriented databases clustering problem;
- ROOT provides an open mode support of schema evolution i.e. backward compatibility (it is always possible to read the old data with the new code). Under certain assumptions ROOT could also provide a forward compatibility (a possibility to read **new** data with the **old** code.

We propose to combine branches into several groups (superbranches). The division presented below may not be the ultimate one, however it illustrates the general idea. One could think, for example, about having the following superbranches:

- RAW data superbranch;
- PAD superbranch;
- DST-PAD superbranch (containing the rest, non-PAD, part of the reconstructed data);

Another alternative is to **drop DST-PAD superbranch**, to keep on tape **only 2 first superbranches** and to be able to reconstruct DST-PAD banks at run-time if necessary.

Different branches are written onto different tapes. Only one copy of the reconstructed data (for each stream) is kept on tape. PAD data is a direct subset of the reconstructed data (unlike Run I PAD's) and do not duplicate DST data.

It is too early to specify which data banks will be written into which branch and how we are going to split them. However it is important for each branch (superbranch) written into separate file to contain event/run numbers, which in this case are duplicated.

ROOT support of splitting the event data into branches and splitting the branches into several files (tapes) is a foundation for the proposed data handling model. We propose to divide the event information into disk-resident and tape-resident branches. Disk-resident branches contain relatively small part of the most oftenly used event data (one could think about equivalent of Run I PAD banks). Each event processed by offline reconstruction has part of its data kept permanently in the central disk repository. As we will see below this part could be sufficient for the most physics analyses which do not require access to the RAW data. The information written into tape-resident branches is kept on robotic storage and is cached on disk by demand.

Switching from the mode in which different branches are written into different files to the mode where all the branches are written into the same file is transparent for users, who know only about the names of requested datasets.

4.2 Data stored in Central Disk Repository

Given the disk space of 20 TBytes available for disk resident datasets we propose to keep on disk (in central disk repository) about 16-18 KBytes of reconstructed event data, which would allow **for all the events** to keep part of their data on disk (disk-resident branches) and at the same time to have about 10% of total disk space available for express needs. We note that 16-18 KBytes of compressed reconstructed data correspond to about 30-35 KBytes of uncompressed data which is pretty close to the characteristic size of Run I PAD's and 2 times larger than the average size of Run I BPAD event. This amount of data is quite sufficient to keep all the mostly used in the analysis data banks on disk.

Based on Run I experience we expect that the data stored in disk-resident branches could satisfy significant part of physics analyses.

A logical unit of data in primary repository is a physics dataset, corresponding to a certain production stream or a certain trigger(s). It is intended to contain all the data collected by the experiment for this stream/trigger. Dataset may consist of many files.

4.3 Data on Robotic Tape Storage

We propose to utilize fully the unique data compression features of ROOT I/O. Given the compression factors presented in the previous section, the overal volume of compressed data to be stored becomes $\mathbf{350/1.2 + 175/2 = 380}$ TBytes.

We propose to keep all the data on robotic storage

As it is not quite clear how non-PAD reconstruction banks have been used in Run I physics analysis (if any), **we could also consider to drop non-PAD part of the reconstructed data and to keep on tape only RAW and PAD data**. In this scenario the non-PAD part of the reconstructed event data would be generated at run-time on user demand.

This would save us of about 45 TBytes, with the overal volume of RAW and PAD data being 335 TBytes.

We emphasize that this number is only about 10% higher then the maximum capacity of the present CDF STK SILO: assuming use of 50 GByte tape cartriges, the capacity if CDF STK SILO is limited by the number of cartridge slots available (60000) and is therefore 300 TBytes.

- So in principle CDF could handle significant part of Run II data without buying a new hardware for robotic tape storage. We only need to buy the new tape media.
- we also should consider more intelligent ways of data compression than "raw" gzipping: for example, SVX data contain 10K of ChipID/ChipStatus bytes, significant part of which could correspond to the chips which have no channels above the readout threshold.
- finally, if we want to save money on buying a new robotic tape storage, putting 10% of RAW data on shelf could also be a feasible option.

4.4 Advantages of the Proposed Data Storage Model

- using ROOT data compression algorithm allows to reduce overall volume of data to be stored by a factor of about 1.4;
- ROOT compression factor for the reconstructed data of about 2 allows to keep on disk equivalent of 30-35 KBytes of uncompressed event data. Based on Run I experience (PAD event size of about 30-40 KBytes, BPAD event size of about 16-18 KBytes) we believe that this amount of data will be sufficient for significant part of the analyses which do not require access to the RAW data. As the examples of such analyses we could take QCD inclusive jet and dijet analyses, exotics missing-Et analyses, inclusive and exclusive analyses involving J/psi's etc. It is quite possible that even one of the most complicated ones - the top cross section analysis - could be satisfied by disk-resident data.
- with the most analyses not using the tape I/O, ROOT-based data handling system allows to reduce significantly overall I/O bandwidth, involving tapes. We expect that tape I/O will be dominated by the data logging and production (about 30 MBytes/sec, which is well within the reach of presently available technology).
- for the reasons above ROOT-based data handling system may not have a problme with to/from tape I/O bandwidth, which could be a bottleneck, limiting data handling abilities for other approaches.

4.5 Event/Run Database

Event/run database allows to coordinate different parts of event for the processing. It contains a section per production output stream. Given event/run number within a stream the Event/Run Database allows to find location of all the disk- and tape-resident branches of this event data. Here is an example of database entry:

```
stream ''AAA''

    run 65100   event 1    -10000     ''65100A''
    run 65100   event 10001-20000     ''65100B''
```

which says that fist 10K events of run 65100 are contained in the file AAA65100A etc (this is just an example of how one could build a file name out of stream name and run number).

Different branches add different extentions, for example, RAW branch (which in its turn may consists of several branches corresponding to different banks) is located in file 65100RAW, PAD branch - in file 65100PAD etc.

It is a ROOT file model which make the structure of the Event/Run database so simple. In a sense ROOT provides essential part of this DB functionality: most oftenly used data branches are disk-resident with event data being accessible in direct access mode.

4.6 File/Dataset Database

File/Dataset database allows to find a tape, corresponding to a given file or to identify all the data files/tapes corresponding to a given dataset.

4.7 Monitoring the Access to the Primary Repository

Efficient management of hierarchical storage system is greatly simplified if the access to the datasets in it is being monitored. For example it may appear that some of the datasets created in the primary repository are not used at all or used extremely rarely whereas for other, intensively used datasets, one would like more than 20 KBytes/event to be disk-resident.

To optimise usage of disk space we propose to monitor access to the datasets stored in the central repository.

Given below is the example of one of the possible implementations.

- all the files belonging to datasets in the central repository are created with group ID which doesn't allow users to read them directly;
- datasets are accessed via a special program - dataset manager. When called from the analysis job dataset manager updates statistics on this dataset usage and dynamically changes group ID of the job after which it can read the datasets stored in the central repository.
- procedure described above allows to monitor access to the central repository. At any point in time statistics on dataset access is available. Based on this statistics the disk space management in the central repository could be greatly facilitated.

4.8 Accessing the Tape-Resident Data

Tape-resident branches are not supposed to be accessed directly, if necessary the data from them are cached onto disk. Disk cache of about 10 TBytes provides the "disk lifetime" of tape-resident data large enough to avoid multiple reading the same data from the tape within the single processing path.

4.9 Network-Distributed Disk Repository: Bandwidth Optimization

Assuming the size of the central disk repository of about 20 (+10) TBytes it is more or less clear that the central disk repository will have a network-distributed architecture. In this case optimization of the network traffic becomes an issue to be dealt with. Consider, for example, an analysis job which has to process the whole inclusive high-pt electron dataset, which are distributed over the several filesystems belonging to different nodes. One could imagine 2 principally different modes of data processing:

- an analysis job is running on a single CPU and it is reading all the necessary data over the network. This approach is a straightforward one, and as we know from Run I practice given the present network condactivity several executables running in this mode could easily paralize the local network;
- alternatively, one could think of "cloning" the executables and distributing them over the network to the nodes where the data are stored. Then each executable runs on its own CPU and reads the data only from the disks mounted to the same bus with the CPU. This mode uses the network much more efficiently, however it is nesessary to coordinate output of many executables.

The second approach is already implemented in PROOF (Parallel ROOT Facility) and undergoes final testing in NA49 experiment at CERN.

5 User, System and Programmer Interfaces

5.1 User Interface and Analysis Model

One of the biggest advantages of ROOT consists in that it provides a wide range of analysis tools integrated with each other on the basis of ROOT data model.

Note that exactly this integration of analysis tools based on ZEBRA data model predefined the great success of CERNLIB analysis tools in 80's-90's.

Each ROOT file could be used for analysis in exactly the same way as PAW ntuple (which is not a surprize - both packages are written by the same author). In other words, ROOT file is an ntuple by definition. ROOT supports 2 major - interactive and batch - analysis modes.

- interactive mode is based on the use of C++ interpreter CINT [6]. From the CINT prompt user does exactly the same things which could be done from PAW interactive prompt (handled by KUIP):
  - issues interactive analysis commands
  - executes interactively C++ scripts, interpreted by CINT

  The major difference between ROOT and PAW here consists in that the input language of CINT interpreter is C++, whereas in case of KUIP it was a specialized command language mixed with the FORTRAN-like scripting language (supported by COMIS interpreter).

  Thus ROOT is using only one input language (C++) instead of 2 (KUIP command language and pseudo-FORTRAN) used by PAW. ROOT does not have a specialized command language - this was a concious design decision

  In general interpreted code runs slower than the compiled one and this could have an impact on physics analyses. CINT interpreter provides a highly optimized byte code, so in many cases using scripts interpreted by CINT is as almost efficient as using the compiled code. CINT also supports some simple debugging features.

- ROOT has a GUI based used interface, it also provides a simple in use yet flexible GUI builder. Usefullness of ROOT GUI builder has already been demonstrated in the prototype of ROOT-based event display for CDF.

- another extremely powerful interactive tool is ROOT **Object Browser** , which allows to browse objects in memory at run-time;

- in batch mode user links his executable against ROOT libraries and uses ROOT tools to fill histograms and trees in exactly the same way as HBOOK tools are used to fill histograms and ntuples.

- A migration path from PAW to ROOT is available. HBOOK-to-ROOT converter **h2root** transforms a HBOOK RZ file into a ROOT tree so ROOT could be immediately used to analyse the existing data. The size of ROOT file produced by **h2root** is typically several (2 and more) times smaller than the size of the corresponding HBOOK ntuple. Therefore switching from PAW/HBOOK to ROOT has a very nice side effect: one could free a significant amount of disk space.

A unique feature of ROOT as an analysis framework consists in that from the CINT prompt user has a completely interactive access to the data stored in ROOT file. To make it possible ROOT generates a special dictionary (a C++ file) for each class of interest. This dictionary is generated automatically by parsing the include files with class definitions, dictionary tells the interpreter how to call methods of this class interactively. A recommended way is to compile the generated dictionaries into a shared library and to load it dynamically prior to reading data. Examples of how to generate and to use ROOTCINT dictionaries could be found on ROOT web cite [1] and also on CDFSGA (see examples in **/cdf/upgrade/root/v1_03_09/tests** area)

5.2 Programmer Interface

As ROOT is written in C++, a programmer interface to its data handling facilities is provided by a set of C++ methods (API) defined in ROOT I/O classes. HTML-based description on all the ROOT classes is available from the web [8]. We also want to emphasize that ROOT is a free software, and all its libraries, complete set of source codes (including I/O classes), tutorials and examples are available on CDFSGA from **/cdf/upgrade/root** area.

5.3 System Interface

It is also important to understand how ROOT software could be used within the CDF software environment and which interfaces (if any) are required for this. As different C++ compilers do name mangling in different ways we need ROOT libraries to be compiled with the same compiler as the rest of CDF C++ code.

- as of today we maintain on CDFSGA ROOT libraries compiled with GCC-2.7.2 and the use of ROOT within the CDF software environment (both - Run I and Run II) has been successfully prototyped.

- In very near future CDF is planning to start using KAI C++ compiler (KCC). Work on porting ROOT to IRIX6-KCC has already started.

- as ROOT dosn't rely on any other external package, it could be easily ported on any other architecture CDF may decide to support in future.

The interfaces required to read/write ROOT tree files from within the CDF analysis framework (AC++) are the specialized input and output modules which would read and write ROOT trees and convert them into CDF transient objects. It is not excluded that we'll find ROOT container classes to be well suited for transient objects, in this case interfacing to the AC++ framework would become even more simple. Each persistent class should have corresponding methods which convert transient (riun-time) representation of the object into its persistent representation and vice versa. These routines (so-called **Streamer** methods) also provide an open-mode support of schema evolution. By default **Streamer** methods are generated automatically by ROOTCINT. ROOT also supports user-written **Streamer** functions, which allows optimisation of their performance.

In conclusion, we expect ROOT libraries to be available on all the hardware configurations supported by CDF (including Intel/Linux) by the spring of 1998.

6 Scalability issues

There are 2 major questions related to the scalability of any data handling system:

- What is the maximal size of the data the system can handle in principle and what are the limitations on the data organization?
- How efficient is the access to the data in case of large data volumes are maintained within it?

6.1 Maximal size of the data and the limitations on the data organization

ROOT version 1.03/09 supports 32-bit filesystems(pointers), thus limiting the maximal size of a single file by about 2 GBytes. This limitation however doesn't affect the total number of files stored in the system which could be arbitrary large.

To allow large data volumes to be processed at the analysis stage ROOT provides a mechanism of so-called chains. Chain is an ordered sequence of files, ROOT reads files defined in the chain one after another. File chains of about 70 GBytes are in routine use in NA49 physics analysis.

Keeping also in mind that the next version of ROOT (expected in spring'98) will support 64-bit pointers, we conclude that there is no practical limitation on the overall volume of the data handled by ROOT.

6.2 Data Access Efficiency

Tree structure of ROOT file allows user job to read in only those branches which are necessary for the analysis thus minimizing the required I/O bandwidth. To understand better how much one could gain from splitting the event data into branches, we compared ROOT to YBOS in "dataset scan" mode, which is typical for many exotics analyses. In "dataset scan" mode large data samples are being scanned in search for very rare events. The absolute majority of events could be rejected by very simple cutes, for example, but cutting on event missing E_t. To make this selection one needs to read only a minor part of event data, in this particular example - either METS or QMET bank (just to remind - QMET bank contains only 6 data words). In the test described in more detail in [2] ROOT scanned the BPAD dataset about 15 times faster than YBOS. Based on this result we conclude that in "dataset scan" mode the data access time could be easily reduced by an order of magnitude and more as compared to the conventional approach.

It is possible to split the event data into as many branches as possible. For example, **h2root** when converting column-wise HBOOK ntuple into a ROOT tree writes each column into a separate branch, in which case each branch contains just one number per event.

ROOT I/O model is extremely flexible and if necessary it could support any degree of granularity in organizing the event data into branches. However we believe that the natural level of granularity of CDF event data is given by a single bank. We expect that splitting the event data into branches corresponding to different banks would provide very close to optimal algorithm of accessing the event data.

7 Data Replication and Distribution

We assume that the most part of high-statistics data analyses will be done using the computing resources located at Fermilab and do not consider a problem of distributing primary datasets to the local computing centers of CDF institutions. However the relatively small (secondary) datasets could be easily distributed and copied (via FTP, for example) to the remote institutions where they could immediately be used for the analysis. The distribution scheme is exactly the same as the one used in Run I. No installation of any special database-like support tools is required in the remote institutions.

8 Handling the Monte Carlo data

Calculation of detector efficiencies for numerous physics processes to be studied in Run II will require generation of huge volumes of MC data. We propose to keep all the centrally maintained MC datasets on robotic storage so they would be cached on disk by demand. If we assume that the overall volume of MC data to be kept on robotic storage is about the same as the overall volume of the reconstructed PAD data, then MC data would fit into the proposed 400 TBytes of robotic storage. Therefore no additional hardware aquisition is required to store and maintain MC data in central repository.

9 Preliminary Cost Estimates

In this section we'll try to get cost estimates for the proposed ROOT-based data handling system for CDF in Run II. 400 TBytes of robotic storage cost (assuming the 50 GBytes cartiges are used and the cost of library slot to be $300, i.e. $6/GByte) amounts to $2.4 Million. As all the data are supposed to be kept on robotic storage, no additional spendings on the tape media are required. 30 TBytes of disk space under the assumption of $60/GByte add another $1.8 Million. Adding (as it has been done in [9]) $0.5 Million for server system we end up with $4.7 Million total required for hardware and media. This number accounts for both data and centrally maintained Monte Carlo datasets.

This estimate ignores that CDF already has one available STK SILO with the maximal capacity of 200 TBytes and this robotic storage has already been paid for.

10 Benchmarking ROOT-based Solution Versus Other Available Solutions

10.1 Benchmarks done by ROOT team

There are several benchmark results available which compare I/O performance of ROOT to that of other I/O packages. Significant part of the benchmarks has been done at CERN by the ROOT team which compared ROOT to ZEBRA [10] and to Objectivity/DB [11]. Here is a brief summary of the results.

- ROOT vs ZEBRA:
  - ROOT writes direct access output file 1.89 times faster than ZEBRA;
  - ROOT reads the data back from direct access file 1.24 times faster than ZEBRA;
  - in mode when ROOT is using split option and compresses only branches with integers (split=1) it still writes the data by about 10% faster than ZEBRA, however the size of ROOT file is 1.84 times smaller than the size of the corresponding ZEBRA file;
- ROOT vs Objectivity/DB

- on the database traversal test ROOT performed more than 6.5 times faster than Objectivity/DB;
- the size of output file written by ROOT was about 3 times smaller than the size of equivalent file, written by Objectivity/DB with the overhead introduced by Objectivity/DB being of the order of several percent.

10.2 Benchmarks done at CDF

There are also results on ROOT vs YBOS benchmarking done at CDF [2]. They could be summarized as follows:

- the size of BPAD file converted into ROOT format (with split=1, which means compression only on integer branches) is 1.94 times smaller than the size of original YBOS file;
- in "raw" mode, when the whole event is being read in, ROOT reads the data out about 3.8 times slower than YBOS (note that no optimization of ROOT I/O has been done and automatically generated **Streamer** functions have been used for all the classes);
- in "dataset scan" mode (with only one branch/event read by ROOT) ROOT reads the file about 15 times faster than YBOS (which independently on the access pattern has to read the whole file)

10.3 Benchmarks done by PHOBOS collaboration at RHIC

There is an independent set of benchmarks on ROOT vs Objectivity/DB done by PHOBOS collaboration at RHIC [12] done under the following conditions.

- a set of runs containing MC events has been simulated and written out with both ROOT and Objectivity/DB.
- within the run each event contained the same number of tracks, for different runs number of tracks was varied from a very small for RHIC number (< 100) up to 30000 tracks;
- output event contained 3 (either ROOT or OBjectivity/DB) data containers with the equal number of tracks/container
- the parameters compared were
  - time to write the output file
  - output file size
  - time to retrieve the information from the datafile
- no I/O optimizations have been done in either case

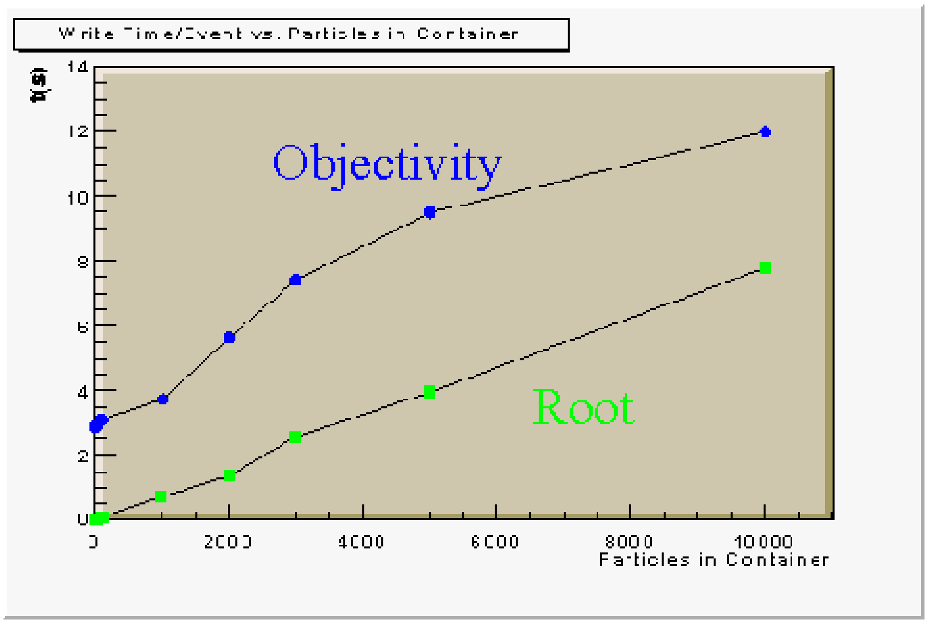
Figure 2 shows that given the amount of information is the same ROOT writes the data out 2-4 times faster then Objectivity/DB

The distribution in Figure 3 shows that under the tested conditions the size of ROOT file is 5 to 10 times smaller than the size of corresponding Objectivity/DB file.

11 Low-luminosity scenario

In this section we'd like to discuss a data-taking scenario corresponding to the integrated luminosity of 2 fb^{-1} taken at instanteneous luminosities below $1 \cdot 10^{32} cm^{-2} sec^{-1}$. In this case CDF could significantly increase the Level 3 cross-section and collect amount of data of about 1 Pentabyte. To be consistent with other proposals we consider a case when overall number of recorded events is $2 \cdot 10^9$. There are some questions, however, which one may want to answer before.

Write time per event

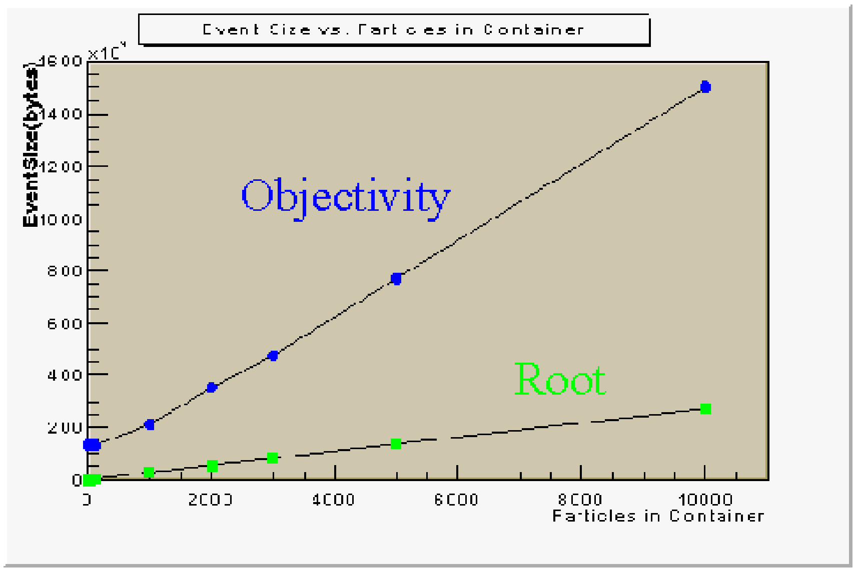


Root write time 2-4 times faster

Jan 13, 1998 Patrick Decowski, Krakow Collaboration Meeting 17

Figure 2:

Size of one Event



Root events are 5-10 times smaller than Objy.

Jan 13, 1998 Patrick Decowski, Krakow Collaboration Meeting 18

Figure 3:

- The high-luminosity scenario is based on the effective L3 cross-section of about 550 nb all over the Run II. If the integrated Run II luminosity is still $2fb^{-1}$ what are the triggers which make the L3 cross-section to be about 1000 nb? Remind that Run 1B "to-tape" cross-section, based on the number of events processed in stream B of about 45 millions, under the same assumption about the dynamic prescales, translates into effective L3 cross-section of about 450 nb.

- are there any alternative ways to reduce data logging rate without sacrificing with physics? Let's consider, for example, an inclusive low-Pt electron trigger, which certainly is of interest for B-group and provides a very high trigger rates the other hand. Typical topology of event of interest taken in this trigger is electron on one side and fully or partially reconstructed B-meson on the opposite side. So at Level 3 one could require presence of the (low-Pt) jet on the side, opposite to the electron. One could also think about requiring total Pt of all the tracks on the side opposite to electron to be greater than some minimal value. These cuts have high efficiency for events of $b\bar{b}$-production whereas they could allow not to write out events for which flavour of the B-meson on the side opposite to electron could not be reconstructed.

- similar considerations are applicable in case one two-track triggers, which are very intensive consumers ot the trigger bandwidth. To study CP-violation in B-mesonsystem in addition to 2 tracks separated from the beam axis and having their effective mass around that of B-meson one also needs to reconstruct flavour of the second B-meson, so a cut on the sum of Pt's of all the tracks in the semisphere opposite to 2-track system applied at Level 3 would have high efficiency for selecting the events which potentially could appear in the final distributions.

Up to this moment we assumed that all the data necessary for most analyses (all PAD's) are disk resident. It could be the case even for this scenario. To simplify the considerations we assume that disk space needs are dominated by the statistics coming from B-triggers, so what is being kept on disk corresponds not to PAD's, but to BPAD's. Average size of Run I BPAD was about 16 KBytes. Even if we assume it to be 25 KBytes in Run II, i.e. by more than 50% larger (which we believe is a very conservative estimate), then given the compression factor of 2 provided by ROOT, the size of the compressed BPAD event would be about 13 KBytes. To keep $2 \cdot 10^9$ events on disk storage one needs about 25 TBytes of disk space, which is still less than 30 TBytes we assumed to be available in total.

It should be mentioned that in this scenario not all the RAW data are expected tp be available from automatic storage (a limitation provided by a budget constraint), which could significantly complicate reprocessing of the shelf-resident data. However, even in this case a ROOT-based data handling system could provide a solution. With RAW data split into several branches, say, tracking RAW data branch, calorimetry RAW data branch, muon RAW data branch etc, one could start from putting on shelf tapes containing the RAW data, which most probably will not require re-reconstruction. For example, it is very unlikely that it will be necessary to do D-to-E conversion for calorimetry data twice in Run II, so after TOWE/QTOW bank is created, the tapes with RAW calorimetry data could be taken from the robotic storage.

In case this is not enough, CDF could consider setting different priorities to different physics analyses in terms of the hardware use. High-priority analyses would still keep all the PAD (BPAD) data on disk whereas the low-priority analyses would have to reconsider the disk-resident part of their datasets. The same approach could be taken when discussing the robotic storage. High priority analyses have all the necessary RAW data on robotic storage, the other analyses would have to keep part of the RAW data shelf-resident. We believe that if properly organized this approach could provide an efficient way of managing the available resources for the CDF data analyses in Run II.

12 Comparison of ROOT-based system to other approaches proposed for CDF

There are several data handling technologies proposed to be used by CDF in Run II. It is interesting to compare ROOT-based system to them and see what are the advantages and disadvantages of different proposals.

- First of all we note that
  - ROOT supports a flat I/O without splitting the event data into branches
  - using branches does not necessaruly require writing different branches to different files, it is still possible to write all teh branches into the same file.

  Thus, ROOT supports an I/O mode, in which system is essentially equivalent to the sequential model proposed in [15] with (TR)YBOS files replaced with ROOT files. Sequential model would provide better performance when the whole event is read in, however, as it has been shown in [2], for a data access pattern when it is necessary to read only a minor part of event data, ROOT-based system could provide an order of magniture improvement in the data access time over the sequential approach.

- Light-weight database proposal [13] describes the proposed system only at the conceptual level, it does not discuss any particular implementation. Comparing this proposal to [13] one could see that many features of the proposed light-weight database, like splitting event data into different files, use of variable granularity "BLOBS" etc are **already implemented** within ROOT project. So in a certain sense a proposal of ROOT-based system could be considered as an example of light-weight database implemented on the basis provided by ROOT I/O technology.

- The available results of benchmark tests demonstrate that ROOT I/O technology provides better performance and higher data compression than the Objectivity/DB I/O technology.

13 Conclusions

A proposal for the ROOT-based data handling system has been presented. We described the major components of the system and considered its configuration for 2 possible scenarios, corresponding to the integrated luminosity of 2 fb^{-1} collected in Run II:

- high-luminosity scenario, in which case CDF would collect about 1 billion events so for all the events a PAD format data would be available from automated storage (disk and robotic tape storage)

- low-luminosity scenario, in which case CDF could significantly increase L3 cross-section, so the overall amount of collected data would not fit to available automated storage;

We showed that in both cases the unique features provided by ROOT I/O technology

- direct access to the different parts of event data;

- splitting of event data into several files with low overhead of object-oriented I/O;

- high compression of the reconstructed data

could provide an efficient solution for the CDF Run II data handling system;

Based on the benchmark results presented in this note, we conclude that even if the only issues considered are those of I/O performance, ROOT technology outperforms its existing competitors in case when large data volumes are considered.

We want to emphasize that this conclusion doesn't depend on the amount of data CDF collaboration is going to collect in Run II

However ROOT provides not only the data model and data handling tools, but the complete set of data analysis tools which are based on this data model and integrated into it. This is a great advantage and it is especially important because the major goal of CDF in Run II is not to collect the data, but to **collect and analyse** them.

All these arguments taken together give us reasons to believe that ROOT technology could provide a reliable basis for the CDF Run II data handling system.

14 Acknowledgements

Authors would like to thank S.Tkaczyk who read the note and made many useful comments. We thank M.Shapiro, J.Lewis and R.Kennedy for many helpful and clarifying discussions. We appreciate discussions with Liz Buckley-Geer which resulted in the separate consideration of 2 data-taking scenarios. We thank M.Baker and P.Decowski (PHOBOS collaboration at RHIC) who provided us with the results of tests done by PHOBOS collaboration.

References

[1] Rene Brun, Nenad Buncic, Valery Fine, Fons Rademakers. ROOT documentation. see http://root.cern.ch for full set of the documentation.

[2] P.Murat, On the possibility to use ROOT Data Model for CDF data storage needs in Run II, CDF internal note 4412

[3] Liz Buckley-Geer and Rob Plunkett, Run II Data Handling Recommendations, CDFNEWS-15416 posted 24-JAN by FNALD::PLUNK

[4] W.D. Dagenhart, K. Karr, S. Rolli,K. Sliwa, S. Lammel, Specification of a data handling approach based on an object oriented datbase system and ....
CDF internal note 4346

[5] B.Blair et al (The CDFII Collaboration),
The CDFII Detector Technical Design Report
Fermilab-Pub-96/390-E

[6] C++ interpreter developed by Masaharu Goto (HP Japan).
See http://root.cern.ch/root/Cint.html

[7] ROOT I/O benchmarks: http://root.cern.ch/root/Ebench.html

[8] ROOT class reference guide: http://root.cern.ch/root/html/ClassIndex.html

[9] Liz Buckley-Geer Stephan Lammel Rob Plunkett, CDF Run II Data Handling
CDF internal note 4072

[10] see http://root.cern.ch/root/Zbench.html

[11] R.Brun and F.Rademakers, Performance comparison between ROOT, Objectivity/DB and LHC++ histOOgrams,
see http://root.cern.ch/root/Benchmark.html

[12] P.Decowski, ROOT vs Objectivity benchmark tests,
http://phobos-srv.mit.edu/~decowski/PhatObjyRoot (slides 15 until the end)

[13] E.Berman, ... S.Lammel et al, Specifications of a Light-Weight Event Database and Application to Fermilab Run II. CDF internal note 4345

[14] G.Bellettini and W.Carithers, Table of trigger cross-sections,
letter of the CDF Co-spokespersons to the Fermilab PAC (Dec 10, 1996)

[15] W.Kowald, S.Lammel, Specifications of a Sequential File Model and Application to CDF Run II. CDF internal note 4352

15 Addendum 1: Analysis of the Data Flow in the System

Data streams in the system could be divided into 2 groups, bandwidths of which are defined principally different requirements:

- data streams related to L3 and offline production
- data streams driven by physics analyses

While requirements for the bandwidths related to the data aquisition and production could be estimated rather reliably starting from the expected peak data logging rate and expected average raw event size, it is more difficult to quantify the technical requirements to the I/O bandwidths driven by the peak activity of the physicists analysing the data.

We start analysis with the data streams related to the data aquisition and production systems.

- With L3 peak output rate being about 70-75Hz and the average event size of 250 KBytes, the L3 output rate should be about 20MBytes/sec. The L3 output is directed to the SILO disk cache, the minimal bandwidth of this data stream should be of about 20 MBytes/sec;
- all the RAW data should be stored on tape, disk-to-tape bandwidth defined by the RAW data stream should be the same - 20 MBytes/sec;
- For the production farm to process the data at the same rate, the input robot-to-disk rate should be of the same order - 20 MBytes/sec. However this level of bandwidth is required only if production is synchronized with the data logging, in which case production could read input data directly from the SILO disk cache. Therefore, provided SILO cache is large enough, production farm could avoid rereading RAW data from tape. Writing the production output into multiple output streams with the overlap of about 40% requires production-to-robot I/O bandwidth of about 30 MBytes/sec. With all 3 components added up the total I/O bandwidth should be of the order of 70 MBytes/sec.
- disk-resident branches of the production output have to be written into the central repository in parallel. This should require less than 10% of production-to-tape bandwidth. We take 10% as an estimate, this tranlates into 3 MBytes/sec of disk-to-disk I/O and doesn't seem to be a problem.

One could approach the issue of the bandwidth of analysis-driven data streams from the opposite end and ask, for example, if an overal bandwidth of 100 MBytes/sec is could be available for the analysis jobs without specifying the consumer needs explicitly. Let's assume that the most part of this bandwidth will be defined by the analysis jobs using only disk-resident branches (remind that this is about 30-35 KBytes/event). The central disk repository is a distributed multi-CPU system with the data disks mounted locally to different CPU's. Taking 10 for the number of CPU's serving the central repository we come to the requirement of 10MBytes/sec per server CPU. Presently available buses (wide SCSI, for example) support data transfer rates which are twice as high (20MBytes/sec), so we conclude that the analysis bandwidth of about 100 MBytes/sec doesn't seem to be a problem for the proposed data handling system. Keeping in mind that in this case the overall bandwidth for the network-distributed disk repository is proportional to the number of server CPU's, one could conclude that at a price of about \$50000 (50 Intel/Linux boxes) an overall bandwidth of 500 MBytes/sec could be provided for the analysis needs.

In any hierarchical storage system involving tape storage a potential bottleneck #1 is tape I/O. One of the important advantages of the proposed system is that it is designed in a way which allows to minimize the tape I/O bandwidth. It should be mentioned, that we expect overall disk-to-tape bandwidth to be dominated by the output of L3 and production (50 MBytes/sec) with the required tape-to-disk bandwidth being significantly lower (20 MBytes/sec+). If it will be possible to run offline reconstruction in real time, we could save 20 MBytes/sec in each direction - in this case only output of the production could be written to tape. Peak I/O rates of about 10 MBytes/sec per tape drive have already been demonstrated (for example by IBM tape drives), average bandwidths could however be substantially lower. Therefore reducing overal tape I/O bandwidth by 40 MBytes/sec would mean reducing it by a factor of about 3 and it would be a very important practical issue.

The table below shows the estimated peak requirements for the I/O bandwidths between the major components of the system. Some of the fields in the table corresponding to the bahdwidths defined by the peak activity of the users were intentionally left blank (see comments below).

I/O Bandwidth requirements between the major compoments of the ROOT-based data storage system (MBytes/sec) estimated for the L3 output rate of 75 Hz and the average RAW event size of 250 KBytes.

from \ to	L3 farm	prod. farm	SILO cache	SILO	disk rep.	user.proc
L3 farm	*********	---------	20	----	--------	--------
prod. farm	------	************	30	----	3	--------
SILO cache	------	20	************	50+g	--------	e
SILO	------	---------	20+f	******	--------	--------
disk rep.	------	d	---------	----	**********	c
user proc.	------	----------	a	----	b	**********

a) most part of the MC generated data goes into the SILO cache and then to SILO
b) bandwidth required for new dataset creation (selections)
c) bandwidth required by the analysis processes
d) bandwidth required for partial data reprocessing (reclustering of the jets, SVX retracking) which doesn't need reading the tapes
e) bandwidth required by the reprocessing jobs (like full retracking) which need reading the tapes and also the analysis jobs which need full event information
f) this is an overall tape-to-disk bandwidth, which could (and should!) be significantly reduced provided SILO cache is large enough
g) bandwidth required for retrieval of tape-resident branches for reprocessing and analysis

16 Addendum 2: Initial Implementation Plan

Initial stage of the implementation plan has several goals, which include both prototyping and building parts of the new system and analysing CDF Run I experience with data handling. The major target is to have the data hanling system fully prototyped and tested in **6 months before** the start of Run II.

- fully prototype major components of the proposed data handling model model before the run II;
- based on Run I experience optimize the event format to minimize the average event size without loosing the information
- study overlap matrix (term belongs to Jonathan Lewis) between the production output streams and, based on this study, define their optimal organization minimizing the overlap;

16.1 Prototyping and building the components of the new system

- implement C++ classes for the ROOT persistent containers, optimize their interfaces to (TR)YBOS banks;
- implement input and output AC++ modules reading and writing ROOT trees and converting them into TRYBOS banks;
- implement user interface code telling ROOT to write different branches into different files
-
  - prototype splitting the RAW data into several streams;
  - write them onto separate tapes and store on SILO;
  - implement a prototype of event/run and dataset databases;
  - prototype a dataset manager software and interface it to AC++ (AC ?)
- we consider it possible to make prototyping of the ROOT-based system to be more than just an exersize in data handling. To do this we propose to reprocess one of Run I streams (Stream A J/psi sample) with the new CTC pattern recognition algorithm, work on which is currently going on. We propose to write the output into ROOT tree in split mode and to use the results for physics analysis.

16.2 Minimization of the overlap between the physics streams

The tasks described in this section are aimed in analysing CDF Run I experience with data handling.

- define overlap matrix between the physics production streams (Run I overlap between the physics production streams has been estimated in [5] to be about 30%)
- identify streams with large overlaps (if those exist) and see if it is possible to split the data into stream in a way optimizing the overlap between them;

16.3 Optimization of the Run II event format

There are several directions of work which should be pursued:

- model Run II RAW data event. This work has to be done in close cooperation with the simulation group and Level 3 group. It includes the following steps:
  - implementation of hit generation code for all the major detector components in CDF Run II Monte Carlo

- implementation of the RAW data bank creation routines in CDF Run II Monte Carlo;
- design and implementation of Run II reconstruction banks formats;
- simulation of **"typical"** Run II events for different physics processes and evaluation of average Run II RAW event size;

- study of "intelligent" ways to compress the RAW data. Here is just one of the directions in which this work could go on:
  - present format of SVX D-banks assumes storing for each event of about 10KBytes of ChipID/ChipStatus data ;
  - present COTD bank format **always** assumes writing for each event of about 10 KBytes of pointers to cells;
- design of optimal (in terms of data compression) output bank formats for the reconstruction. For example, we may consider to write out **only** Q(compressed) banks rather than uncompressed (QTRK vs TRKS);
- when the reconstruction code is ready, run it on MC simulated data and evaluate the average size of the reconstructed event data (separately PAD and DST)

16.4 Manpower requirements

Manpower requirements could be estimated starting from the following considerations.

- the work is planned to be done in close cooperation with the ROOT team, which would provide the support and future development of the ROOT system.
  We feel it important to mention that ROOT team has already expressed interest in participation at the next stages of this proposal
- one computer professional is required for developing event/run/dataset database software
- one physicist is required to analyse overlap between Run I production streams;
- work on estimating the RAW event size is planned to be done in close cooperation with Monte Carlo and online groups, which have to implement I/O routines for all the necessary RAW data banks.
- optimization of RAW event format should be done in cooperation with CDF online and offline groups;
- implementation of specialized AC++ I/O modules and interfacing to CDF software environment has to be done together with CDF Offline Upgrade group
- prototyping the data handling model itself and the data analysis model will require 2 more physicists.

Therefore the estimate of the manpower required is 3 physicists + 1 computer professional (all working full time on the project). This estimate doesn't include the work which is supposed to be done by the physicists/computer profeccionals involved with the project on part-time basis. Assuming for this a scale factor of about 1.3-1.5 we estimate that 4 physicists and 2 computer profeccionals working on the project full time will be sufficient for successful completion of the initial implementation plan.