Data Handling at CDF : ROOT-based approach

From the recommendations of the CDF Data Handling Technical Review Committee released on May 01 1998:

5.0 Recommendations:

[1] Attention needs to be paid to all elements of data handling and an architecture needs to be developed that defines all the pieces and their interactions.

[2] In light of the limited manpower available to the collaboration, we recommend:

* Discontinue the development of the LWDB approach

* Stop work on objectivity for CDF; design a modular system that does not preclude future use of Objectivity.

* Integrate Root I/O mechanisms into the Trybos I/O package.

[3] Formalize collaboration with Root team and CDF/D0/CD joint working group as soon as practical.

P.Murat, CDF collaboration, June 16, 1998

1 Some Interesting Numbers

- **Curent generation of HEP experiments is the first one, where the Data Handling systems are budget-constrained and their design to much extent budget-driven**
- CDF in Run I : about 40-50 TBytes of data on tapes
- CDF in Run II: use 1 PByte for all the estimates (20 times more than in Run I), system should scale up to 3-5 PBytes
- assuming $2/GByte of tape media: **$2 M** only for tape media,
- scaling up by a factor of 20 the disk resources CDF had in Run I results in about the same number - about $2 M...
- could we afford having a backup copy of the RAW data? - it is of the order of $1M..
- FNAL (CDF + D0) DH project: about $5M per experiment, comparable with the costs of detector subsystems: CDF intermediate tracker has been initially estimated below $5 M...
- **strong demand for new I/O technologies:**
  - estimated size of CDF RAW event in Run II is about 250 KBytes, peak data logging rate 75 Hz, with EXB 8500 5 GByte tapes a shift crew has to put a new tape into a tape drive every 4 minutes...
  - peak data logging rate about 20 MBytes/sec. Today no one tape drive avalable on the market provides such a bandwidth, Eliant Exabyte - about 1 Mbyte/sec

2 Overview of CDF Run I Data Handling System

- Level 3 output - 3 streams:
  - express-stream ("stream A") : 10% of all the data, the most important triggers, used for troubleshooting and monitoring of detector physics performance
  - all the data data ("stream B")
  - some of prescaled L3 triggers ("stream C") filling the output L3 bandwidth
- production output - about 20-30 physics-oriented streams
- having multiple streams allowed physics groups at the analysis stage to deal with the data dense in the events of interest - don't need to spin 100 tapes with only each 100-th event being of potential interest for the particular analysis
- saves tape I/O bandwidth and number of tape mounts
- data reprocessing: a serious issue for any tracking detector (new calibration/alignment constants, new track reconstruction algorithm)
- splitting all the data into physics-oriented streams greatly simplified reprocessing of the data - not all the streams needed to be reprocessed.
- the price: overlap between the production streams: 30 %. With the express-stream added the total overlap was about 40%.

- we had robotic tape storage (STK SILO) in Run I with tape library capacity of about 200 TBytes, had very positive experience with it.
- in principle we could put all the Run I data into SILO, didn't do it though
- data handling system worked quite well, it didn't limit physics analyses
- the most important lessons from Run I:
  - luminosity bookkeping, event/run cataloges have been implemented at a very primitive level, need to **improve bookkeping**

    given event/run numbers need to be able to find corresponding file(s)/tape(s)
  - data reprocessing: we were able to reprocess all the Run I data several times in a centralized way, but were not able to organize reprocessing of the large datasets needed by the physics groups

    retracking of inclusive low-Pt lepton datasets by B-group with new SVX alignment constants has not been done)

    need to **keep all the data necessary for the reprocessing on automated storage**

3 A bit of History

DH saga lasted about 6 months

- Dec'97: first workshop on data handling
- end of April'98: 2-day workshop on data handling
- May'98: recommendations by the Technical Review Committee
- in between:
  - a Technical Review Committee was set up
  - Specifications to the CDF Run II DH system by the Physics Groups
  - each of the proponents received a list of 40-60 **very technical** questions to be answered in a written form

For me it was the first time when the software technology has been treated as it should be - as a technology. The choice has been made after the technical review

4 Solutions considered for Run II CDF DH System

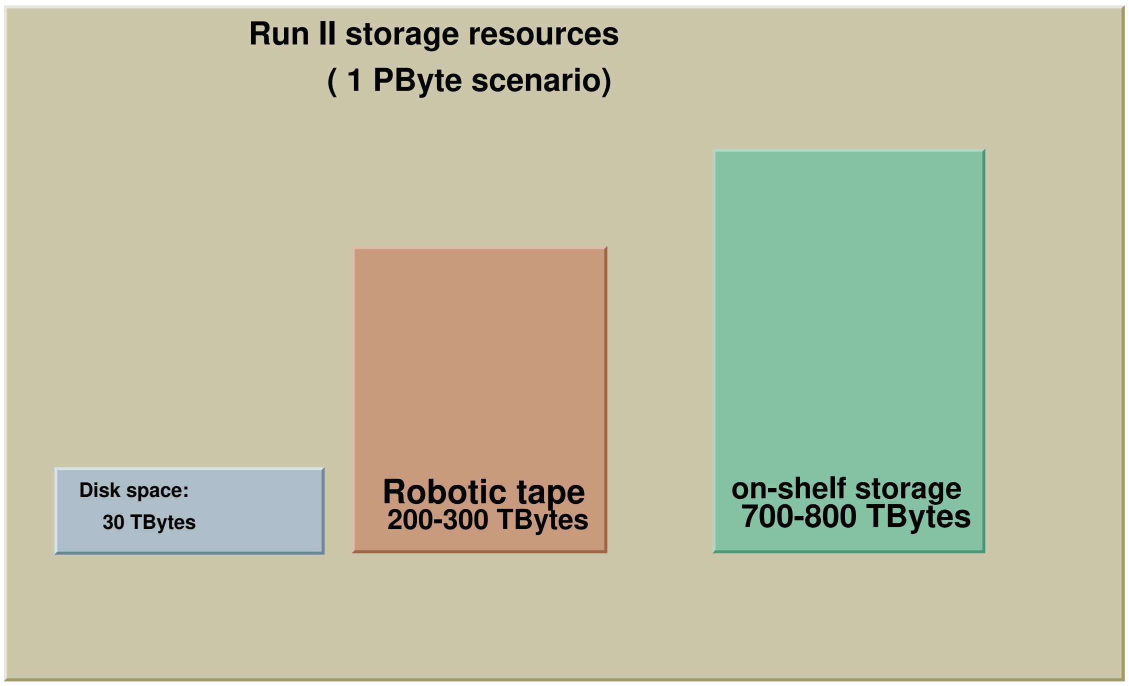
- sequential file model of Run I.
- Light-Weight DB: an approach based on new, to be developed file format.
- Objectivity/DB-based solution
- ROOT-based solution

5 Identified elements of DH system for CDF

- Resource Manager:
  - how to avoid a situation when one student allocates all the available tape drives or submits 100 jobs into a batch queue (in Run I CDF used NQS/IRIX batch system with "FIFO" batch queues)
  - how to manage efficiently the disk space available: pretty bad experience from Run I, where the physics groups didn't have any tools for resource management and significant part of the disk space has been filled up with MC datasets which nobody has been using
  - automatic spooling to tape
- Bookkeping/Cataloging System: Run DB, Luminosity DB
- HSM Interface
- Production Interface
- Analysis Interface
- Event I/O

Note : many components of the Data Handling System are common for all the proposed solutions.

6 Sequential Run I File Model



- simple but inefficient from the point of view of data access
- many high-statistics analyses (exotics, for example) are I/O rather than CPU bound. The characteristic data access pattern for these analyses is that the very first simple cut rejects significant part of the events ($\not{E}_t > 25GeV$) Provided $\not{E}_t$ is precalculated it is not necessary to read **all** the event data to make this cut. By reading only the necessary part of event data one could speed the analysis job up by an order of magnitude.
- reprocessing of the B-datasets: Run I approach required not just a volunteer, but a **kamikadze**.
- inefficiency calls for strong administration
- **So we believe that not from the very beginning of the Run II, but somewhere closer to the end the sequential approach could become a major limitation for many analyses and analysis-related jobs.**

7 LWDB

Conclusions by the Technical Review Committee:

```
* The committee believes this approach requires extra
  manpower compared to the other options.

* The committee believes that most of the good ideas
  contained in the LWDB approach could be implemented
  in the other approaches that are further along.

* This approach offers no commonality with the rest of
  the HEP community.
```

8 Objectivity/DB: Lessons From BaBar

There is a significant effort put into Objectivity/DB by HEP community (RD45, BaBar). We've been communicating with BaBar people and learning from their experience.

- to provide any reasonable level of tape I/O a multilevel hierarchical storage system should be file-based (to read a single event need to prefetch a file from tape)
- If offline code at user level talks to DB daemon, every student has a chance to write something into the DB and students are especially inventive when it comes to something they are not supposed to do.. BaBar introduced a "firewall" between the offline code and Obj/DB. Only I/O modules talk to Obj/DB, offline code doesn't know about the persistency mechanism
this is a very good idea in general
- clustering issue (object size should match to page size) has not been easily resolved
- objects written to the DB should be reasonably large, don't want to write each single hit as an object into DB
- persistent vs transient objects. Offline code deals with transient objects, the ones which go to tape are persistent objects. Use good old technique of packing the data when writing them out
- schema evolution (changing the bank formats): Obj/DB claims to support schema evolution, but they use this term for something different from what we use it for. BaBar builds schema evolution support on the top of Objectivity. Each change the format of transient objects leads to a creation of a new class.
- because of low e^+e^- cross-section the performance issues are

much less critical for BaBar than for CD, what could be acceptable for BaBar may not work for CDF.

- ongoing efforts (RD45, BaBar) have been concentrated only on file I/O issue, the other components - resource management, production interface, analysis interface - have not been worked on. Time scale of RD45 is very different from that of CDF.

Observations by the Technical Review Committee:

```
* Exists. Extensive HEP field testing of disk I/O is
  underway. Tape I/O later this year.

* Ongoing HEP development efforts have not concentrated
  on areas key for CDF applications - resource management,
  bookeeping/catalog, tape access, etc.

* Not clear how physics groups will utilize increased
  flexibility offered by this approach.

* Remaining technical challenges appear difficult
  to surmount on RunII timescale with resources available
  for CDF.
```

9 ROOT-based solution

- technology by ROOT project at CERN (lead by **R.Brun**) has several **unique features**:
  - supports both **sequential access** and **direct access** I/O modes;
  - parallel multibuffer I/O (a small revolution in I/O technique) provides a **transparent switch** in between two I/O modes
  - parallel multibuffer **data compression**, which gives compression ratios of about 2 for the reconstructed data
  - an **analysis framework** with all the necessary analysis tools is integrated into **ROOT data model**
- according to the available benchmarks ROOT data handling technology is the most advanced among the technologies known in HEP.

Observations by the Technical Review Committee:

```
* Proposed solution for I/O seems to migrate smoothly
  from sequential files to split files with branches
  on different media to direct access of branches on
  disk.

* I/O portion of Root field tested in HEP.

* Relatively low additional overhead for remote users.
```

10 What is ROOT (users point of view

In 0-th approximation one could think of ROOT as of C++ implementation of CERNLIB, which already includes KERNLIB, PACKLIB, HBOOK, HPLOT, PAW, ZEBRA etc...

i.e. as of

- a set of libraries to link against;
- an analysis (PAW-like) framework providing access to the data (ROOT files). From the command prompt of the framework a user can interactively invoke his own routines (equivalent to PAW **kumac's**) to operate on the data

ROOT is an object-oriented (C++) software system which provides a :

- **data model** : run-time data model, file model, persistency (I/O) mechanism

and integrated on the basis of this data model

- physics analysis tools (histogramming, fitting)
- graphics and presentation tools
- tools for automatic documentation generation
- networking tools
- more ...

Present implementation of ROOT (see **http://root.cern.ch**) already provides the level of functionality which **supercedes that of CERNLIB in many areas** (including I/O and analysis tools) and ROOT has a very good potential to become a successor of CERNLIB for the next generation of HEP experiments.

ROOT is being developed at CERN for already 3 years, the first public public release was announced in winter'97

11 Overview of ROOT I/O Model

- Container for storing the event: a ROOT **tree** ;
- In simple words **tree** is a column-wise ntuple with its element being an **object**, for example a **CDF data bank**.
- being an ntuple, ROOT tree could be used for the data analysis in exactly the same way as we are using HBOOK ntuples in PAW with ROOT providing an interactive analysis framework;
- **tree** is a single object written to a file, file may contain other objects like **run conditions record** which could be accessed directly, w/o reading the whole file
- ROOT supports I/O mode in which all the event data are written sequentially into the same file;
- event data could be split into **branches** which are written out via different I/O buffers **(principally new in HEP concept of parallel multibuffer I/O.)** - this provides a direct access mode
- contents of each branch could be read back independently without the reading the total file - **deferred input**
- branches could be written out to one or to several different files in a way transparent for user
- support of schema evolution
- supports writing out pointers to objects (significant part of functionality of OO-databases).
- doesn't have typical for commercial object-oriented databases clustering problem;
- branch-based data compression gives compression ratios of about 1.2-1.3 (**gzip**) and of about 2 for the reconstructed data

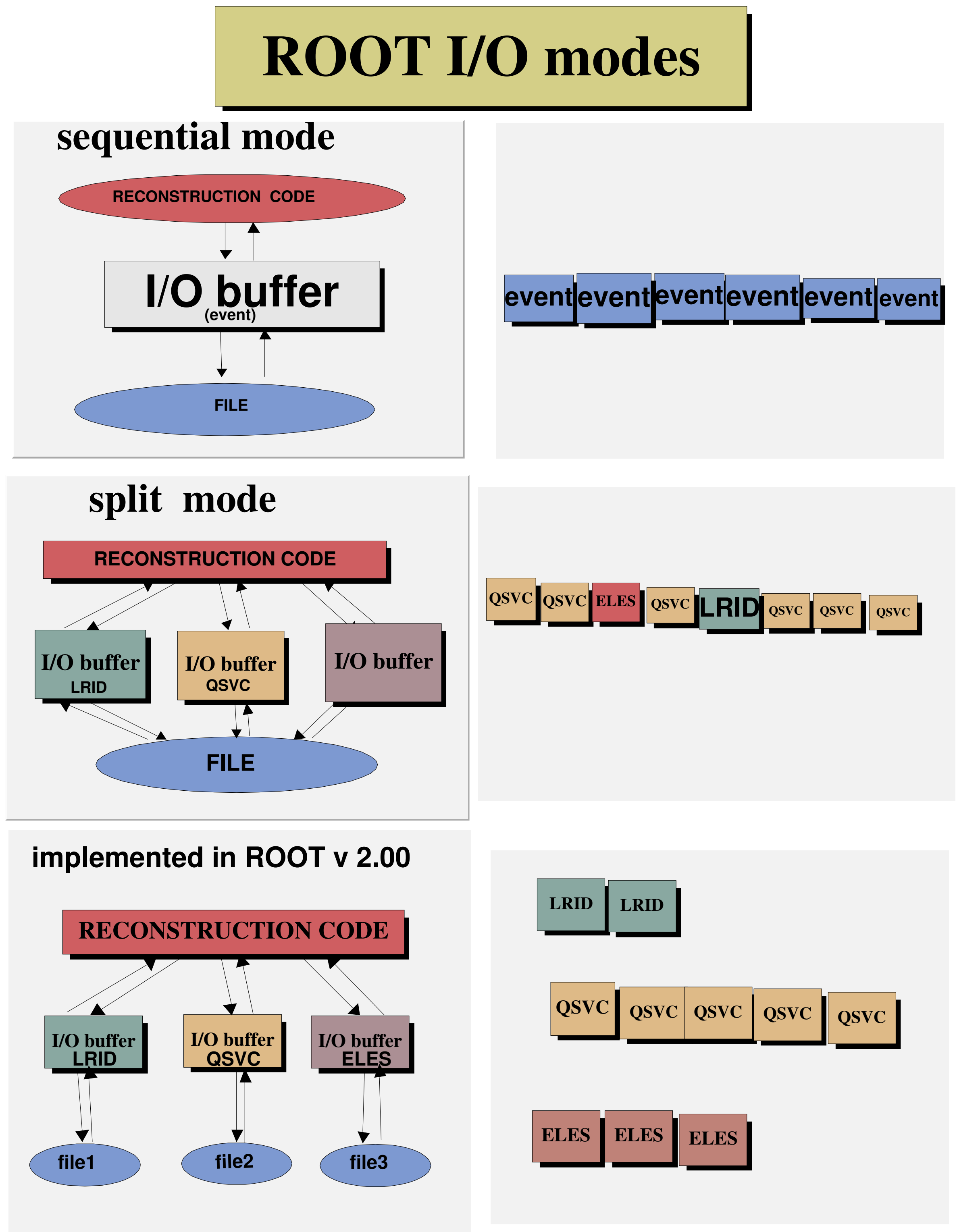


Figure 1: ROOT I/O modes

Support of Schema Evolution

```
class T {
  int a;
  ClassDef(T,1)
};
```

```
void T::Streamer(TBuffer R__b)
{
  // Stream an object of class T

  if (R__b.IsReading()) {
     Version_t R__v = R__b.ReadVersion();
     R__b >> a;
  } else {
     R__b.WriteVersion(T::IsA());
     R__b << a;
  }
}
```

```
class T {
  int a;
  int b;
  ClassDef(T,2)
};
```

```
void T::Streamer(TBuffer R__b) {
  if (R__b.IsReading()) {
     Version_t R__v = R__b.ReadVersion();
     R__b >> a;
     if (R__v == 2) {
       R__b >> b;
     }
  } else {
     R__b.WriteVersion(T::IsA());
     R__b << a;
     if (T::IsA() == 2) {
       R__b >> b;
     }
  }
}
```

Figure 2: support of schema evolution in ROOT

12 Overview of Run II DH System

12.1 Data volumes in the System

- integrated luminosity 2 fb^{-1}, 1.8×10^9 events, average size of RAW data: **250 KBytes/event**
- RAW data: allow separate reprocessing of different physics streams by the cost of replication of the production output 1.4 (30% overlap + 10% express stream), overal amount of RAW data: **630 TBytes**
- no DST format, only PAD, reconstruct missing banks on the fly
- PAD data: concervatively (25% of the RAW data), 60 KBytes/event, overal amount of reconstructed data: **160 TBytes** (only PAD's)
- overall amount of the data: **790 TBytes**

12.2 Data clustering

- data are organized in banks (RAW) and bank-like objects (try to avoid writing pointers out)
- RAW data: tape-resident superbranch (one or several branches - may want to keep tracking data separately)
- PAD superbranch: mostly disk-resident
- to the moment: no backup copy of the RAW data
- **multistep transition to ntuple-like event format**
  - step 1: write CDF event out as a TObjArray of YBOS banks (practically completed)
  - step 2: reconsider event model and split data into branches (undesrtand how this has to be done)
  - step3: learn to work with the data split into several files

13 ROOT Benchmarks Done at CDF

- data compression

filename	PAD size	ROOT size	compression
/data63/qcd/jet96/j200/pad/qj0b_ccr598.pad	35137536	18812254	1.87
/data61/top/ana/run1b/wevents/topfnd_w4jet.pad	3872768	1711437	2.26
/data61/top/ana/run1b/wevents/topfnd_w3jet.pad	11927552	5228739	2.28
/data61/top/ana/run1b/wevents/topfnd_w2jet.pad	69574656	30351968	2.29
/data61/top/ana/run1b/wevents/topfnd_w1jet.pad	399017984	177220173	2.25
/data17/bottom/psia_7b/a59640a_.psia_7b	12793856	6600864	1.94
/data17/bottom/psia_7b/a59985a_.psia_7b	12161024	6688514	1.92

- performance comparison vs YBOS. Use PAD file with 1216 events, corresponding ROOT file written in split mode (about 35 branches)
- YBOS: 2.35 CPU sec
- ROOT (read all the data) : 8.88 CPU sec
- ROOT (read only 1 small branch) : 0.15 CPU sec

13.1 Data on Tape

We propose to utilize fully the unique data compression features of ROOT I/O. Given the compression factors presented above, the overal volume of the (compressed) data to be stored becomes

$$\mathbf{630/1.2 + 160/2 \simeq 600TBytes}$$

This is equivalent to the maximal capacity of
2 STK SILO (assuming 50 GByte cartriges are use),
which in principle could be affordable

790TBytes - 600TBytes = 200TBytes
200TBytes of tape storage in today's prices is about $400K

If there is not enough robotic storage for all the data:

- split RAW superbranch into several files put on shelf those branches which most probably will not be needed the second time :
  - RAW calorimetry data (QTOW is enough)
  - RAW muon data (reconstructed segments should be enough)
- the only part of RAW data which for sure will be necessary to store on robotic storage is tracking D-banks
- squeeze RAW banks. Example: silicon D-banks contain 10K chipID/chipStatus data for **empty event**

13.2 Data on Disk

- central disk repository: 30 TBytes total
  - 25 TBytes for permanent datasets
  - 5 TBytes: cache of the robotic tape storage
- Don't duplicate events on disk: 25 TBytes of disk space translate into about **14 KBytes/event**, with ROOT compression this means about **28 KBytes** of uncompressed data in average;
- Run I PAD size: **32 KBytes** in average, Run I BPAD size: **16 KBytes** in average

assume (but this has to be proved) that 28-30 KBytes per event should safisfy most part of Run II physics analyses

- **if there is no overlap between the physics streams on disk makes it unclear how to do reprocessing** ;
- 40% overlap between the physics streams on disk: **about 70% of all the PAD data could be kept on disk permanently**, the rest part is kept on robotic storage (PAD superbranch)
- decision on which PAD's are kept on tape has be made by the physics groups
- large size of robotic storage cache will provide reasonably large "disk life time" for these data

14 Data Access Modes

14.1 Tape-Resident Data (RAW)

Tape-resident branches are not supposed to be accessed directly, if necessary the data from them are cached onto disk. Disk cache of about 5 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.

14.2 Disk-Resident Data (PAD)

The central disk repository of about 25 (+5) TBytes will most probably have a network-distributed architecture (several CPU servers with disks mounted locally to the servers). Assume most analyses to use data stored in the central repository.

In this case optimization of the network traffic becomes an issue to be dealt with.

Example: an analysis job, processing the whole inclusive high-pt electron dataset, data of which are distributed over the several filesystems belonging to different nodes. 2 principally different processing modes:

- job runs on a single CPU and reads the necessary data over the network (several users could easily paralize the local network);
- **clone the executables** and distributing them over the network to the nodes where the data are stored. **No data traffic over the network**. Nesessary to coordinate output of many executables.

The second approach: PROOF (Parallel ROOT Facility) undergoes testing in NA49 experiment at CERN.

15 Scalability of the Data Model Supported by ROOT

What is the maximal size of the data in the system? what are the limitations on the data organization?

- ROOT 2.0 supports 64-bit filesystems, so there is **no practical limitation on the overall volume of the data**.

How efficient is the access to the large datasets? Where the phase transition occurs?

- **direct access to each branch:** read in only those branches which are necessary for the analysis and minimizing the required I/O bandwidth (deferred input)
- keep different branches on different levels of data storage hierarchy
- whereas it is possible to split the event data into as many branches as possible, expect that branches corresponding to different named banks would provide an efficient data access with minimal overhead;

16 Data Replication and Distribution

- assume most part of high-statistics data analyses will be done using the computing resources located at Fermilab
- do not consider a problem of distributing primary datasets to the local computing centers of CDF institutions.
- 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 in Run I, no installation of any special database-like support tools is required in the remote institutions.

17 Calibration DB / Run DB Replication and Distribution

- with the exception of write locks ROOT I/O is an OODB I/O, thus ROOT file model is ideally suited for offline readonly databases
  - ROOT file may have directories and subdirectories
  - objects retrieved by name (key includes the directory name)
  - multiple versions (cycles) of the same object
- central CDF DB in Run II will be driven by a commercial engine (Oracle), the same for online and offline
- interface to Oracle: TSql class (discussed with ROOT team)
- create read-only extractions from the CDF DB in ROOT format and distribute them to the remote institutions.
- no licensing problem, universities will not need to buy and maintain commercial DB software, no installation problem, no need to have system privilleges

18 Summary

- ROOT-based system provides a smooth transition from sequential to DB-based I/O model
- unique data compression features could could allow to keep most part of the PAD data on disk and significantly reduce tape I/O
- non-negligible savings on the media
- less secondary datasets: use event/run selection lists
- field tested (NA49)
- integrates DH and analysis tools
- available on all the platform/compiler configurations of interest
- is free, available in source codes, thus no dependence on the vendor
- more and more experiments start using ROOT, so this solution could provide commonality with the rest of HEP community

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