NEW TECHNIQUES FOR DATA ANALYSIS
IN HIGH-ENERGY PHYSICS

Richard P. Mount
California Institute of Technology, Pasadena, California, USA

ABSTRACT
If Thomson and Rutherford were given a tour of CERN, SLAC or Fermilab, they would have little difficulty in comprehending the principles of the huge accelerators as awesome, but logical, extensions of their own experimental techniques. However, the path leading from the fundamental collisions produced by the accelerators to the publications in Physics Letters would amaze, confuse, and quite possibly distress them.

In an attempt to alleviate such confusion and distress I will describe the physical and logical foundations of data analysis techniques in high energy physics experiments. Particular stress will be placed on new techniques involving interaction, graphics, workstations and databases, while not ignoring older ideas which continue to be important.

1. INTRODUCTION

These lectures will concentrate on new techniques in HEP data analysis. However, since the intended audience includes students with little or no HEP background, I have provided a brief introduction to HEP itself, and a summary of the logic behind all physics analysis techniques, old and new. Unlike the less fortunate students at the school, the reader is invited to skip over the sections which insult his intelligence or his erudition.

1.1 Why HEP?

Our fundamental understanding of the universe has two frontiers:

1. High Energy Physics,
   very small space-time scales,
   high energy densities.

2. Cosmology,
   very large space-time scales (at least now).
We believe that HEP re-creates the conditions in the universe when it was only nanoseconds or picoseconds old. Thus HEP and Cosmology are not separate studies, since they merge in the early stages of the evolution of our universe.

HEP experiments have revealed the tantalisingly simple structure of matter and forces shown in Fig. 1. All matter is made up of half-integer spin particles known generically as fermions. Quarks carry ‘baryon number’ and leptons don’t. All the forces are explained by the exchange of integer spin particles known generically as bosons. Figure 2 shows how the exchange of bosons can explain both scattering and new particle production using $e^+e^-$ collisions as an example. There is an appealing symmetry between the three ‘generations’ of quarks, and the three ‘generations’ of leptons. There is also an appealing similarity between the way in which all forces work, since they are all described by ‘gauge theories’ which require physical observables to be unaffected by a very general set of transformations. (For gravitation this is still only a conjecture).

<table>
<thead>
<tr>
<th>THE WORLD CONTAINS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATTER (FERMIONS)</td>
</tr>
<tr>
<td>quarks and leptons</td>
</tr>
<tr>
<td>u c t e μ τ</td>
</tr>
<tr>
<td>d s b ννν</td>
</tr>
<tr>
<td>FORCES (GAUGE BOSONS)</td>
</tr>
<tr>
<td>Electroweak force</td>
</tr>
<tr>
<td>γ, W⁺, W⁻, Z⁰</td>
</tr>
<tr>
<td>Strong force</td>
</tr>
<tr>
<td>gluon</td>
</tr>
<tr>
<td>Gravitation</td>
</tr>
<tr>
<td>? graviton ?</td>
</tr>
<tr>
<td>AND</td>
</tr>
<tr>
<td>The Higgs</td>
</tr>
<tr>
<td>H⁰</td>
</tr>
</tbody>
</table>

*Figure 1* The simple structure of matter and forces.
Figure 2  How $\gamma$ and $Z^0$ bosons make $e^+e^-$ interactions happen.

This structure is tantalising because, while it is fairly simple and consistent with everything we observe, it contains too many arbitrary numbers for intellectual comfort. The most offensive arbitrary numbers are the masses of the fermions. The natural mass scale of the ‘world’ shown in Fig. 1 is of the order $10^{15}$ to $10^{19}$ GeV, and the observed masses of the order $10^0$ GeV must either be due to outrageous coincidental cancellations, or something new. There is also no explanation of why the 6 quark ‘flavours’, or the apparently corresponding 6 types of lepton should exist at all.

The masses of the bosons are in marginally better shape. The photon and the gluons have zero mass which is a comfortable value giving theorists few problems. The $W$ and
Z bosons are the heaviest particles known, and their theoretical co-existence with the photon is allowed by the, as yet undiscovered, Higgs boson, whose interactions with $W$s and $Z$s makes them massive. The Higgs itself has problems maintaining its respectability because, although it is an essential component of the dramatically successful Glashow-Weinberg-Salam electroweak theory, the theory places almost no constraint on its mass.

Fortunately for experimental HEP, it seems that enlightenment will be accessible. Most theorists firmly believe that both the Higgs, (perhaps in a more complicated form than one $H^0$ particle), and some new particles casting light on fermion masses, must be accessible to the current (LEP/SLC) or the next (LHC/SSC) generation of experimental facilities.

1.2 Making Measurements in the Quantum World

The fundamental interactions are not deterministic. For example, if an electron and a positron collide, we have no way of knowing in advance which of the processes shown in Fig. 2 will take place. Even if the physics of $e^+e^-$ collisions were totally understood, and we knew the initial conditions perfectly, we could not predict any more than the probability of producing each possible final state.

Since we believe that the perfect physical truths we seek would only allow us to calculate probabilities, we can reverse the argument and say that only by careful measurement of probabilities can we gain an understanding of the perfect physical truths. To measure probabilities accurately we must record and analyse many collisions. The high energy electron-positron and proton-proton collisions, which will be our windows on to new physics in the next decades, are themselves extremely complex events, and analysing them in large numbers will require massive data handling and computing resources.

There are further problems which complicate our measurements of the probabilities resulting from fundamental physics, and which make precise, high statistics, simulation an integral part of physics analysis.

Physics Smearing

Figure 3 shows a typical way in which physics itself makes it more difficult to measure fundamental processes. The figure shows three snapshots of an $e^+e^-$ collision in which the ‘fundamental’ process is the production of a $b\bar{b}$ quark-antiquark pair. Within a time during which we cannot possibly make any measurements at all, the $b$ and $\bar{b}$ quarks ‘fragment’ into jets of other particles. Even if we could calculate the complex physics of fragmentation with arbitrary precision, and measure all the particles produced, we could not identify jets from $b$ and $\bar{b}$ quarks on an event-by-event basis. On a statistical basis, $b$ and $\bar{b}$ jets can be distinguished from $u, d, s, c$ jets and $\tau$ decays, because, among
other features, they tend to be broader.

This example reflects a general problem: the new, high energy (short time-scale) physics we try to study will almost inevitably be smeared out by low energy processes acting on a somewhat longer time-scale. If we are lucky, we may understand these low energy processes rather well, but their effects are, of course, probabilistic, and the smearing cannot be removed on an event-by-event basis. Even worse, it is usually impossible to 'unsmear' the measurements without making rather rigid assumptions about the nature of the more fundamental process which was smeared. It is therefore not surprising that the normal approach to physics analysis is to perform several Monte-Carlo simulations of the smeared data, starting from various possible hypotheses about the underlying physics.

**Detector Smearing**

We have already seen that physics analysis is difficult even with perfect detectors. Detectors are far from perfect for many reasons:
1. Dead regions needed for mechanical supports, cables, the beam tube etc.

2. Dead regions due to lack of money. For example, the L3 detector has a very high resolution electromagnetic calorimeter made out of Bismuth Germanium Oxide (BGO) crystals. One third of the 12,000 crystals will not be installed for the first two years of running due to lack of money.

3. Limited Resolution. Every device has a finite resolution, and some, such as hadron calorimeters at moderate energy, are inherently imprecise (resolution $\leq 50\% / \sqrt{\text{Energy}}$).

4. Confusion. Detectors do not always measure what we want to know. For example:
   
   - a particle emerging from 1 metre of iron is probably a muon, but it may be a ‘punch through’ pion,
   
   - a combination of a charged track (measured in a wire chamber) and light in a BGO crystal may signal an electron, but could also be due to a photon and a pion,
   
   - it is normally impossible to disentangle (important) electrons embedded in (boring) jets,
   
   - etc.

Real physics detectors smear and lose information even more effectively than the physics smearing mentioned in the previous section. In most cases, the only effective physics analysis technique is to make assumptions about the underlying physics, and see if they result in simulated measurements which are consistent with the observations.

There is an important exception to the ‘no physics results without simulation’ rule. If new physics shows itself by producing mono-energetic particles or particle pairs, the first discovery can often be made purely on the basis of the observed events. Nevertheless, the first discovery of ‘something new’ must be followed by detailed studies involving simulation before the full implications can be understood.

1.3 Outline of Physics Analysis

Figure 4 shows the principal building blocks of physics analysis. The data are typically 100 kilobytes per event of highly compressed information gathered by a data acquisition system from up to $10^6$ sensitive devices. The simulation must include models of (what may be) the underlying physics, models of the physics smearing processes such as fragmentation, and a full treatment of the way particles traverse and interact in the detector, and how their passage finally produces electrical signals. In most studies it is also important to perform a detailed simulation of possible background signals.
Background can come from 'boring' physics processes, from beam-gas collisions, or even from electrical noise.

![Diagram showing the outline of physics analysis strategy](image)

**Figure 4** An outline of physics analysis strategy.

The human mind is not equipped to decide whether $10^7$ real events, each 100 kilobytes long, are consistent with $3 \times 10^7$ simulated events. To overcome this frailty, the data are first 'reconstructed' reducing the 100 kilobytes to a few numbers for each event describing concisely what features of the event the detector has been able to measure. The term 'reconstruction' was first used in the days of bubble chambers. The charged particles emerging from few GeV collisions in a bubble chamber could be individually reconstructed with almost perfect efficiency. In today's high energy
detectors ‘reconstruction’ is often an optimistic term. Frequently the best that can be done is to measure unresolved jets or energy clusters. This is usually not a serious problem, since the unresolved particles in a jet carry mainly information about the ‘boring’ process of quark fragmentation.

Reconstruction reduces the data volume while attempting to preserve all significant measured information. The data volume is still enormous, so the next stage, often called ‘Physics Analysis’, reduces the data volume discarding information which may not be relevant to the particular effects under study. Both real and simulated data are reduced to a few simple distributions which can be overlayed to compare reality and hypothesis. Reducing the data volume in a way that preserves maximum sensitivity to new physics and maximum immunity to noise is the key to successful analysis. It requires intuition and intelligence, but also the computer tools to allow many wild ideas to be tried out quickly.

1.4 Two Physics Analysis Examples

Both the analyses described below may be ways to uncover important new physics.

**Higgs Search**

One of the most eagerly sought reactions at LEP will be:

\[ e^+e^- \rightarrow Z^0 \rightarrow H^0Z^0(\text{virtual}) \rightarrow \text{jets} + e^+e^- \]

The Higgs is expected to decay into jets of hadrons, so the measured final state will contain two hadron jets, an electron and a positron, all normally well separated in angle. If the Higgs is to be granted the status of ‘particle’ it must have a defined mass, and thus the \( e^+e^- \) pair produced together with it must always have a reconstructed mass equal to the centre-of-mass energy of LEP minus the Higgs mass.

The physics analysis technique is very simple:

1. Use ‘cuts’ to enhance the signal/background ratio. Some of the signal is, of course, lost.
2. Plot a histogram of the effective mass of the \( e^+e^- \) pair.
3. Simulate the signal to see how much was lost by the cuts.
4. Simulate all known backgrounds to see how much fake signal they contribute.

Although there may be a few tens of Higgs events hidden in a sample of millions of \( Z^0 \) decays, selection of events with an isolated electron and positron removes nearly all the background while preserving most of the signal. Figure 5 shows how a Higgs in the
mass range 20 to 40 GeV would stand out above the remaining background. If the mass is above 50 GeV, Higgs events would still be measured, but could not be identified.

Muon Charge Asymmetry

In addition to the ‘bump hunting’ technique, signs of new physics can also be found by a painstaking analysis of very common events. Figure 6 shows one such class of events, and their interpretation in terms of particle exchange. Interference between $\gamma$ and $Z^0$ exchange leads to an asymmetry in the angular distribution of muon pairs. If there are other $Z^0$-like particles they will distort this angular distribution, even though their mass is too high for direct discovery at LEP. Figure 7a shows what the data (points with error-bars) and simulation (histogram) might look like in the absence of new $Z^0$s, and Fig. 7b shows the small deviations from the simulation that might be produced by new high mass $Z^0$.

The effects I have fabricated in Fig. 7b are several times greater than the smallest effects we hope to detect at LEP. They are almost undetectable by casual inspection of the figure but show up clearly in a statistical analysis comparing its left and right halves. However, unless systematic losses of events due, for example, to dead regions of the detector, are understood almost perfectly, all the statistical precision is worthless and the potential discoveries may be lost. An even more terrifying spectre for an experimental physicist is the possibility of ‘discovering’ something which is not really there.
The interpretation of $e^+e^- \rightarrow \mu^+\mu^-$ in terms of $\gamma$ and $Z^0$ exchange.

### 1.5 Physics Analysis and Accelerators

My examples so far, and my more detailed descriptions of techniques later in these lectures, are all drawn from $e^+e^-$ physics at the LEP accelerator. Although the principles of physics analysis at other machines are just the same, there can be large differences in data rates and experimental techniques. For completeness I now compare the LEP environment with some of the more extreme experimental environments of the past and the future.
Figure 7  Comparing the muon angular distribution for data and simulation.

Bubble Chamber Physics
Many physicists of my age worked for some years on reactions like

\[ K^- p \rightarrow XXX \]

using a hydrogen bubble chamber as both target and detector. These 'good old days' now seem very different from LEP physics. The key features which explain this difference are:

1. Bubble chambers have an excellent acceptance for charged particles.

2. In the 'good old days' there was no serious underlying theory (at least of hadronic physics).

3. As a result of 1) and 2), there was very little need for simulation.
Physics at LEP
For its first years of operation LEP and the LEP detectors will be used to study

\[ e^+e^- \rightarrow Z^0 \rightarrow X. \]

The main features of such physics are:

1. Electrons are ‘pointlike’ particles, that is they have no known substructure.

2. A few million events per year in which ‘matter is created’ will be observed. This rate is due to the combined effects of pointlike cross-sections which fall like the square of the energy, and an enhancement by a factor of over a thousand due to \( Z^0 \) production.

3. New pointlike (fundamental) particles are easily seen. The production rates of all pointlike particle-antiparticle pairs are similar once their production threshold is exceeded.

4. Precise predictions can be made for most features of the events. Deviations of less than 1% from the standard model predictions can be clear signs of new physics. This sort of physics analysis requires high statistics and great care in both data analysis and simulation.

LHC/SSC - Future Hadron Colliders
The process studied at hadron colliders can be written

\[ pp \rightarrow X. \]

It might be more correct to write this like

\[
(u + u + d + g + g + g + u + \bar{u} + d + \bar{d})
+ (u + u + d + g + g + g + g + s + \bar{s} + c + \bar{c} + u + \bar{u} + d + \bar{d})
\rightarrow \text{XXXXX},
\]

which emphasises the complexity of protons and of their high energy collisions. Experimentation at tomorrow’s (or even today’s) hadron colliders is very different from life at LEP:

1. A few million events per second in which matter is created will be observed.

2. Most events are too complex for precise theoretical understanding.
3. Nevertheless, millions of 'hard' events occur each year. In a 'hard' event a pointlike constituent of one proton, carrying a large fraction of its energy, hits a similarly energetic constituent of another proton. Hard events are, in principle, as valuable as pointlike scatters in $\sqrt{s}$ physics, but the absence of a known constituent energy, and the presence of millions of unwanted events, makes life more difficult for the experimenter.

CLIC/TLC - Future $e^+e^-$ Colliders
There are no firm proposals to build $e^+e^-$ colliders at $\sim 1TeV$, but studies are proceeding on both sides of the Atlantic. Physics at these machines would be a total reversal of the frantic LHC/SSC era. The pointlike cross-sections at very high energy and (probably) without any enhancement like that of the $Z^0$ at LEP would reduce rates to about ten thousand events per year. At such rates it would be easy to provide the experimenters with adequate computing facilities.

2. Physics Analysis and Computing for a Large HEP Experiment

Figure 8 shows the diagram of the analysis structure for the L3 experiment. If I changed the title to 'HEP Analysis Structure' the figure would be correct for almost all modern experiments. In particular, the emphasis on providing a database service, and an interactive graphics service, both used at all stages in the analysis, is typical of today's experiments. The programs, such as the calibration processors, and the various components of the Monte Carlo and the reconstruction, may comprise up to 1,000,000 lines of code written specially for the experiment.

The physics analysis outlined in Fig. 8 requires many resources. The modern view is that software, data handling capacity, networks, and CPU power are all important tools for physics analysis. I would go even a little further, and say that I have written them in order of their importance, at least in so far as their need for manpower and money are concerned.

In an attempt to give some feeling for the resources required, table 1 shows some information about typical data volumes in a LEP experiment. The main message is that the data volumes are large, and making any fraction of these data available to the hundreds of physicists in a LEP collaboration necessarily requires expensive data handling and networking. Before going on to talk in detail about the latest software techniques and tools, I will describe a typical hardware and networking environment in which these tools are used.
2.1 The L3 Computing Environment

Figure 9 shows the overall structure of the L3 computing environment. The figure is now about four years old and only recently has begun to describe a reality rather than a goal. I will describe the L3 computing environment in some detail, because in most respects it is not at all specific to the L3 experiment and can be regarded as a good example of an environment for HEP data analysis.

Workstations were chosen several years ago as the best way to write and debug the hundreds of thousands of lines of code. Interactive graphics using workstations is the best available debugging tool, both for the software and for the detector itself.

Data handling capability for L3 is provided by the IBM mainframe component of LEPICS (the L3 Parallel Integrated Computing System) and by a share of CERN’s
Table 1

Typical LEP data Volumes per Experiment

| Event rate ($Z^0 \rightarrow$ hadrons) | 0.2 per second |
| Event rate (junk) | 0.8 per second |
| Size of good events | 200 kilobytes |
| Running time | 4,000 hours/year |
| Total 'raw' data | 10,000 tapes/year |
| Reconstructed Data | 10,000 tapes/year |
| Other data (eg. simulated) | 20,000 tapes/year |

central facilities. Networking on the CERN site and worldwide is necessary to make the real progress on physics analysis approach the sum of individual efforts. Finally, the little boxes hanging off the bottom of 'LEPICS' reflect my statement about the relatively minor importance of CPU power. These boxes provide most of the CPU power for L3, and cost much less than the data handling equipment.

I will now describe in a little more detail the main elements of the L3 computing environment.

Graphics Workstations
I will give some examples of the importance of graphics in a later section. Here it is necessary to emphasise what is meant (in experimental HEP) by a workstation:

1. CPU equivalent to between 1 and 12 VAX 11/780 equivalents. (Tomorrow's workstations will, of course, be faster.)

2. Large, high-resolution bitmapped screen. Machines with $1280 \times 1024$ colour screens are now normal. The hardware and software of the workstation support graphics and multiple windows.

3. A good Fortran environment. This includes a Fortran compiler up to mainframe standards, the ability to run the largest HEP programs, and a good debugging system.

4. Good communications with the VAXes and IBMs used for data acquisition and data handling. An isolated workstation with the power of a CRAY-4 is not useful to an experimental high energy physicist.
LEPICS

The LEPICS configuration at the end of 1988 is shown in Fig. 10. Most of the figure shows an 'ordinary' IBM system with all the usual peripherals and network connections. The main role of the IBM part of LEPICS is to give jobs and physicists access to L3 data stored on tens of thousands of tapes complemented by many gigabytes of disk space. Although the 3090-180E processor is not a negligible CPU resource, it is far too small (perhaps by a factor 12) to meet L3's CERN-site CPU needs. The first indications of how the computing power will be provided appear at the lower right corner of Fig. 10. The VICI is an interface between an IBM I/O channel and the industry standard VME bus. A string of 3081/E emulators is connected to the VME bus, and the IBM can copy data to or from the 3081/E memory at close to 3 megabytes/sec. The 3081/E is a processor built by CERN and SLAC which emulates the IBM system/370 instruction
set and uses the same data format as the IBM. The string of 3081/Es in Fig. 10 more than doubles the total CPU power of the configuration. Work is now in progress to connect more modern (and very cost-effective) processors such as the Apollo DN10000 to the IBM via VME to Channel interfaces.

Figure 10  The LEPICS computer system in December 1988.

This assembly of hardware and its interconnections can be the foundations of a powerful HEP computing system, if we can provide system and user software to exploit it. Fortunately nearly all the CPU cycles used by high energy physicists are employed to process 'events' — the real or simulated results of single collisions. Events are independent of each other, and so the 1000 events on a tape can be processed forward, backward, or in parallel. In practice, if we want the results of parallel processing (including histograms and statistical summaries) to be identical to those from a serial job, the software has to be rather well organised. However, adapting HEP software for par-
allel processing is relatively simple, and the resulting code may be even more intelligible than the original serial-only version.

Up to now, efforts to use parallel computing for HEP data analysis have stopped at the point where production programs had been made to run on a dedicated 'farm' of processors. The 'production team' would take control of the farm for a night or a week, and perform a long series of reconstruction or simulation jobs. Unfortunately, organised production is only part of the load, and in a collaboration such as L3, the other 395 physicists would also like to be able to run some of their jobs on cost-effective processors. It is necessary to take the large step from a 'processor farm' to a parallel computing centre. This step requires that the attached processors become managed dynamically in much the same way as the host mainframe CPU cycles are managed by its operating system. Figure 11 outlines how L3 will do this within the framework of the VM/CMS operating system.

Figure 11  L3 plans for attached processor resource management under IBM VM/CMS.
The virtual machine (process) BMON in Fig. 11 is part of the SLAC Batch Monitor system used widely within HEP. BMON manages the queue of batch jobs, and when appropriate, starts a 'batch worker' (JOB VM) virtual machine and gives it a job to execute. The attached processor resources are managed by the additional virtual machine EMUMAN, which releases jobs in the BMON queue when AP resources are available to execute them. Dynamic management means that the APs allocated to a job may vary, and EMUMAN accomplishes this by telling low-level code in the JOB VMs how to map a fixed number of virtual processors on to a varying number of real allocated devices. The Interface VMs provide a stable software interface to heterogeneous attached hardware. EMUMAN sets up communication paths from the JOB VMs to the attached processors, but does not itself handle this main data flow.

LEP3NET

The L3 experiment involves collaborators from the USA, Western Europe, Eastern Europe, the USSR, China and India. The network connections which now link most of these collaborators were in the main set up through the efforts of L3 members. Figure 12 shows the current configuration of 'LEP3NET' together with the ESNET-X.25 network which was recently created using LEP3NET as a model. (ESNET-X.25 links SLAC, LBL, Fermilab, BNL and MIT and has a satellite link from Fermilab to CERN.) Most LEP3NET lines now run at speeds between 9.6 kilobits/sec. and 64 kilobits/sec. supporting remote log-on to computers, and the exchange of software. Later in these lectures I will describe how much higher speed networking could revolutionise the way in which universities are involved in physics analysis. L3 members are currently expending considerable (mainly political) efforts to achieve megabits/second as soon as possible.

3. Examples of Software Foundations

I will now describe some of the modern tools which support physics analysis. I will concentrate on the approach used by L3 simply because I can explain this approach better than the equally valid (and usually similar) approaches used by other experiments. I will give two complementary examples of data management, the ZEBRA system and the DBL3 database system. Then I will describe the GEANT3 system which takes most of the hard work out of describing the complex geometry of modern detectors. Finally I will give a sales talk about the importance of interactive graphics, and describe the PAW system which provides the foundations for interactive physics analysis on workstations.
3.1 ZEBRA: Data-Structure Management and I/O

High energy physicists continue to program in the ugly language called FORTRAN. Computer scientists could probably list 100 reasons why this is a stupid idea, but from the strictly practical viewpoint, FORTRAN brings two big problems:

1. It has no data structures, only fixed-dimension arrays.
   HEP data usually has a very complex structure. For example Fig. 13 shows a simplified version of part of the data structure used to describe L3 events.

2. It has no efficient machine independent I/O.
   L3 data will be acquired by a VAX, first processed on an IBM, and then looked at by physicists sitting at Apollo workstations. All these machines have different data formats.

These problems are augmented by one constraint:

3. Physicists want direct read/write access to data in memory; returning the wanted data as a subroutine argument is (supposed to be) far too inefficient. In other words, physicists insist on being able to overwrite any part of their data by mistake.

Neither of these problems is new, and many packages providing solutions have appeared in the last 15 years. I will describe the ZEBRA [1] system but many of the ideas are common to its predecessors/competitors such as HYDRA, ZBOOK, BOS etc.

ZEBRA manages chains and trees of data objects in memory, and allows the I/O characteristics of each object to be recorded, so that any part of a data structure can be
translated into a compact, machine independent, format and sent to another computer. Figure 14a shows the logical structure of a chain, which is normally used to relate objects which are all of the same type, such as the tracks emerging from a vertex. Figure 14b shows how chains and trees can be used to build more general structures including hierarchical relationships. To allow efficient navigation around complex structures, ‘links’ (addresses) must be stored in data objects pointing to other related objects. Figure 14c shows the links automatically created by ZEBRA to help both the user and the ZEBRA system itself.

ZEBRA manages data by creating ‘banks’ in large Fortran Common blocks known as ZEBRA Stores. The banks include space into which the user can put data, and also space in which system and user links are maintained. For example, the ‘reference links’, shown as the upward-right pointing arrows in Fig. 13, are maintained with the structural links within the bank. Figure 15 shows the format of a ZEBRA bank. In addition to the features already described, note the I/O control byte and the I/O descriptor words which support automatic data translation for machine independent I/O.

I will not replicate the entire ZEBRA manual, and will conclude this section on ZEBRA by giving a few commented examples of the code which you might write when using ZEBRA. Of course, the point I am trying to make is that it is ‘not really all that unpleasant to use’, but I will let you judge for yourselves.
a) A linear chain of banks

b) A more general ZEBRA structure

c) Links automatically created by ZEBRA

Figure 14 Examples of ZEBRA data structures.

Initialise a ZEBRA Store

COMMON/name/IFENCE(10),LINK,ISTORE(100000)
DIMENSION LQ(999),IQ(999),Q(999)
EQUIVALENCE (LINK,LQ(1)),(LQ(9),IQ(1),Q(1))
CALL MZEBRA(0)
CALL MZSTOR(IXSTOR,'/name/',',',IFENCE,LINK,
Figure 15  The format of a ZEBRA bank.

*      ISTORE(1),ISTORE(1),ISTORE(20000),
      ISTORE(100000))

The nice little EQUIVALENCE trick allows integer and floating point data to co-exist in the data structures and ensures that LQ(bank_address - 5) is the address of the 5th structural link whereas IQ(bank_address + 5) is the address of the 5th data word. I gave a full set of arguments for MZSTOR to make my example realistic, but I will not burden the reader with an explanation of each one.

Create a Link Area

COMMON/GLINKS/LGEVNT,LGVRTX,LGTRAK,LGWIREF
CALL MZLINK(IXSTOR,'/GLINKS/',LGEVNT,LGVRTX,LGWIREF)

When using complex data structures it is convenient to store the address of frequently used banks, rather than to re-navigate through the structure whenever the
address is required. However, ZEBRA may 'garbage collect' at almost any time. Any addresses you might want to maintain should be declared to ZEBRA, so that it can update the addresses for you if it moves any of your banks around. This example shows the declaration of one 'structural link', LGEVNT, followed by three 'reference links'.

Create a Bank

CALL MZBOOK(IXSTOR,LGVRTX,LGEVNT,-1,'VRTX',6,3,100,2,0)

This shows the creation of a bank 'VRTX', returning its address LGVRTX. The new bank 'hangs off' link number 1 in the existing bank whose address is LGEVNT. The new bank has 6 user links, of which 3 are structural, implying that up to 3 banks may 'hang off' this new bank. The new bank has 100 data words, all of which are integers and all of which are cleared to zero when the bank is created.

Insert Data

Q(LGVRTX+1) = X
Q(LGVRTX+2) = Y
Q(LGVRTX+3) = Z

Notice that this example involves no subroutine call. The example should be compared with the corresponding 'pure Fortran' ('VRTX(1)=X etc.'). Although I promised that you could judge for yourselves, I am sure that you can see that, in this example, ZEBRA is only marginally more CPU consumptive, and only marginally more difficult to code and understand, than 'pure Fortran'.

Use Data

PRINT *, 'Vertex position', (Q(LGVRTX+I), I=1,3)

Once again, access to the data requires no subroutine call, and it is easy to make the code as readable (or unreadable) as normal Fortran.

Drop a Bank

CALL MZDROP(IXSTOR,LGVRTX, ' ')

Unlike Fortran arrays, ZEBRA banks, or the data structures supported by banks, can be dropped when no longer needed. Dropped banks are marked, but the space is only recovered by compressing the data structure if a 'garbage collection' is provoked by some operation which is unable to find enough immediately available free space.

Output a Data-Structure

CALL FZOUT(1,IXSTOR,LGVRTX,1, ' ',2,10,IHEADR)
The data structure supported by the bank whose address is LGVRTX within store IXSTOR is written out to logical unit 1. The record is flagged as the start of a new event and is preceded by a 10 word integer header read from the array IHEADR. An initialization call to FZFILE will have already chosen machine independent binary, machine independent ASCII, or machine dependent ‘native’ output mode.

3.2 DBL3: HEP Database Management

I am not giving a lecture course on database theory so, in this section, I will allow myself to take an HEP (or even L3) viewpoint on database systems. From this viewpoint, I define a database system to be:

“A system for the storage and retrieval of data in which data are identified by the attributes which are most relevant for the user or application”

Here are two typical ‘queries’ to database systems conforming to my definition. The first example is formulated in the SQL query language:

```
SELECT NAME, PHONE FROM L3_COLLABORATORS
WHERE RESPONSIBILITY LIKE '%%TEC%%GAS_SYSTEM%%'
ORDER BY DATE_LAST_CALLED_IN_THE_MIDDLE_OF_THE_NIGHT
```

The second example is typical of the functions required of a database for HEP calibration information:

```
CALL GET_VALID_CALIBRATION('BGO/BARREL', '03:14:55 19-MAR-1990')
```

When retrieving data from a database system it is not necessary to specify tape numbers, or device addresses, and in this respect, database systems are similar to modern file systems. In addition, a database system is normally expected to be efficient even when the size of the objects being stored and retrieved is very small.

There is an enormous range of systems which can be described as ‘database management systems’. At the most sophisticated end of the range are systems like ORACLE and SQL/DS which are widely marketed. Although no attempt has been made to optimise these systems for HEP, the main restriction on their use is cost rather than the quality and flexibility of the software. At the lower end of the range are systems like the ZEBRA RZ package, which computer scientists would probably call an ‘access method’ rather than a ‘database management system’.

Why should HEP be so interested in database systems? Let me answer this indirectly by pointing to Fig. 16 which shows the structure of the part of the L3 database referring to the alignment and calibration of the L3 muon chambers. The muon chamber system is just one of the high precision detector systems in L3. The intrinsic precision of these systems can only be realised by frequent calibrations of all their components.
The calibration information shown in Fig. 16 is not static; some information has a life of months, but many of the database objects are refreshed every few minutes. Unless these data are handled automatically it is quite impossible to exploit the detector we have built.

![Diagram](image)

**Figure 16** Structure of the L3 database for muon chamber alignment and calibration.

The L3 Collaboration recognised that a database system was required over four years ago. As a first step, the performance and features of existing products were assessed by using them to build a ‘model calibration database’ for L3. Table 2 shows the numerical results of tests using different underlying systems. Table 2 shows that the HEP-written packages KAPACK and ZEBRA-RZ were faster than commercial database systems. However, the HEP systems also lacked many of the useful features of the
commercial systems, and L3’s choice of ZEBRA-RZ was finally based almost entirely on the near impossibility of persuading over 40 institutes to buy an expensive commercial system.

Table 2: Results of the L3 Model Database Tests

<table>
<thead>
<tr>
<th></th>
<th>ORACLE IBM 3090</th>
<th>ORACLE VAX 8600</th>
<th>SQL/DS IBM 3081 (night)</th>
<th>KAPACK IBM 3081 (day)</th>
<th>RZ IBM 3090</th>
<th>RZ VAX 8650</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database Size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Megabytes</td>
<td>120</td>
<td>6</td>
<td>120</td>
<td>420</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>‘Rows’</td>
<td>7000</td>
<td>350</td>
<td>7000</td>
<td>20000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td><strong>Insert Performance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elapsed Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>small row, msec</td>
<td>100-300</td>
<td>200-1000</td>
<td>↓</td>
<td>↓</td>
<td>&lt;10</td>
<td>&lt;10</td>
</tr>
<tr>
<td>large row, msec/kbyte</td>
<td>6-17</td>
<td>13-77</td>
<td>330</td>
<td>26</td>
<td>6.5</td>
<td>3-7</td>
</tr>
<tr>
<td>CPU Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>small row, msec</td>
<td>↓</td>
<td>25</td>
<td>↓</td>
<td>↓</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>large row, msec/kbyte</td>
<td>0.7</td>
<td>3.5</td>
<td>77</td>
<td>1</td>
<td>0.65</td>
<td>0.4-0.7</td>
</tr>
<tr>
<td><strong>Read Performance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elapsed Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>small row, msec</td>
<td>50-80</td>
<td>100-150</td>
<td>55</td>
<td>20</td>
<td>59</td>
<td>78</td>
</tr>
<tr>
<td>large row, msec/kbyte</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>CPU Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>small row, msec</td>
<td>↓</td>
<td>25</td>
<td>↓</td>
<td>↓</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>large row, msec/kbyte</td>
<td>1</td>
<td>2.7</td>
<td>0.7</td>
<td>0.13</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td><strong>Read model L3 reconstruction environment:</strong></td>
<td>6 secs</td>
<td>10 secs</td>
<td>6 secs</td>
<td>3 secs</td>
<td>6 secs</td>
<td>5 secs</td>
</tr>
<tr>
<td>~40 rows, ~500 kbytes.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The RZ system supports a UNIX-like hierarchy of 'directories'. Directories can contain other directories and 'objects' where an object has a 'keys' and 'data'. The keys are used to identify objects, and are usually just a few words per object. The data part of an object can be retrieved once its keys are known. The DBL3 system, constructed on top of RZ, offers the following principal features:

1. Data objects are identified by pathname (e.g. //L3/BGO/BARREL/CALIB), and
by validity time. On insertion the range of validity times must be given. On retrieval, the system returns the most recently inserted object which is valid for the requested time.

2. Data objects can be compressed, either by applying compression algorithms to individual objects, or by comparing a new object with an earlier one and recording only the differences.

3. DBL3 manages a memory resident 'cache' of recently retrieved or entered data as shown in Fig. 17. The cache management is steered by two user callable subroutines:

\[
\text{CALL DBUSE(''/\text{DBL3/\text{MUCH/\text{MCALB/\text{ALIG/BEAC/}}}\),}
\]
\[
\text{Time-date, Other-Keys, Address, ....)}
\]

This means, 'I intend to use this database object. Please load it if it is not in memory (or if the copy in memory is no longer valid). Give me its address'.

\[
\text{CALL DFREE(''/\text{DBL3/\text{MUCH/\text{MCALB/\text{ALIG/BEAC/}}}\}')}
\]

This means, 'I won't be using this object for a while. You can overwrite it in memory if you need the space'.

4. Normally, if some DBL3 data are superseded, or even found to be wrong, they are not deleted. Figure 18 shows how old and new versions of calibration data can co-exist in the database. By default, the more recently inserted data will be retrieved by DBUSE, but it remains possible to make the database behave exactly as it would have done at some earlier time.

In commercial database systems, such as airline reservation systems, a correct and high performance handling of concurrent and conflicting write requests is of great importance. In an HEP experiment, if two conflicting groups are given responsibility for a calibration task, then the problems cannot be solved by database technology alone. In other words, conflicting write access to a calibration database is not a problem. This opens up the possibility of creating many simultaneously valid copies of the database which keep each other up-to-date by sending all changes over network links. The L3 collaboration will use a pair of communicating 'server processes' to ensure the (almost) simultaneous validity of the database on the data-acquisition VAX cluster, and on the off-line LEPICS system. Generalisation of this idea, to ensure coherence of L3 databases worldwide, is an intriguing and challenging prospect.
3.3 GEANT3: Geometry and Tracking for Complex Detectors

High statistics physics using a complex detector needs a Monte Carlo simulation with several major components:

1. Generation of the particles emerging from the collision assuming some combination of known physics and more speculative hypotheses.

2. Precise representation of the detector geometry. Typical detectors have over 1,000,000 components with which particles interact.

3. Precise simulation of all secondary interactions and 'showers'.

4. Precise simulation of the response of detectors (gas, plastic, glass, crystals, etc.) to the passage of the simulated particles.

The GEANT3 system [2] provides the core of a simulation program, leaving only the particle generation and description of the detector response to the user.

An economical but powerful description of detector geometry is supported by a hierarchy of 'volumes'. A volume can be split up into sub-volumes by positioning other volumes inside it, or by slicing it up in a repetitive way. The hierarchy is represented by a (ZEBRA) tree data structure. The symmetry of HEP detectors makes it unnecessary that the tree end in 1,000,000 'leaves' describing individual components, since most components are identical, apart from a spatial transformation, to many others.

For example, the L3 muon chamber system which consist of 176 chambers and tens of thousands of sensitive wires is described by some 60 calls to GEANT3 subroutines.

Graphics are vital in checking the correctness of this description of detector geometry. GEANT3 can produce a three dimensional display of the geometry that has been created and examples of these displays will be given in the next section.

Once GEANT3 'knows' all about the detector geometry, it can track particles through the detector and record their paths and energy losses in the sensitive volumes. However, energy loss is only one possible fate of a particle travelling through material. Even in empty space, many particles decay spontaneously into others, and in solid or liquid matter, most particles interact after a short distance. GEANT3 includes a detailed simulation of the 'electromagnetic showers' which are the usual fate of electrons or photons, and a painstaking simulation of the much more complex and varied processes involved in hadronic showers. This latter simulation originated in the GHEISHA simulation system [3] but is now also available as the GEANH package within GEANT3.

335
3.4 Interactive Graphics

The outline of GEANT3 provides a good introduction to my sales pitch for interactive graphics. GEANT3 is itself a package of some 60,000 lines of code, not including the other CERN library routines it invokes. To make GEANT3 into a complete simulation for a detector like L3, a further 50,000 lines of code are required, mainly concerned with the details of detector response. Programs containing more than 100,000 lines of Fortran code are common in HEP simulation, reconstruction and physics analysis. There is no way that a simple inspection of printed output from such programs can convince anybody that reasonable things are being done by all 100,000 lines of code.

Electronic wizards cannot build equipment without oscilloscopes. Their software counterparts would be foolish to build even more complex programs without the equivalent instrumentation. I will illustrate the importance of graphics produced interactively by means of a few examples – many of them drawn from the L3 simulation program based on GEANT3.

![Diagram of LEP3 and sub-detectors]

*Figure 19* The upper two levels of the GEANT3 data structure representing the L3 geometry.

It is natural to think of graphical displays of physical objects such as real detectors. However, for the software builder, a graphical display of an abstract object, such as a data structure can be at least as useful. Figures 19 and 20 show the result of giving the interactive ‘DTREE’ command to GEANT3. This command displays the tree data-structure which has been built up to represent the L3 detector. To facilitate checking the logic, small pictures of each geometrical object described by the data structure are added to the picture. Figure 19 shows just the top levels of the representation of the L3 detector, whereas Fig. 20 shows the complete structure representing the muon chambers. It should be stressed that, having written the 60 lines of code that describe the muon
chambers, no further information is needed to generate the graphical description of the chambers or of the tree data structure.

Figure 20  The GEANT3 data structure representing the L3 muon chambers.

Having satisfied ourselves that the logical relationships between the components of our detector are correct, it may also be a good idea to check that the right dimensions were coded. The 'DSPEC' command draws a 'specification' sheet for any component of the geometrical structure, such as the L3 inner muon chamber MBI shown in Fig. 21 or the more complicated hadron calorimeter barrel structure shown in Fig. 22.

As a real example of debugging using graphics look at Fig. 23. This figure shows a 'zoom' in on one corner of one of the 144 trapezoidal modules which make up the L3 hadron calorimeter. The individual brass tubes and uranium or stainless steel plates
Figure 21  The GEANT3 ‘Specification Sheet’ for an L3 inner muon chamber.

Figure 22  The GEANT3 ‘Specification Sheet’ for the L3 hadron calorimeter barrel.
are clearly visible. When first coded, the module description included an error which made one of the stainless steel plates extend just outside the module's case. The error was clearly visible on such a graphical display.

![Figure 23 A corner of an L3 hadron calorimeter module.](image)

The examples shown so far were mainly two dimensional projections. When probing the performance of the hardware or reconstruction software of a three dimensional detector, it is natural to try to use three dimensional displays. To get three dimensional information into the brain of a physicist it is usually sufficient to display an image on a high resolution screen (e.g. 1280 x 1024) and allow the physicist to rotate or wobble the three dimensional object. High performance workstations from several manufacturers have the hardware capabilities needed for these displays; acquiring the workstations only costs money. Much more challenging is the production of three dimensional displays which convey useful information. As an example of the problems compare Figs. 24a and 24b. Figure 24a shows GEANT3’s two dimensional view of an L3 muon chamber ‘octant’, one of 16 seven tonne structures each supporting 5 muon chambers. The display looks simple enough. Of course, GEANT3 knows all about the three dimensional structure of this octant, so why not rotate it and really understand the structure? Figure 24b shows a rotated view, but far from being clearer, an excess of (already existing) detailed information has made the picture unintelligible.
Figure 24  Views of an L3 muon chamber ‘octant’: a) end-on, b) rotated.

As a general principle, three dimensional graphics require a completely fresh approach; attempts to generalise the displays that work well in two dimensions are rarely satisfactory. Several experiments have found three dimensional displays a vital component of their physics analysis, and experience with prototype three dimensional displays for L3 has already shown their value. Unfortunately, reproducing 3D colour as a black and white drawing on paper is no way to convince you of this. I will limit myself to the single example of Fig. 25 which shows a composite display of an event in the L3 detector. The four quadrants of the display include three 3D views and one 2D presentation of muon chamber residuals. reconstructed

3.5 PAW: The Physics Analysis Workstation

In spite of its name, PAW [4] is purely a software project. The main aim of PAW is to provide a physics analysis environment which is portable, but can make effective use of the facilities offered by modern workstations. PAW can also run in a mainframe + dumb terminal environment, but the prospective users of such systems should be aware that almost all PAW development starts on workstations.

The components of the PAW system are shown in Fig. 26. The observant will notice that ZEBRA has already been mentioned in these lectures, and the old will notice that HBOOK and H PLOT have been around (in earlier versions) for many years. Thus PAW
Figure 25  A composite display of an event in the L3 detector.  
Three of the four display windows show 3-dimensional views.

attempts to collect existing tools into a coherent environment, and to add to them extra utilities to make the environment complete.

The components in Fig. 26 merit individual descriptions:

- **KUIP**: The Kernel User Interface Package [5] is a new package providing the user interface to PAW. Like all the components of the PAW environment, it is designed to be used alone if appropriate. Thus interactive control of the L3 reconstruction program is now provided by KUIP, but we do not consider that the reconstruction is under the control of PAW. The reason why KUIP can be used outside PAW is that it allows complete flexibility in command definitions. Commands are organised in a hierarchical structure. For example

**HISTOGRAM/FIT/GAUS**
performs a fit of a Gaussian curve to an HBOOK histogram whereas
Figure 26  The components of the PAW system.

L3/REL3/MUCH/MUTK displays numerical information about tracks in the L3 muon chambers found in the REL3 reconstruction program.

Commands are created by typing a ‘Command Definition File’ in a relatively intelligible format. The CDF is then turned into ‘gobbledygook’ Fortran by the KUIP compiler. Figures 27a and 27b show an extract from a CDF file and the corresponding output of the KUIP compiler. Frequently used KUIP commands can be stored in ‘macro files’, and it is even possible to make a macro file from the transcript of a KUIP session in which you have achieved something particularly clever.

- **HBOOK**: The HBOOK histogram package [6] is over ten years old. The original version supported booking, filling and printing (on a line-printer) of histograms and scatter plots. All HBOOK functions are invoked by calls to Fortran subroutines. The latest version of HBOOK (V4) supports a hierarchical directory of histograms and has many additional features of which the most important is
(a) KUIP Command Definition File

> N HISDEF
> Menu HISTOGRAM
> Guidance
> Manipulation of histograms, ntuples.
> Interface to the HBOOK package
> Command LIST
> Parameters
> +
> CHOPT 'Options' C D=' ', 'R=' ', 'I'
> Guidance
> List the histograms in the current directory (memory or disk).
> Histograms are all HBOOK objects including ntuples.
> If CHOPT='I' a verbose format is used (batch: HINDEX).
> Action PAHIST

(b) Output of the KUIP Compiler

SUBROUTINE HISDEF
PARAMETER (MGUIDL=99)
CHARACTER*80 GUID
COMMON /KCGUID/ GUID(MGUIDL)
EXTERNAL PAHIST
CALL KUNWG( 19)
CALL KUCMD(' ','HISTOGRAM','C')
GUID( 1)='Manipulation of histograms, ntuples.'
GUID( 2)='Interface to the HBOOK package'
CALL KUGUID('HISTOGRAM',GUID, 2,'S')
CALL KUCMD('HISTOGRAM',' ','SW')
CALL KUNWG( 46)
CALL KUCMD(' ','LIST','C')
CALL KUNDPV(  1,  1,  1,  1,  1)
CALL KUPAR('LIST','CHOPT','Options','CO','S')
CALL KUFPVAL('LIST','CHOPT',0,0.,' ',D')
CALL KUFPVAL('LIST','CHOPT',0,0.,'I','V')
GUID( 1)='List the histograms in the current directory'//
+ 'tory (memory or disk).'
GUID( 2)='Histories are all HBOOK objects including'//
+ 'ng ntuples.'
GUID( 3)='If CHOPT='I' a verbose format is used'//
+ '(batch: HINDEX).'
CALL KUGUID('LIST',GUID, 3,'S')
CALL KUACT('LIST',PAHIST)

Figure 27 Source and compiled versions of a KUIP command definition file.

probably the support for 'N-tuples'. An N-tuple can be regarded as a simple fixed-format event-file, in which N words are used to store the key quantities describing an event. Of course, this representation is only appropriate at the final stages of
physics analysis, and it is very important that a physicist can change his mind about what should be put into the N-tuples several times a day. Having created an N-tuple file, HBOOK and other components of the PAW system provide tools for making a wide variety of displays based on the N-tuple components.

- **HPlot**: The HPlot [7] package, allowing the graphical (as opposed to line-printer) display of HBOOK histograms, also has its origins in the mists of time. The latest version (V5) is fully integrated with the other components of the PAW environment.

- **HIGZ**: The High Level Interface to Graphics and Zebra [8] is an attempt to present users with a stable interface to the graphics jungle. In spite of (or because of) the many graphics standards (GKS-2D, GKS-3D, PHIGS, ....) the way in which graphical information can be stored and displayed still varies widely. HIGZ offers a GKS-like interface to higher level programs, together with some additional 'macroprimitives' to simplify the drawing of commonly needed objects such as axes and boxes. HIGZ can also store pictures in an RZ-based picture database. The graphics functions are then implemented on top of various graphics packages which may either be standards, or efficient manufacturer-specific products.

The original HIGZ within the PAW system only handled two dimensional graphics. Recent extensions to support three-dimensions have made it appropriate to use HIGZ as a general tool to ensure the portability of all programs using graphics.

- **COMIS**: The Compilation and Interpretation System [9] is a Fortran 77 interpreter which can be invoked from within PAW. Over the last 20 or 30 years, several attempts have been made to create 'standard' physics analysis software where a few simple instructions on data cards would produce the desired histograms. These systems usually became more and more complex before being abandoned in favour of the flexibility of making cuts and calculating quantities in pure Fortran. The problem with pure Fortran is that a simple change to the code requires re-compilation of the code, and re-linking the program which can take a total of minutes or hours. In the PAW environment a command like

```
NTUPLE/PLT 30.X
```

will draw a histogram of variable 'X' in N-tuple file number 30.

```
NTUPLE/PLT 30.X SELECT.FOR
```

will produce the same plot but subject also to the selections made by subroutine SELECT.FOR which is interpreted in real time by COMIS. If SELECT.FOR is not doing the right thing, it can be edited from within PAW and the command can be re-issued.

- **ZEBRA**: I will not start to explain ZEBRA again. The only point I want to make here is that ZEBRA provides the data structure and file management for KUIP, and for the latest versions of HBOOK and HIGZ.
Finally I should show you an example of PAW running on a workstation. Figure 28 shows an Apollo workstation screen during a PAW session in which the performance of the L3 muon reconstruction was under study.

![Typical PAW session on an Apollo workstation](image)

**Figure 28** Typical PAW session on an Apollo workstation.

4. HEP Data Analysis in the 1990s

Up to this point I have examined the more forward-looking techniques currently employed in HEP data analysis. Since 1990 is in the far future, the title of this chapter entitles me to explore more idealistic ideas without the constraint of having it all working by next summer.

4.1 HEP Sociology

Figure 29 summarises the problem which we must try to solve. Experimental HEP is an intensely collaborative science. If collaboration also means that everybody has to move to the experimental site, then all intellectual exchanges with university colleagues and university students will be lost. Most people are sure that this loss would be fatal for HEP on both scientific and political grounds.
300 physicists (at CERN, RAL, Caltech, Liverpool, Paris etc.)
All having 1 VAX-equivalent workstations today.
All having 10 VAX equivalent workstations in 1991.

Figure 29 The problem of data management for experimental HEP.

In the ‘good old days’ of bubble chamber physics, the problem was less severe. Analysis of bubble chamber data took years wherever it was done, and universities were not at a disadvantage. For example, I wrote my Ph.D. thesis on a bubble chamber experiment which was analysed, in its entirety, at Cambridge University by myself and one other graduate student.

In modern high statistics experiments, data can and should be analysed within hours of data-taking to ensure that all the money and effort is not being wasted on acquiring un-reconstructable junk. This immediate reconstruction, and the subsequent first look at the physics, can be done most effectively by people at the site where the data are acquired. Of course this is ‘unfair’ to physicists at a remote institute connected by a dial-up modem.

If nothing could be done about this problem, I would not bother to talk about it. However, in the last few years, the technology of network infrastructure has reached
the point at which it is technically feasible to give remote physicists the same access to an experiment's data as that enjoyed by their 'fortunate' colleagues who have to take shifts on the experiment.

4.2 The Hierarchy of Data Sets

Before looking at the possibilities of modern networking, we should understand the sort of data to which physicists need access. Most physics studies, or even technical software developments, do not start from the raw tapes written by data acquisition systems, but from tapes or data sets which are the output of a reconstruction program. The traditional name for these data sets, even if they are huge, is 'Data Summary Tapes' or 'DSTs'.

The names of DSTs are by no means standardised, but most large experiments have DST categories similar to those outlined below. In each case the data volumes I give are those expected after a few years of running a single LEP experiment.

Master DST
This is the output of the reconstruction program. The events will contain not only reconstructed 4-vectors (energy plus three components of momentum) for particles, clusters or jets, but also a more-or-less complete history of the reconstruction process. Many experiments add the original raw data for good measure so that the 'summary tape' is longer than the input tape. A Master DST may amount to 20,000 tapes. It can take minutes or hours to access a single event, and months to process all the events systematically. Consequently a Master DST may wait for months or years (or even for ever) before a new version is produced using better calibrations and reconstruction algorithms.

Clearly a Master DST is an unwieldy object. The data can be made more accessible either by selecting a subset of the events, or by selecting a subset of the data for each event.

Subset DSTs
Different physics studies need different event samples, and some physics studies are much more urgent (and perhaps more interesting) than others. There is no point in repeatedly reading through 10,000,000 events if the only events you look at are the 1,000 with three or more muons. Thus, in addition to writing a Master DST, most experiments write subsets of the events to separate files. Typical subsets for a LEP experiment would be:

- Events with one or more muons,
- Events with two or more muons,
- Events with two or more electrons,
- Events with missing energy transverse to the beam,
- Events with isolated photons,
- Events with many jets or broad jets.

The size a Subset DST could range from 1 to 1000 tapes. The smaller DSTs would be kept on disk – the larger DSTs perhaps in an automatic tape cartridge handler. The smaller, or more important, Subset DSTs may have quite a short life. For example as soon as a better calibration of the electromagnetic detector was available, the electron/photon Subset DSTs might be re-processed.

**Mini DSTs**

In my arbitrary nomenclature, a Mini DST contains less information for each event than the Master DST. Typically the raw data and the detailed reconstruction history are discarded, leaving only a summary of what was found. Normally it is impossible to correct a calibration error, or re-run a reconstruction algorithm using the data on the Mini DST. The Mini DST events may be around 5% of the size of those on the Master DST, and a Mini Subset DST will usually be something that you could carry in your luggage. Nevertheless, the Mini DST for the whole event sample could still be 1000 tapes, and could only be read infrequently.

**Micro DST**

Micro DSTs are likely to contain the N-tuples which can be manipulated by PAW. If a complete LEP data sample ($10^7$ events per experiment) were to be compressed to the size of one tape cartridge (200 megabytes), each event could be represented by only 5 words. Micro DSTs at this level of compression become very specialised. A particular selection of variables might be adequate for one afternoon’s study by one or two physicists, after which a new selection would have to be made from the Mini DST. Part of the challenge of high statistics physics analysis is to achieve the best balance between a large DST (containing everything that might be needed) and a compact DST (which can be read in seconds rather than hours).

### 4.3 Network Access to Data

It would be extremely costly and manpower intensive to ship all Master DSTs, Subset DSTs, Mini DSTs and Micro DSTs by freight to 40 remote institutes. Furthermore, except in the case of the Master DST, the tapes would often be out of date by the time they had cleared customs. Thus if a remote physicist is to be at no disadvantage, he must have network access to much of these data. To give an idea of what the network requirements are, Fig. 30 shows how long a physicist sitting at a remote workstation
would have to wait for a 200 kilobyte event if he had access to various levels of network performance. The hypothetical network links range from the 9.6 kilobits/sec. ‘phone line’, which many physicists would, even now, consider a luxury, to the 100 (or more) megabits/sec which can easily be carried by a 10 micron thick glass fibre. With 2 megabits/sec to himself, or with a share of a higher speed network, the remote physicist ceases to care where the data are located.

<table>
<thead>
<tr>
<th>Line Speed</th>
<th>Time to Transmit one 200 kbyte event</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dedicated</td>
</tr>
<tr>
<td>9.6 k bits/sec.</td>
<td>3 mins.</td>
</tr>
<tr>
<td>64 k bits/sec.</td>
<td>30 secs.</td>
</tr>
<tr>
<td>2 M bits/sec.</td>
<td>1 sec.</td>
</tr>
<tr>
<td>10 M bits/sec.</td>
<td>200 msec.</td>
</tr>
<tr>
<td>100 M bits/sec.</td>
<td>20 msec.</td>
</tr>
</tbody>
</table>

Remote Access becomes more appropriate than file-transfer or air-freight.

*Figure 30* Access times for a 200 kilobyte event at different levels of network performance.

These technically possible network speeds bring other benefits. In a collaboration linked by multi-megabit lines it becomes possible to use computing and data handling resources wherever they are located without compromising the speed and flexibility of the physics analysis.
4.4 Optimising Remote Access to Data

Figure 31 shows a simple approach to remote data access. A physicist runs a program on his workstation using data obtained from a remote data handling system which is probably located at the experimental site. If the task involves several minutes of computing for each event, then this approach is optimal. On the contrary, if the 'event selection' rejects 99% of the events after looking at one bit in the event header, then this approach is very inefficient.

![Diagram showing the simple approach to remote data access](image)

*Figure 31* The simple approach to remote data access.

A remote access strategy which optimises performance given finite resources must also include the more difficult approaches shown in Figs. 32a and 32b. The physicist's application must be split into local and remote parts so that the communication load is minimised and the performance is consequently maximised. This sort of splitting is best thought of from the very beginning, so that physics analysis programs are constructed as a set of communicating modules and any module can run at either end of the network.

Figures 31 and 32 are still over-simplifications. In reality, there may be several data handling centres accessible over the network, and overall performance will be optimised if copies of frequently accessed data migrate automatically to a center which is close to the remote physicist. It is hard to say which will be most difficult to achieve, the technically possible but largely unfunded network infrastructure, or the software for distributed data management which is a technical challenge and will also require close cooperation between autonomous HEP computing centres. Those of us who believe that large HEP experiments will be exciting and rewarding in the next decades are committed to a solution of these problems.
Figure 32 More difficult approaches to remote data access involving split application programs.

REFERENCES


