CERN 2000–001
29 February 2000

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

1999 CERN SCHOOL OF COMPUTING

Stare Jabłonki, Poland
12–25 September 1999

PROCEEDINGS

Editor: C.E. Vandoni

GENEVA
2000

ISSN 0304–2898
ISBN 92–9083–161–8

CERN 2000–001
29 February 2000

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

1999 CERN SCHOOL OF COMPUTING

Stare Jabłonki, Poland
12–25 September 1999

PROCEEDINGS

Editor: C.E. Vandoni

GENEVA
2000

CERN–Service d'information scientifique–RD/998–1500–février 2000

Abstract

The programme of the School was arranged around four themes.

The theme "Advanced Topics" explored the field of Quantum Technologies: quantum computing and quantum cryptography. Quantum computation is a new computational paradigm invented by Richard Feynman and others in the early 1980s in which numbers would be represented by quantum mechanical states of some suitable atomic-scale systems. This idea brings a new feature to computation: the ability to compute with quantum mechanical superpositions of numbers. For certain types of computational problems this new feature would make quantum computation very much more efficient than classical computation.

Other lectures introduced some of the technologies that could be used for storing and managing the many PetaBytes of data that will be collected and processed at the Large Hadron Collider (LHC) accelerator, which is scheduled to begin operation at CERN in 2005. The state of the art of the current mainline hardware technologies was described, with a discussion of the likely evolution during the next five years. The notes also introduced some more exotic techniques, and discussed the storage capacity and performance requirements of the experiments which will use the LHC accelerator.

In addition, there were presentations of the results of the CERN investigation and survey of the market offering and of the technology evolution in the field of System Managed Mass Storage Systems, including a summary of the practical experience with CERN developed systems (SHIFT and STAGE), the HPSS activity, a survey of existing alternatives (Nstore, Eurostore, SHIFT++ etc.).

The design of a trigger and data acquisition system for a general-purpose experiment at the Large Hadron Collider poses a challenge that has no precedent in the history of experimental physics. The physics requirements, the detector characteristics and the high collision rate expected at LHC luminosities of 10**32 to 10**34 cm-2 s-1 inherently constrain many aspects of the architecture of a high-efficiency data acquisition system. The detector signals must be amplified, shaped and eventually digitised. The analogue or digital information for each channel must be held in local buffers during the decision time of the event selection system, operating at the bunch crossing frequency of 40 MHz. Then the data fragments must be synchronised, collected and compressed to form a full event while the rate of storable events is reduced by subsequent trigger levels. Telecommunication and computing networks will be extensively used to interconnect the signal digitizers to a large complex of computing elements used to analyse event data and select the collisions of physical interest. The course introduced the requirements and the basic concepts of trigger and readout systems at LHC experiments

The goal of the "Software Building" track was to combine exposure to software engineering principles coupled to the software technologies and packages that are relevant for LHC experiments. It is also to give students a taste of working on large software projects that are typical of LHC experiments. The idea is to use the software engineering and LHC++ lectures as a framework with a single case study, based on LHC++, for all phases of the software process. The lectures explained the different phases of the software process, components of the LHC++ software suite and the exercises use an LHC++ application as a case study.

The "Internet Software Technologies" track explored the general field of the Software-based technologies in use or planned over Internets and Intranets. The track was composed of three distinct though connected topics: Distributed Computing using Agents, Transaction Technologies, and Advanced Web Software Topics. The first course focused on a promising

technique for supporting distributed computing: the use of agents written in Java. This method is applied to the specific field of Distributed Physics Analysis. The second course "Web-based Transaction Technologies" described the mechanisms and techniques for supporting client-server interactions based on web forms. It started with a presentation of the HTML language, then carried on with scripting languages (JavaScript) and the CGI interface. The third course was devoted to a selection of more advanced web-based software topics. This included a presentation of the Cascaded Style Sheet (CSF) system and the Dynamic HTML (DHTML). The course also addressed the XML language as well as the SMIL language for the support of synchronised multimedia documents.


Hotel Anders

Preface

The 22nd CERN School of Computing took place at the Hotel Anders, Stare Jabłonki, Poland from 12-25 September 1999. The School was organised in collaboration with the Faculty of Physics, Warsaw University and the Department "Internet for Schools" of the Foundation in Support of Local Democracy.

Thirteen lecturers, of which seven were from CERN, one from the USA, and five from Europe, were invited to give courses at the School. Forty-five students (coming from 34 institutes, 21 countries and of 22 nationalities) attended the School of which 13 in total were funded by UNESCO; of these 13, 12 also received funding for their travel. Two of the funded students (Malkov, Bordeanu) were also partially sponsored by the "Fundacja Popierania Nauki, Kasa im. Jozefa Mianowskiego".

The programme consisted of 37.50 hours of lectures (including 2 evening lectures) and 18 hours of exercises. There were 13 lecturers including evening lecturers. One student session of 1/2 hour was organised on the last Friday morning.

Two evening lectures took place. The first "Feynman and Computation" was given by A. Hey. The second was given by A. Wróblewski and was entitled "Physics in the 1900's".

The programme of the Schools was organised round four themes:

- Internet Software Technologies
- Software Building
- LHC Experiments Data Communication & Data Processing Systems
- Advanced Topics

The School was opened in the presence of:

Prof. Jerzy Niewodniczanski, Chairman, National Atomic Energy Agency,

Prof. Hans Hoffmann, Director of Technology and Scientific Computing, CERN,

Prof. Ryszard Sosnowski, Polish Delegate to CERN,

Dr. Adam Soltan, Director, Department of International Cooperation and European Integration, National Atomic Energy Agency,

Dr. Jacek Gajewski, Internet for Schools, and University of Warsaw,

Sergio Cittolin, and Robert Jones, Members of the Advisory Committee -CERN.

In the absence of C.E. Vandoni, R. Jones and J. Gajewski acted as co-Directors of the School. The following members of the Advisory Organising Committee attended the School for a few days at various times: F. Etienne (Chairman), F. Flückiger, A.J.G. Hey and S. Cittolin (who were also lecturers). Prof. K. Chatasińska-Macukow, Dean of the Physics Faculty, Warsaw University and Prof. M. Kicinska, Deputy Dean of the Physics Faculty, Warsaw University, together with Prof. A. Wróblewski, member of the CERN Council and former Rector of Warsaw University (evening lecturer) visited the School on the last day.

F. Collin of the CERN, IT-PDP group, and A. Pacheco of IFAE Barcelona were the System Managers of the CSC computer centre and their extremely hard work and efficient management, both before, during and after the School was a major contributing factor in the success of the School. The computing and peripheral equipment was provided by CERN. Our Polish colleagues provided the network connection from Stare Jabłonki to CERN and the following Polish technicians were on site, either partially or full-time: Lukasz Blecki (part-time), M. Kacprzak (part-time), F. Kosla, D. Techmanski. A. Skoczylas was also on-site as an additional aide. Miss Sylwia Rudnik gave secretarial assistance.

C. Markou and K. Zachariadou, members of the Local Organising Committee for the 2000 School Committee, attended for one week each, so as to obtain a better idea of the organisation and setting up of a CERN Computing School.

The Internet connectivity for CSC'99 was sponsored by the Polish State Committee for Scientific Research, grant KBN 115/E/343/S/99

For the first time in the history of the Schools, a subset of the lectures were "webcasted", and made available on the CERN Web Site (http://webcast.cern.ch/Projects/CSC99/).

Very special thanks must go to the lecturers for the enormous task of preparing, presenting and writing up their courses.

Both G.V. Frigo and M. Ruggier are warmly thanked; G.V. Frigo for having designed an excellent poster and M. Ruggier for his work in producing the CSC Web page.

We express our gratitude to our secretary and administrator, J. Franco-Turner, not only for the efforts made during the preparation of the School, but also for her invaluable help in preparing these Proceedings.

The participation of so many people coming from many different parts of the world, is a convincing proof of the usefulness and success of this School. We should also thank the students for their enthusiastic and conscientious participation in the life of the School in all its aspects.

EDITOR'S NOTE

To his great regret, the Editor did not succeed in obtaining written contributions from a number of lecturers, in spite of repeated efforts.

However, a number of presentations in electronic form, and, where applicable, some material used for the exercises is available on the web, under

http://www.cern.ch/CSC

A CD-ROM containing a compilation of this material is available on request.

Advisory Committee

W. Carena, CERN, Geneva

S. Cittolin, CERN, Geneva

M. Delfino, CERN, Geneva

F. Etienne, CPPM, Marseille

F. Flückiger, CERN, Geneva

J. Franco-Turner, CERN, Geneva

L.O. Hertzberger, University of Amsterdam, Amsterdam

A.J.G. Hey, University of Southampton, Southampton

R.G. Jacobsen, University of California, Berkeley

R. Jones, CERN, Geneva

G. Kellner, CERN, Geneva,

P. Palazzi, CERN, Geneva

C. Vandoni, CERN, Geneva

D.O. Williams, CERN, Geneva

Local Organising Committee

Z. Ajduk, Warsaw University, Warsaw

K. Doroba, Warsaw University, Warsaw

M. Krzyżanowski, Dept. "Internet for Schools" FRDL, Warsaw

J. Gajewski, Warsaw University

S. Rudnik, Dept. "Internet for Schools" FRDL, Warsaw)

R. Sosnowski, Institute for Nuclear Studies, Warsaw

A. Wróblewski, Warsaw University, Warsaw

Lecturers

S. Cittolin, CERN, Geneva, Switzerland

M. Dönszelmann, CERN, Geneva, Switzerland

A. Dunlop, Open Text AG, St. Gallen, Switzerland

B. Ferrero-Merlino, CERN, Geneva, Switzerland

F. Gagliardi, CERN, Geneva, Switzerland

A.J.G. Hey, University of Southampton, Southampton, U.K.

B. Ibrahim, University of Geneva, Geneva, Switzerland

R. Jones, CERN, Geneva, Switzerland

M. Nowak, CERN, Geneva, Switzerland

M. Podgorny, Syracuse University, Syracuse, USA

L. Robertson, CERN, Geneva, Switzerland

Z. Szkutnik, University of Mining and Metallurgy, Cracow, Poland

A. Wróblewski, University of Warsaw, Warsaw, Poland

Contents

EVENING LECTURE

NOTES ON QUANTUM COMPUTING AND RELATED TOPICS

D.A. Ross and A.J.G. Hey
Quantum Technology Group, University of Southampton, Southampton, SO17 1BJ

Abstract

These notes are intended as a simple introduction to the new field of quantum computing, quantum information theory and quantum cryptography. Undergraduate level quantum mechanics and mathematics is required for an understanding of these lectures. After an introduction to qubits and quantum registers, we introduce the key topics of entangled states and quantum logic gates. For two qubit states, we introduce the four Bell states as a change of basis. The essentials of quantum cryptography are then described, although this is just a straightforward application of quantum mechanics. The characters of Alice, Bob and Eve are first introduced here. Two qubit Bell states are used to demonstrate a novel 'dense coding' technique. Finally, in these communication applications, quantum teleportation is explained in detail, again making use of entangled Bell states. The technique of magnetic spin resonance is used as a familiar example to illustrate how qubit operations could in principle be realised. This leads on to the specification of quantum devices that can encode functions. All this is preparatory to a detailed discussion of two of the most significant quantum algorithms discovered to date, namely, Peter Shor's factorization algorithm and Lov Grover's quantum database search algorithm.

1. INTRODUCTION

The basic unit of a classical computer is a bit. This is a device that can be in one of two states. Usually this is a wire which is in the state $|1>$ if the wire carries a voltage and $|0>$ if it does not (more precisely the two states are distinguished by the electrode having a high or low voltage respectively). Thus such a bit can carry one binary digit, the two states representing the numbers 0 and 1. By assembling L such bits one can store numbers from 0 to $2^L - 1$. The memory of a modern computer contains of the order of 10^9 bits and the disk storage contains of the order of 10^{11} bits.

In early computers a memory device to store a bit consisted of a small toroid of ferromagnetic material with an electric coil wrapped around it. If the bit was "set" (i.e. in the state representing the number 1) then a current passed through the coil and the toroid produced a magnetic field. For the state representing 0 there was no current and consequently no magnetic field. Clearly the total number of such bits was limited by constraints of both size and cost and computer with more than 10^6 bits were rare.

Since then we have seen the revolution in semiconductor technology and a great deal of effort has been put into reducing the size and costs of these binary bits. Nowadays a flat microchip with a surface area of order 1 cm^2 can hold of the order of 10^8 bits. The small size of these memory chips has also had the effect of speeding up the rate at which computers can run; essentially this is because the electromagnetic signal has less distance to travel between components.

The original motivation for imagining a "quantum computer" was based on pushing these improvements in technology to their physical limit. The smallest device one can imagine, that can exist in two states, is a single electron which has the property of spin whose component in a given direction (usually taken to be the $z-$direction) can take one of two values, $\pm\frac{1}{2}\hbar$. We could take these two states to represent the two states of a binary bit. The spin of an electron can be flipped by the application of an oscillating magnetic field with the correct (resonant) frequency, and can in principle be measured by

applying a constant magnetic field in the z direction and observing the energy change. If this were a single outer electron of a molecule that represented one lattice point on the surface of a crystal, a surface area of order 1 cm^2 could hold of order 10^{16} such bits. The difficulty, of course, is that to store and read different numbers we would need to be able to apply or measure magnetic fields that differentiated between two neighbouring spins which were only 10^{-8} cm apart.

There is, however, an important qualitative difference between a classical bit, which is an electronic component, and a quantum bit, such as the spin of an electron. Whereas a classical bit can only be in one of two possible states (high or low voltage) and must be in one of these two states, a quantum bit need not be in one or other of the two allowed states but can in general be in any linear superposition of these states. The electron does not have to be in an eigenstate of the z component of spin, for which the value is definitely either $+\frac{1}{2}\hbar$ or $-\frac{1}{2}\hbar$, but in a linear superposition of these. For a register of L such quantum bits this gives us the opportunity of storing all possible numbers between 0 and $2^L - 1$, *simultaneously* and performing operations on these numbers and storing the result of applying such operations on all arguments simultaneously. The difficulty now arises of how to project from this linear superposition the particular value that we are interested in. This is where algorithms for quantum computing are used and there are cases in which these algorithms can significantly enhance the rate at which a computation can be performed. One particular example of this is a database search for which the time taken to carry out the search grows linearly with the size of the database if classical computational algorithms are used, but only as the square root of the size of the database if a quantum algorithm is used on an initial quantum state, which consists of a superposition of the entire database. The database in question must be a quantum version of the classical database.

The practical difficulties in constructing such quantum computers are enormous. So far the various algorithms have only been carried out on samples of at most two or three quantum bits. Nevertheless a theoretical study of the potential power of a quantum computers is a worthwhile enterprise, albeit in anticipation of significant improvement in the required engineering techniques.

2. DEFINITIONS ETC.

a. **qubit:**
A qubit is a quantum system which can be in one of two states. We shall think of these as spin-$\frac{1}{2}$ particles, the two states being two eigenstates of S_z, although it is likely that in practice a photon will be used, the two states being the state of polarization (horizontal or vertical) with respect to some chosen axis. The qubit can take 2-values - 0 or 1, which are associated with the two eigenstates as follows:

$$|1> \equiv \ |\uparrow>$$

$$|0> \equiv \ |\downarrow>$$

In general a qubit can be in a superposition of these two states with complex coefficients α and β,

$$\alpha|0> + \beta|1>, \quad (|\alpha|^2 + |\beta|^2 = 1)$$

and it is this property that distinguishes them from classical bits used in conventional computers. In mathematical terms, we say that since the general state of a qubit can be a superposition of the two pure states, with arbitrary complex coefficients, then the state a described as a vector in the two dimensional complex space $\mathcal{C}^2$.

b. **L-bit register:** A register is a set of L qubits. Such a register can be used to store an integer number, J, between 0 and $2^L - 1$. The state of the register is denoted by this number, e.g.

$$|J> \equiv \ |\uparrow\uparrow\downarrow\uparrow\downarrow \cdots \downarrow\downarrow> \ .$$

For example in the case of a 2-bit register

$$|0> \equiv |\downarrow\downarrow>$$
$$|1> \equiv |\downarrow\uparrow>$$
$$|2> \equiv |\uparrow\downarrow>$$
$$|3> \equiv |\uparrow\uparrow>$$

Once again, a register can be in a superposition of states

$$|\psi> \equiv \sum_{J=0}^{2^L-1} a_J |J> .$$

The interpretation of the (complex) coefficients a_J is that $|a_J|^2$ is the probability that a measurement of the state of the system will yield the value J. Clearly by conservation of probability we have

$$\sum_{J=0}^{2^L-1} |a_J|^2 = 1.$$

Such states are also known as "coherent" states.
In mathematical terms, the state of an L qubit register is a vector in a space which is the outer product $C^2 \otimes C^2 \cdots \otimes C^2$, one for each of the L qubits.

c. **Entangled pair:**
This is a pair of qubits which is in a superposition of eigenstates of S_z i.e. some superposition of the states $|0>$, $|1>$, $|2>$, $|3>$ defined above, in such a way that the state *cannot* be written as the product of states for each qubit.
Thus, for example the state

$$\frac{1}{2}(|0> + |1> + |2> + |3>)$$

is *not* an entangled pair, since it can be written as

$$\frac{1}{2}(|\downarrow> + |\uparrow>) \otimes (|\downarrow> + |\uparrow>),$$

whereas the state

$$\frac{1}{\sqrt{2}}(|0> + |3>) = \frac{1}{\sqrt{2}}(|\downarrow\downarrow> + |\uparrow\uparrow>)$$

is an example of an entangled pair. In general a superposition (with coefficients a_i, $i = 0 \cdots 3$)

$$a_0|0> + a_1|1> + a_2|2> + a_3|3>$$

is an entangled state *unless*

$$\det \begin{vmatrix} a_0 & a_1 \\ a_2 & a_3 \end{vmatrix} = 0.$$

A specific example of entangled pairs occurs in the total spin of multi-electron atoms. In the case of He, for example, the two electrons can be in a total spin state $S = 1$ with three allowed values for the $z-$component of spin, $S_z = -1$, 0, 1, or in a total spin state $S = 0$. In terms of the individual spins of the two electrons these are given by

$$|\downarrow\downarrow>, \quad S = 1,\ S_z = -1$$
$$|\uparrow\uparrow>, \quad S = 1,\ S_z = 1$$
$$\frac{1}{\sqrt{2}}(|\downarrow\uparrow> + |\uparrow\downarrow>), \quad S = 1,\ S_z = 0$$
$$\frac{1}{\sqrt{2}}(|\downarrow\uparrow> - |\uparrow\downarrow>), \quad S = 0,\ S_z = 0.$$

The two states $S = 1, S_z = 0$ and $S = 0, S_z = 0$ are examples of entangled states.
The concept of entangling can easily be extended to L qubits. The state is entangled *unless* it can be written as a product of states for each of the L qubits. Regarding the state as a vector in the space $\mathcal{C}^2 \otimes \mathcal{C}^2 \cdots \otimes \mathcal{C}^2$, a state is said to be entangled if it *cannot* be expressed as a single outer product of vectors in each $\mathcal{C}^2$ space, but only as a linear superposition of such outer products (known as a "tensor product").

d. **Unitary transformations:**
A Unitary transformation is a transformation which takes the superposition (coherent) state of L qubits

$$\sum_{J=0}^{2^L-1} a_J |J>$$

to

$$\sum_{J=0}^{2^L-1} a'_J |J>,$$

where

$$a'_J = \sum_{K=0}^{2^L-1} U_J^K a_K,$$

the matrix $\mathbf{U}$ being unitary

$$\mathbf{U}^\dagger \mathbf{U} = I.$$

From this unitarity property one can show that the new coefficients a'_J also obey the conservation of probability relation

$$\sum_{J=0}^{2^L-1} |a'_J|^2 = 1$$

and so the new state is also a superposition in which the probability of a measurement yielding the value J is $|a'_J|^2$. This is also a coherent state so the unitarity operator preserves the coherence.
A unitary transformation might only act on one qubit, leaving the other qubits in the register alone or alternatively it might act on two or more qubits simultaneously.

Examples of **U**:

The unitary transformations on a single qubit can be written in terms of four matrices, each depending on a single parameter, θ.

- $$\mathbf{u}_x(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & i\sin(\frac{\theta}{2}) \\ i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}.$$

  For a spin-$\frac{1}{2}$ particle, this corresponds to a rotation through angle θ about the $x-$ axis.

- $$\mathbf{u}_y(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2}) \\ -\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}.$$

  For a spin-$\frac{1}{2}$ particle, this corresponds to a rotation through angle θ about the $y-$ axis.

- $$\mathbf{u}_z(\theta) = \begin{pmatrix} e^{i\theta/2} & 0 \\ 0 & e^{-i\theta/2} \end{pmatrix}.$$

For a spin-$\frac{1}{2}$ particle, this corresponds to a rotation through angle θ about the $z-$ axis.

$$\mathbf{u}_0(\theta) = \begin{pmatrix} e^{i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}.$$

This corresponds to multiplication by an overall phase factor. The identity matrix, $\mathbf{I}$, is $\mathbf{u}_0(4\pi)$. Spin representations of the rotation group are double-valued: a rotation by 2π generates an overall minus sign and 4π is required for the identity operation.
A general unitary 2×2 matrix can always be obtained from a product of these transformations.

Now consider 2 qubit states. The unitary matrices $\mathbf{U}$ are 4×4 matrices. For example

- $$\mathbf{U}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

  This flips both qubits. It is a "NOT" gate, denoted by $\mathbf{U}_{NOT}$, i.e.

  $$\mathbf{U}_{NOT}|\uparrow\uparrow> = \mathbf{U}_1|3> = |0> = |\downarrow\downarrow> .$$

- $$\mathbf{U}_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

  This flips qubit 2 only, e.g.

  $$\mathbf{U}_2|\uparrow\uparrow> = \mathbf{U}_2|3> = |2> = |\uparrow\downarrow> .$$

- $$\mathbf{U}_3 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

  This flips qubit 2 if and only if qubit 1 is in the state $|1>$. This is a "controlled NOT gate", and is usually denoted as $\mathbf{C}_{NOT}$, e.g.

  $$\mathbf{C}_{NOT}|\uparrow\uparrow> = \mathbf{C}_{NOT}|3> = |2> = |\uparrow\downarrow>,$$

  but

  $$\mathbf{C}_{NOT}|\downarrow\uparrow> = \mathbf{C}_{NOT}|1> = |1> = |\downarrow\uparrow>,$$

- $$\mathbf{U}_4 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

  This flips qubit 1 if and only if qubit 2 is in the state $|1>$. It is also a controlled NOT gate and we denote it by $\mathbf{C}'_{NOT}$. Thus

  $$\mathbf{C}'_{NOT}|\uparrow\uparrow> = \mathbf{C}_{NOT}|3> = |1> = |\downarrow\uparrow>,$$

  but

  $$\mathbf{C}'_{NOT}|\uparrow\downarrow> = \mathbf{C}_{NOT}|2> = |2> = |\uparrow\downarrow>,$$

- $$\mathbf{U}_5 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

  This interchanges the two qubits and is denoted by $\mathbf{U}_{switch}$. This can be obtained from a combination (product) of $\mathbf{C}'_{NOT}$ and $\mathbf{C}'_{NOT}$, i.e.

$$\mathbf{U}_{switch} = \mathbf{C}_{NOT}\,C'_{NOT}\,C_{NOT}$$

Any L qubit unitary matrix can be constructed out of outer products of these single qubit unitary matrices (or their matrix products). However they will not in general be a single outer product of these 2×2 unitary matrices, but may be a sum of such outer products (this is known as a "tensor product").
In the above examples of 2 qubit unitary operators we have

$$\mathbf{U}_1 = \mathbf{u}_x(\pi) \otimes \mathbf{u}_x(-\pi)$$

$$\mathbf{U}_2 = \mathbf{I} \otimes (\mathbf{u}_0(\pi)\mathbf{u}_x(-\pi))$$

$$\mathbf{C}_{NOT} = \frac{1}{2}\left(\mathbf{I} \otimes \mathbf{I} - (\mathbf{u}_0(\pi)\mathbf{u}_z(-\pi)) \otimes \mathbf{I} + \mathbf{u}_0(\pi) \otimes \mathbf{u}_x(-\pi) + (\mathbf{u}_z(\pi)) \otimes \mathbf{u}_x(-\pi)\right).$$

The last is an example of such a tensor product.
A unitary matrix representing the transformation of an l qubit system is a matrix in the outer product space $C^2 \otimes C^2 \cdots \otimes C^2$. If the transformation acts on each qubit separately then the matrix can be written as an outer product of a (2×2) matrix on each C^2 space for each qubit. If, on the other hand, the transformation involves the interaction between qubits, as is the case for the controlled NOT gate, then the unitary matrix is not a single outer product of matrixes acting on each qubit, but a linear superposition of such outer products.
In general, a physical device can in principle be constructed that performs any of these unitary transformations. In the case of spin-$\frac{1}{2}$ particles we use the techniques of NMR (Nuclear Magnetic Resonance) to illustrate the construction of 'gedanken' devices, as is described later.

e. **Hadamard transformation:**
This is a unitary transformation which acts on each qubit with the matrix

$$\mathbf{u}_H = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}$$

In terms of the fundamental single qubit transformations described above we have

$$\mathbf{u}_H = \mathbf{u}_0(\pi)\mathbf{u}_z(\pi)\mathbf{u}_y(-\pi/2).$$

The L qubit Hadamard transformation is represented by the outer product

$$\mathbf{U}_H = \mathbf{u}_H \otimes \mathbf{u}_H \otimes \mathbf{u}_H \cdots$$

It is often more convenient to use the pseudo-Hadamard transformation represented by the matrix

$$\mathbf{u}_{pH} = \mathbf{u}_y(-\pi/2) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}.$$

Either of these transformations has the effect that it transforms the lowest state $|0>$ into the sum of all states with equal coefficients,

$$\mathbf{U}_H|0> = \sum_{J=0}^{2^L-1} |J>$$

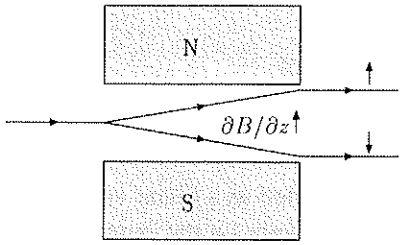


Fig. 1: Stern-Gerlach apparatus. A particle of spin-$\frac{1}{2}$ is passed between the poles of a magnet, which produces a non-uniform magnetic field in the $z-$direction. The particle is displaced upwards or downwards according to the $z-$compoent of its spin being $+\frac{1}{2}$ or $-\frac{1}{2}$ respectively. If the particle is initially in a superposition of these states then this apparatus forces it into one or other of the allowed states.

or

$$\mathbf{U}_{pH}|0>=\sum_{J=0}^{2^L-1}|J>\,.$$

The Hadamard gate is idempotent, i.e. it is equal to its own inverse, whereas this is not the case for the pseudo-Hadamard transformation. On the other hand a pseudo-Hadamard transformation can be achieved by a single rotation about the $y-$axis.

f. **Measurement:**
A measurement of a coherent state is an operation which "collapses" the state into a pure state. For a superposition (coherent state) we have for each value of K (0 to 2^L-1)

$$\sum_{J=0}^{2^L-1}a_J|J>\rightarrow\ |K>,$$

with probability $|a_K|^2$. This operation destroys the coherence of the state by collapsing it into one of the allowed pure states. The operation cannot be described by a simple matrix multiplication.
In the case of spin-$\frac{1}{2}$ particles such a collapse is effected by the simultaneous measurement of the $z-$component of spin of each of the particles. The $z-$component of spin of a single electron, S_z, may be measured using a "Stern-Gerlach" apparatus. The electron is passed through a region of non-uniform magnetic field in the $z-$direction. This causes a displacement of the path of the electron in one of two directions depending on the $z-$component of the spin of the electron (which is proportional to the $z-$component of the magnetic moment of the electron). From this displacement, the $z-$component of the spin can be deduced. If the electron was *not* in a pure eigenstate of S_z but a superposition of such eigenstates, then the operation of passing it through a Stern-Gerlach apparatus forces the electron into one of the two eigenstates of S_z.

g. **Bell states:**
These are four states for a 2 qubit system, which are specific examples of entangled pairs. They are labelled B_0, B_1, B_2, B_3 and may be defined as

$$|B_0>\equiv\ \frac{1}{\sqrt{2}}(|\uparrow\uparrow>+i|\downarrow\downarrow>)\ =\ \frac{1}{\sqrt{2}}(|3>+i|0>)$$

$$|B_1>\equiv\ \frac{1}{\sqrt{2}}(|\downarrow\downarrow>+i|\uparrow\uparrow>)\ =\ \frac{1}{\sqrt{2}}(|0>+i|3>)$$

$$|B_2>\equiv\ \frac{1}{\sqrt{2}}(|\downarrow\uparrow>-i|\uparrow\downarrow>)\ =\ \frac{1}{\sqrt{2}}(|1>-i|2>)$$

$$|B_3>\equiv\ \frac{1}{\sqrt{2}}(|\uparrow\downarrow>-i|\downarrow\uparrow>)\ =\ \frac{1}{\sqrt{2}}(|2>-i|1>)$$

These Bell states form an orthonormal set

$$< B_I|B_J >= \delta_{IJ},\ \ I,J = 0\cdots 3$$

so that any two qubit state can be expanded as a linear sum of Bell states.
They can be obtained by acting respectively on the pure state $|0>$, $|1>$, $|2>$, or $|3>$ with the unitary transformation

$$\mathbf{U}_{Bell} = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 0 & i & 1 \\ 1 & -i & 0 & 0 \\ -i & 1 & 0 & 0 \\ 0 & 0 & 1 & i \end{pmatrix}$$

e.g.

$$|B_J> = \mathbf{U}_{Bell}|J>,\ \ J = 0,\cdots 3$$

It is useful to invert these Bell states, i.e to write the pure states as superpositions of Bell states. This gives

$$|0>=|\downarrow\downarrow>=\frac{1}{\sqrt{2}}\left(|B_1> -i|B_0>\right)$$

$$|1>=|\downarrow\uparrow>=\frac{1}{\sqrt{2}}\left(|B_2> +i|B_3>\right)$$

$$|2>=|\uparrow\downarrow>=\frac{1}{\sqrt{2}}\left(|B_3> +i|B_2>\right)$$

$$|3>=|\uparrow\uparrow>=\frac{1}{\sqrt{2}}\left(|B_0> -i|B_1>\right)$$

These may be written

$$|J> = \sum_{K=0}^{3}\left(U_{Bell}^{-1}\right)_J^K |B_K>$$

Thus a Bell state may be measured by passing it through a device which performs the inverse transformation of $\mathbf{U}_{Bell}$ and then measuring the $z-$components of spin of the two spin-$\frac{1}{2}$ particles. These Bell states have the remarkable property that they can be transformed into each other by transforming *only one* of the two qubits, i.e. the 4×4 transformation matrix which transforms $|B_I>$ into $|B_J>$ is of the form

$$\mathbf{I}\otimes\mathbf{v}_{IJ}.$$

for example

$$\mathbf{v}_{30} = \mathbf{u}_y(-\pi)$$

$$\mathbf{v}_{20} = \mathbf{u}_x(-\pi)$$

$$\mathbf{v}_{10} = \mathbf{u}_z(\pi)$$

$$\mathbf{v}_{00} = \mathbf{I}$$

This property of the Bell states is the crucial property on which applications such as quantum teleportation depend. Entanglement, as expressed in these Bell states, is the essence of the mystery of quantum mechanics. These states embody the non-local, 'faster-than-light' property of quantum mechanics, that Einstein so detested and which the EPR paradox was intended to highlight. Quantum algorithms make essential use of this non-local property to deliver their spectacular improvements over classical algorithms.

3. QUANTUM CRYPTOGRAPHY

This is not really quantum computing but rather the use of quantum mechanics to transmit a key which is only known to the encoder (Alice) and the decoder (Bob). A better name than quantum cryptography would be quantum key distribution since quantum mechanics is used to create a method of cryptographic key distribution which can detect the presence of an eavesdropper (Eve) listening in.

A message N, which can be stored in an L-bit register is encoded with the use of a key K which is also a number between 0 and $2^L - 1$. The encoded message M is simply

$$M = N \oplus K$$

($\oplus$ means exclusive or - XOR).

The decoding is effected by again performing the XOR operation with K

$$M \oplus K = N \oplus K \oplus K = N \oplus 0 = N$$

The key is transmitted from encoder to decoder (or vice versa) by transmitting a large number of qubits (usually one will need at least 2L of these). The qubits are either in one of the two eigenstates of S_z *or* in one of the two eigenstates of S_x. These are chosen at random, but with equal probability by the encoder. For each qubit the encoder, Alice, records the eigenvalue of the qubit as well as the direction of spin (z or x) in which the qubit was an eigenstate. The decoder, Bob, measures either the z-component or the x-component of the spin of each qubit (at random, but with equal probability) and records the result as well as which direction of spin was measured.

In about half the cases Bob will have measured the spin in the same direction as Alice prepared it ("good" qubits). For such qubits Bob will obtain a result for the eigenvalue which is always equal to the eigenvalue corresponding to the eigenstate in which it was transmitted. In the remaining half, in which Bob measured the spin in a different direction from the direction in which they were prepared ("bad" qubits) the result will have equal probability of being equal or opposite to the eigenvalue of the prepared state. These "bad" bits must be discarded, but it is safe to build a key, K, from the remaining "good" qubits.

It is therefore sufficient for Bob to tell Alice (on an open line if necessary) in which direction the spin of each qubit was measured it but not the result. Alice can then tell Bob (again on an open line) which are the "good" qubits and which are the "bad" ones. Although this is public information, no third party can reconstruct the key, since the third party still does not know the eigenvalues of the "good" qubits.

One important feature of this technique is that the presence of an eavesdropper, Eve, can be detected. If Eve intercepts the signal from Alice, she does not know which setting, z or x, that Alice used. She must therefore choose a setting at random and then retransmit this result, using her setting, to Bob. Since Eve will not guess correctly every time, when Alice and Bob first make contact over the phone, they compare not only the settings but the results. If there is an eavesdropper then Alice and Bob will find that there are some "good qubits" on which they disagree. They then know that the security of the quantum channel is compromised. If they find perfect agreement, and can conclude there is no eavesdropper, they can then go ahead and exchange only setting information as described above.

Quantum key distribution, both over optical fibres and in free space, has been successfully demonstrated by a number of different groups.

4. DENSE CODING

This is a technique which can be used to send a message consisting of an integer between 0 and $2^{2L} - 1$, by transmitting L qubits only.

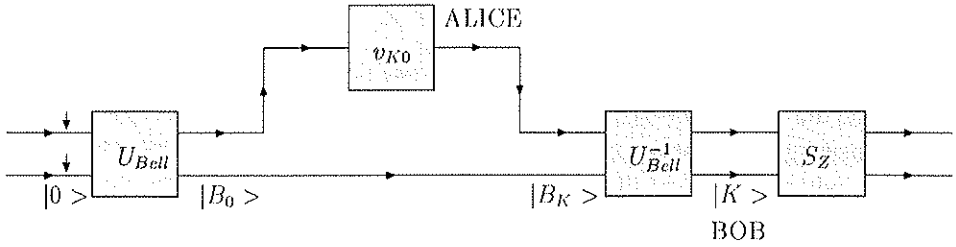


Fig. 2: Alice sends the number K ($K = 0 \cdots 3$) to Bob. Alice takes one particle from an entangled pair in Bell state $|B_0>$, performs the transformation $\mathbf{v}_{K0}$ on it, and sends it to Bob, who then measures the Bell state of the transformed entangled pair.

We consider just one qubit and use it to transfer a number, K, between 0 and 3.

The technique uses the fact that a transformation between Bell states can be effected by acting on one qubit only. Thus the sender, Alice, and receiver, Bob, each take one qubit from a state which is in a well defined Bell state, $|B_J>$. Alice then performs a transformation $\mathbf{v}_{JK}$ on her qubit and transmits it to Bob. Bob then measures the Bell state of the pair of qubits (the qubit that was sent plus the qubit from the original entangled pair) and deduces the value of K between 0 and 3.

As an example we assume that the sender and receiver both receive a qubit from an entangled pair which is in the state $|B_0>$.

The entangled pair starts in the state

$$|B_0> \equiv \frac{1}{\sqrt{2}}(|\uparrow\uparrow> + i|\downarrow\downarrow>)$$

Now Alice performs one of the following unitary transformations on her qubit

$$\mathbf{v}_{00} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

(no operation)

$$\mathbf{v}_{10} = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}$$

(rotation by π about the z-axis)

$$\mathbf{v}_{20} = \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix}$$

(rotation by $-\pi$ about the x-axis)

$$\mathbf{v}_{30} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

(rotation by -π about the y-axis).

The entangled pair is now in the state

$$|\psi_K> = \frac{1}{\sqrt{2}}(|\uparrow\ (\mathbf{v}_{K0}\uparrow)> + i|\downarrow\ (\mathbf{v}_{K0}\downarrow)>)$$

for some value of K between 0 and 3.

Now use

$$\mathbf{v}_{00}|\uparrow\rangle = |\uparrow\rangle$$
$$\mathbf{v}_{00}|\downarrow\rangle = |\downarrow\rangle$$
$$\mathbf{v}_{10}|\uparrow\rangle = i|\uparrow\rangle$$
$$\mathbf{v}_{10}|\downarrow\rangle = -i|\downarrow\rangle$$
$$\mathbf{v}_{20}|\uparrow\rangle = -i|\downarrow\rangle$$
$$\mathbf{v}_{20}|\downarrow\rangle = -i|\uparrow\rangle$$
$$\mathbf{v}_{30}|\uparrow\rangle = |\downarrow\rangle$$
$$\mathbf{v}_{30}|\downarrow\rangle = -|\uparrow\rangle$$

to see that $|\psi_K\rangle$ are once again Bell states, i.e.

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}(|\uparrow\uparrow\rangle + i|\downarrow\downarrow\rangle) = |B_0\rangle$$

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(i|\uparrow\uparrow\rangle + i(-i)|\downarrow\downarrow\rangle) = |B_1\rangle$$

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(-i|\uparrow\downarrow\rangle + i(-i)|\downarrow\uparrow\rangle) = |B_2\rangle$$

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle - i|\downarrow\uparrow\rangle) = |B_3\rangle .$$

After performing one of these unitary operations on her electron, Alice sends the transformed electron to Bob. Bob now measures the new Bell state of the entangled pair and deduces the value of K from the result of that measurement.

5. QUANTUM TELEPORTATION

If a qubit is in a pure eigenstate then one can measure the z-component of spin and communicate the result of the measurement to a recipient. However, if the qubit is in some superposition

$$|\psi\rangle = \alpha|\uparrow\rangle + \beta|\downarrow\rangle,$$

then any measurement of S_z will collapse the state into one of the two pure eigenstates. Thus a superposition state cannot be measured without destroying information about the original state. This result goes by the name of the 'Quantum No Cloning Theorem'.

The theorem is proved as follows:

Suppose that U_c is a unitary cloning operator, such that for any arbitrary quantum state $|\alpha\rangle$,

$$U_c|\alpha\rangle|0\rangle = |\alpha\rangle|\alpha\rangle .$$

Likewise for a different quantum state $|\beta\rangle$ we would have

$$U_c|\beta\rangle|0\rangle = |\beta\rangle|\beta\rangle .$$

Now let $|\psi\rangle$ be another quantum state which is a linear superposition of $|\alpha\rangle$ and $|\beta\rangle$,

$$|\psi\rangle = a|\alpha\rangle + b|\beta\rangle .$$

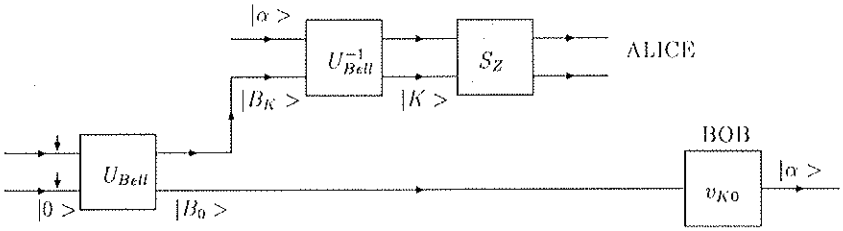


Fig. 3: Alice and Bob each take one qubit from an entangled pair in Bell state $|B_0>$. Alice then measures the Bell state of the entangled pair consisting of the qubit taken from the original Bell state pair and the unmeasured qubit she wishes to teleport to Bob, which is in the state $|\psi>$. She communicates the result of this measurement, K to Bob, who then performs the transformation $\mathbf{v}_{K0}$ on his qubit, thereby tranforming it into the state $|\psi>$

Operating on $|\psi>|0>$ with the cloning operator leads to

$$U_c\,|\psi>|0> = \; a\,|\alpha>|\alpha> \; + b\,|\beta>|\beta> \, .$$

This is *not* the state $|\psi>|\psi>$, which contradicts the postulate that the operator U_c clones *any* arbitrary quantum state.

Now, although, as we have seen, a quantum state (qubit) cannot be copied, we will now show that it can be transported from Alice to Bob, but only at the expense of destroying the original state. The method relies on the same properties of Bell states as the algorithm for dense coding. In the case of dense coding, Alice first performs a transformation on a single qubit of a two qubit Bell state, then sends the transformed qubit to Bob who finally measures the final Bell state of the resulting pair. For quantum teleportation, Alice and Bob again start with one qubit of an entangled Bell state pair. Alice measures the Bell state formed by her qubit and the unknown qubit and then tells Bob which transformation to make on his qubit to regenerate the original unmeasured qubit state.

It is necessary for both Alice and Bob each to take one of two qubits which have been prepared in some Bell state. Again we shall consider the state $|B_0>$ for convenience, although this can easily be generalized. Alice now has two qubits - the qubit in the state $|\psi>$ that she wishes to transport and the qubit obtained from the device that produced the entangled pair in Bell state $|B_0>$.

The three qubit state can therefore be written

$$|\phi> = |\psi> \otimes |B_0> = \; \frac{1}{\sqrt{2}}\,(\alpha|\uparrow\uparrow\uparrow> + i\alpha|\uparrow\downarrow\downarrow> + \beta|\downarrow\uparrow\uparrow> + i\beta|\downarrow\downarrow\downarrow>)$$

For convenience we shall write this as

$$|\phi> = \; \frac{1}{\sqrt{2}}\,(\alpha|\uparrow\uparrow> \otimes |\uparrow> + i\alpha|\uparrow\downarrow> \otimes |\downarrow> + \beta|\downarrow\uparrow> \otimes |\uparrow> + i\beta|\downarrow\downarrow> \otimes |\downarrow>)$$

where we have separated out the third qubit, which is the one taken from the Bell state $|B_0>$ by Bob. After Alice has measured the Bell state of her two qubits, Bob's qubit can be transformed into a 'copy' of the original qubit in the state $|\psi>$ by performing one of the four unitary transformations $\mathbf{v}_{K0}$, $K = 0 \cdots 3$ used in the section above on dense coding. Which of the four unitary transformations needs to be used depends on the result of the Bell state measurement.

To see this, we expand the above expression for $|\phi>$ into a sum of Bell states for the first two qubits, using the expressions for the inversions of the Bell states given below the definition of the Bell states. After collecting terms this gives

$$|\phi> = \frac{1}{2}\left(|B_0>\otimes(\alpha|\uparrow>+\beta|\downarrow>)+|B_1>\otimes(-i\alpha|\uparrow>+i\beta|\downarrow>)\right.$$
$$\left.+|B_2>\otimes(-\alpha|\downarrow>+\beta|\uparrow>)+|B_3>\otimes(i\alpha|\downarrow>+i\beta|\uparrow>)\right)$$

Applying the inverses of the operators $\mathbf{v}_{K0}$, $K = 0\cdots 3$ to the state $|\psi>$, which we wish to teleport, we can see that this may be written

$$|\phi> = \frac{1}{2}\Big(|B_0>\otimes(\mathbf{v}_{00})^{-1}|\psi>+|B_1>\otimes(\mathbf{v}_{10})^{-1}|\psi>$$
$$+|B_2>\otimes(\mathbf{v}_{30})^{-1}|\psi>+|B_3>\otimes(\mathbf{v}_{20})^{-1}|\psi>\Big)$$

A measurement of the Bell state by Alice collapses the wavefunction into one of these components. In particular, it forces the qubit taken by Bob into the state $\mathbf{v}_{K0}^{-1}|\alpha>$ [1]. The result of the measurement tells Alice into which component the wavefunction has collapsed. She then communicates this information to Bob who performs the relevant unitary transformation on his qubit which is then transformed into the required state $|\psi>$.

Note that although the wavefunction collapses immediately upon the measurement of the Bell state by Alice - so that the Bob's qubit is also instantaneously collapsed, the information required to reproduce the initial state $|\psi>$ has to be communicated from the sender to the recipient at a velocity less than or equal to the velocity of light.

6. A 'GEDANKEN REALISATION: MAGNETIC RESONANCE

We start by showing how magnetic resonance can be used to effect the transformations $\mathbf{u}_x$, $\mathbf{u}_y$, $\mathbf{u}_z$ on a single qubit, which is taken to be the spin part of the wavefunction of a spin-$\frac{1}{2}$ particle.

We take the example of $\mathbf{u}_y$ and work (for convenience) in a system of units where $\hbar = 1$.

First we imagine the spin-$\frac{1}{2}$ placed in a uniform magnetic field of magnitude B_0 in the $z-$ direction.

The part of the Hamiltonian that depends on the spin is then given by

$$H = \mu B_0 \mathbf{S}_z,$$

where for a particle of charge e and mass m, and gyromagnetic ratio g (=2 for an electron), the magnetic moment (vector) is given by

$$\underline{\mu} = g\frac{e}{2m}\underline{S},$$

the operators for the components of $\underline{S}$ being represented by the 2×2 matrices

$$\mathbf{S}_x = \frac{1}{2}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix},\ \mathbf{S}_y = \frac{1}{2}\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix},\ \mathbf{S}_z = \frac{1}{2}\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$$

This leads to a ("Zeeman") energy splitting between the two pure states ($S_z = \pm\frac{1}{2}$) with energy difference $\omega_0 = \mu B_0$.

Now we apply an oscillating magnetic field with angular frequency ω_0 and amplitude B', ($B' \ll B$) in the *negative* y-direction, so that the (spin dependent part of the) Hamiltonian becomes

$$H = \mu B\mathbf{S}_z - \mu B'\cos(\omega_0 t)\mathbf{S}_y.$$

[1] This collapse of the state of Bob's qubit due to a measurement performed by Alice is an example of the Einstein-Podolsky-Rosen (EPR) paradox.

We write the time dependent spin-part of the wavefunction as

$$\begin{pmatrix} a(t)e^{-i\omega_0 t/2} \\ b(t)e^{i\omega_0 t/2} \end{pmatrix},$$

where we have displayed explicitly the time dependence of the two states in the absence of the applied oscillating magnetic field. Defining $\omega' = \frac{1}{2}\mu B'$ the Schroedinger equation is

$$i\frac{\partial}{\partial t}\begin{pmatrix} a(t)e^{-i\omega_0 t/2} \\ b(t)e^{i\omega_0 t/2} \end{pmatrix} = \omega_0 \mathbf{S}_z \begin{pmatrix} a(t)e^{-i\omega_0 t/2} \\ b(t)e^{i\omega_0 t/2} \end{pmatrix} - 2\omega' \cos(\omega_0 t)\mathbf{S}_y \begin{pmatrix} a(t)e^{-i\omega_0 t/2} \\ b(t)e^{i\omega_0 t/2} \end{pmatrix},$$

which upon writing $\cos(\omega_0 t) = \frac{1}{2}\left(e^{i\omega_0 t} + e^{-i\omega_0 t}\right)$ and a little algebra simplifies to

$$i\frac{\partial}{\partial t}\begin{pmatrix} a(t) \\ b(t) \end{pmatrix} = -\omega' \mathbf{S}_y \begin{pmatrix} a(t) \\ b(t) \end{pmatrix} - \omega' \mathbf{S}_y \begin{pmatrix} a(t)e^{-2i\omega_0 t} \\ b(t)e^{2i\omega_0 t} \end{pmatrix}.$$

Now we make the approximation that since we shall apply the oscillating field for a time which is large compared with $1/\omega_0$, the last term in the above equation oscillates very rapidly and averages out to a very small quantity over this time interval and may therefore be neglected. We thus end up with

$$i\frac{\partial}{\partial t}\begin{pmatrix} a(t) \\ b(t) \end{pmatrix} = -\omega' \mathbf{S}_y \begin{pmatrix} a(t) \\ b(t) \end{pmatrix}.$$

This is a pair of first order differential equations whose solution is

$$\begin{pmatrix} a(t) \\ b(t) \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{1}{2}\omega' t\right) & \sin\left(\frac{1}{2}\omega' t\right) \\ -\sin\left(\frac{1}{2}\omega' t\right) & \cos\left(\frac{1}{2}\omega' t\right) \end{pmatrix} \begin{pmatrix} a_0 \\ b_0 \end{pmatrix},$$

where a_0, b_0 are the initial values of $a(t)$ and $b(t)$. Thus we see that if we set $\theta = \omega' t$ $(= \frac{1}{2}\mu B' t)$ then this pulse of oscillating magnetic field in the (negative) $y-$ direction effects the transformation represented by the matrix $\mathbf{u}_y(\theta)$. The transformations $\mathbf{u}_x(\theta)$ and $\mathbf{u}_z(\theta)$ are similarly effected by applying the oscillating magnetic fields in the negative $x-$ and $z-$ directions respectively.

In most cases the spin-$\frac{1}{2}$ particle is a nucleus and this method is known as "Nuclear Magnetic Resonance" (NMR).

When there is more than one spin-$\frac{1}{2}$ particle present, they will interact with each other through the magnetic moments associated with their spins. Now, in addition to the energy shifts produced by the applied uniform magnetic field in the $z-$direction, there is a shift which depends in general on the mutual orientation of the various spins in the system, e.g for a two qubit system there will be a contribution to the energy whose sign depends on whether the spins are of the same sign or of opposite sign. It is this contribution to the energy which is used to construct devices which effect transformations on system consisting of more than one qubit and which are not single outer products of transformations on each bit (such as a controlled NOT gate).

Now consider NMR devices which operate on a two qubit system.

Notation:

$\phi_w^{(l)}$ means a pulse which rotates the spin state of qubit l through an angle ϕ about the w-axis. The inverse of this operation is written $\phi_{-w}^{(l)}$. Thus $\phi_w^{(l)}$ is a pulse which performs the transformation $\mathbf{u}_w(\phi)$ on qubit l. Note that in usual NMR notation the angle ϕ is usually quoted in *degrees*.

If $w = z$ then these operators effect a phase change through angle $\phi/2$ with sign depending on the spin of the qubit.

A further operator which effects a phase change is written $\phi^{(12)}$. This is just a time delay in which the state of the two qubits evolves under the influence of the coupling of the mutual spins, which may be written $\lambda S_z^{(1)} S_z^{(2)}$. The time delay occurs for a period $2\phi/\lambda$ such that the phase change of the state is $+\phi/2$ if both the spins have the same z-component and $-\phi/2$ if the two spins have opposite z-component.

In terms of 4×4 matrices for a 2 qubit (bit 1 is the most significant bit and bit 2 is the least significant bit) system these pulses may be represented as

$$\phi_y^{(2)} = \begin{pmatrix} \cos\left(\frac{\phi}{2}\right) & \sin\left(\frac{\phi}{2}\right) & 0 & 0 \\ -\sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\phi}{2}\right) & \sin\left(\frac{\phi}{2}\right) \\ 0 & 0 & -\sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) \end{pmatrix}$$

$$\phi_y^{(1)} = \begin{pmatrix} \cos\left(\frac{\phi}{2}\right) & 0 & \sin\left(\frac{\phi}{2}\right) & 0 \\ 0 & \cos\left(\frac{\phi}{2}\right) & 0 & \sin\left(\frac{\phi}{2}\right) \\ -\sin\left(\frac{\phi}{2}\right) & 0 & \cos\left(\frac{\phi}{2}\right) & 0 \\ 0 & -\sin\left(\frac{\phi}{2}\right) & 0 & \cos\left(\frac{\phi}{2}\right) \end{pmatrix}$$

$$\phi_x^{(2)} = \begin{pmatrix} \cos\left(\frac{\phi}{2}\right) & i\sin\left(\frac{\phi}{2}\right) & 0 & 0 \\ i\sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\phi}{2}\right) & i\sin\left(\frac{\phi}{2}\right) \\ 0 & 0 & i\sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) \end{pmatrix}$$

$$\phi_x^{(1)} = \begin{pmatrix} \cos\left(\frac{\phi}{2}\right) & 0 & i\sin\left(\frac{\phi}{2}\right) & 0 \\ 0 & \cos\left(\frac{\phi}{2}\right) & 0 & i\sin\left(\frac{\phi}{2}\right) \\ i\sin\left(\frac{\phi}{2}\right) & 0 & \cos\left(\frac{\phi}{2}\right) & 0 \\ 0 & i\sin\left(\frac{\phi}{2}\right) & 0 & \cos\left(\frac{\phi}{2}\right) \end{pmatrix}$$

$$\phi_z^{(2)} = \begin{pmatrix} e^{i\phi/2} & & & \\ & e^{-i\phi/2} & & \\ & & e^{i\phi/2} & \\ & & & e^{-i\phi/2} \end{pmatrix}$$

$$\phi_z^{(1)} = \begin{pmatrix} e^{i\phi/2} & & & \\ & e^{i\phi/2} & & \\ & & e^{-i\phi/2} & \\ & & & e^{-i\phi/2} \end{pmatrix}$$

and

$$\phi^{(12)} = \begin{pmatrix} e^{i\phi/2} & & & \\ & e^{-i\phi/2} & & \\ & & e^{-i\phi/2} & \\ & & & e^{i\phi/2} \end{pmatrix}$$

Thus for example a series of pulses which flips the sign of the state $|3>$ but leaves the others unchanged is given (up to an irrelevant overall phase) by

$$90_z^{(2)}\, 90_z^{(1)}\, 90^{(12)} \;=\; e^{-i\pi/4} \begin{pmatrix} e^{i\pi} & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix}$$

A controlled NOT gate, which flips the spin of the least significant qubit if and only if the most significant qubit is set

$$|j> \otimes |k> \rightarrow |j> \otimes |j \oplus k>$$

$$|0> \rightarrow |0>, \quad |1> \rightarrow |1>, \quad |2> \rightarrow |3>, \quad |3> \rightarrow |2>$$

This has a 4×4 matrix representation

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Up to an overall phase, this may be reproduced by the series of pulses (sequence obtained by reading from right to left)

$$C_{NOT} \;=\; 90_{-y}^{(2)}\, 90_z^{(2)}\, 90_{-z}^{(1)}\, 90^{(12)}\, 90_y^{(2)}$$

To see this we first consider the three middle terms

$$\begin{aligned} 90_z^{(2)}\, 90_{-z}^{(1)}\, 90^{(12)} \;&=\; \begin{pmatrix} e^{i\left(-\frac{\pi}{4}+\frac{\pi}{4}+\frac{\pi}{4}\right)} & & & \\ & e^{i\left(-\frac{\pi}{4}-\frac{\pi}{4}-\frac{\pi}{4}\right)} & & \\ & & e^{i\left(+\frac{\pi}{4}+\frac{\pi}{4}-\frac{\pi}{4}\right)} & \\ & & & e^{i\left(+\frac{\pi}{4}-\frac{\pi}{4}+\frac{\pi}{4}\right)} \end{pmatrix} \\ &=\; e^{i\left(\frac{\pi}{4}\right)} \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \end{aligned}$$

and

$$\begin{aligned} 90_{-y}^{(2)} \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} 90_y^{(2)} \;&=\; \frac{1}{2} \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \\ &=\; \frac{1}{2} \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \\ &=\; \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

Thus we see that $C_{NOT} = 90^{(2)}_{-y}\, 90^{(2)}_{z}\, 90^{(1)}_{-z}\, 90^{(12)}\, 90^{(2)}_{y}$ reproduces the required controlled NOT gate up to an overall phase.

Likewise $C'_{NOT} = 90^{(1)}_{-y}\, 90^{(2)}_{-z}\, 90^{(1)}_{z}\, 90^{(12)}\, 90^{(1)}_{y}$ reproduces the other type of controlled NOT in which the most significant bit is flipped if and only if the least significant bit is "set".

Therefore the following combination of pulses will interchange the two qubits

$$U_{switch} = 90^{(2)}_{-y}\, 90^{(2)}_{z}\, 90^{(1)}_{-z}\, 90^{(12)}\, 90^{(2)}_{y} 90^{(1)}_{-y}\, 90^{(2)}_{-z}\, 90^{(1)}_{z}\, 90^{(12)}\, 90^{(1)}_{y} 90^{(2)}_{-y}\, 90^{(2)}_{z}\, 90^{(1)}_{-z}\, 90^{(12)}\, 90^{(2)}_{y}$$

Consider the following combination of pulses

$$U_{Bell} \equiv C'_{NOT} 90^{(1)}_{z} 90^{(2)}_{z} 90^{(12)} 90^{(2)}_{-x} 90^{(2)}_{z} 90^{(1)}_{-z} 90^{(12)}$$

This has a matrix representation

$$\begin{aligned}
\mathbf{U}_{Bell} &= \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \begin{pmatrix} 1 & -i & 0 & 0 \\ -i & 1 & 0 & 0 \\ 0 & 0 & 1 & -i \\ 0 & 0 & -i & 1 \end{pmatrix} \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix}, \\
&= \frac{-i}{\sqrt{2}} \begin{pmatrix} 0 & 0 & i & 1 \\ 1 & -i & 0 & 0 \\ -i & 1 & 0 & 0 \\ 0 & 0 & 1 & i \end{pmatrix}
\end{aligned}$$

which is the matrix (up to an overall phase) that converts pure states into Bell entangled states.

A two qubit Hadamard gate (a device that performs a two qubit Hadamard transformation) can be constructed (up to an overall phase) as

$$H = 90^{(1)}_{y} 90^{(2)}_{y} 180^{(12)}.$$

In terms of the matrix representation

$$\mathbf{H} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} i & & & \\ & -i & & \\ & & -i & \\ & & & i \end{pmatrix} = \frac{i}{2} \begin{pmatrix} 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

7. QUANTUM LOGIC GATES USING MAGNETIC RESONANCE

We consider a two qubit system. A Controlled NOT gate (exclusive or - XOR) is a device into which one sends a pair of qubits in the state

$$|j> \otimes |k>, \quad (j,k = 0,1)$$

and the output state is

$$|j> \otimes |k \oplus j>$$

In general the input states could be superpositions

$$|\psi> = \sum_{j=0}^{1} a_j |j>$$

$$|\phi>=\sum_{k=0}^{1} b_k|k>$$

In this case the device performs the operation

$$|\psi>\otimes|\phi>\rightarrow \sum_{j,k=0}^{1} a_j b_k|j>\otimes|j\oplus k>$$

Such a device could consist of a proton (or other nucleus) trapped at some site in a semiconductor (a "quantum dot") in spin state j with magnetic moment $g_I\mu_N$, and an electron in spin state k trapped at some other site in the semiconductor (or a nucleus with a very much larger magnetic moment). The magnetic moment of the electron is $2\mu_B$, which is much larger than that of the proton by a factor of the ratio of the proton to electron mass. A constant magnetic field B_0 is applied in the z-direction. The proton is the first qubit and the electron is the second qubit There is a mutual interaction between the two magnetic moments, which depends on the distance between the two qubits and the relative orientation of their spins.

The Hamiltonian for the system has a part which is proportional to the applied magnetic field, which we may write as

$$H_{mag} \;=\; B_0\left(g_I\mu_N(j-1/2)+2\mu_B(k-1/2)+(-1)^{(j+k)}\,\lambda\right)$$

Where λ encodes the mutual interaction and is multiplied by a sign which is positive if the spins are aligned and negative otherwise. The energy levels between the two allowed states for the electron differ by

$$B_0\,(2\mu_B+2\lambda) \;\equiv\; \omega_0+\Delta\omega,$$

if $j=1$, and

$$B_0\,(2\mu_B-2\lambda)) \;\equiv \omega_0-\Delta\omega,$$

if $j=0$.

By applying an oscillating magnetic field in the $y-$direction with frequency $\omega_0+\Delta\omega$ and amplitude B' one can induce oscillations in the spin state of the electron *provided the proton is in the state* $j=1$. If the proton is *not* in this state then the probability for inducing transitions is negligibly small. Likewise the probability for inducing transitions in the proton is negligibly small. If this oscillating magnetic field is applied for a time

$$t=\frac{\pi}{\mu_B B'},$$

then the electron will (almost) always flip its spin state.

Thus we have constructed a device which performs the following operations

$$|0>\otimes|0>\rightarrow |0>\otimes|0>$$
$$|0>\otimes|1>\rightarrow |0>\otimes|1>$$
$$|1>\otimes|0>\rightarrow |1>\otimes|1>$$
$$|1>\otimes|1>\rightarrow |1>\otimes|0>$$

We see that the second output qubit contains the exclusive XOR of the two input qubits. Note that this is just a Controlled NOT gate, (C_{NOT}).

Very simply we can construct a NOR gate which performs

$$|j>\otimes|k>\rightarrow\; |j>\otimes|*j\oplus k>,$$

simply by changing the frequency of the oscillating magnetic field to $\omega_0-\Delta\omega$

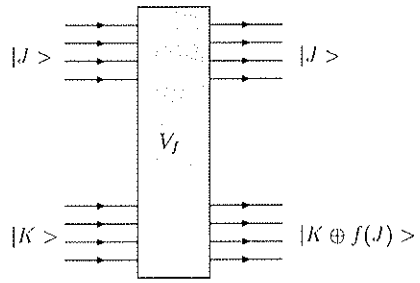


Fig. 4: V_f is a function device which performs the transformation $|J> \otimes |K> \rightarrow |J> \otimes |K \oplus f(J)>$.

8. FUNCTIONS

The device described above can easily be used to produce quantum states that encode functions. In the case of a function $f(j)$ which maps a single qubit onto a single qubit ($j = 0, 1,\ f(j) = 0, 1$) then the applied magnetic field in this device should be set to

$$B' \sum_{j=0}^{1} f(j) e^{i\omega(j)t} \theta(\pi - B'\mu_B t)$$

where

$$\omega_j = \omega_0 - (-1)^j \Delta\omega.$$

This then performs the operation

$$|j> \otimes |k> \rightarrow\ |j> \otimes |k \oplus f(j)> .$$

If k is taken to be 0, then the second bit just contains $f(j)$ at output.

This device is easily extended to a function which maps an integer between 0 and $2^L - 1$ onto a single bit. The first qubit is replaced by an L qubit register known as the "control register". This consists of L protons trapped at different sites on the semiconductor. Now each proton will have a different mutual interaction term with the electron because the distance between the magnetic dipoles is different for each of the protons. The magnetic part of the Hamiltonian now becomes

$$H_{mag} \ = \ B_0 \left(\sum_{l=1}^{L} g_I \mu_N (j_l - 1/2) + 2\mu_B (k - 1/2) + \sum_{l=1}^{L} (-1)^{(j_l + k)} \lambda_l \right)$$

and the energy difference between the two electron states is

$$\omega(J) \ = \ B_0 \left(2\mu_B - \sum_{l=1}^{L} 2\,(-1)^{j_l} \lambda_l \right) \ \equiv \ \omega_0 + \sum_{l=1}^{L} \Delta\omega_J,$$

The input register (the protons) is in the state $|J>$ where

$$J = \sum_{l=1}^{L} j_j 2^{l-1}.$$

Now by applying an oscillating magnetic field

$$B' \sum_{J=0}^{2^L - 1} f(J) e^{i\omega(J)t} \theta(\pi - B'\mu_B t)$$

the device will perform the transformation

$$|J> \otimes |k> \rightarrow\ |J> \otimes |k \oplus f(J)> .$$

Again if we set $k = 0$ initially then the device will return $f(J)$ in the electron qubit ("target qubit").

Generalizing this to a function which maps an integer between 0 and $2^L - 1$ to an integer in the range 0 to $2^{L'} - 1$ presents severe practical difficulties. Now as well as L protons at different sites in the semiconductor we need L' electrons at different sites and we need to be able to access each of these with a different frequency oscillating field.

Writing a function $f(J)$ as

$$f(J) \equiv \sum_{l'=1}^{L'} f_{l'}(J) 2^{l'-1},$$

the oscillating magnetic field applied to the l' electron must be

$$B' \sum_{J=0}^{2^L-1} f_{l'}(J) e^{i\omega(J)t} \theta(\pi - B'\mu_B t).$$

If the "target register" (the L' electrons) are initially in the state K then this device performs the operation

$$|J> \otimes |K> \rightarrow |J> \otimes |K \oplus f(J)> .$$

Setting $K = 0$ thus generates $f(J)$ in the target register.

Note that this is an example of a "reversible gate", i.e. if we pass the output through the apparatus we recover the input.

$$|J> \otimes |K \oplus f(J)> \rightarrow |J> \otimes |K> .$$

This quantum device can produce a quantum state which is a superposition of functions of several inputs. In particular, if we set all the L qubits (protons) of the "control" register to be eigenstates of S_x with eigenvalue $+\frac{1}{2}$, i.e. each in the state

$$\frac{1}{\sqrt{2}} (|\uparrow> + |\downarrow>),$$

then the register is in the superposition [2]

$$\frac{1}{\sqrt{2^L}} (|0> + |1> + |2> + \cdots) = \sum_{J=0}^{2^L-1} |J> .$$

If the target (electrons) register was initially in the state 0 (i.e. all electrons in the state $|\downarrow>$) then upon exit the target register is would the state

$$\sum_{J=0}^{2^L-1} |f(J)> .$$

We thus have a state which contains all of the possible values of $f(J)$ simultaneously. However, once a measurement is made on the spin in the z-direction of spins of all the electrons the state collapses into one of the allowed values of J for the protons and the corresponding $f(J)$ for the electrons. There is equal probability for obtaining each value of the pair $(J, f(J))$.

An "oracle" is a device which reverses the sign of a wavefunction of the register is in some particular (marked) state $|J_0>$. This can be achieved by constructing a function device in which the target

[2] This is equivalent to the state obtained by operating on the state $|0>$ with a Hadamard transformation.

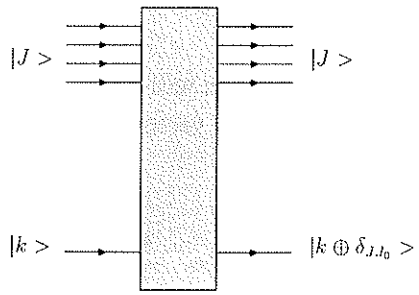


Fig. 5: The oracle operating on a control register and a single qubit target register performs the transformation $|J> \otimes |k> \rightarrow |J> \otimes |k \oplus \delta_{JJ_0}>$.

register is a single qubit ($L' = 1$) which is flipped if and only if the control register is in the state $|J_0>$, i.e.

$$|J> \otimes |k> \rightarrow |J> \otimes |k \oplus \delta_{JJ_0}> .$$

If $|k>$ is initially in the state

$$|k>= \frac{1}{\sqrt{2}} (|0> - |1>) ,$$

then

$$|k \oplus \delta_{JJ_0}> = (-1)^{\delta_{JJ_0}} \frac{1}{\sqrt{2}} (|0> - |1>) ,$$

so that the device flips the sign of the wavefunction (for the entire system) if and only if the control register is in the state $|J_0>$.

9. DISCRETE FOURIER TRANSFORMS

We consider the example of taking the Fourier transform of a 2 qubit quantum system, which is in the state

$$|\psi>= \sum_{J=0}^{3} a_J |J> .$$

The Fourier transform state is

$$|\phi>= \sum_{K=0}^{3} b_K |K>,$$

where

$$b_K = \sum_{J=0}^{3} e^{i\pi JK/2} a_J .$$

For this we need the following quantum devices which can perform unitary operators.

a. The first is a unary operator which acts on one qubit only

$$\mathbf{u}_y \left(\frac{\pi}{2}\right) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix},$$

which corresponds to a rotation through $\pi/2$ about the y-axis.

$$\mathbf{u}_y \left(\frac{\pi}{2}\right) |\uparrow> = \frac{1}{\sqrt{2}} (|\uparrow> - |\downarrow>)$$

$$\mathbf{u}_y \left(\frac{\pi}{2}\right) |\downarrow> = \frac{1}{\sqrt{2}} (|\downarrow> + |\uparrow>)$$

This can act on *either* of the two qubits so we will write these as $\mathbf{U}^A_{(1)}$ and $\mathbf{U}^A_{(2)}$. Thus acting on the two qubit system represented by the bases vectors $|0>$ to $|3>$, these operators may be written as the 4×4 matrices

$$\mathbf{U}^A_{(1)} = \mathbf{u}_y\left(\frac{\pi}{2}\right) \otimes \mathbf{I} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{U}^A_{(2)} = \mathbf{I} \otimes \mathbf{u}_y\left(\frac{\pi}{2}\right) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

b. A binary operator, $\mathbf{U}^B(\phi)$, which acts on a two qubit system, leaving all states alone except that the state $|1> \equiv |\downarrow\uparrow>$ is multiplied by a phase $e^{i\phi}$.
In 4×4 matrix notation, therefore

$$\mathbf{U}^B(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\phi} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

c. The binary operator, $\mathbf{U}_{NOT}$ which flips both qubits

$$\mathbf{U}_{NOT} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

d. The binary operator that interchanges the two qubits

$$\mathbf{U}_{switch} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Now the operation of the Fourier transform consists of the sequential application of the following operations:

a. Applicaton of the operator $\mathbf{U}_{NOT}$
b. Application of operator $\mathbf{u}_y(\pi/2)$ on qubit (1), $(\mathbf{U}^A_{(1)})$
c. Application of $\mathbf{U}^B(\pi/2)$
d. Application of operator $\mathbf{u}_y(\pi/2)$ on qubit (2), $(\mathbf{U}^A_{(2)})$
e. Application of the operator $\mathbf{U}_{switch}$.
In other words

$$\mathbf{F}_T \equiv \mathbf{U}_{switch}\mathbf{U}^A_{(2)}U^B(\pi/2)\mathbf{U}^A_{(1)}\mathbf{U}_{NOT}$$

$$= \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

Thus we see that

$$\mathbf{F}_T|J> = \frac{1}{2}\sum_{K=0}^{3} e^{i(JK\pi/2)}|K>,$$

as required.

This can be generalized to an L-bit system, using appropriate combinations of $\mathbf{U}^A_{(j)}$ and $\mathbf{U}^B_{jk}(\pi/2^n)$, $(i,j,k,n = 0\cdots(2^L-1))$.

10. FACTORIZATION

Factorization algorithm:

To factorize the number N, we can use an algorithm known as "Shor's algorithm".

Let a be coprime with N (no common factors).

The function defined by

$$f_{a,N}(J) \equiv (a^J, \text{ MOD } N)$$

has a period, P.

Provided P is even and $(a^{P/2}, \text{ MOD } N) \neq N-1$, then the greatest common divisors of the pairs

$$(a^{P/2}+1,\ N) \ and \ (a^{P/2}-1,\ N)$$

are factors of N.

Example:

$$N = 21,\ \ a = 2$$

$$(2, \text{ MOD } 21),\ (2^2, \text{ MOD } 21)\ (2^3, \text{ MOD } 21)\ (2^4, \text{ MOD } 21)\ (2^5, \text{ MOD } 21)$$

$$(2^6, \text{ MOD } 21)\ (2^7, \text{ MOD } 21)\ (2^8, \text{ MOD } 21)\ (2^9, \text{ MOD } 21)\ \cdots$$

$$= 2,\ 4,\ 8,\ 16,\ 11,\ 1,\ 2,\ 4,\ 8,\ \cdots$$

The period of the function $P = 6$, so that $2^{6/2} = 8$.

The greatest common divisor of 9 and 21 is 3.
The greatest common divisor of 7 and 21 is 7.
Thus the factors of 21 are 7 and 3.

Now finding the greatest common divisor of two numbers can be achieved by a very fast algorithm (due to Euclid). The difficulty is finding the period, P, of the function $f_{a,N}(J)$. By classical computers this is the same level of complexity as any other factorization algorithm.

However by quantum encoding the function $f_{a,N}(J)$ the period can be found relatively rapidly.

First we construct a device that performs the operation

$$|J>\otimes|K> \rightarrow |J>\otimes|K \oplus f_{a,N}(J)>,$$

using the magnetic resonance method described above. Then (by polarizing the spins of all the qubits in the control register in the $x-$direction, with eigenvalue $+\frac{1}{2}$), we consider the input

$$\sum_{J=0}^{2^L-1} \frac{1}{\sqrt{2^L}}|J>\otimes|0>$$

and obtain upon output

$$\sum_{J=0}^{2^L-1} \frac{1}{\sqrt{2^L}} |J> \otimes |f_{a,N}(J)> .$$

Next we make a measurement of the spin state (in the $z-$direction of the target register . This returns $f_{a,N}^q$, $q = 0, \cdots (P-1)$, which is one of the P allowed values of the function $f_{a,N}(J)$.

This immediately collapses the control register into a superposition (unnormalized)

$$\sum_r |rP + q>,$$

where r runs from zero to the integer below $2^L/P$. The problem is that q can take any non-negative integer value up to $P-1$. However if we take the Fourier transform of this state this effect is 'washed away'. In more detail the Fourier transform of the above function (again unnormalized) is

$$\sum_{K=0}^{2^L-1} \sum_r e^{i(K(rP+q)\pi/2^L)} |K> .$$

If P is an integer divisor of 2^L then the factor

$$\sum_{r=0}^{2^L/P} e^{i(KrP\pi/2^L)}$$

vanishes unless K is an integer multiple of $2^L/P$. This means that any subsequent measurement of the z-component of the spin of the control register will yield a result which is an integer multiple of $2^L/P$. Thus after a few such measurements the value of P can be determined with high confidence.

In the more realistic case where P is *not* an integer divisor of 2^L the result of the Fourier transform is a superposition which is very highly peaked around integer multiples of $2^L/P$. Thus several measurements of the spin state of the Fourier transformed control register have to be taken. However once again the value of P can be deduced with a high level of confidence after a number of measurements which is far fewer than the number of operations required to factorize a number using classical computers.

If P turns out to be one of the forbidden values, the the process must be repeated using a different value of a (a separate function device). However the probability of P being allowed is greater than 50%.

11. GROVER'S ALGORITHM

The objective is to force a register which is a superposition of all allowed states (with equal coefficient) into a particular "marked state". The state is marked by passing the system through an "oracle", which reverses the sign of the wavefunction if and only if the register is in the marked state. With a classical computer one must systematically compare all the states with the marked state, a process which grows as the maximum allowed marked number (i.e. as 2^L for an L bit register), whereas using Grover's algorithm this process only grows as the square root of the maximum allowed number (i.e. as $2^{L/2}$ for an L bit register).

First an example using two qubits:

Initially the qubits are in the state $|0>$. We apply a (pseudo-) Hadamard transformation, **H**, which performs the operation

$$|0> \rightarrow \mathbf{H}|0> = \frac{1}{2}(|0> + |1> + |2> + |3>)$$

$\mathbf{U}_J$ is a matrix which flips the sign of the state $|J>$, but leaves all other states alone, i.e.

$$\mathbf{U}_J|\Psi> = |\Psi> -2<J|\Psi>|J> .$$

The device which performs such an operation is the oracle.

Now consider the sequence of operators $\mathbf{H}\,U_0H^{-1}\mathbf{U}_JH$ acting on $|0>$

$$\begin{aligned}\mathbf{H}\,U_0H^{-1}\mathbf{U}_JH|0> &= \sum_{K=0}^{3}\frac{1}{2}\mathbf{H}\,U_0H^{-1}\mathbf{U}_J|K> \\ &= \sum_{K=0}^{3}\frac{1}{2}\mathbf{H}\,U_0H^{-1}|K> -\mathbf{H}\,U_0H^{-1}|0> \\ = \sum_{K=0}^{3}\frac{1}{2}\mathbf{H}\,H^{-1}|K> - \sum_{K=0}^{3} &< 0|\mathbf{H}^{-1}|K>\mathbf{H}|0> -\mathbf{H}\,H^{-1}|J> +2<0|\mathbf{H}^{-1}|J>\mathbf{H}|0>\end{aligned}$$

Now

$$\sum_{K=0}^{3}\frac{1}{2}\mathbf{H}\,H^{-1}|K> = \mathbf{H}|0>$$

and

$$<0|\mathbf{H}^{-1}|K> = \frac{1}{2} \text{ for all K}$$

This leaves

$$\mathbf{H}\,U_0H^{-1}\mathbf{U}_JH|0> = -|J> .$$

This device forces the state $|0>$ into the state $|J>$ (up to a sign) after a single pass.

This has been achieved experimentally by Jones using a solution of the base cytosine in D_2O. This results in a molecule with two unpaired protons forming a two spin system. Selective NMR pulses can be applied to each proton.

Now consider L bits.
Again a Hadamard transformation is applied to the state $|0>$,

$$\mathbf{H}|0> = \frac{1}{\sqrt{2^L}}\sum_{K=0}^{2^L-1}|K> .$$

The oracle flips the sign of the state $|J>$, but leaves all other states unchanged

$$\mathbf{U}_J|\Psi> = |\Psi> -2<J|\Psi>|J>$$

$$<0|\mathbf{H}^{-1}|K> = \frac{1}{\sqrt{2^L}}, \text{ for all } K$$

Let $|\Psi> \equiv \mathbf{H}|0>$.

Now consider the operator $\mathbf{H}\,U_0H^{-1}\mathbf{U}_J$ acting on the state $|\Psi>$

$$\begin{aligned}\mathbf{H}\,U_0H^{-1}\mathbf{U}_J|\Psi> &= \sum_{K=0}^{2^L-1}\frac{1}{\sqrt{2^L}}\mathbf{H}\,U_0H^{-1}|K> -\frac{2}{\sqrt{2^L}}\mathbf{H}\,U_0H^{-1}|J> \\ = \sum_{K=0}^{2^L-1}\frac{1}{\sqrt{2^L}}|K> -2\sum_{K=0}^{2^L-1}\frac{1}{\sqrt{2^L}} &< 0|\mathbf{H}^{-1}|K>\mathbf{H}|0> -\frac{2}{\sqrt{2^L}}\mathbf{H}\,U_0H^{-1}|J> +\frac{4}{\sqrt{2^L}}<0|\mathbf{H}^{-1}|J>\mathbf{H}|0> \\ &= \mathbf{H}|0> -\frac{2}{2^L}2^L\mathbf{H}|0> +\frac{4}{2^L}\mathbf{H}|0> -\frac{2}{\sqrt{2^L}}|J> \\ &= -\left(1-\frac{4}{2^L}\right)|\Psi> -\frac{2}{\sqrt{2^L}}|J>\end{aligned}$$

Now consider the operator $\mathbf{H}\,U_0 H^{-1}\mathbf{U}_J$ acting on the state $|J>$

$$\begin{aligned}\mathbf{H}\,U_0 H^{-1}\mathbf{U}_J|J> &= -\mathbf{H}\,\mathbf{U}_0 H^{-1}\mathbf{U}_J|J> \\ &= -|J> +2<0|\mathbf{H}^{-1}|J>\mathbf{H}|0> \\ &= -|J> +\frac{2}{\sqrt{2^L}}|\Psi>\end{aligned}$$

Consider the space of the two (non-orthogonal) states $|\Psi>$ and $|J>$. In this subspace the operator $\mathbf{H}\,U_0 H^{-1}\mathbf{U}_J$ has the matrix representation

$$-\begin{pmatrix} \left(1-\frac{4}{2^L}\right) & \frac{2}{\sqrt{2^L}} \\ -\frac{2}{\sqrt{2^L}} & 1 \end{pmatrix}$$

For large L we may approximate this by

$$-\begin{pmatrix} \cos(\alpha/2) & \sin(\alpha/2) \\ -\sin(\alpha/2) & \cos(\alpha/2) \end{pmatrix}$$

where

$$\alpha = \frac{4}{\sqrt{2^L}}.$$

If we perform this operation N times where N is the nearest integer to

$$\pi\frac{\sqrt{2^L}}{4}$$

then we get (approximately) the matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

In other words the state is converted from pure $|\Psi>$ which we obtain by passing $|0>$ through a Hadamard gate, to the required state $|J>$ in less than $\sqrt{2^L}$ passes.

$$\left(\mathbf{H}U_0 H^{-1}\mathbf{U}_J\right)^N \mathbf{H}|0> \approx |J> .$$

The error in this approximation is of order $1/\sqrt{2^L}$.

It may be more convenient to use orthogonal states. We therefore define the state $|\Phi>$ which is orthogonal to $|J>$ by

$$|\Phi> = \sqrt{\frac{2^L}{2^L-1}}\left(|\Psi> -\frac{1}{\sqrt{2^L}}|J>\right) = \frac{1}{\sqrt{2^L-1}}\sum_{K\neq J}|K>$$

so that

$$|\Psi> = \sqrt{\frac{2^L-1}{2^L}}|\Phi> +\frac{1}{\sqrt{2^L}}|J> = \cos\beta|\Phi> +\sin\beta|J>,$$

where $\sin\beta = 1/\sqrt{2^L}$, i.e.

$$\begin{aligned}<0|\mathbf{H}|\Phi> &= \cos\beta \\ <0|\mathbf{H}|J> &= \sin\beta\end{aligned}$$

The operator $\mathbf{U}_0$ may be expressed as

$$\mathbf{U}_0 \;=\; \mathbf{I} - 2|0><0|,$$

where $\mathbf{I}$ is the identity operator, so that

$$\mathbf{H}\,U_0 H^{-1} \;=\; \mathbf{I} - 2\mathbf{H}|0><0|\mathbf{H}^{-1}$$

Now the operator $\mathbf{O} \equiv \mathbf{H}\,U_0 H^{-1}\mathbf{U}_J$ acts on the orthogonal $|\Phi>$, $|J>$ subspace as follows.

$$\begin{aligned}
\mathbf{O}|J> &= -\mathbf{H}\,U_0 H^{-1}|J> = -|J> +2\mathbf{H}|0><0|\mathbf{H}^{-1}|J> \\
&= -|J> +2\sin\beta\,\cos\beta|\Phi> +2\sin^2\beta|J> \\
&= -\left(\cos(2\beta)|J> -\sin(2\beta)|\Phi>\right)
\end{aligned}$$

$$\begin{aligned}
\mathbf{O}|\Phi> &= \mathbf{H}\,U_0 H^{-1}|\Phi> = |\Phi> -2\mathbf{H}|0><0|\mathbf{H}^{-1}|\Phi> \\
&= |\Phi> -2\cos^2\beta|\Phi> -2\sin\beta\,\cos\beta|J> \\
&= -\left(\cos(2\beta)|\Phi> +\sin(2\beta)|J>\right)
\end{aligned}$$

Thus in this subspace the operator $\mathbf{O}$ has the (unitary) representation

$$\mathbf{O} \;=\; -\begin{pmatrix} \cos(2\beta) & \sin(2\beta) \\ -\sin(2\beta) & \cos(2\beta) \end{pmatrix}$$

The precise number of times one need to apply the operator O in order to obtain a pure state $|J>$ is given by

$$(2N+1)\beta = \frac{\pi}{2}$$

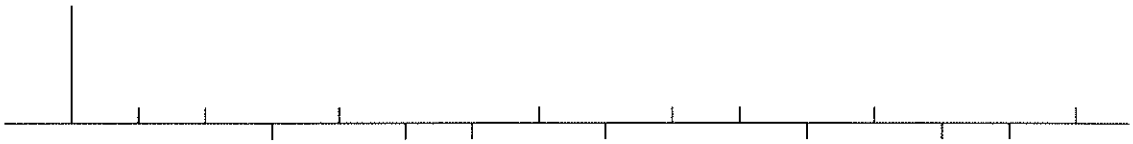
$$N = \frac{\pi}{4\sin^{-1}\left(\frac{1}{\sqrt{2^L}}\right)} - \frac{1}{2}$$

Note that for $L = 2$, the exact solution is $N = 1$.

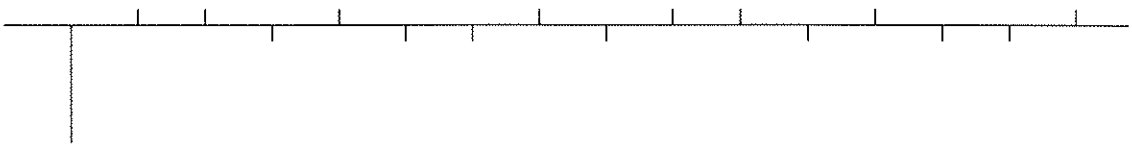
In the following example we take the case of four qubits, so that the register store a number between 0 and 15. We assume that the marked state is the number 7. We begin by taking the state $|0>$ and performing a Hadamard transformation so that we have a superposition of all states with equal coefficients. Now we pass the state four times, though the series of transformations $\mathbf{U}_7$, H, U_0, and $\mathbf{H}$. Note that after three iterations the state is almost purely in the state $|7>$, as required. In fact the coefficient of the component $|7>$ is 0.96 rather than unity and there is still a small component from the other states. Recalling that the probability to find the system in a given state is the square (modulus) of the coefficient, we see that there is a 99.8% probability to find the system in the required state after three iterations. We note also that upon application of a fourth iteration the purity of the state is lost - the components of the other states has increased considerably.

$\mathbf{U}_7$

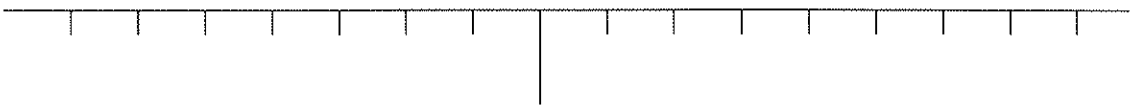
$\mathbf{H}$



$\mathbf{U}_0$

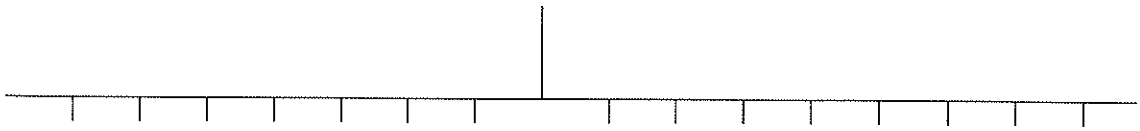


$\mathbf{H}$

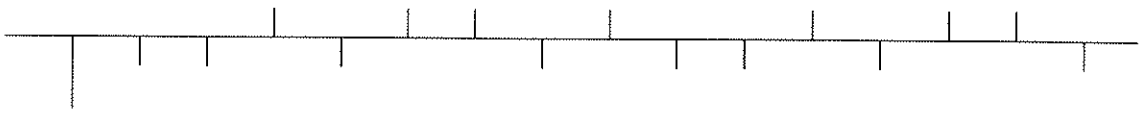


Iteration 2

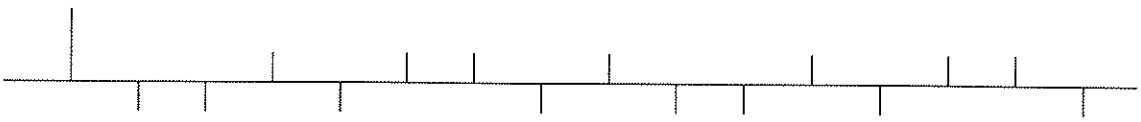
$\mathbf{U}_7$



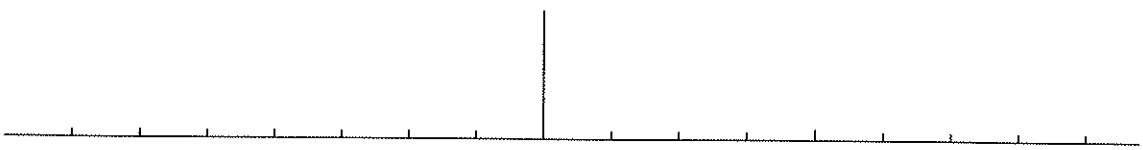
H



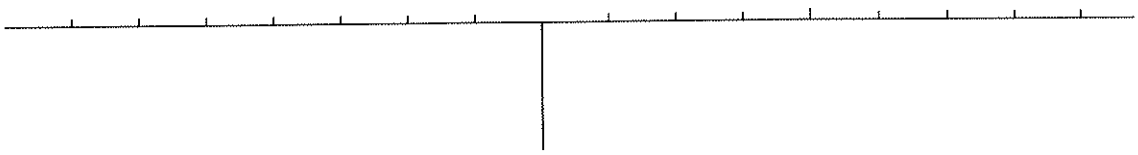
$\mathbf{U}_0$



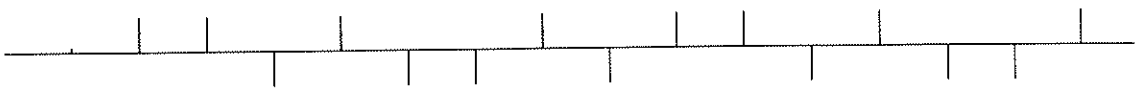
H



$\mathbf{U}_7$

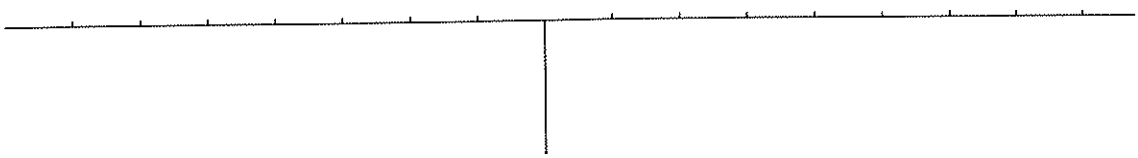


$\mathbf{H}$



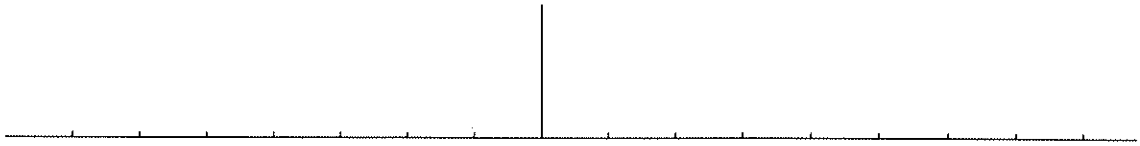
$\mathbf{U}_0$

$\mathbf{H}$

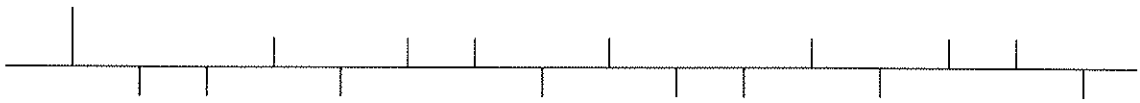


Iteration 4

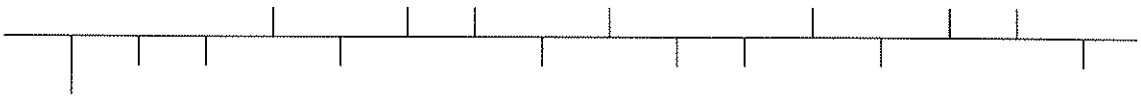
$\mathbf{U}_7$



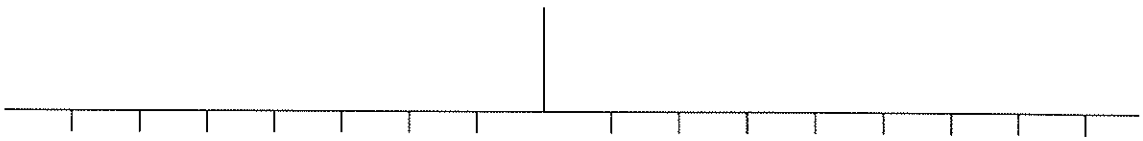
$\mathbf{H}$



$\mathbf{U}_0$



$\mathbf{H}$



11.1 Using Grover's Algorithm to Search a Database

Consider a function $f(J)$ which maps an integer J to an integer $F = f(J)$. For simplicity assume that the map is one-to-one. The objective is to force a state into the state $|J> \otimes |F>$, given knowledge of F but not J. Once again we need an oracle which reverses the sign of the wavefunction for a state if the second (target) register is in the state $|F>$, but otherwise leaves the state unchanged.

Using a sequence of NMR pulses we can construct a device which performs the operation V_f, such that

$$|J> \otimes |K> \rightarrow \mathbf{V}_f|J> \otimes |K> \equiv |J> \otimes |K \oplus f(J)> .$$

The first register is the "control" register and the second register is the "target register". Since we assume that the map is one-to-one we are assuming that these registers both contain L qubits.

Note that V_f is an idempotent operator, i.e. the device is reversible. In particular,

$$\mathbf{V}_f|J> \otimes |0> = |J> \otimes |f(J)>$$

$$\mathbf{V}_f|J> \otimes |f(J)> = |J> \otimes |0>$$

Now define an device W which is a Hadamard gate acting on the control register *only* followed by the device V_f, with the matrix representation

$$\mathbf{W} \equiv \mathbf{V}_f H$$

$$\mathbf{W}^{-1} = \mathbf{V}_f H^{-1}$$

where H is a Hadamard gate acting on the control register only. Thus

$$\mathbf{W}|0> \otimes |0> = \frac{1}{\sqrt{2^L}} \sum_{K=0}^{2^L-1} \mathbf{V}_f|K> \otimes |0> = \frac{1}{\sqrt{2^L}} \sum_{K=0}^{2^L-1} |K> \otimes |f(K)>$$

Since the map is one-to-one we can rewrite this last expression as

$$\frac{1}{\sqrt{2^L}} \sum_{F=0}^{2^L-1} |f^{-1}(F)> \otimes |F> .$$

Now let U_F be a device which flips the sign of the quantum state if and only if the target register is in the state $|F>$. From the above discussion on Grover's algorithm for L qubits it follows that

$$\left(\mathbf{W}U_0 W^{-1}\mathbf{U}_F\right)^N \mathbf{W}|0> \otimes |0> \approx |f^{-1}(F)> \otimes |F>,$$

where N is the nearest integer to $\sqrt{2^L}\pi/4$.

12. DEUTSCH'S ALGORITHM

Consider a one bit function ("true" or "false"), $f(I)$, where I is an integer between 0 and $2^L - 1$, but $f(I)$ can only take the values 0 or 1. If $f(I) = 0$ for all values of I or $f(I) = 1$ for all values of I, then the function is said to be "even". If $f(I) = 1$ for $2^L/2$ values of I and $f(I) = 0$ for the remaining $2^L/2$ values then the function is said to be "balanced".

For a classical computer, if we want to establish whether a function is even or balanced (or neither) we would need to sample the function for all values of the argument, I. Using Deutsche's algorithm we can construct a state which is a function of a superposition of all possible arguments and with a single

enquiry we can establish either that the function is not balanced or (exclusive) that the function is not even.

Single qubit:

Let U_f be a quantum logic gate which perform the operation on a single control bit and a single target bit

$$|j> \otimes |k> \rightarrow \mathbf{U}_f|j> \otimes |k> = |j> \otimes |k \oplus f(j)>,$$

where $f(j)$ is a single bit function of the single bit j, e.g.

$$\mathbf{U}_f|j> \otimes |0> = |j> \otimes |f(j)>$$

Now let $|k>$ be

$$|k> = \frac{1}{\sqrt{2}}(|\downarrow> - |\uparrow>) = \frac{1}{\sqrt{2}}(|0> - |1>)$$

such that

$$\mathbf{U}_f|j> \otimes |k> = \frac{1}{\sqrt{2}}(|j> \otimes |f(j)> - |j> \otimes |0>), \text{ if } f(j) = 1,$$

and

$$\mathbf{U}_f|i> \otimes |j> = \frac{1}{\sqrt{2}}(|j> \otimes |f(j)> - |j> \otimes |1>), \text{ if } f(j) = 0.$$

In other words

$$\mathbf{U}_f|j> \otimes |k> = \frac{1}{\sqrt{2}}(-1)^{f(j)}|j> \otimes |k>$$

Now let $|j>$ be

$$|j> = \frac{1}{\sqrt{2}}(|\uparrow> + |\downarrow>) = \frac{1}{\sqrt{2}}(|1> + |0>)$$

$$\begin{aligned} \mathbf{U}_f|j> \otimes |k> &= \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0> + (-1)^{f(1)}|1>\right) \otimes |k> \\ &= (-1)^{f(0)}|\left(|0> + (-1)^{(f(1)+f(0))}|1>\right) \otimes |k>. \end{aligned}$$

Now the control bit is in an eigenstate of S_x with eigenvalue $-\frac{1}{2}$ if $f(0) \oplus f(1) = 1$ (balanced) and $+\frac{1}{2}$ if $f(0) \oplus f(1) = 0$ (constant).

Extension to L bits for the control register:

U_f is a device that performs the operation

$$|J> \otimes |k> \rightarrow \mathbf{U}_f|J> \otimes |k> = |J> \otimes |k \oplus f(J)>.$$

Here $f(J)$ is a single bit function of the integer J.

We start with the control register in the state

$$\mathbf{H}|0> = \frac{1}{\sqrt{2^L}} \sum_{K=0}^{2^L-1} |K>$$

and the target bit in the state $(|0> - |1>)/\sqrt{2}$, as before.

Upon output from the device the control register is in the state

$$\frac{1}{\sqrt{2^L}} \sum_{K=0}^{2^L-1} (-1)^{f(K)} |K> .$$

If f is balanced then this state is orthogonal to $\mathbf{H}|0>$, i.e the overlap

$$\frac{1}{\sqrt{2^L}} \sum_{K=0}^{2^L-1} (-1)^{f(K)} < 0|\mathbf{H}|K>$$

is zero. On the other hand if the function is constant then the overlap has modulus unity.

Thus we pass the sample through a Stern-Gerlach apparatus that only allows the particle to pass if $S_x = L/2$. From this measurement we can deduce the following with absolute certainty

- If the sample passes through then the function is NOT balanced (the overlap is not zero).
- If the sample does not pass though then the function is NOT constant (the modulus of the overlap is not unity).

FURTHER READING

The following references contain full references to the literature and alternative presentations to the topics discussed in these notes:-

1. "The Feynman Lectures on Computation" by Richard P. Feynman, edited by Anthony J.G. Hey and Robin W. Allen, (Perseus Books, 1996; Penguin Books 1999).
2. "Explorations in Quantum Computing" by Colin P. Williams and Scott H. Clearwater (TELOS, Springer-Verlag, 1997).
3. "Introductions to Quantum Computation", edited by Hoi-Kwong Lo, Sandu Popescu and Tim Spills (World Scientific, 1998).
4. "Feynman and Computation: Exploring the Limits of Computes", edited by Anthony J.G. Hey (Perseus, 1999).

MANAGED STORAGE SYSTEMS AT CERN

Ingo Augustin and Fabrizio Gagliardi
CERN, Geneva, Switzerland

Abstract
The amount of data produced by the future LHC experiments requires fundamental changes in the management of the mass storage environment. At present the contents of the CERN tape libraries are not transparently managed. The experimental data rates and volumes will grow by more than an order of magnitude in future. This implies major changes in the management of centrally stored data. Several efforts to address this challenge are described in this paper.

1. INTRODUCTION

Traditionally the majority of High-Energy Physics experiments have recorded their data locally at the experimental sites and later transferred the tapes to the computer center. Since 1995 experiments like NA48 send their data online to the computer center via dedicated fibers where it is stored centrally (Central Data Recording). The sheer amount of data (100 TB/year for NA48) and the data rates require high performance storage devices, which are too expensive for individual collaborations.

LHC will exceed the present requirements by orders of magnitude. ALICE plans to take data at more than 1GB/sec. Others will produce data at 100MB/sec. Each of these experiments will collect at least 1 PB/year (1 PB = 1 PetaByte = 10^{15} Bytes ~ 1.5 million CD-ROMs or a bookshelf of a length of 5000 km). Although CERN will be one of the biggest storage facilities in the world, the storage itself of this data is not a problem. However, large storage facilities are usually used for backup or archives. This implies that the data is written once, and rarely read back. In our environment the situation is reversed. We write data once, but improved calibrations and reconstruction will require more than one pass reading the raw data. Efficient data retrieval becomes important. In order to achieve this, optimized access to resources (disks, tapes...) has to be guaranteed. This article describes some of the efforts CERN has undertaken to tackle this task.

2. THE CERN PETABYTE PROBLEM

The maximum data rate of the LHC experiments is 1-2 GB/sec. Taking into account that the data has to be read several times, a network (and of course storage) bandwidth of several GB/sec seems necessary. Current tape drives and disks operate with a bandwidth of at most tens of MB/sec, which implies that hundreds of devices are necessary to achieve the needed throughput. Even with tapes (or disks) of a size of 100 GB per piece, at least 10'000 of them are needed for each of the LHC experiments every year. Global collaborations and the more and more systematic production of analyzed data leads to round-the-clock operations of all computing systems. This, and the sheer amount of data, requires automated operations. Human resources are also scarce everywhere.

The future experiments will all used asynchronous readout models. Pipelining of data, buffering and highly sophisticated event filtering allows (or even requires) parallel streams of data. These streams can be handled and stored independently. The number and throughput of these streams can be adjusted to network and storage capabilities and is therefore highly scalable. Sufficient disk-buffers can de-couple the data acquisition performance from the central data recording, thus ensuring the highest possible performance of the tape system.

Usually the data has to be reconstructed several times due to improved calibrations and reconstruction software. The large amount of data requires centralized procedures to do this. A systematic reconstruction of the whole data can be viewed as the reversal of central data recording.

The classical HSM model features the migration from expensive, fast and small devices, such as RAM to inexpensive, slow and large devices (tapes).

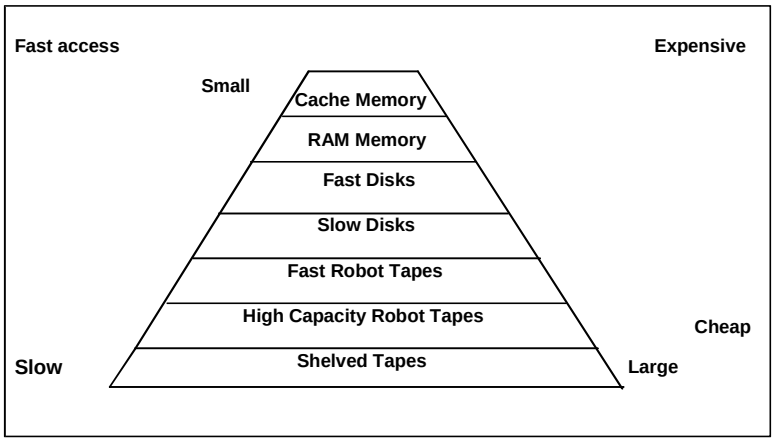


Figure 1: Classical storage model where the data migrates from high performance storage devices to cheaper and slower ones.

During the recent years this pyramid has begun to be distorted. Disks are getting bigger and bigger, and their prices have dropped, but the data rates did not change accordingly. At the same time the tapes became bigger and faster (with still more progress announced by the vendors), but the price is relative stable. Therefore, the relative financial share of tape storage in a HSM installation increases. The steady accumulation of large amounts of data makes it to costly to keep the data on disks for a long time. On the other hand, the data will be reprocessed several times during the first few years. Therefore access to the mass storage system is required. Optimization of these accesses is mandatory. Nowhere in the whole storage chain can performance be more easily degraded than at the tape level.

For example: the current workhorses of the CERN tape system are StorageTek *Redwood* drives. Transfer rates of about 10 MB/s and 50 GB capacity per tape volume are their principal characteristics. In terms of operations the time which is required to mount, load and position the tape is also important. If one transfers a file of 50GB to (or from) a tape, one achieves 10MB/s reduced slightly (2%) by the operational tape handling overhead (typically 100 sec for this type of drives). During the time of the overhead the drive can not be used for transfers. Many of the experiments use Linux as their standard operating system. At present Linux restricts the filesize to 2GB. This filesize appears reasonably large, but the impact on the performance of tape operations is now dramatic: 100 sec for tape loading and positioning, 200 sec for the transfer of 2 GB. This means the effective tape throughput came down from 98% to 66% of its design maximum. One third of the capacity is lost. With hundreds of drives being necessary to operate at the LHC data rate, this means hundreds of thousands of Swiss Francs that have to be invested additionally.

In this example the impact of the user (experiments) data model on the mass storage was shown. Access patterns like random access to the tapes are potentially even worse as the overhead easily becomes the dominant constraint of the input and output of the data. Therefore, the sequential mass storage on tapes has to be de-coupled from the user by using large disk pools (staging) or the users (experiments) have to get involved closely into the operations of the mass storage system. It is unlikely that the experiments will be keen to adjust their data models to the needs of hardware, which will certainly change during the decades of LHC operations.

The current model of LHC computing is built around computing farms, consisting of thousands of PCs. Although the assembly of such an amount of computers in a networking environment is not a problem per se (CERN already has this amount), the fact that they are working on the same data in a coordinated way provokes severe problems. Maintenance, configuration management and monitoring will be challenges. The computing farms can be used for online filtering or reconstruction during central data recording and for systematic reconstruction or analysis at later stages. In either way they will have to access the mass storage system, either to act as a data source or as a sink. Without optimization this kind of access will present itself as random access to the mass storage system. As described before this immediately introduces problems.

CERN investigated several routes to overcome this mismatch between the volatile PC-farm environment and the relatively inflexible mass storage environment. These will be described in a later section.

3. CERN STORAGE REQUIREMENTS

The CERN storage requirements can be separated in two general areas. First there are the objective requirements, which have to be fulfilled in order to do the job:

- Aggregate transfer rate at least tens of gigabytes/second with hundreds of streams in the 10-100 MB/s range.
- Continuous operation (operator attendance < 8 hrs/day)
- Storage capacity in the 10 - 100 PB range
- Thousand of simultaneous clients
- Large total number of files (~2^{64})
- File sizes only limited by operating system
- Reliable and error free transactions
- All this has to be achieved within the available budget (not yet defined)

It is standard CERN policy not to rely on a single provider for a system, if possible. Therefore a storage system should support different computer platforms and storage hardware. It is very likely that most of the computing for LHC is done on PC-farms, hence a support of these (at least as clients) is mandatory. Ideally the storage system itself would be running on PCs.

This last point leads to the second set of requirements. They are more a result of the need to achieve the goals within the given financial framework and the available human resources.

The experience with previous and current systems shows that easy manageability and efficient monitoring are key issues for the operation of a large storage system. Especially when users are accessing data on tape, resources like tape drives are easily wasted due to inefficient scheduling. Priorities, quotas, user groups fall in the same category.

The storage system should operate in a fully distributed environment.

These operational aspects are weaker in the sense that these topics can be used to compare and evaluate different systems, but are not 'show-stoppers'.

4. THE IEEE MASS STORAGE MODELS

In the early nineties the need for large storage systems became noticeable. The IEEE Storage System Standards Working Group (SSSWG) tried to identify the high-level abstractions that underlie modern storage systems [1,2]. The model should be applicable in a heterogeneous distributed environment. Many aspects of a distributed system are irrelevant to a user of the system. As a result, transparency

became the key point of the model. In this context, transparency means that the user accesses the data always in the same way: he does not know where it is or whether other users are using it. Behavior of operations and parameters should be always the same, regardless where the data is physically stored.

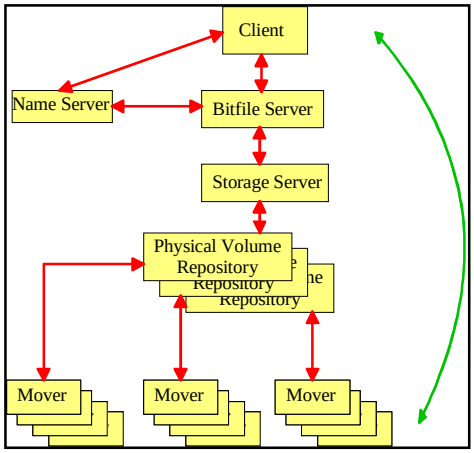


Figure 2: Components of the IEEE Mass Storage Systems Reference Model (V4)

4.1 Mass Storage System Reference Model Version 4

4.1.1 Mover

The Mover changes or monitors the read/write state of a device (e.g., positioning within the physical volume or performing loading and unloading operations).

4.1.2 Physical Volume Repository (PVR)

The PVR is a module that manages individual objects (cartridges, slots and transfer agents such as robots). It locates, mounts and dismounts cartridges on drives.

4.1.3 Storage Server

The Storage Server maps the physical volumes (cartridges, disks) to logical ones. The Bitfile Server as its client sees only a uniform logical space for each of the user or storage groups. The Storage Server consists of several sub-modules that allocate a PVR for a given data transfer, administrate the user/storage groups, enforce quotas. These modules become quite complicated if optimized tape, tape-drive or network usage is desired.

4.1.4 Bitfile Server

A bitfile server handles the logical aspects of bitfiles that are stored in one or more storage servers of the storage system. It creates the appropriate attributes of the bitfile like account-ID, access-control information, storage class, priority, backup information… Additionally the bitfile server maintains access statistics and records the real location of the file.

4.1.5 Name Server

If a file is submitted into a storage system, the bitfile server creates a unique bitfile-ID. The human-readable ID (/usr/xys/public/blabla/phd.tex) is convenient for people, but ambiguities can easily occur. The name server maps these filenames to the unique bitfile-IDs. This allows a storage system to be viewed as a global space rather than as space of host computers containing locally identified files.

4.2 Reference Model for Open Storage System Interconnection (MSS Version 5)

It is the successor of the previously described MSS model version 4. It is much more abstract than the old version. Mover and PVR are maintained, but the Physical Volume Library (PVL) and the Virtual Storage Service (VSS) have replaced everything else. The PVL manages physical volumes, such as tapes and disks. The VSS takes over the remaining functionality in order to present an abstract view of storage. Internally the old bitfile is now seen as composition of transfer units. This allows composition schemes, such as concatenation, replication, striping and various RAID methods. The big advantage is the possible redundancy and throughput during data migration. A file can be stored much faster when it is striped over several tapes (of course using the same number of drives). Unfortunately this involves the same number of tape mounts when the file has to be read back. In most tape libraries the data is read never or only occasionally. The situation in High-Energy Physics is different. Although one of the systems at CERN (HPSS) is capable of striping over tapes, this feature is not used.

5. CURRENT CERN PROJECTS

CERN investigated possible commercial solutions for quite some time and HPSS, a joint development by US DoE and IBM seemed the only potential "product" with enough performance and scalability to fulfill HEP requirements for the future generation of experiments. An HPSS evaluation project was therefore started in the fall of 1997. In parallel an internal project to develop further the in-house STAGER product was started. This was felt essential given the time scale of the COMPASS experiment scheduled to take data in 2000-2001. It was also decided to collaborate with DESY and a consortium of research and industrial partners in the EU supported Eurostore project.

5.1 HPSS Evaluation

HPSS [3] (High Performance Storage System) was first installed in the fall of 1997 on a test IBM system. The original test configuration included RS6000 AIX machines with IBM 3590 tape drive drives in IBM 3494 tape robots.

For HPSS to be successfully adopted by CERN multi-vendors support was essential. High level discussions with the HPSS consortium were therefore started while a joint project with digital was initiated to develop a port to Digital Unix of the HPSS data movers. This was carried out by a joint team of CERN and Digital experts during 1997/98 and delivered to the HPSS support team in IBM Houston for inclusion in the successive base line delivery kits.

A first prototype production service was started in August 1998. This included a user tape hierarchy for the Chorus experiment and Central Data Recording storage for the NA57 experiment. More than 2TB and 3000 files were managed.

An interesting test for the Objectivity/HPSS Interface was also implemented. This was particularly important given the strategic interest for Objectivity [4] and somehow the conflictual nature of the two products. This test stored 1TB for Atlas in 2000 files. Small testbeam setups for LHCB & Alice were also implemented.

Data is separated by *Class of Service.* The class of services (COS) defined were:

- COS 1 (User Tapes): 230GB AIX disk on 3590 tapes
- COS 2 (Raw Data on Redwood): 243GB DUX disk on Redwood tapes
- COS 4 (Testbeam): 32GB AIX disk on 2 Redwood copies (25GB, 50GB)
- COS 5 (Atlas 1TB Milestone): COS 1 disks on Redwood tapes
- COS 6 (Compass Test): COS 2 disks on Redwood tapes

The current tape mount rate (without NA57 data acquisition running) is about 100 per day.

5.1.1 Ongoing Work

A joint IT/Alice Mass Storage Project was started to investigate the use of a commercial HSM to manage their high acquisition rate of sequential data.

Milestones for recording “raw data” to tertiary storage were agreed:

- 30MB/s sustained in 1Q1999 (achieved with 4 sources & 2 sinks - IBM only)
- 100MB/s by 4Q1999 (need to “borrow” ~12 Redwoods when Alpha performance improved) - interesting to compare with the in house CASTOR project.

5.1.2 Experience and problems

- Architecture: random data access performance is slow via the standard provided API.
- Networking: it was difficult at the beginning to get HIPPI working, but it is mostly stable by now. HPSS software reliability is high but the overall service reliability is much dependent on network and media problems.
- Manageability: hard in the CERN computing environment. It assumes experienced operators to be available and to this we must add the cost of maintaining a DCE environment, which is otherwise not needed at CERN. The future of DCE is also questionable in the open commercial market.
- Operation: frequent reconfigurations caused service interruptions.
- Portability: this is very critical for CERN and HEP. The first implementation of the Digital Alpha 4100 data mover was slow (HIPPI to Redwood) in SMP configurations.

5.1.3 Successes

Excellent support from the HPSS IBM team in Houston was verified. The HPSS product itself has been stable, no data were lost because of HPSS. Sequential performance using the complex interface is only limited by hardware performance.

“Retirement” of old media to new products can migrate data gracefully. This is very important now that the lifetime of tape technology is not more than 4-5 years.

A single name space for all files is a good concept and works well.

HPSS allowed an efficient use of the shared disk pools and tape storage. Tapes are filled completely with consequent reduction of media consumption. The interface to HPSS has been developed using CERN standard libraries such as rfcp and the CERN stager. This allows new user applications to be added quickly and with minimum effort from the users.

5.1.4 CERN HPSS Plans

The original idea was to decide by end of 1999 if to commit to HPSS (i.e. use it for Compass in 2000) or drop it. However the Alpha port support and packaging by the HPSS consortium is not complete yet.

The first components of the Sun Solaris port in development at SLAC are now in the product. The BABAR experiment has started with HPSS and Objectivity at SLAC and at IN2P3 in Lyon, therefore we will be able to learn much soon.

The current strategy is therefore to continue a low level production use of HPSS to gain more experience and be ready to reconsider it as the final solution in case of positive and conclusive product developments.

5.2 CASTOR

In January 1999, CERN began to develop CASTOR ("CERN Advanced Storage Manager"). Its main goal is to be able to handle the COMPASS and NA48 data (25 + 35 MB/s) in a fully distributed environment and in a completely device independent fashion. Scalability should be good so we could also handle LHC data (100 MB/s per experiment) starting in 2005. Sequential and random access should both be supported with good performance.

CASTOR objectives are therefore:

High performance, good scalability, high modularity (to be able to easily replace components and integrate commercial products as far as they become available and show a better total cost of ownership and price performance factors).

CASTOR will provide HSM functionality with a single large name space. Migrate/recall functions are all focussed on HEP requirements, therefore keeping the design and implementation simple and less expensive.

It will be available on all Unix and NT platforms and will support most SCSI tape drives and robotics. System dependencies are grouped in few files to ease portability.

A user/operator/administrator graphical interface (GUI+WEB) is foreseen, but a command line interface will be retained for more automatic production use.

In the spirit of the CERN developed software it should remain easy to clone and deploy CASTOR outside CERN. CASTOR aims at using as much as possible commodity hardware components such as inexpensive PCs as tape servers.

The first version of CASTOR will be deployed at CERN during winter 1999/2000 and a large test (100MB/s during 7 days) will be attempted for ALICE in February 2000.

Support for Storage Area Networks (SAN) will be integrated with the goal of decreasing the number of data movers and servers. In the SAN model CPU servers are directly connected to the disks and share data. This is a move away from the traditional tape and disk server model, eliminating data copies between disk servers and CPU servers. SAN uses native filesystems, which give much better performance than NFS. It is important to acquire expertise in the area of emerging technology but even with SAN some HSM functionality will still be needed.

Support for different data storage models is being planned:

- disk pools
- local caches
- Storage Area Networks
- local disk and tape drives.

5.3 EuroStore

The third project with CERN participation is EuroStore [5], an European Union funded ESPRIT project. CERN, QSW (a supercomputer manufacturer) and DESY formed together with various smaller European enterprises a consortium to develop a scalable, reliable and easy manageable storage system almost entirely based on Java. QSW developed the Parallel File System (PFS), which is used in their high performance computer systems. DESY was the developer of the HSM system. The current storage system of DESY will reach its limits of scalability with the appearance of HERA B. The similarity of requirements for a storage system at CERN and DESY made a collaboration in this field desirable. The role of CERN and the commercial collaborators was the definition of user requirements and the assessment of the developed software according to these requirements.

The excessive requirements of LHC in terms of scalability and reliability, together with the necessity of flexible administration and maintenance, made up the bulk of the user requirements for the EuroStore software. These user requirements have been used also for HPSS, CASTOR and MONARC [6].

Similar to HPSS the EuroStore HSM is based on the IEEE mass storage standard. The complete HSM service is built out of sub-services implemented as separate processes or threads running on the same or different heterogeneous platforms. The communication mechanism between all these sub-services is done with a secure message passing environment, called Cell-Communication.

The HSM supports the notion of Storage Groups to allow a single Store to be divided into several sub-domains containing specific user groups and/or dataset types. The Store represents the root of the internal HSM object structure, which is built out of Storage Groups. The Storage Group is further subdivided into Volume Sets, which act as the source and destination for the HSM internal migration of datasets. The Volume Set is itself built out of Volume Containers defining the set of physical volumes belonging to a single physical library. To describe and control the internal HSM migration there exists an object, called Migration Path, which encloses the migration condition and the source/destination Volume Set. Each dataset stored in the HSM has a link to an existing Migration Path describing the dataset migration characteristics.

The HSM provides a simple service to the PFS (or other clients), namely storing and retrieving complete datasets (or files in the PFS nomenclature) sequentially. A future version of the EuroStore HSM might support read operations on parts of datasets (partial reads). This simplicity is mirrored in the data access API in that it contains only 3 functions: create/write a dataset, read an existing dataset and remove an existing dataset. In addition, the API will support simple query operations (ask for all files on a given volume, etc.) for its clients (like PFS). The data access API is implemented as a C based thread safe library.

The PVL supports additional functions:

- Priorities, specified by the client application. This was an important requirement of the EuroStore collaborators of the Health sector.
- Configurable numbers of write operations on a given Volume Set. This allows the choice between storage in chronological order, as in Central Data Recording, and the policy based selection of available resources (the PVL would choose a volume and a drive according to the current situation).
- Regular expression assigned to a storage device (drive). The PVL will manage a defined set of mainly request dependent variables that can be used to construct a regular expression. For example, a drive might be available during the time between *3:00* and *4:00* only for a user called *oracle_backup* on the host *oracle_server.cern.ch*. During all other times other users could use the drive.
- Virtual library partitioning allows dynamic resource allocations like "*20% of the tape drives are given to a certain client/user-group*".

The modular design of the EuroStore HSM provides the necessary scalability. Every component (e.g. movers, PVRs, PVLs, Bitfile servers) can be located on a different computer. The implementation in Java will provide the necessary portability to cover a wide range of heterogeneous hardware platforms.

The EuroStore prototype was deployed at CERN during April 1999. The hardware platform consists of four dual processor SUN Enterprise 450 servers. Each of the servers is equipped with four 8 GB hard disks, which build the components of one or more Parallel File Systems. The PFS can be

striped over several nodes of the cluster. The data is transferred between the nodes via a switched ELAN/ELITE network (max. 250 MB/s). Each of the E450s is connected to the CERN LAN with Gigabit Ethernet. At present the prototype uses two StorageTek 9840 tape drives, located in one of the automated tape libraries of CERN.

During the initial assessments many configurations have been tested and, except for the usual programming bugs, no conceptual problem could be found. The GUI based administration and management of the HSM system proved to be very effective and flexible. The implementation of an HSM in Java has been successfully demonstrated, although the issue of performance and reliability could not be really addressed yet, due to the ongoing development of the prototype. The EuroStore project will continue until summer 2000. DESY intends to deploy the EuroStore HSM as a production system at the end of the project.

6. FUTURE DEVELOPMENTS

The current plans at CERN are to continue the lines of development described above while exploring ways to increase the interoperability of HEP HSM systems.

7. CONCLUSION

It is clear that while commodity components computing seems to offer scalable and affordable solutions for LHC computing, the management of the data will remain a difficult challenge to tackle.

Disk storage is the only component which seems so far to follow the Moore price evolution curve of PCs. Tape and robotics seem to stagnate or have a very slow evolution at best.

The HSM commercial market doesn't seem to match HEP requirements, although some analysts predict tremendous growth in this field in a near future.

Until this happens probably we need to take a conservative approach and develop simple in house solutions in collaboration with other major HEP centres which share our needs.

We should in parallel continue to monitor technology and market evolution to spot trends, which we could exploit, and commercial products, which we could acquire.

The impact of large distributed data access models such as the ones investigated by MONARC should be taken into appropriate consideration.

ACKNOWLEDGEMENTS

The authors want to acknowledge the contributions of the Data Management section of the PDP group of the CERN IT-division. In particular J.-P. Baud, C. Curran and H. Renshall provided valuable input by reviewing the manuscript.

REFERENCES

[1] S. Coleman and S. Miller, ed., Mass Storage System Reference Model Version 4, May 1990, Technical Committee on Mass Storage Systems and Technology, Institute of Electrical and Electronics Engineers.

[2] A. Abbott et al., Reference Model for Open Systems Interconnection, September 1994, IEEE Storage Systems Standards Working Group (P1244), Institute of Electrical and Electronics Engineers.

[3] http://www.sdsc.edu/hpss/hpss.html

[4] http://www.objectivity.com

[5] http://www.quadrics.com/eurostore

[6] http://www.cern.ch/MONARC

RELATED WEB PAGES

http://nicewww.cern.ch/~fab/default.htm

http://www.cern.ch/eurostore/

http://www.ssswg.org/

http://home.cern.ch/~hepmss/

http://www.nsic.org/

http://www.snia.org/

DATA STORAGE TECHNOLOGIES FOR LHC

Les Robertson
CERN, Geneva, Switzerland

Abstract
The paper introduces some of the technologies that could be used for storing and managing the many PetaBytes of data that will be collected and processed at the Large Hadron Collider (LHC) accelerator, which is scheduled to begin operation at CERN in 2005. The state of the art of the current mainline hardware technologies is described, with a discussion of the likely evolution during the next five years. This paper is a summary of the material provided in the lecture notes, which include more detailed descriptions of some of the fundamental technologies and costs issues. The notes also introduce some more exotic techniques, and discuss the storage capacity and performance requirements of the experiments that will use the LHC accelerator.

1 MAGNETIC HARD DISKS

The principal technology used for permanent storage of digital computer data for the last thirty years is the magnetic hard disk. The state of the art hard disk product today can store 72 GigaBytes of data in a container 10 cm wide and 4 cm high, the format of the standard disk slots in a personal computer. The data is stored on twelve rigid disk platters mounted on a spindle that is rotated by an electric motor at around 10,000 rpm. There is one recording head for each disk surface, the heads being mounted on a set of arms (the *accessor*) that can move the heads across the recording surface of the disk. The data is recorded in concentric *tracks* as the disk platter rotates beneath the head.

Magnetic disk platters must provide a magnetic layer suitable for high-density recording and a surface layer that is smooth and durable to support very low head-flying heights. The platter is usually made by sputtering a thin layer of material with high magnetisation and coercivity[1] characteristics on to a rigid aluminium or glass substrate. This is followed by applying a very thin protective layer (diamond like carbon) and a lubricant.

The magnetic layer is usually a cobalt alloy (CoCrTa, CoPtCr, …). The aluminium substrate is generally first coated with a chromium layer before the magnetic layer is applied. When the disk is powered off, the heads "land" on an area near the centre of the disk, which is laser textured to avoid the heads sticking to the otherwise perfectly smooth disk surface. Current production disks are recorded at a density of 3-4 Gbits per square inch (Gb/in$^{2)}$, while laboratory prototypes have been demonstrated at over 35 Gb/in^2. The areal density of delivered products is increasing at around 60% per year, requiring continuous improvements in materials, manufacturing techniques and head technology.

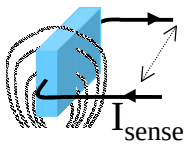
The area used to store individual bits decreases as the storage density increases. These smaller bit sizes imply smaller grains in the magnetic material, and higher coercivity (to sharpen the transitions between the bits). But there are limits beyond which the magnetic polarisation becomes unstable, when the fluctuations in thermodynamic energy at operating temperatures have a moderate probability of causing magnetic state changes. This is called the *super-paramagnetic* limit. This is not

[1] coercivity: The coercive force is the magnetic field which must be applied to neutralise the magnetic orientation of a material. Coercivity is a measure of how stable the magnetic orientation is in the material.

a fixed limit, as it depends on the magnetic properties of the materials, the recording method, shape of the bit pattern, etc. The limit with today's materials and heads is in the region of 40 Gb/in^2, but it is expected that this will be pushed progressively up as new materials, heads and techniques are applied to the problem. Some industry experts think that 100 Gb/in^2 will be reached before magnetic recording in the current sense will have to be replaced.

Data is recorded by passing a current through a tiny coil in the record head to generate a magnetic field of sufficient strength to overcome the coercivity of the recording material. Until a few years ago an inductive head was used to read back the information: the fluctuations in the magnetic field as the magnetised bits pass beneath the read head induce a current in the sensing coil. With reducing bit sizes, progressively weaker magnetic fields are available for the read head to sense and during the 1990s inductive read heads were abandoned in favour of magneto-resistive (MR) technology. MR heads use a material such as an NiFe alloy, which has the property that, placed in a magnetic field, its resistance changes with the strength of the magnetic flux. In an MR head such a material is used to form a conductor placed perpendicular to the plane of the recorded medium. A sense current is passed through the conductor, and the signal appears as a voltage drop proportional to the strength of the magnetic field.

magneto-resistive read head

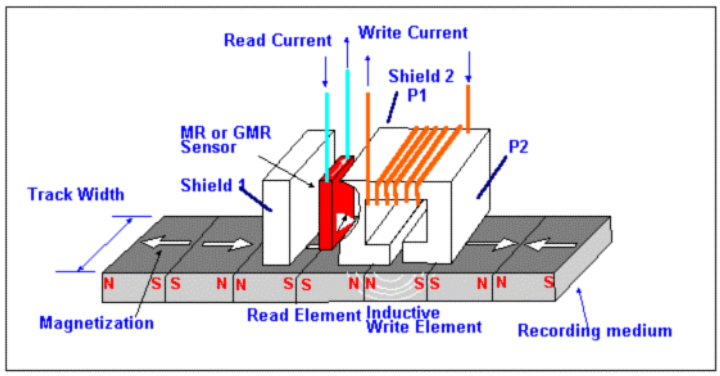


$\Delta R \alpha H$

$\Delta V \alpha I \Delta R$

While inductive heads detect the rate of change in the magnetic field, MR heads sense directly the field strength. The output signal strength is proportional to the sense current, giving several advantages in overcoming noise in high density, high data rate systems. MR heads have an even greater advantage over inductive sensors in low-velocity recording applications such as low power disk, and magnetic tape.

An MR read head still needs an inductive head for writing and erasing. The MR element is sandwiched between two magnetic elements that shield it from the field of neighbouring recorded bits - a bit like the way the tube of a telescope keeps out stray light from the surroundings. To reduce the weight on the accessor the dual head is often manufactured as a single thin film composite head.

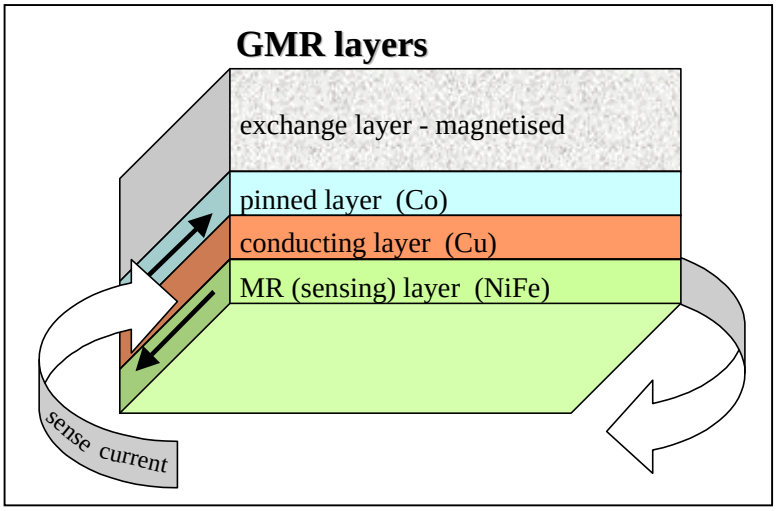


picture: IBM Research - Almaden

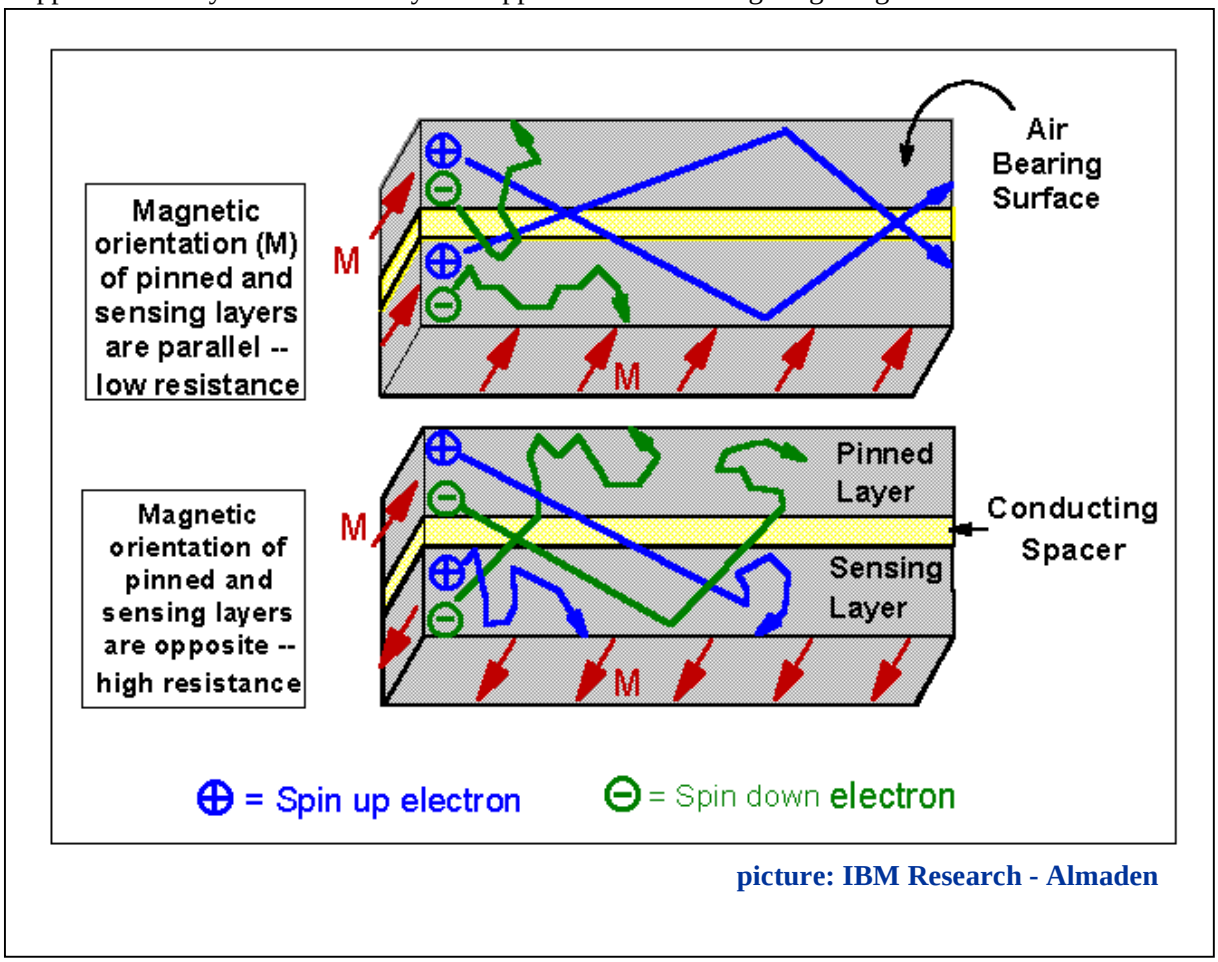
Composite head: inductive write, MR read

Today's most advanced heads use the Giant Magneto-Resistive (GMR) effect. GMR sensors are made of four layers. The *sensing*, or *magneto-resistive* layer changes magnetic orientation according to the field of the recorded material. This is covered by a *conducting* layer, separating it from the *pinned* layer, which has a fixed magnetic orientation maintained by the permanently magnetised *exchange* layer.

The sensing current passes through the first three layers, electrons moving freely within these layers. The physical mechanism used depends on the electron spin. Electrons with a direction of spin parallel to the magnetic field encounter low resistance, while electrons with a spin opposite to the magnetic orientation of the material encounter higher resistance.



Because a GMR head exploits the different behaviour of conduction electrons with spin parallel to or opposed to the magnetic orientation of the field it is also referred to as a *Spin Valve*. The following diagram explains the operation in more detail. When both pinned and sensing layers have the same magnetic orientation, some electrons are stopped in both materials, while electrons of opposite spin flow freely in both materials. On the other hand, when the pinned and sensing layers have opposite orientations, electrons of both spins tend to be stopped when they flow into the layer of opposite orientation - giving a higher resistance.



picture: IBM Research - Almaden

The effective density of recording is measured as the number of bits per square inch of recording surface. Today the bits are much wider than they are long because the narrowness of the track is limited by various mechanical factors concerned with positioning of the heads and track-

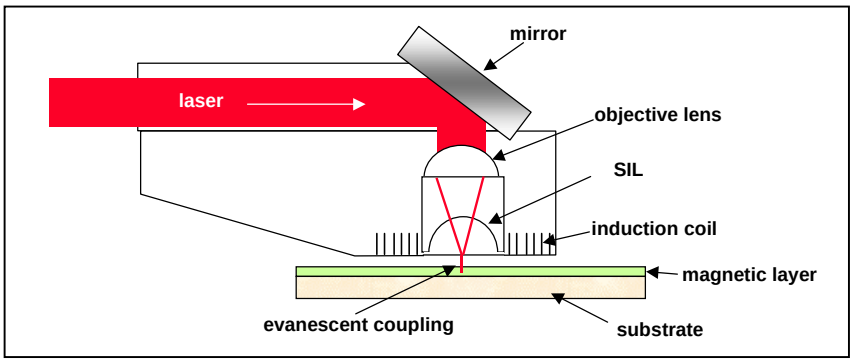
following. The length of the bit is less constrained, limited only by the electro-magnetic properties mentioned above, and the size and sensitivity of the head. At an areal recording density of 10 Gb/in^{2,} the bit is about 1 micron wide and 0.06 microns long.

The speed with which data can be recorded and read back from a hard disk is proportional to the linear recording density (number of bits per inch measured along the track) and the rotational speed of the disk. The linear density is increasing at about the square root of the annual rate of increase of the areal density, or some 26% per year. However, the rotational speed has increased by barely 50% in the past ten years. We shall therefore see a growing discrepancy between data rate and disk capacity. It has not been found practical or cost-effective to use parallel heads to increase the data transfer rate, and this problem is more likely to be alleviated by the continued decrease in the overall size (capacity) of the disk unit.

Before beginning to transfer data from the disk the head must be positioned over the correct track (*seek* time), and then we must wait for the beginning of the data segment to come under the head (called the *latency*). The *seek* time has hardly improved by 50% over the past ten years. The *latency*, the time taken for half a revolution of the disk, has also not improved very much in that time. Without a data cache in the disk controller this would lead to very poor performance for small data transfers. The data cache, several MBytes in size in current disks, uses techniques like read-ahead buffering, and speed matching between the disk and the data channel to improve performance.

There is some way to go in further developing the current hard disk technology, before the physical limits are reached, but several alternative technologies are being explored which could ultimately lead to much higher recording densities. One such line of development uses materials developed for magneto-optical devices. These are composites of rare earths and transition metals (such as GdFeCo, TbFeCo), which have very stable magnetisation at normal operating temperatures, and a low Curie point[2]. The material is heated above its Curie point by a laser, and then magnetised using an induction coil. The data is read back using the conventional magneto-optic technique - the magnetic field changes the polarisation of the reflected laser beam (Kerr effect).

However, the recording density of laser-based systems is limited by the resolution of the optics to about one half of the wavelength, which is 0.35 microns for red light. A novel technique used to reduce the laser spot size combines a *Solid Immersion Lens* (SIL) with the *near field* effect, or *evanescent coupling*. The SIL, based on work performed at Stanford University, focuses the laser beam internally on the surface of the lens, which is made from a material with a very high refractive index. The energy of the laser is then transferred between the lens surface and the recording layer using the near field effect which operates within one wavelength of the radiating surface. The TeraStor Corporation has demonstrated a disk using this technique.



[2] *Curie point*: The temperature above which a material loses its magnetic orientation

2 MAGNETIC TAPE

Magnetic tape has traditionally been used as an inexpensive storage medium for backup of disk data, and for archiving old versions of files. It used to be a clear two orders of magnitude cheaper than disk, and had excellent shelf storage characteristics in terms of volumetric storage density, and ruggedness. More recently, for the past fifteen years or so, large reliable tape robots have enabled gigantic quantities of data to be available cheaply with an access time of a few minutes. StorageTek, the leader in the development of high volume tape robots, uses the name Nearline to describe this class of storage. However, in recent years, hard disk and optical storage have begun to provide strong competition for magnetic tape in terms of both volumetric storage density and cost.

Magnetic tape is made from a thin flexible substrate, about 10 microns thick, made from Polyethelyne[3]. The recording layer used by most high quality magnetic tapes today is microscopic metal particles held in a binder which coats the substrate - MP tape. Metal Evaporated (ME) tape is manufactured using a process in which the substrate is passed through cobalt or cobalt-nickel vapour, which condenses in a thin layer on the surface. ME tape has better recording characteristics at very high frequencies.

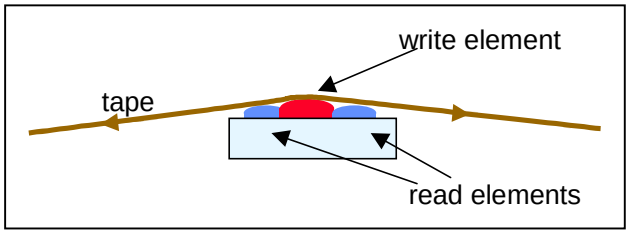
The tape is usually stored in a single-reel cartridge. When mounted on a drive the end of the tape is pulled out of the cartridge and attached to a take-up spool. The drive then moves the tape back and forth past the recording heads by controlling the cartridge reel and the take-up spool. The tape path is carefully designed to minimise friction and tape tension. Some systems use cassettes where both reels are mounted in the container. This avoids extracting the tape and is usually much faster in loading. However, the cassette hold less tape than can be packed into a single reel cartridge.

The recording and read technologies use developments from the higher performance and higher density hard disk industry, but there are major mechanical problems with tape which are not present in hard disks.

Magnetic tape is basically a sequential access medium. The tape is loaded at the beginning, and must then be positioned by spacing across intervening data until the desired position is found. This is not entirely true, as modern tapes are recorded in parallel bands recorded along the length of the tape (as we shall see later), but data written on a tape cannot be deleted, or re-written (there is no update possibility). New data must be appended to the current end of data. Most tapes maintain a directory that is written in a special area at the start of the tape, and updated before the tape is dismounted. This contains the addresses of the beginning of each file on the tape, and allows the drive to use positioning information in pre-recorded servo tracks to space relatively quickly to the start of a specific file.

There are two basic techniques for recording data on magnetic tape: *linear*, where the tracks are recorded parallel to the length of the tape; and *helical*, where the tracks are recorded at an angle across the width of the tape.

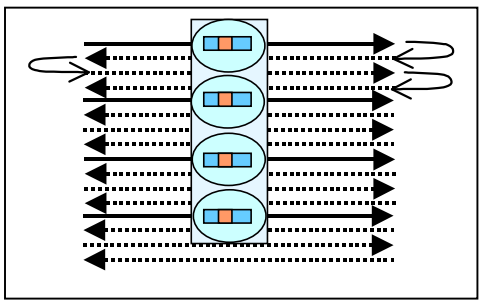
2.1 Linear Recording



In linear tape recording, the tape passes over a fixed head, which usually contains several sets of recording elements to enable several tracks to be recorded or read back in parallel in order to increase the effective data transfer rate. Each set comprises a write and read element, so that recorded data can immediately be re-read and checked for errors. In order to support bi-directional recording, a third element (read) is required on

[3] PET: polyethylene terephthalate, or PEN: polyethylene naphthalate

the other side of the write element. Erase elements may also be required. In linear recording the tape barely touches the head, reducing wear of the head elements and the risk of damage to the tape.

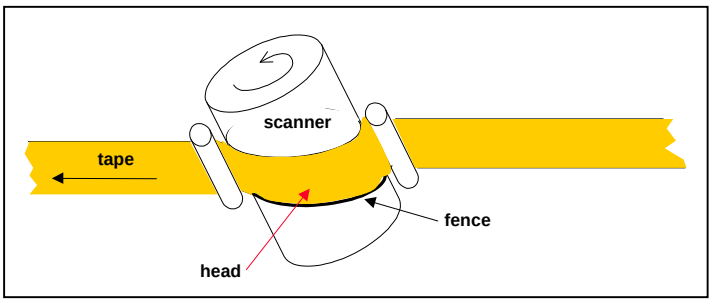


The multi-track head writes a series of parallel tracks along the tape until the end-of-tape marker is detected. Then the head is moved slightly across the tape, and recording proceeds with the tape running in the opposite direction. This continues until all of the data bands have been written. Since the head elements are generally wider than the track, track groups actually comprise a set of parallel tracks, separated by tracks that belong to another track group. This multi-track-group method of recording is called *serpentine* recording.

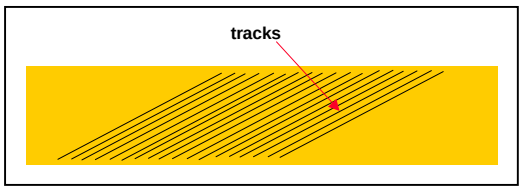
There are a number of practical problems that must be addressed by the designers and manufacturers of magnetic tape systems. Friction between the tape and head leads to head wear and the risk of tape damage. Dust and debris from the slitting process used to cut half-inch tapes from broad strips of medium can exacerbate this problem. The path followed by the tape as it moves from one reel to the other must be as simple as possible, to ensure that minimum tension is required to maintain the extremely thin medium in contact with the heads. The capstans in the path usually employ air bearings to reduce friction. As the temperature and humidity change, so the substrate expands and contracts. The lateral expansion/contraction of the tape poses a major problem for parallel track recording, as the distance between tracks may vary between the time the tracks are written and read. Finally, when the tape is stored on a shelf or in a robot slot gravity slowly distorts the reel of tape, making it more difficult for the tape guidance system to position it. Tape should always be stored vertically, and should be re-spooled regularly to minimise this problem.

2.2 Helical scan recording

Helical scan recording was developed for the entertainment business, in order to obtain higher recording densities - both higher track densities and higher linear densities.



In these systems (used in the mass market VCR) the tape is partially wrapped around a scanner mounted at an angle to the direction of the tape. The tape moves slowly past the scanner, which contains a set of head elements that spin at a much higher speed. The effective relative speed of head and tape is therefore increased without incurring the mechanical difficulties of moving the tape along the tape path at high speed. Data transfer rate can be maintained by increasing the rotational speed of the scanner or by including more than one set of recording elements in the scanner. But it is not necessary to resort to parallel track recording.



There are some fundamental problems encountered with helical scan technology, which make it more difficult to achieve the level of data integrity that can be obtained with linear techniques. The main problems are:

- Head wear due to the increased contact pressure coming from the tape wrap on the scanner (capstan effect).

- The helical wrapping around the scanner is achieved by riding the tape on a fence or step that protrudes from the scanner. This requires that the tape edge has been formed very accurately during slitting, and there is a high risk of edge damage. Once the edge is damaged the recorded tracks develop a curvature which makes it difficult for the read head to follow. In contrast, linear tapes reserve a guard band at each edge of the tape to minimise problems arising from manufacturing or usage damage.

Helical scan tape systems achieve today substantially higher track densities than that possible with linear recording methods. For example, the StorageTek *Redwood* drive achieves 2,800 tracks per inch, enabling it to record 50 GBytes of user data on 1,200 feet of half inch tape. In contrast the 9840 linear tape system from the same company only stores 20 GBytes of user data on a 900 foot tape, at about 600 tracks per inch. New techniques adapted from hard disk technology will allow linear recording to narrow this gap.

2.3 Data compression

One significant advantage of sequential recording systems like magnetic tape is that data compression can be used by the recording system to improve the apparent recording density. This is not possible with a random access system like hard disk, where the physical address of any data block must be able to be computed directly from its logical position in the file. Early compression systems used algorithms that were effective for commercial data streams, but gave little advantage in the case of the random patterns encountered in scientific data or data where straightforward compression techniques had already been applied - such as raw physics events.

More recently, the availability of inexpensive but high performance processors has enabled more sophisticated compression algorithms to be used. In common use today is DLZ1. This is one of the set of Lempel-Ziv compression algorithms which map variable length input strings to variable length output symbols. During compression, the algorithm builds a dictionary of strings which is accessed by means of a hash table. Compression occurs when input data matches a string in the table and is replaced with the output symbol. Advanced DLZ1 algorithms perform well even on pre-compressed scientific data.

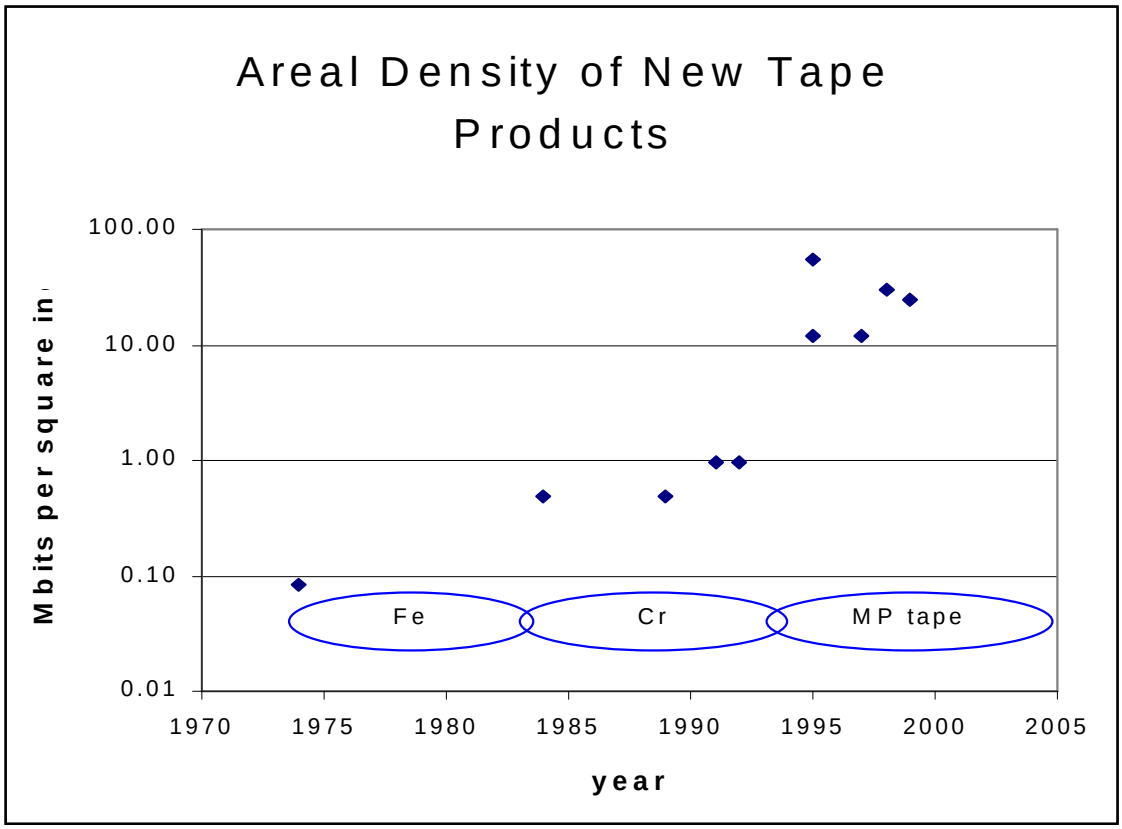
2.4 Technology trends

Unlike in the case of the hard disk market it is rather difficult to identify clear trends in magnetic tape developments. Historically there have been long periods in which a single standard and recording density has dominated the market. During the past 25 years, as can be seen from the diagram below, there have been three distinct generations of data centre quality tape systems.

- the 9-track "6250 bpi" technology using iron oxide which was introduced in 1974;
- the 3480/90 technology, using 18 or 36 tracks on a chrome dioxide medium, which was first introduced in 1984;
- in 1995 the first of a number of new products appeared, all incompatible, but using MP tape, and giving a leap from 800 MByte cartridges to multi-GByte systems

Helical scan has proven to be too difficult a technology for the quality required at the high end - data applications have a completely different definition of reliability from video applications, where helical scan is the standard. Products with acceptable reliability for digital applications are unacceptably expensive. We are not likely to see this change, as linear technology is steadily achieving higher areal densities.

The major applications for magnetic tape are backup and archiving - which require a different level of reliability from applications which keep the master copy of the current data on tape - like high energy physics applications. The model of using tapes as a storage medium for active data needs to be reconsidered.



Since it is difficult to generalise about industry trends, we have to look at specific manufacturers and products in order to have an idea about how things may develop over the next 5 years. The situation at the top end of the market is quite satisfactory, with three or four competitors using proven linear technology aimed at the data centre. The manufacturers have announced “road maps”, explaining how they can realistically develop their systems to improve the capacity and performance by a factor of 2-4 over the next 5 years, without requiring fundamental changes in technology.

At the low end of the market the major products at present use helical scan. As we have seen this is a more difficult technology than linear recording. More important however is the competition that these products, aimed at the mass market, will experience from optical systems.

3 OPTICAL RECORDING

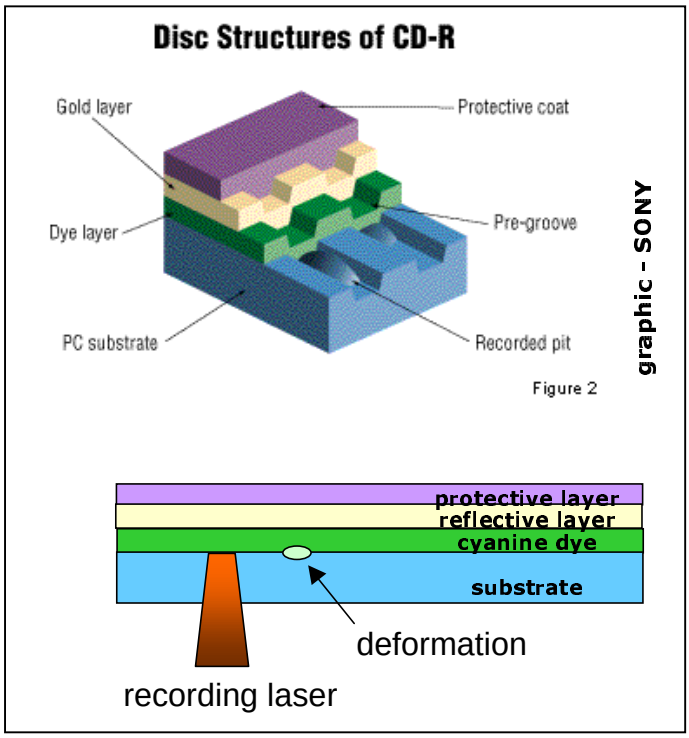
For over twenty years claims have been made that optical recording will displace magnetic recording for secondary or at least tertiary storage. The reason is (or was) that optical systems have the potential for much higher areal densities - up to 5 Gbits per square inch in the case of a red laser. There have been many products using optical techniques, almost all write-once systems, but the achieved density (user data bits per square inch) has been disappointing, performance modest, and costs for media and drives too high. Recordable optical disk was used only for special applications - such as those where the fact that it was write-once was an advantage. Meanwhile magnetically recorded hard disk has steadily improved in density, performance and cost, to the point where it has now exceeded the maximum recording density of purely optical systems.

Paradoxically, although optical recording has lost this fundamental advantage, for the first time we are beginning to see real competition, at least for magnetic tape, coming from optical products. The reason is that recordable CDs and DVDs, products emerging from the mass market of entertainment, have achieved reliability and price levels which place them in direct competition with

low end tape systems. In addition, as these are random access systems, they offer a substantially different (better) level of functionality.

3.1 Write Once - CD-R and DVD-R

Recordable (write once) CDs and DVDs use a disk made from a polycarbonate substrate. The recording layer uses heat-sensitive material, usually Cyanine dye. The heat of the recording laser chemically decomposes spots in the dye layer and physically deforms the adjacent substrate. During playback, the decomposed spot absorbs light from the playback laser, and the substrate deformation scatters the light, together making the spot look like a pit which can be read by a CD or DVD ROM drive. The disk is mechanically pre-formatted with a flat-bottomed helical groove. The ridge between adjacent grooves is called the land, which has a flat surface. Address information (sector headers, etc.) is embossed on the disk during manufacturer.



The disks can be made double sided, and DVD-R disks can store up to 4.7 GBytes on each side. This technology does not support the dual layer per side technique used by mechanically formed DVD-ROM disks. The recording speed of current products is about 1.3 MBytes/second.

3.2 Erasable DVD-RAM

Re-writeable optical disks use phase change technology. The recording layer consists of an alloy of tellurium, germanium and antimony. A laser beam heats the material above the 600°C melting point. The layer cools to a crystalline or amorphous state depending on the power level of the laser. With high power the material melts quickly and cools quickly forming an amorphous spot. At lower power the process is slower and the surface assumes its crystalline state. When reading takes place the amorphous spots have much lower reflectivity than the crystalline spots - so they look like “pits”.

The DVD-RAM format uses a substrate with pre-formed groove and sector headers. Data is recorded in the groove and on the “land”, or ridge between the grooves, to increase the capacity. This involves re-focussing the laser. The original DVD-RAM format has a capacity of 2.6 GBytes per side, but a second version of the standard has been agreed by the DVD Forum which uses a reduced spot size to hold 4.7 GBytes per side.

BIBLIOGRAPHY

The Complete Handbook of Magnetic Recording, Finn Jorgensen, Tab Books 1995, ISBN 0-07-033045-X

Magnetic Storage Handbook - Second edition, C. Denis Mee, Eric D. Daniel, McGraw-Hill 1996, ISBN 0-07-041275-8

Magnetic Recording Technology, C. Denis Mee, Eric D. Daniel, McGraw-Hill Publishing Company 1996, ISBN 0-07-041276-6

Digital Magnetic Recording, Albert S. Hoagland, James E. Monson, John Wiley & Sons 1991, ISBN 0-47-140144-7

DVD Demystified, Jim Taylor, McGraw-Hill 1997, ISBN 0-07-064841-7

SOFTWARE BUILDING

A.N.Dunlop, B.Ferrero Merlino, R.Jones, M.Nowak, Z.Szkutnik

Overview
This track combined software engineering lectures with exposure to the software technologies and packages relevant for LHC experiments. It showed, in a practical sense, how software engineering can help in the development of HEP applications based on the LHC++ software suite and also gave a taste of working on large software projects that are typical of LHC experiments. The lectures provided an overview of LHC++ and covered those aspects of software engineering most relevant for HEP software development.

1. EXERCISES

The hands-on tutorial introduced a series of exercises to solve given problems. The tutorials followed the natural progression of physics analysis exploring the major LHC++ packages on the way. The students completed the tutorials in groups of two.

Essentially, the students were required to develop several C++ programs in succession starting from skeletons:

i) Populate an event database using HepODBMS according to a defined object model.

ii) Build an event tag database from data prepared in I. Select some interesting event attributes and copy them to the event tag database.

iii) Use the Gemini minimisation package to find the minima values for a given set of problems.

iv) Read event tag database built in II and display the contents. Use the LHC++ inter-active graphical tools to apply more cuts.

2. LECTURES

2.1 Track Introduction

Speaker: R.Jones

The goal of this track was to provide an overview of LHC++, which is a comprehensive, mainstream and modern software suite for development of physics analysis software. This overview was coupled with an introduction to those aspects of software engineering that are considered relevant for physics analysis software development. The exercises provided practical hands-on experience of using the major LHC++ packages. The intention was that students should come away with practical knowledge of how to develop physics analysis software in an organised manner. It was not a goal of this track to teach C++, object-oriented concepts, WNT or give details of the physics involved in LHC experiments.

The LHC++ lectures are documented as separate papers in these proceedings. Below is a short summary of the track introduction, closing and software engineering lectures.

2.1.1 Introduction to Software Engineering and OO methodologies

Speaker: A.N.Dunlop

A definition of what is meant by software engineering gave a starting point for this lecture which then went on to explain how the scale of the software project determines the software process required to successfully run the project to completion.

Various processes exist for OO software (OOSE, OMT, Booch, Fusion, Martin-Odell, Unified etc.) and have varying definitions for the phases involved during the project but the Unified process is centred around the architecture and follows a number of iterations driven by use-cases.

2.1.2 An Overview of UML and use-cases

Speaker: A.N.Dunlop

This lecture covered the basic structure of UML, the notation and types of diagrams that can be used to describe the software under development. Emphasis was put on the use cases as a means of driving the development and how they are used at various phases.

2.1.3 Software Design

Speaker: R.Jones

The task of design was introduced as consisting of three levels: architecture, mechanistic and detailed based on the scope of the decisions made. Each level was further defined to show its goals, techniques and deliverables. The UML class, sequence and collaboration diagrams were explained and examples drawn from the exercises. The concept of patterns, how they can be applied to analysis and design and examples from LHC++ packages and the exercises were given.

2.1.4 Software Testing

Speaker: R.Jones

This lecture covered the basic principles of software testing and \why programs have defects. The cost of defect removal and the classification of defects were addressed. The use and basis of software inspections as a means for removing defects was shown to be the most effective way of improving the quality of software. The different types of testing (unit, integration, regression and acceptance) were described and how CASE tools can help in these tasks. The lecture finished with a set of axioms about testing that can improve the way most software developers approach the subject.

2.1.5 Wrap-up on Software Engineering Issues

Speaker: R.Jones

This aim of this lecture was to look at some aspects that affect the long-term well being of development projects. It started by asking three questions:

• Why is the software process so important?

• What is so good about iterative development anyway?

• Why can’t we just get on with writing the code?

To answer the first question, the most common reasons for failure of software projects were listed with how activities, such as adequate analysis and design, can be used to avoid them. The second question was addressed by giving an example of what iterative development means and by showing the unfortunate results of not using it.

Hopefully the students understood that by answering the first two questions the answer to the third becomes clear. As a means of supporting iterative development cycles, configuration management systems were introduced and the lecture finished by emphasising that software always costs something (time or money): either *some* up-front by investing in analysis and design or *more* later to fix all the problems.

2.2 Feedback session

The track finished with a feedback session during which the students asked questions about how to apply the software engineering techniques in different situations, including how to introduce software inspections and how to motivate developers to worry about those issues concerned with software maintenance. There were detailed questions about the use of Objectivity and Gemini in LHC++.

READING LIST

Software Engineering

1. R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw Hill, 4th Edition, 1996, ISBN: 0070521824
2. H.E. Eriksson & M. Penker, UML Toolkit, Wiley, 1998, ISBN: 0471191612
3. M. Fowler, Uml Distilled: Applying the Standard Object Modelling Language, Addison-Wesley, 1997, ISBN: 0201325632
4. Software Engineering Resources http://www.rspa.com/spi/

THE LHC++ ENVIRONMENT

Bernardino Ferrero Merlino
CERN IT Division - CH 1211 Geneva 23 Switzerland

Abstract
The LHC++ project is an ongoing effort to provide an Object-Oriented software environment for future HEP experiments. It is based on standards-conforming solutions, together with HEP-specific extensions and components. Data persistence is provided by the Objectivity/DB Object Database (ODBMS), while the IRIS Explorer Visualization system is the foundation for the Interactive Analysis environment. To complement the standard package, a set of C++ class libraries for histogram management, ntuple-like analysis (based on Objectivity/DB) and for presentation graphics (based on Open Inventor) have been developed.

1. INTRODUCTION

Over a period of many years, CERN, in conjunction with other laboratories, built up a large collection of routines and programs oriented towards the needs of a physics research laboratory. This software – almost entirely written in Fortran, is referred to collectively as the CERN Program Library or CERNLIB [1]. For many years, it was assumed that CERNLIB would simply be migrated from Fortran 77 to Fortran 90. However, in the early ’90s an important change took place, namely the adoption of object-oriented techniques and programming languages such as C++ and – more recently – Java. As a result of these changes, the need for the “C++-equivalent of CERNLIB” arose. The LHC++ project was initiated in 1995 to address these issues. Given the falling manpower envelope of the laboratory, it was clear that there would be insufficient resources to develop and support everything in-house and so alternatives, such as collaborative development and the use of commercial components, were investigated.

The current LHC++ [2] strategy relies on both commercial and HEP-specific components. It’s noteworthy that the LHC++ environment is built using a ‘layered’ approach, where all basic functionality are implemented as standalone C++ class libraries that are then integrated using a more sophisticated Modular Visualization System (MVS). A sketch of the LHC++ components is given in Table 1 below:

Description	Components
Data Analysis	IRIS Explorer - HEPExplorer
Custom graphics	MasterSuite - HEPInventor
Basic graphics	OpenInventor - OpenGL
HEP math	HEPFitting – GEMINI - CLHEP
Basic math	NAG C library
Histograms	HTL
Database	HepODBMS
Persistency	Objectivity/DB
C++	Standard Libraries (STL)
HEP specific	CLHEP

Table 1 - LHC++ Components

2. LHC++ COMMERCIAL COMPONENTS

Many factors contributed to the choice of the commercial components of LHC++. These included the functionality of the individual packages, their adherence to standards – either *de-facto* or *de-jure* – their interoperability, their market share (including other HEP laboratories) and of course cost! Several of the suppliers chosen already had a long-established relationship with CERN from previous software packages and the systems themselves were "interrelated". This is important as it not only guarantees their interoperability but simplifies the issues related to ensuring consistent releases across multiple platforms – these issues having been already addressed by the vendors concerned.

2.1 Objectivity/DB ODBMS

In order to study solutions for storing and handling the multi-PB data samples expected with LHC, the RD45 Project [3] was established in 1995. The proposed solution should also be able to cope with other persistent objects, such as histograms, calibration and monitoring data, and so forth. It was found that the best candidate for handling this problem is an Object Database Management Group (ODMG) [4] compliant object database used together with a mass storage system, based upon the IEEE reference model for mass storage systems [5]. After considering a few alternatives, the presently favored solution is built upon Objectivity/DB [6] and HPSS (High Performance Storage System) [7].

2.2 IRIS Explorer

IRIS Explorer [8] is a toolkit for visualization of scientific data, which can be manipulated via visual programming tools. Users define their analysis application by connecting building blocks, called modules, into a so-called map (see Figure 1 below). Modules act like filters: they read one or more streams of input data and produce one of more streams of output data. The behavior of modules is controlled (interactively) by a set of parameters. IRIS Explorer comes with a rather complete set of modules for performing basic data transformations and it is straightforward to create new modules. IRIS Explorer is built on top of recognized graphics standard such as OpenGL [9] and Open Inventor [10], thus making possible to integrate third party packages based on the same standards, e.g. GEANT-4 [11].

2.3 OpenGL

OpenGL is an industry standard for graphics. It is vendor-neutral and multi-platform, and is optimized for building environments for developing 2D and 3D visual applications. Several vendors already offer a hardware implementation of the standard, thus ensuring that rendering speed will be optimal.

2.4 Open Inventor

Open Inventor is an object-oriented 3D toolkit built on top of OpenGL, providing a comprehensive solution to interactive graphics programming. Its programming model is based on a 3D scene database optimized to ease building graphics applications. It includes a large set of objects, such as cubes, polygons, text, materials, cameras, lights, track-balls, handle boxes, 3D viewers, editors and defines a standard file format (IV) for 3D data interchange files, that is the basis for the Virtual Reality Modeling Language (VRML) [12] standard.

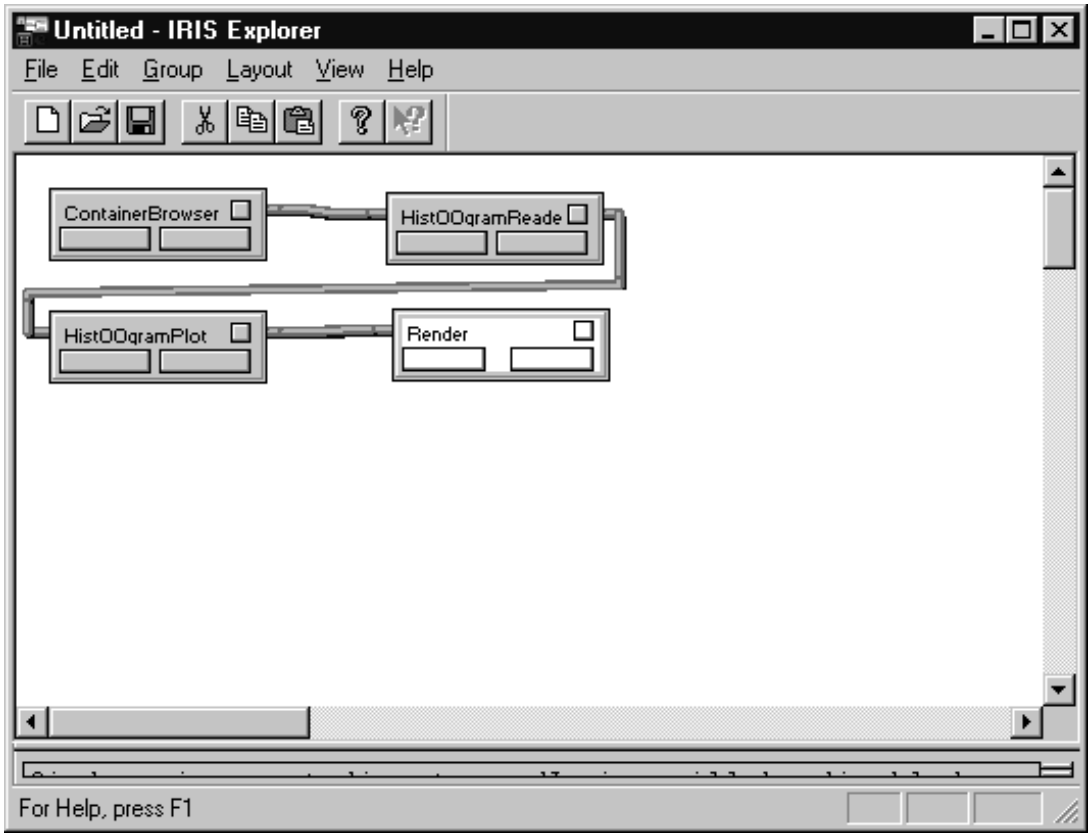


Figure 1 – example of a "Map" in IRIS Explorer

2.5 MasterSuite

MasterSuite [13] is a C++ toolkit for data visualization, containing class libraries with extension nodes to Open Inventor. These extensions cover both 2D (drawing, charting, etc.) and 3D (drawing, legends, etc.). In addition, it provides a set of classes to develop viewers for scientific data for output on screen as well as in vector-PostScript format.

2.6 NAG C Library

The NAG C Library [14] is a collection of about 400 user-callable mathematical and statistical functions. The library includes facilities in the area of minimization, ordinary differential equations, Fourier transform, linear algebra, zeros of polynomial, statistics, time series etc. The library uses double precision throughout to ensure maximum accuracy of results. The correctness of each library function is evaluated and verified by specially written test programs performed on each of the machine ranges for which the library is available.

3. LHC++ HEP-SPECIFIC COMPONENTS

Although the commercial components on which LHC++ is built offer a solid foundation, they do not – in general – provide the complete functionality that is required in the HEP community. To cater for such needs, small extensions – typically some 2-3K lines of code – are provided.

3.1 HepODBMS

HepODBMS [15] is a set of class libraries built on top of the ODMG C++ interface. Their purpose is to provide a higher level interface than is specified by the ODMG, to simplify the porting of existing applications and provide a minimum level of support for transient-persistent switching. Furthermore, these libraries help to insulate applications against changes between releases from a given vendor and between the products of different vendors. The HepODBMS libraries provide classes to deal with session management, clustering hints, tag and event collections.

3.2 The Histogram Template Library (HTL)

The Histogram Template Library (HTL) [16] is a C++ class library that provides powerful histogram functionality. As the name suggests, it exploits the template facility of C++ and is designed to be compact, extensible, modular and fast. As such it only deals with histograms (summary data representing the frequency of values) and not with the whole set of values. Furthermore, although simple file-based I/O and "line printer" output are supported, it is not coupled to more advanced I/O and visualization techniques. In the context of LHC++, such capabilities are provided by other components that fully interoperate with HTL.

HTL itself offers the basic features of HBOOK [17] as well as a number of useful extensions, with an object-oriented (O-O) approach. These features include the following:

- booking and filling of 1D, 2D and profile histograms;
- computation of statistics such as the mean or r.m.s of a histogram;
- support for operations between histograms;
- Browsing of and access to characteristics of individual histograms.

3.3 HEPInventor

HEPInventor [18] proposes an easily understandable and user-friendly way to present data in physics programs. It is implemented as a graphical class library built on top of MasterSuite to provide an interface between HTL and its presentation graphics.

3.4 HEPExplorer

HEPExplorer [19] is a set of HEP-specific IRIS Explorer modules, which help a physicist set up an environment to analyze experimental data, produce histograms, fit models and prepare data presentation plots. IRIS Explorer Maps that implement simple analysis-related tasks, such as visualize and fit a histogram, produce histograms out of tag data (see section 4), etc. are part of the package as well.

3.5 Gemini/HEPFitting

Gemini [20] is a class library providing basic minimization/fitting capabilities. The library integrates under the same interface both MINUIT [21] and NAG minimizers, although classes are provided to access features that are unique to one minimization engine (such as NAG support for linear and non-linear constraints).

HEPFitting [22] is a utility library to fit either HTL or vectors of data, with a handy interface to specify complex fit functions assembling gaussian, polynomial or exponential terms, as well as user defined functions.

3.6 CLHEP

A set of HEP-specific foundation and utility classes such as random generators, physics vectors, geometry and linear algebra is packaged in the CLHEP class library [23]. CLHEP is structured in a set of packages independent of any external package (interdependencies within CLHEP are allowed under certain conditions).

4. ANALYSIS SCENARIO

The analysis scenario can be split in two parts. The first part concerns populating the database with reconstructed event data and is usually done in a C++ program, typically running in batch jobs. The second part implies using an interactive tool, such as the IRIS Explorer framework, to actually produce summary data, usually as histograms, out of the event data. Histograms can then be manipulated, fitted using an appropriate tool and eventually printed in a PostScript file to embed in a paper or a slide presentation.

4.1 The 'batch' part

The main task of this part is populating the Objectivity data store with event information coming, very likely, from a former reconstruction phase. Most new HEP experiments assume that it will be possible to make both raw data and reconstructed data available on-line thanks to the integration between Objectivity/DB and HPSS. Each experiment will have its own data model and physicists should be able to navigate through it. This is a major problem for a general-purpose Interactive Analysis environment, since, unlike the Ntuple case, a common and pre-defined data model, shared amongst all experiments, is no longer imposed.

Since all data needed for analysis is supposed to be on-line, the role of the Ntuple replacement could be quite different. While reasonably small personal data collections will still exist, the main concern will probably be how to index large event stores to speed up the analysis.

The RD45 Project suggested one approach to deal with both problems. The idea it to speed up queries by defining for each event a Tag, i.e., a small set of its most important physics attributes plus an association with the event where the Tag data come from. A collection of tag objects is saved together in a Tag Database, something intermediate between an Event Directory and an Ntuple. Since they are globally defined for the whole experiment, concrete tags can be optimized so that they offer a very efficient way to make initial cuts on attributes, thus achieving a high degree of selectivity. On top of that, at any moment users can cross the association to the event to retrieve any other details about the full event, which are not contained in the Tag.

In general the experiment or group will make the selection of key attributes characterizing events, so that concrete tags are mostly defined for experiment-wide or workgroup-wide data sets. However, individual physicists have the possibility to define their own simpler data collection by using the Generic Tag mechanism. This second lightweight procedure allows users to define a tag on the fly, without creating a persistent class. Compared to the concrete tag, there is, of course, a small performance penalty, but this is most of the time balanced by an increased flexibility, since at any time new fields can be added to the tag and the association to the complete event data remains available.

The set of individual tags is called an Explorable Collection, i.e., a collection of objects implementing an interface for access from IRIS Explorer.

4.2 An example: creating a Tag collection out of existing events

The Event we want to create the Tag from is composed by two kinds of information:

- Tracking information, represented by a variable size array of tracks
- Calorimeter information, represented by a variable length array of clusters

```
/// persistent Tracker class
class Tracker : public ooObj {
public:
ooVArrayT<Track> tracks;
private:
};
/// persistent Calo class
class Calo : public ooObj {
public:
ooVArrayT<Cluster> clusters;
};
/// persistent Event class
class Event : public ooObj {
private:
  int evtNo;
public:
  d_Ref<Tracker> tracker;
  d_Ref<Calo>    calo;
};
```

So, for each event, we will have a collection of tracks and a collection of clusters, plus a unique event identifier.

The classes implementing a single track or cluster will contain information related to the particle traversing the two sub-detectors:

```
// Basic track: persistent by embedding
class Track {
public:
double getPhi()   { return phi;}
  double getTheta() { return theta;}
  double getPt()    { return pt;}
private:
  double phi;
  double theta;
  double pt;
};
// Basic cluster: persistent by embedding
```

```
class Cluster {
public:
 double getPhi()    { return phi; }
  double getTheta()  { return theta; }
  double getEnergy() { return energy; }
private:
  double phi;
  double theta;
  double energy;
};
```

The tag we want to create will contain the pT and phi attribute of the tracks having maximum and minimum pT, plus the event unique identifier. Hence the Tag description will be something like:

```
HepExplorableGenericTags highPt; // create a tag collection
// define fields all fields that belong to genTag
   TagAttribute<long>  eventNo (highPt,"eventNo");
   /* track with highest pT*/
   TagAttribute<double> ptPlus (highPt,"ptPlus");
   TagAttribute<double> phiPlus(highPt,"phiPlus");
   /* track with lowest pT*/
   TagAttribute<double> ptMinus (highPt,"ptMinus");
   TagAttribute<double> phiMinus(highPt,"phiMinus");
```

It’s now possible to scan the events, identify the tracks with minimum/maximum pT and replicate their pT and phi attributes in the Tag:

```
ooItr(Event) eventItr;
eventItr.scan(container("Events"));
while( eventItr.next())
  {
   HepRef(Tracker) aTracker = eventItr->tracker;
   int maxTrack = 0,  minTrack = 0;
   for (int track=0; track < aTracker->getNoOfTracks(); track++) {
    if (aTracker->tracks[track].getPt()
       > aTracker->tracks[maxTrack].getPt() )
      maxTrack = track;
    if (aTracker->tracks[track].getPt()
       < aTracker->tracks[minTrack].getPt() )
      minTrack = track;
   }
highPt.newTag();      // create a new tag (all fields have default values)
 eventNo  = eventItr->getEventNo();
 ptPlus   = aTracker->tracks[maxTrack].getPt();
 phiPlus  = aTracker->tracks[maxTrack].getPhi();
 ptMinus  = aTracker->tracks[minTrack].getPt();
 phiMinus = aTracker->tracks[minTrack].getPhi();
}
```

It is noteworthy that the Tag’s attribute are managed exactly as standard C++ variables: the overloaded assignment operator will take care of putting the values in the Tag that will be stored in the database.

4.3 The interactive part

Interactive Analysis in IRIS Explorer is implemented by a set of HEPExplorer modules. Generally speaking, the current set of modules allows users to extract data from an Objectivity/DB data store and put them in one or more HTL histogram(s). In particular the user can select an Explorable Collection, define a set of cuts over the collection as a C++ expression, define the input streams for the HTL histogram(s) to produce and automatically generate and compile C++ code that implements the cuts and fill the histograms.

Apart from accessing the data in the tag, users can invoke C++ functions that implement, e.g., common physics or access the experiment specific event object (by traversing the association between a tag and its related event). User-defined functions can be used whenever a C++ expression is allowed. This means, for example, that reconstruction C++ code can be used in the analysis module (and the other way round).

Since there's no interpreter involved, the analysis code can use any C++ feature supported by the local compiler (templates, STL, exceptions, etc.)

An alternative to code generation/dynamic compilation is the use of a restricted C++ syntax to specify the cuts. Such restricted syntax is then interpreted to filter the data that will fill the histogram. An example of such approach is the TagViewer module (see Figure 2 below):

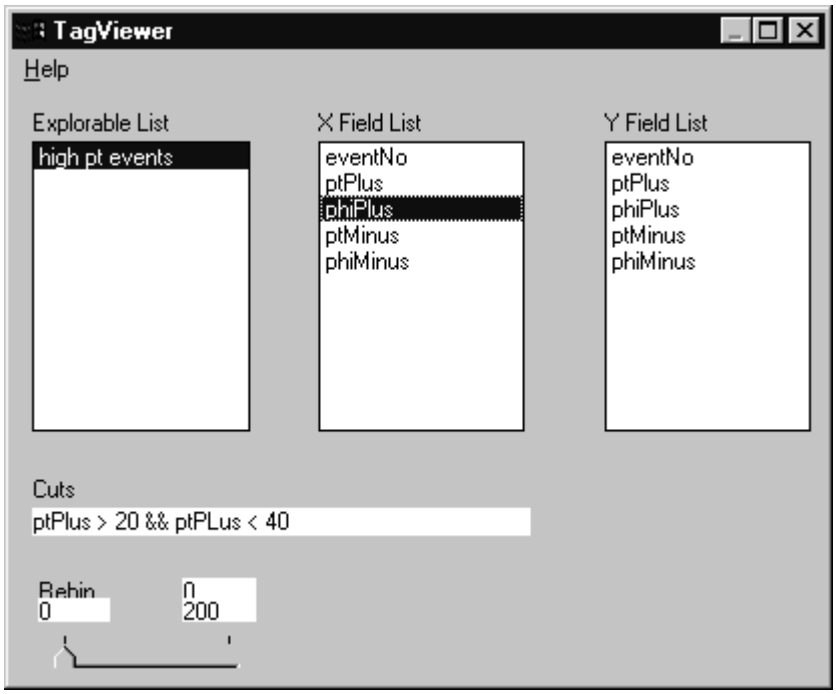


Figure 2 – The TagViewer Module

The cuts are expressed as a simple C++ expression involving only Tag variables, relational and logical operators.

5. CONCLUSIONS

The HEPExplorer package is a successful effort to integrate IRIS Explorer and Objectivity/DB so that high energy physicists can 'exercise' the analysis chain, from event to paper, on data stored in an object oriented database.

We believe IRIS Explorer is a good environment for data analysis and visualization: its compliance with graphics standard, its simple development environment, its robustness and modularity being certainly the main good points.

The layered approach has proved to be an effective way to cope with change.Since the first release of LHC++ (July 1998) we have already changed the basic C++ libraries (from Rogue Wave's Tools.h++ to STL) and the whole histogram package (from HistOOgrams to HTL) without major impact on any other part of the package.

REFERENCES

[1] CERN Program Library (CERNLIB): see http://wwwinfo.cern.ch/asd/index.html.

[2] LHC++: see http://wwwinfo.cern.ch/asd/lhc++/index.html

[3] RD45 - A Persistent Object Manager for HEP, see http://wwwinfo.cern.ch/asd/rd45/index.html.

[4] The Object Database Management Group (ODMG): see http://www.odmg.org/.

[5] The IEEE Storage System Standards Working Group: see http://www.ssswg.org/.

[6] Objectivity/DB: see http://www.objectivity.com.

[7] The High Performance Storage System: see http://www.sdsc.edu/hpss/.

[8] IRIS Explorer User Guide, ISBN 1-85206-110-3, 1995.

[9] OpenGL Reference Manual, ISBN 0-201-63276-4, Addison Wesley, 1992.

[10] OpenInventor Reference Manual, ISBN 0-201-62491-5, Addison Wesley, 1994.

[11] An Object-Oriented Detector Simulation Toolkit (GEANT-4): see http://wwwcn.cern.ch/pl/geant/geant4.html.

[12] Virtual Reality Markup Language (VRML): see http://vrml.wired.com/.

[13] 3DMasterSuite: see http://www.tgs.com/.

[14] NAG C library: see http://www.nag.co.uk/.

[15] HEPODBMS: see http://wwwinfo.cern.ch/asd/lhc++/HepODBMS/user-guide/ho.html.

[16] HTL: see http://wwwinfo.cern.ch/asd/lhc++/htlguide/htl.html.

[17] HBOOK: see http://wwwinfo.cern.ch/asdoc/hbook_html3/hboomain.html.

[18] HEPInventor: see http://home.cern.ch/~couet/HEPInventor_doc/.

[19] HEPExplorer: see http://wwwinfo.cern.ch/asd/lhc++/HepExplorer/index.html.

[20] GEMINI: see http://wwwinfo.cern.ch/asd/lhc++/Gemini/.

[21] MINUIT - Function Minimization and Error Analysis Package, CERN Program Library Long Writeup D506: http://wwwinfo.cern.ch/asdoc/minuit_html3/minmain.html.

[22] HEPFitting: see http://wwwinfo.cern.ch/asd/lhc++/HepFitting/.

[23] CLHEP - A Class Library for HEP, see http://wwwinfo.cern.ch/asd/lhc++/clhep/index.html

[24] PAW - the Physics Analysis Workshop - CERN Program Library Long Writeup, Q121

GEMINI AND HEPFITTING COMPONENTS OF LHC++

*Zbigniew Szkutnik *)*
Department of Applied Mathematics, University of Mining and Metallurgy, Cracow, Poland

Abstract
Design concepts and usage of Gemini and HEPFitting components of LHC++ environment are presented. Gemini's approach to error analysis and the relation of Minos-type errors and Hessian-based errors are briefly discussed.

1. INTRODUCTION

Gemini [1] is a GEneral MINImization and error analysis package implemented as a C++ class library. It provides a unified object oriented Application Programming Interface to various minimizers. The currently supported set of minimizers (Minuit [2] and NAG C [3]) can be extended. For the common subset of functionality, it is up to the user which minimization engine does the work. The user can easily switch between various minimizers without essential changes in the application code. Gemini finds a minimum of an objective function, possibly subject to general constraints, and performs an error analysis. While being a part of LHC++, Gemini only depends on the actual minimizer and may thus be used without the other LHC++ components.

HEPFitting [4] is a collection of C++ fitting classes, based on Gemini. It allows for loading data, defining a model and a fitting criterion, performing a fit and obtaining fit results, including error analysis. Basic HEPFitting classes are derived from Gemini classes and thus inherit Gemini's basic minimization and error analysis features. Additional, special features of HEPFitting include a simple way of setting parameters for the error analysis and a mechanism for building a model out of predefined components as well as for defining an arbitrary model. A suitable objective function to be minimized is automatically created, according to the fitting method chosen. Models can be fitted both to HTL histograms [5] and to arbitrary data points which can be loaded via user's arrays.

The intended primary area of application of both packages is batch data processing. Thus, apart from a simple text-mode printout of Minos- and Hessian based confidence regions, the packages do not provide any other visualization tools.

2. THE CONCEPT AND IMPLEMENTATION OF GEMINI

2.1 Prerequisites

Minuit, a FORTRAN minimization package, has been in successful use in HEP for more than 30 years. Especially, its error analysis has become a *de facto* standard in physics data analysis. An analysis of users' requirements at the early stage of the Minuit-replacement project [6] showed that a Minuit's successor should provide Minuit's functionality in a Minuit-style and, additionally, that some new features like, e.g. general constraints, should be provided. It has also been stressed on many occasions that Minos-type error analysis must be possible with the new package, in order that it can be accepted by the HEP community.

No single commercially available product could fulfill all the requirements, although we identified the family of NAG C minimizers as the most prospective candidate, both with respect to functionality and to performance. We thus decided to write a small (currently about 3200 lines of code) C++ open-ended package which could internally use various working engines and which would

---

*) The author was working for LHC++ while being a Scientific Associate at CERN in 1997 and 1998 .

provide all the requested functionality in the requested style. Switching between various minimizers should not require any essential changes in the user's code in order to ease the cross-checks and the transition period for the Minuit-users. This is how Gemini came to life.

Special effort proved to be necessary in order to implement Minos-type errors. Being standard within the HEP community, this type of errors is by no means standard in the non-HEP world. To the best of our knowledge, Minuit was the only package which implemented this type of errors. We thus decided to write a special Minos analysis module for Gemini, so that any type of minimizer could be plugged-in, regardless on whether it is able to perform the Minos analysis or not.

2.2 Functions and objects

The basic types of objects used within Gemini are: the objective function object, the minimization object and the contour object.

The objective function is the function to be minimized. It is defined by the user as a C/C++ function, which computes the objective function value and, optionally, its gradient, for a given vector of function arguments. The function is then captured into an objective function object, which is of the type *OBJfun*.

The minimization object is the main object, which contains the complete problem definition. It also provides methods for assigning an objective function object, defining the objective function arguments and their admissible regions, setting minimization options, running a minimizer, obtaining the current status of the minimization process, obtaining results and error analysis. If the minimization object is declared as being of type *CMinuit*, then Minuit is used as the minimization engine. Similarly, if it is of type *NAGmin*, then NAG C minimizers are used. A generic pointer of the type *(GEmini *)* can point both to *CMinuit* and *NAGmin* type objects and can be used, if minimization objects are dynamically allocated on the heap. This allows the user to select the minimizer at run time rather than at compilation time.

A contour is a set of points from the boundary of a (bounded) set in a two-dimensional subspace. In Gemini, it typically represents either an elliptical boundary of a Hessian-based confidence region for a selected pair of parameters, or a Minos contour, i.e. the curve on which the minimum of the objective function, with respect to all the remaining parameters, equals the current minimum plus a user-defined value. Contours are implemented in Gemini as an abstract data type with overloaded assignment and addition operators. Addition means, in this case, merging and can be used for overlaying the contours. A public method *plot()* produces a text-mode printout of, possibly overlayed, contours.

2.3 Example

The following example is a complete C++ program. It minimizes the Powell's quartic function of four arguments defined as

$$f(x,y,z,u) = (x+10y)^2 + 5(z-u)^2 + (y-2z)^4 + 10(x-u)^4$$

subject to one nonlinear constraint

$$x^2 + y^2 + z^2 + u^2 = 1$$

and one linear constraint

$$x + y + z + u = 0\,.$$

Note that, for the unrestricted problem with the Powell's objective function, the Hessian becomes singular at the origin and the minimum point is not uniquely defined. With added constraints, the Hessian singularity point is excluded from the admissible region. Also note that this

nonlinear optimization problem cannot be solved with Minuit, which only allows for simple bound constraints to be imposed.

```
#include "gemini.h"

inline double square(double x) { return(x*x); }
const int nop=4; // number of parameters

// objective function (Powell's quartic function of 4 vars)

void myfun(int n, double g[], double *objf, const double parms[], int code)
{
  // define aliases for convenience
  const double &x=parms[0], &y=parms[1], &z=parms[2], &u=parms[3];

  *objf = square(x+10*y) + 5*square(z-u) + square(square(y-2*z)) +
                                                  10*square(square(x-u));
  if(code == 2){
     // gradient components
     g[0] = 2*(x+10*y) + 40*(x-u)*square(x-u);
     g[1] = 20*(x+10*y) + 4*(y-2*z)*square(y-2*z);
     g[2] = 10*(z-u) - 8*(y-2*z)*square(y-2*z);
     g[3] = -10*(z-u) - 40*(x-u)*square(x-u);
  }
}

// non-linear constraint function (sum of squares equals 1)

void nlf(int nop, double g[], double *val, const double parms[], int code)
{
  const double &x=parms[0], &y=parms[1], &z=parms[2], &u=parms[3];

  *val = square(x) + square(y) + square(z) + square(u);

  if(code == 2){
     // gradient components
     g[0]=2*x; g[1]=2*y; g[2]=2*z; g[3]=2*u;
  }
}

int main()
{
  // capture objective function into objective function object
  OBJfun fcn(myfun);

  // create main minimization object
  NAGmin nlp("Non-linear optimization example", nop, &fcn);

  // impose non-linear constraint
  if( nlp.setNlinConstraint(11, nlf, 1.0, 1.0) ) exit(1);

  // impose linear constraint
  double lincoef[nop];
  lincoef[0] = lincoef[1] = lincoef[2] = lincoef[3] = 1.0;
  if( nlp.setLinConstraint(1, lincoef, 0, 0) ) exit(1);

  nlp.printSetup();
  if( nlp.minimize() ) exit(1);
  nlp.printResults();

  return(0);
}
```

3. ERROR ANALYSIS IN GEMINI

3.1 General concept of errors

The general concept of 'errors' or 'uncertainties' in Gemini is the same as in Minuit. For a given objective function $F(\theta)$ to be minimized, with $\theta = (\theta_1,...,\theta_p)$, and for a given error parameter UP, the 'uncertainty set' US of the solution $\tilde{\theta} = (\tilde{\theta}_1,...,\tilde{\theta}_p)$ is defined as

$$\text{US} = \{\theta : F(\theta) - F(\tilde{\theta}) \leq \text{UP}\} \tag{1}$$

For any sub-vector of $\tilde{\theta}$, the uncertainty set is constructed as the orthogonal projection of US onto the corresponding plane spanned by the selected components.

This purely geometrical concept is meaningful, in *qualitative* sense, for arbitrary objective functions. 'Errors' or 'uncertainties' are related to the shape of the objective function in a neighborhood of the minimum.

Well-defined *quantitative* meaning, in probabilistic terms, can be assigned to such defined 'errors' or 'uncertainties' in statistical problems, when the objective function is a fit criterion, for example a chi-squared, log-likelihood or least squares loss function.

Error analysis based on the plain difference $F(\theta) - F(\tilde{\theta})$ is called Minos analysis, as in Minuit. In this context, we also use terms like Minos error and Minos confidence region. Minos analysis can be computationally very costly, however, as it requires multiple function minimization to find points on the boundary of US or of its projection. It will be seen below, how Minos analysis can formally be justified in statistical terms. For maximum likelihood estimators and standard minimum chi-squared estimators, for example, it can be done via the asymptotic chi-squared distribution of a suitably transformed likelihood ratio.

A standard way to overcome the computational difficulty of Minos analysis is to approximate $F(\theta) - F(\tilde{\theta})$ with $0.5 \cdot (\theta - \tilde{\theta})^T H (\theta - \tilde{\theta})$, with H being the Hessian of F at $\tilde{\theta}$. One obtains this approximation via the standard Taylor expansion of F around $\tilde{\theta}$ and using the fact that the gradient of F at the minimum is zero. With this approximation, approximate versions of both US and its projections can be found analytically, so that multiple function minimization can be avoided. This leads to the standard Hessian-based error analysis and is related to asymptotically normal distributions of estimators.

In the following sections, those two approaches are described in more detail and their links to standard statistics exposed. Only unconstrained minimization problems are discussed here. For a discussion of error analysis for problems with constraints, see e.g. [1]. It is always assumed that the problem is regular enough for the underlying mathematical theory to be applicable. The aim of this description is to expose the main ideas rather than to present technical details. Relevant mathematical results can be found, for example, in the books [7-9].

Ref. [10] is a standard statistical reference for HEP-physicists. It contains, in particular, a discussion of the Minos idea in less formal terms of an 'implicit transformation to linearity and back', which provides further insight into the idea of Minos.

3.2 Minos error analysis

The Minos uncertainty set US for the whole vector θ is defined above in (1). In order to obtain an uncertainty set for two components only, say θ_1 and θ_2, we have to project US onto the plane spanned by those components. This projection is a set of points (θ_1, θ_2) such that $F(\theta) \leq F(\tilde{\theta}) + \text{UP}$, for some $\theta_3,...,\theta_p$. Equivalently, it is the set of points (θ_1, θ_2) such that the

minimum of $F(\theta)$ with respect to $\theta_3,...,\theta_p$ and with θ_1 and θ_2 fixed is not greater than $F(\tilde{\theta})$ + UP . The boundary of this set is thus the contour of the function

$$\tilde{F}(\theta_1,\theta_2) = \min_{\theta_3,...,\theta_p} F(\theta)$$

which corresponds to $\tilde{F}(\theta_1,\theta_2) = F(\tilde{\theta})$ + UP .

For a single parameter, say θ_1, we define a function

$$\hat{F}(\theta_1) = \min_{\theta_2,...,\theta_p} F(\theta)$$

and construct the uncertainty set, or the projection of US, as $\{\theta_1 : \hat{F}(\theta_1) \le F(\tilde{\theta}) + \mathrm{UP}\}$. For a regular function *F*, genuine local minimum $\tilde{\theta}$ and 'small' UP, this will be an interval $[\underline{\theta}_1, \overline{\theta}_1]$, say. The positive and negative Minos errors are then defined as, correspondingly, $\overline{\theta}_1 - \tilde{\theta}_1$ and $\tilde{\theta}_1 - \underline{\theta}_1$.

In order to give Minos errors a quantitative, statistical meaning, let us assume first that *F* equals *-2*log-likelihood* for a regular statistical model. The unrestricted minimum $\tilde{\theta}$ of *F* is then a maximum likelihood estimator of θ. Let further $\hat{\theta}$ be the minimum of *F*, subject to *r* independent restrictions on θ. It is well-known that, for any true θ which satisfies the restrictions, $F(\hat{\theta}) - F(\tilde{\theta})$ is asymptotically chi-squared distributed with *r* degrees of freedom - a fact used for the construction of the so-called asymptotic likelihood ratio test (λ-test).

It follows immediately that, for any true θ with the given values of the first two components, $\tilde{F}(\theta_1,\theta_2) - F(\tilde{\theta})$ is asymptotically chi-squared distributed with two degrees of freedom (we impose two constraints by fixing the values of θ_1 and θ_2) and, for any true θ with the given value of the first component, $\hat{F}(\theta_1) - F(\tilde{\theta})$ is asymptotically chi-squared distributed with one degree of freedom (we fix the value of θ_1 only).

A standard Neyman asymptotic (1-α)-confidence region for (θ_1,θ_2) can then be constructed as

$$\{(\theta_1,\theta_2) : \tilde{F}(\theta_1,\theta_2) - F(\tilde{\theta}) \le c_\alpha\}$$

with c_α being the (1-α)-quantile of the chi-squared distribution with two degrees of freedom. This is exactly the projection of US, with UP = c_α, onto the plane spanned by the first two components.

Similarly, an asymptotic Neyman (1-α)-confidence region for θ_1 is

$$\{\theta_1 : \hat{F}(\theta_1) - F(\tilde{\theta}) \le c_\alpha\}$$

with c_α being the (1-α)-quantile of the chi-squared distribution with one degree of freedom. Again, this is the projection of US with UP = c_α onto the first axis.

With obvious modifications, similar argument applies, of course, to any subset of the components of θ, which leads to the following conclusion:

> If *F* equals *-2*log-likelihood*, then Minos confidence regions for *r* components of θ have the asymptotic coverage probability 1-α, if UP is the (1-α)-quantile of the chi-squared

distribution with *r* degrees of freedom. With *r* = 1 and UP = 1, the coverage probability corresponds to that of a '± one-sigma error bar' for a single parameter.

The scale factor of *F* is essential. Additive terms, which do not depend on θ can be dropped, however.

In Gaussian models, *-2*log-likelihood* equals, up to a constant, additive term, the chi-squared fit criterion and the whole analysis applies. In many other cases, the equality holds asymptotically, thus validating the Minos analysis with *F* being the chi-squared fit criterion. In particular, this is true for the Poisson histogram cells counts model (see e.g. [1]).

3.3 Hessian-based error analysis

In the Hessian-based error analysis, $F(\theta) - F(\tilde{\theta})$ is approximated with $0.5 \cdot (\theta - \tilde{\theta})^T H (\theta - \tilde{\theta})$, with *H* being the Hessian of *F* at $\tilde{\theta}$. The approximate version US' of the uncertainty set US, corresponding to a given value of the UP parameter takes then the form

$$\text{US}' = \{\theta : 0.5 \cdot (\theta - \tilde{\theta})^T H (\theta - \tilde{\theta}) \le \text{UP}\}. \tag{2}$$

The orthogonal projection $\text{US}_r^{'}$ of US' onto the plane spanned by, say, the first *r* components of θ consists of all points $(\theta_1,...,\theta_r)$ such that the minimum of $(\theta - \tilde{\theta})^T H (\theta - \tilde{\theta})$ with respect to $\theta_{r+1},...,\theta_p$ is not greater than 2*UP. Let us split $\theta - \tilde{\theta}$ into two sub-vectors: θ_I consisting of the first *r* components and θ_{II} consisting of the remaining *p-r* components. Correspondingly, we can write

$$H = \begin{bmatrix} H_1 & H_{12} \\ H_{12}^T & H_2 \end{bmatrix}$$

with H_1 of size *(r, r)* and H_2 of size *(p-r, p-r)*. Looking for a minimum with respect to θ_{II} and with θ_I fixed, we have then

$$(\theta - \tilde{\theta})^T H (\theta - \tilde{\theta}) = \theta_I^T H_1 \theta_I + 2\theta_I^T H_{12} \theta_{II} + \theta_{II}^T H_2 \theta_{II} = G(\theta_{II})$$

and

$$\text{grad}\, G(\theta_{II}) = 2H_{12}^T \theta_I + 2H_2 \theta_{II}.$$

The equation $\text{grad}\, G(\tilde{\theta}_{II}) = 0$ gives the minimum point $\tilde{\theta}_{II} = -H_2^{-1} H_{12}^T \theta_I$. The minimum value is

$$G(\tilde{\theta}_{II}) = \theta_I^T (H_1 - H_{12} H_2^{-1} H_{12}^T) \theta_I$$

which gives

$$US_r^{'} = \{\theta_I : 0.5 \cdot \theta_I^T (H_1 - H_{12} H_2^{-1} H_{12}^T) \theta_I \le \text{UP}\}.$$

On the other hand, using the symmetric, block matrix inversion formula, we have

$$\begin{bmatrix} H_1 & H_{12} \\ H_{12}^T & H_2 \end{bmatrix}^{-1} = \begin{bmatrix} (H_1 - H_{12} H_2^{-1} H_{12}^T)^{-1} & X \\ X^T & H_2^{-1}(I - H_{12}^T X) \end{bmatrix} \tag{3}$$

with $X^T = -H_2^{-1} H_{12}^T (H_1 - H_{12} H_2^{-1} H_{12}^T)^{-1}$. This means that, with $S = (0.5 \cdot H)^{-1}$, we can write

$$\mathrm{US'} = \{\theta : (\theta - \tilde{\theta})^T S^{-1} (\theta - \tilde{\theta}) \leq \mathrm{UP}\}$$

and, denoting by S_r the upper left *(r, r)* portion of *S* , the projection $\mathrm{US}_r^{'}$ takes the form

$$US_r^{'} = \{\theta_I : \theta_I^T S_r^{-1} \theta_I \leq \mathrm{UP}\}. \tag{4}$$

In order to set this in relation with statistics, recall that if *F* equals *-2*log-likelihood*, then the maximum likelihood estimator $\tilde{\theta}$ is, in regular cases, asymptotically normally distributed

$$\tilde{\theta} \sim \mathrm{AN}(\theta, \mathrm{I}_\theta^{-1})$$

where $I_\theta^{-1} = 0.5 \cdot \mathrm{E}_\theta H$ is the Fisher information matrix for the whole data set. Again, in regular cases, one can reasonably assume that $\mathrm{E}_\theta H \approx H(\tilde{\theta})$ and use the inverse of $0.5 \cdot H(\tilde{\theta})$ as an estimate of the asymptotic covariance matrix of $\tilde{\theta}$.

This is clearly related to (2) and means that, since $(\theta - \tilde{\theta})^T S^{-1} (\theta - \tilde{\theta})$ is asymptotically chi-squared distributed with *p* degrees of freedom, in order to have the asymptotic 1-α coverage probability for US', one should set UP to the (1-α)-quantile of the chi-squared distribution with *p* degrees of freedom.

Further, S_r in (4) can be interpreted as the covariance matrix of the marginal distribution of $(\theta_1, ..., \theta_r)$ and, in view of its asymptotic normality, setting UP in (4) to the (1-α)-quantile of the chi-squared distribution with *r* degrees of freedom, we get the asymptotic coverage probability 1-α for $\mathrm{US}_r^{'}$.

For the chi-squared fit criterion, the approximation argument from the previous section applies. More generally, the above argument can be extended to any M-estimator, in which case the asymptotic covariance matrix needs not to be the inverse of the Fisher information matrix, but continues to be the (properly normalized) inversed Hessian.

For practical recommendations on how to set the value of the error parameter UP, see [1]. Note, however, that if the HEPFitting package is used, then the value of UP is automatically set, according to the fitted model and the fitting method.

4. THE CONCEPT AND USAGE OF HEPFITTING

4.1 Basic features

HEPFitting is a small C++ package (currently about 1200 lines of code) built on top of Gemini with the aim to ease the most common fitting applications of the minimizers. HEPFitting can internally use both Minut and NAG C minimizers. No change in the application code is needed, the switch being done when the application code is being compiled. If _MINUIT_ is defined at compilation time, then Minuit is used. Otherwise, NAG C minimizers are used.

The main fitting object of the type *HEPHistofit* contains a complete definition of the fitting problem and provides methods for defining the problem, loading data, performing the fit and obtaining results. The fitting region can be restricted by imposing interval limits on space variables, as well as by including/excluding any single histogram bin or data point.

Currently implemented fitting criterions are chi-squared and Poisson maximum likelihood. Various options are provided for handling empty bins and zero-errors data points. The user only defines the model itself. A suitable objective function to be minimized is automatically defined by HEPFitting, according to the selected fitting criterion, and the error parameter is properly set so that

the computed errors are the standard one-sigma-errors. All available components of the objective function gradient are constructed from the model function gradient, as provided by the user in the model function. The package checks, whether the data set and the model assigned to the fitting object are compatible with respect to the space dimensionality.

Special methods are provided for obtaining both elliptical (or Hessian-based) and Minos confidence regions for a selected pair of parameters, which only require that the user specifies the requested confidence level.

4.2 Defining the model

The fitted model is encapsulated in an object of the type *MODELfun*, which can be used to capture an arbitrary user's model, defined as a C/C++ function, as well as to compose a model out of predefined elementary models, like Gaussian, polynomial and exponential ones. Any additional functionality needed in the user's application can be added through the inheritance mechanism, in which case the user derives his/her own class from *MODELfun* and overrides the virtual member function *modelfun* with the actual model function.

When a model is fitted to a histogram, the model function values in bin reference points are multiplied by bin volumes, before being compared to bin contents. The model function thus represents the intensity function of the underlying Poisson process or a density function and is independent of the particular histogram binning. The user can ignore bin volumes by selecting special options, as described in [4].

Various ways of defining a model are described in detail in [4]. Here, we only discuss the way a model can be constructed out of predefined components.

A single-component model may easily be defined with a specialized constructor of *MODELfun* which takes the component name as an argument (G for Gaussian, E for exponential, P*n* for polynomial of degree *n*). The model object created this way may immediately be assigned to a fitting object. For example, in order to fit a Gaussian and then a second-degree polynomial, one can proceed as follows

```
 ...
MODELfun model1("G");
MODELfun model2("P2");
HEPHistoFit myFittingObject;

myFittingObject.setModel( &model1 );
// perform the fit
...
myFittingObject.setModel( &model2 );
// perform the fit
...
```

In order to compose a multi-component model from the built-in standard models, the user has to derive his/her own class from *MODELfun* and override the virtual member function *modelfun*. The overriden function defines the expression. The components used in the expression may be added *en bloc* using the *MODELfun* public method *setComponents(char *string)*, where *string* is composed of blank- or comma-separated components symbols. For other techniques which may be used, see [4].

The order, in which the components are added or, equivalently, the order in which they are placed in *string* is significant. The global vector of parameters consists of the components' parameters stored in the order, in which the components have been added. Similarly, the indices, by which the components are referenced in the expression, correspond to the order, in which the components have been added.

The component functions are referenced in the expression as *f(i,x,p)*, where *i* stands for the component's index and takes values 0, 1, 2, ..., the vector of space coordinates is denoted by *x* and *p* is the global vector of parameters, as passed through the arguments of the *modelfun(...)* function.

If only the four basic operations (+ - * /) are used to build the expression, then there is a simple way to also provide the gradient with respect to parameters, which can significantly improve the performance of the minimizer. The rule can be formulated as follows: "*Look at the components f(i,x,p) as if they were all functions of the same single variable p, use df(i,x,p) to denote derivatives, apply well-known differentiation rules and assign to g the resulting expression*". A suitable gradient vector will then automatically be created and used.

4.3 Example

In the following example, the model is defined as the product of a second-degree polynomial and a Gaussian. Since the parametrization of such a model is redundant, the Gaussian's 'mass' parameter has to be fixed, before performing a fit. Otherwise, the model would not be identifiable.

```
class myModelObject : public MODELfun{
public:
   // define the epression by overriding modelfun()
   double modelfun(const double x[], const double p[], array_n<double>& g,
                                                                    int code){
       // compute gradient, if requested
       if(code==2)
           g = df(0,x,p)*f(1,x,p)+f(0,x,p)*df(1,x,p);

       // return model function value
       return f(0,x,p)*f(1,x,p);

   }
};

int main(){
    // create empty fitting object
    HEPHistoFit myFittingObject;

    // create model object and define model's components, then assign
    myModelObject P2Gmodel;
    P2Gmodel.setComponents("P2,G");
    myFittingObject.setModel( &P2Gmodel );

    // load data, set initial parameters' values e.t.c.
    ...

    // fix the first parameter of the Gaussian at 1 (It's
    // the 4th parameter preceded by 3 parameters of P2
    myFittingObject.parmDef(4,"Gmass",1,1,1,1);

    // perform a fit
    myFittingObject.perform(PoissonMLfit);
    myFittingObject.printResults();

    // 90% confidence regions for the free parameters of the Gaussian
    GeminiContour c1, c2;
    MyFittingObject.ellipticalConfidenceRegion(5,6,c1,0.90);
    MyFittingObject.minosConfidenceRegion(5,6,c2,0.90);
    (c1+c2).plot(); // overlay and plot
}
```

5. CONCLUSIONS

With Gemini, we believe to have a flexible and open framework for function minimization. Numerous tests have proved that the family of NAG C minimizers can satisfy the requirements of the HEP community. However, with its own Minos analysis module, Gemini becomes independent of the minimizer actually used, so that NAG C may easily be replaced with another minimizer, without affecting the users' code.

Using HEPFitting helps the users to keep to a minimum the amount of code written in order to perform standard fits. The implementation of the build-in elementary models and the internal mechanism of computing all available derivatives of the objective function result in considerable improvements in the minimizer's performance, by keeping the number of the model function calls at the minimum.

ACKNOWLEDGEMENTS

The author kindly acknowledges CERN's hospitality while working on the CSC 99 lecture.

REFERENCES

[1] Gemini - A minimization and error analysis package in C++, User's & Reference Guide (1998), see: http://wwwinfo.cern.ch/asd/lhc++/Gemini/.

[2] MINUIT - Function minimisation and error analysis package, CERN Program Library Long Writeup D506 (1994), see: http://wwwinfo.cern.ch/asdoc/minuit_html3/minmain.html.

[3] NAG C Library Manual, Mark 5, Vol. 2, The Numerical Algorithms Group Limited (1998), see: http://www.nag.co.uk/.

[4] HEPFitting - A C++ fitting API for HEP based on Gemini, User's & Reference Guide (1998), see: http://wwwinfo.cern.ch/asd/lhc++/HepFitting/.

[5] Histogram Template Library, User Guide (1999), see:http://wwwinfo.cern.ch/asd/lhc++/htlguide/htl.html

[6] Minuit++, Replacement for Minuit and HEP statistical tools. User Requirements Document (1997), see: http://wwwinfo.cern.ch/asd/lhc++/requirements/stable/URD/html/

[7] S.D. Silvey, Statistical inference (Chapman and Hall, 1975).

[8] Y. Bard, Nonlinear parameter estimation (Academic Press, 1974).

[9] R.J. Serfling, Approximation theorems of mathematical Statistics (Wiley, 1980).

[10] W.T. Eaddie, D. Drijard, F. James, M. Roos and B. Sadoulet, Statistical methods in experimental physics (North-Holland, 1971).

DATA STORAGE AND ACCESS IN LHC++

Marcin Nowak
CERN IT Division, RD45 project - CH 1211 Geneva 23 Switzerland

Abstract
This paper presents LHC data requirements as well as some features of HEP data models and explains how an ODBMS can be used to address them. Essential features of object databases will be discussed, followed by those specific to Objectivity/DB, which is the database currently used in LHC++. The differences between transient and persistent data models will be given with some rules for how to convert the former into the latter. Next, the paper will focus on HepODBMS layer, which is a set of HEP specific classes extending the functionality of a database and forming an interface used by other LHC++ software components. The concept of event collections and object naming will be discussed.

1. INTRODUCTION

Experiments at the Large Hadron Collider (LHC) at CERN will generate huge quantities of data: roughly 5 petabytes (10^{15} bytes) per year and about 100 PB over the whole data-taking period (15+ years). Data will be collected at rates exceeding 1GB/s and later analyzed, perhaps many times. The analysis frameworks of the new experiments will be developed using object-oriented (OO) technologies and consequently their data will be represented in object-oriented data models, often of significant complexity.

These factors form a challenging data storage and management problem and it seems clear that the traditional solutions based on sequential Fortran files would not be adequate. In 1995 the RD45 project was initiated at CERN to investigate new solutions and technologies. The emphasis was put on commercial products, with the hope of minimizing development costs and maintenance effort over the very long period of use. The evaluation of different technologies such as language extensions for persistency, light-weight object managers, object request brokers and object databases led to the recommendation of an Object Database Management System as the main data management product, together with a Mass Storage System to provide physical storage.

Studies of the various ODBMS products on the market, particularly with respect to their ability to satisfy LHC data management requirements, resulted in the selection of a particular database: currently Objectivity/DB.

Experiment	Data Rate	Data Volume
ALICE	1.5 GB/sec	1 PB/year (during one month)
CMS	100 MB/sec	1 PB/year
ATLAS	100 MB/sec	1 PB/year
LHC-B	20 MB/sec	200 TB/year

Table 1 Expected Data Rates and Volumes at LHC

2. OBJECT DATABASES

2.1 Data Model

In OO programming style the data is represented as a set of objects interconnected with each other in various ways, depending on the object model. Figure 1 shows a simple example of the data model for a HEP Event.

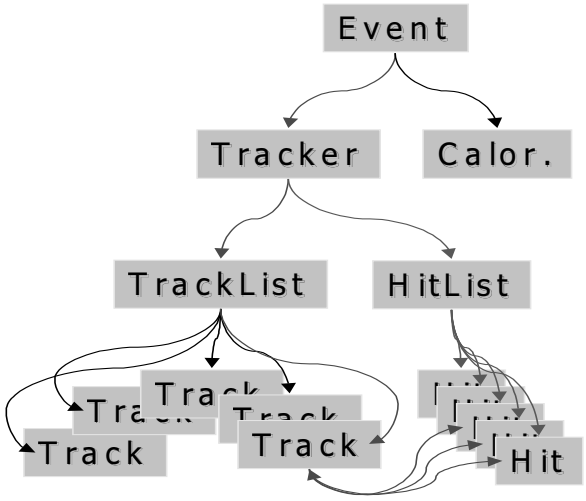


Figure 1: Simple Logical Model of Event Data Objects

An application working with a given data model would traverse the connections between objects to find the data it needs or to modify their contents. It may also modify the network of objects by adding or removing some of them. In the Event model example, the application could navigate from the main Event object to its tracking detector, retrieve a particular track from the track list, then navigate to all associated hit objects to perform a track refit.

2.2 Transient Objects

In the traditional run-cycle, an application would first create in memory the objects it needs and then fill them with some data. Next, it would perform the actual task it was designed for: working with its data representation. Finally, the program would store the results and delete objects from memory. In this scenario, the lifetime of an object is rather short and always bound to the application execution time – the objects exist only within the context of a single program. In the database terminology such objects are called *transient* (i.e. temporary).

OO languages support transient objects and navigation between them (the latter via pointers and references in C++). Creating and traversing in-memory networks of objects is very efficient and type-safe even for polymorphic classes. This, however, assumes that the entire network of objects is maintained in the memory. There is little support from today's languages regarding I/O operations on such networks of objects.

2.2.1 Object Input / Output

Providing I/O for complex data models is a difficult task for the programmer. In the first place, 2 different data formats have to be maintained for every class that is to be stored, namely:

- Class definition used by the application, including pointers to other objects
- Data encoding format used when storing in a file

The formats must be assumed to be different, as the run-time format is tightly coupled to the operating system and even compiler version. Thus, even if we start with an exact memory copy in a file, the possibility of handling different run-time formats must be provided: the application code has

to perform conversions between the two representations. The problem increases further since any individual class definition is likely to change over the long run period of LHC, leaving some objects in the old format stored on tape.

The programmer has also to decide:

- How to perform the conversion.
  The conversion of object attributes may require byte-swapping or alignment adjustment, which is a time-consuming, but rather straightforward operation. What is more difficult is storing the connections between objects, which constitute the shape of the object network. This requires translating pointers and references into a storable format and a special code that will be able to rebuild the network later.
- When to perform the I/O.
  All data transfers have to be initiated explicitly. Typically, some amount of data has to be read from disk when the application starts and all useful results have to be stored at the end. During the execution time, additional I/O operations may be required when the program follows a link referencing an object that is not yet in memory. In a multi-user environment, part of the data in memory may become stale as a result of an update performed – by another user or process – upon the corresponding object on disk. Such situations must be detected to avoid data corruption.
- How much data to transfer.
  In a complex HEP application it is difficult to predict which data items will be used. In many cases all of the event data is read, just in case it is needed. This approach may result in degraded performance.

Code that deals with object I/O often constitutes a large part of the application. Maintaining this code is both tedious and error-prone. Consistency between the disk and memory representation is not performed automatically and errors in this layer may lead to obscure software problems. In addition, large amounts of I/O related code in a class makes programs less understandable and may obscure the actual services provided by the class.

2.3 Object Persistency

Persistent objects are the opposite of transient objects. They do not disappear when the application finishes (they *persist*). This is possible because they do not belong to the application context, but rather to the context of a persistent store. In the case of an ODBMS, they belong to a database context. A persistent object will disappear only when explicitly deleted from the store.

Programs working with persistent objects do not “own” them – they receive only a copy from the store. It is possible for more then one program to access the same object at the same time in a “read” mode.

Object databases maintain the consistency between objects on disk and in memory. The programmer never deals with the disk representation – but sees only the run-time definition of the class. This feature is called “tight language binding”. The ODBMS also takes care of all I/O that has to be performed to retrieve an object. All the problems discussed in section 2.2.1 are handled by the system and not by the application programmer.

Persistent objects are real objects. They support run-time type identification, (multiple) inheritance, polymorphism (virtual functions) and abstract data types. In C++ they can also be instances of templated classes.

2.4 Transactions

Object databases provide transactions in a similar way that relational databases do. The transactions are atomic, consistent, isolated and durable (so-called A.C.I.D. [2] properties) and are usually not nested. All data access is done inside a transaction – otherwise the store is not accessible. The

standard transaction types are “read” and “update”. Some systems provide additional types of transactions that e.g. allow simultaneous read and write to the same objects An example of such a transaction type is the multiple reader, one writer (MROW) transaction supported by Objectivity/DB.

As all data access occurs inside a transaction, all I/O operations are transaction bound. At the start of a transaction, only the connection to the database is established. As the application proceeds to navigate in the data network and access objects, the relevant pieces of data are retrieved. The ODBMS tries to ensure that there are no unneeded data transfers, in order to optimise performance. If the application modifies objects or creates new ones, the changes may be kept in memory or written to disk, but they are not immediately visible to other clients. Only when the transaction is committed, all modifications are flushed to disk and registered in the database.

Transactions in database systems are the main tool to ensure data consistency. If a transaction is interrupted (aborted) in the middle, the database status is not changed.

2.5 Navigational Access

As described above, the main method of finding an object in the network is by navigation. Transient objects use pointers and references as links. A pointer is a memory address and uniquely identifies an object in the application context (or virtual memory address space). Persistent objects, which exist in the database context, need a different kind of identification.

When a new persistent object is created, the ODBMS assigns to it a unique Object Identifier (OID). The actual implementation of the OID varies between different systems, but they have common functionality – they allow the object to be found in the disk store. OIDs that point directly to the object are called physical and OIDs that use indirection are called logical. Logical OIDs give more flexibility at the cost of performance and scalability.

Object Identifiers replace pointers and references in persistent objects. They are used to create uni-directional (pointing in one direction, like a C++ pointer) associations between them. In most products they also enhance the idea of pointers by allowing:

- bi-directional associations
  bi-directional association is a relation between 2 objects. From an implementation point of view it may look just like 2 objects pointing to each other, but the ODBMS makes sure that pointers on both sides are set correctly (or reset) at the same time. It is not possible to modify only one of them, thus ensuring consistency.
- 1-to-n associations
  1-to-n association is a relation between one object and an arbitrary number of objects on the other side. It may be uni- or bi-directional.

The OID is typically hidden from the programmer by wrapping it in a *smart pointer* implementation. Smart pointers are small objects that behave semantically in the same way as normal pointers, but they also provide additional functionality. If the smart pointer provided by ODBMS is dereferenced (in C++ by using “*” or “->” operator on it) the system is able to check if the object pointed to is already in memory, and if not, read it from disk using the OID contained in the smart pointer. After that, the smart pointer behaves just like a normal pointer. All this happens without any additional code in the application.

The ODMG standard [1] defines ODBMS smart pointer as a templated class d_Ref<T>. Figure 2 shows an example program using d_Ref<> in the same way as normal C++ pointer.

```
Collection<Event> events;           // an event collection
Collection<Event>::iterator evt;    // a collection iterator

// loop over all events in the input collection
for(evt = events.begin();  evt != events.end(); evt++)
      {
      // access the first track in the tracklist
      d_Ref<Track> aTrack;
      aTrack  = evt->tracker->tracks[0];

      // print the charge of all its hits
      for (int i = 0; i < aTrack->hits.size(); i++)
            cout  << aTrack->hits[i]->charge
                << endl;
      }
```

Figure 2: Navigation using a C++ program

As a consequence of the tight binding of ODBMS to the programming language the application programmer perceives the database store as a natural extension to application memory space. Using the database one can create networks of objects much larger than would be possible in memory, with indefinite lifetime and the possibility to efficiently navigate among them.

2.6 Database Schema

If the ODBMS is to be able to perform automatic conversion between object representation on disk and in memory, it has to have detailed information about the object. It has to know the type, name and position of every attribute in the object. This information needs to be registered in the database before any object of a given class can be stored. All class definitions known to the ODBMS are called the *database schema*.

The schema registration process depends on the ODBMS and on the programming language. In Objectivity/DB a C++ class is entered into the schema by a program that pre-processes the header files. The headers may contain normal C++ classes, with the exception that object associations should replace pointers.

2.7 Concurrent Access to Data

ODBMS products provide support for multiple clients working on the same data store and concurrently updating it. Usually ODBMSs introduce a central “lockserver” that co-ordinates the updates by keeping a lock table for the whole system. To ensure data consistency in the system, all data changes are part of a transaction. If a transaction accesses part of the database, this region is locked with an appropriate lock mode (read, write or MROW). Subsequent clients trying to operate on the same region must first contact the lockserver to determine what type of access is allowed at a given time. All locks that a transaction has acquired last until the end of the transaction (either by commit or abort).

Locking and transactions are the mechanisms that allow concurrent access to a data store. Without them it would not be possible to guarantee data consistency.

3. CHOOSING ODBMS FOR LHC++

The next section describes specific features of Objectivity/DB - the ODBMS system that the RD45 project currently recommends as the data storage and management product for LHC experiments. The following list mentions requirements that were considered the most important for the selection:

- Standard compliance – ODMG [1]
  The use of a standards compliant API may make it easier to replace one ODBMS component with another system, if such need arises
- Scalability to hundreds of PB
- Mass Storage System interface
  LHC experiments will require a database able to store 100 PB of data, a large part of which will have to be kept in MSS (on tapes)
- Distributed system
- A centralised system will not be able to efficiently deal with such large amounts of data and serve many client applications accessing it concurrently
- Heterogeneous environment
  Research institutes have very diverse computing environments – a system that will be used by all of them should be interoperable between most of them
- Data replication
  Replicating the most frequently used data to remote institutes may have a big impact on performance
- Schema versioning
  The system should allow changes in the class definitions that will inevitably happen in the long run period of LHC
- Language heterogeneity
  LHC++ is written in C++, but there are graphical presentation tools implemented in Java that would profit from direct access to the database
- Object versioning
  This feature is used by various applications, such as a detector calibration database package

4. OBJECTIVITY/DB SPECIFIC FEATURES

This chapter focuses on specific features of Objectivity/DB.

4.1 Federations of Distributed Databases

The Objectivity/DB ODBMS supports a so-called *federation* of distributed databases. Each database within a federation corresponds to a filesystem file and may be located on any host on the network. There is one central federation (FDB) file containing the catalogue of all databases and the class schema. Hosts on which database files are located run the Objectivity data server (ooams). In addition, there is a central lockserver program located on a selected machine.

Client applications may use one Objectivity federated database at a time. To access the data within a federation, the database client software first reads the FDB catalogue to find where the data is located and then connects directly to the data server on a machine hosting the right database.

Before any data is read or modified, the client contacts the lockserver to obtain a lock. These operations are all performed transparently to the user, who only deals with (networks of) objects. Figure 3 shows an example of a Federated Database with 2 client applications accessing it from different hosts.

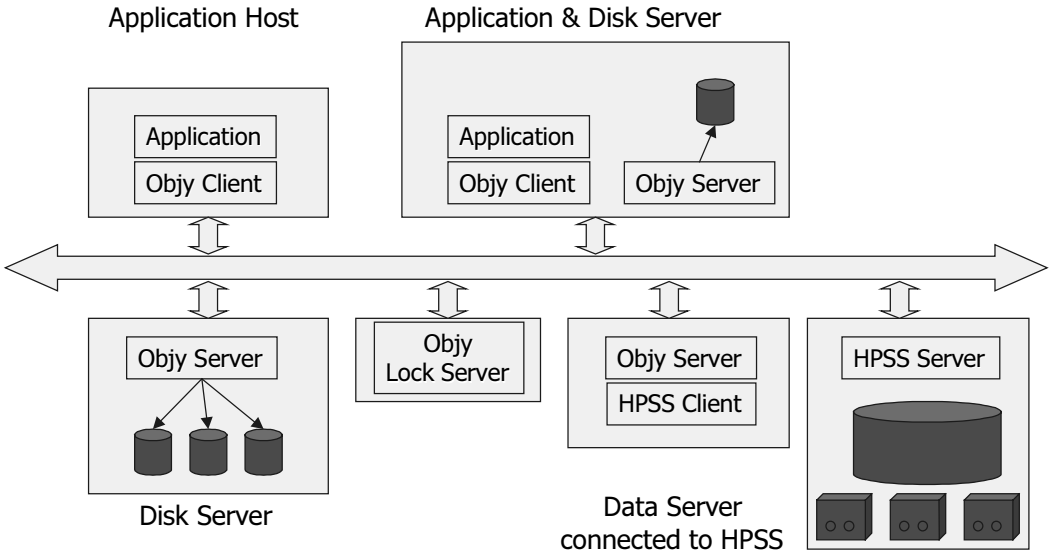


Figure 3 Distributed Applications Sharing a Federated

4.2 Physical Store Implementation

All ODBMS products use a multilevel hierarchy to implement the possibly distributed physical store. Objectivity/DB uses a hierarchy of five different levels. The topmost level - the Federated Database - keeps the catalogue of physical location of all databases that constitute the federation. Each database is structured internally into “containers” - contiguous areas of objects within a database file. Containers consist themselves of database “pages” – regions of fixed size determined at the federation creation time. Every page has “slots” for actual object data (but objects larger then a single page are allowed). Figure 4 illustrates the physical storage hierarchy in Objectivity/DB.

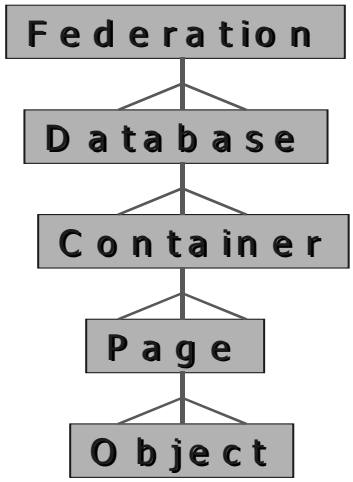


Figure 4 Storage Hierarchy in Objectivity/DB

The structure of the physical store hierarchy is directly reflected by the internal structure of the OID implementation. A 4-tuple of 16-bit numbers that represent database, container, page and slot number is used to uniquely references any object within the store.

Figure 5 Object Identifier Implementation used by Objectivity/DB

4.2.1 Separation of Logical and Physical Storage Model

The concept of OIDs allows any object to be accessed directly in the potentially large distributed store without requiring the application programmer to know the details of the store implementation, such as file and host names. Since information about the physical layout of the store is kept in a central place by the ODBMS, it is much easier to change the storage topography without compromising existing applications. One may change the location of a particular file to a new host by moving the data and changing the catalogue entry. Since the catalogue is shared by all database applications, they will use the data from the new location without any modifications.

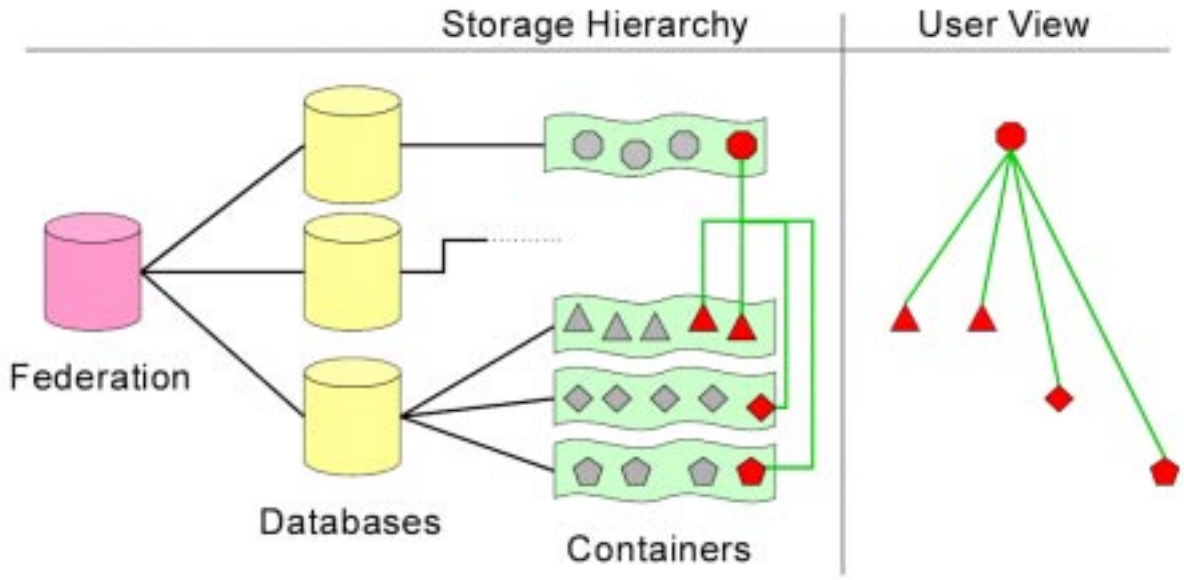


Figure 6: Physical Storage Hierarchy and Logical User View

4.2.2 Data Clustering and Re-Clustering

An important feature offered by several ODBMS products is the support for object clustering. When a persistent object is created, the programmer may supply information where the object should be placed within the physical storage hierarchy. In C++ a clustering hint may be passed as an argument to the new operator. For example, the statement

```
d_Ref<Track> aTrack = new(event)  Track;
```

instructs the database to create a new persistent track object physically close to the event object. This ability to cluster data on the physical storage medium is very important for optimising the performance of applications which access data selectively.

The goal of this clustering optimisation is to transfer only useful data from disk to the application memory (or one storage level below: from tape storage to a disk pool). Grouping data close together that will later be read together can drastically reduce the number of I/O operations needed to acquire this data from disk or tape. It is important to note that this optimisation requires some knowledge about the relative contributions of different access patterns to the data.

An simple clustering strategy is the "type based clustering" where all objects of some particular class are placed together: e.g. Track and Hit objects within an event may be placed close to each other since both classes will often be used together during the event reconstruction.

For physics analysis this simple approach is probably not very efficient since the selection of data that will be read by a particular analysis application depends more on the physics process. In this case one may group the analysis data for a particular physics process together.

4.3 Data Replication

Objectivity/DB supports the replication of all objects in a particular database to multiple physical locations. The aim of this data replication is twofold:

- To enhance performance:
  Client programs may access a local copy of the data instead of transferring data over a network.
- To enhance availability:
  Clients on sites which are temporarily disconnected from the full data store may continue to work on the subset of data for which local replicas are available.

Figure 7 shows a simple configuration where one database is replicated from site 1 to two other remote sites over a wide area network.

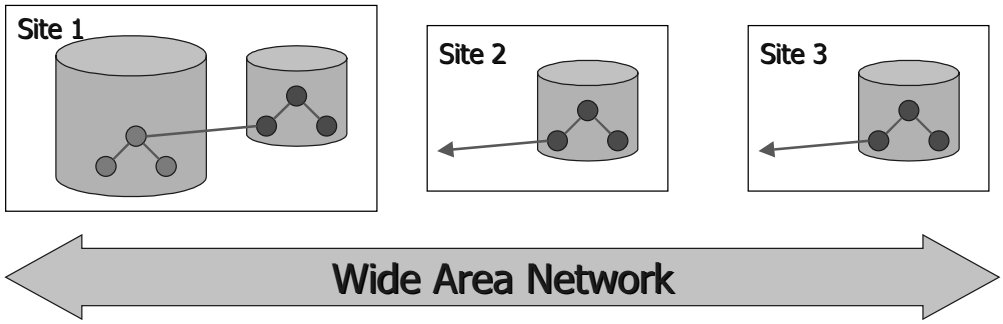


Figure 7 Database Replication

Any state changes of replicated objects on either site are transparently propagated to all other replicas by the database system. In the case that some of the replicas are not reachable, a quorum-based mechanism is used to determine which replica may be modified and a backlog of all changes is kept until other replicas become online again.

The data replication feature is expected to be very useful, for example to distribute central event selection data to multiple regional data centres.

4.4 Schema Generation

The schema generation for C++ classes in Objectivity/DB is performed using a pre-processor program (see Figure 8). The program scans class definitions of persistent classes in Objectivity's Data Definition Language (DDL) and generates C++ header and implementation files for persistent classes. The generated header files define the class interface for clients of a persistent class. The generated implementation files contain C++ code which implements smart-pointer types and various collection iterators for each persistent class. All generated files are then compiled together with any

other application code and linked against the Objectivity library to form a complete database application. The database schema is stored centrally in the federation file.

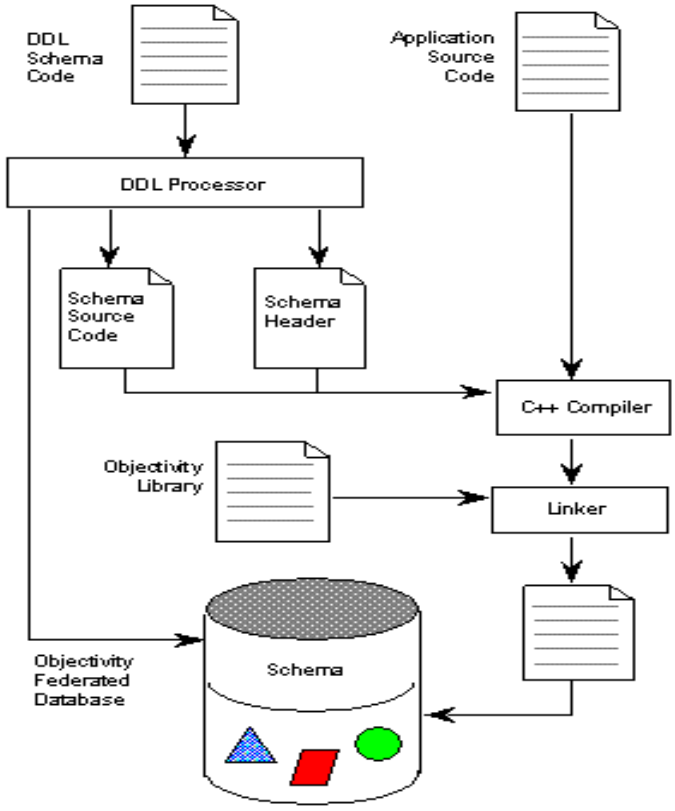


Figure 8 Schema Capture and Build Process

4.5 Creating Persistent Classes

The Data Definition Language used by Objectivity/DB is the C++ programming language extended with object associations. This makes it easy to convert transient applications and eliminates the need to learn a new programming language.

Classes become persistent by inheriting from d_Object class:

```
Class Event : public d_Object { ... };
```

Persistent classes should not contain pointers – memory pointers are meaningless in the persistent store address space. They should be replaced by references to persistent objects:

```
Event   *event_pointer;     // Event is transient
d_Ref<Event>  event_reference;   //Event is persistent
```

The “event_reference” follows the same semantic rules as the C++ pointer “event_pointer”.

The notion of pointers is further enhanced with 1-to-n and bi-directional associations. Below, the “events” attribute is a set of object references:

```
d_Ref<Event>  events[];
```

Bi-directional association is a two-directional link between objects. It has to be declared in both classes, but modification to it is an atomic operation that changes the values on both ends at the same time:

```
d_Ref<Event>  event <-> tracker;     // in Tracker
```

```
d_Ref<Tracker>  tracker <-> event;    // in Event
```

4.5.1 Persistent STL

There are some standard C++ classes that contain and use pointers internally, such as all STL containers. These classes can not be used directly with a database. Objectivity provides a special version of STL that can be used in persistent objects. The names of classes are the standard ones preceded by "d_", e.g. d_vector, d_map. Here is an example declaration of a vector of Events:

```
d_vector<Event>   my_events;
```

4.6 Object Naming

The normal way of working with a network of objects is navigation. However, the navigation has to start somewhere! Objectivity/DB allows any given object to be named and later located using this name. Objects can be named in different scopes:

- on the global level of the federation
- in scope of database or a container
- in scope of any other persistent object

Using different scopes enables the creation of personal namespaces.

4.7 Object Collections

It is very common to group objects into collections. Collections can be physical, logical or a mix of the two:

- Physical grouping is achieved by placing objects into one of the physical containers or databases of the federated database. The size of the collection is then limited by the size of the physical container it is located in.
- Logical collection is a group of references to persistent objects. The references may be stored in one of the container classes, such as a vector. The size of the collection is limited by the capacity of the collection class.
- Mixed collection is a logical collection of physical containers. The size of such a collection is practically infinite.

5. HEPODBMS LAYER

HepODBMS is a software layer that is located between the ODBMS and all other LHC++ modules. Its two main functions are to provide insulation from the database API and HEP specific storage classes.

5.1 API Independence

During the lifetime of the LHC, new versions of commercial components will be released and maybe even new products will be adopted. To make transitions between them easier, the dependence on the API of a specific vendor should be minimized. This can be achieved by using standard compliant products. However, many software products use a proprietary API that makes most efficient use of their internal architecture or are simply not fully standard compliant.

In Objectivity/DB, the structure of the federated database does not exactly reflect the ODMG database - for example, there is no notion of federation or containers in the ODMB standard. Hence, the API that deals with them is non-standard. HepODBMS tries to minimize dependence on these non-standard features by providing naming indirection and providing a higher-level database session control class.

5.2 API Enhancements

5.2.1 Database Session Control

HepODBMS contains a session control class **HepDbApplication** that provides:

- Easy transaction handling
- Methods to create databases and containers and to find them later by name
- Job and transaction level diagnostics
- The ability to set options through environmental variables

Figure 9 shows an example of a simple application using the HepDbApplication class to initialize the connection to a federated database, start a transaction and create a histogram.

```
Main(){
     HepDbApplication dbApp; // create an appl. Object
     dbApp.init("MyFD");     // init FD connection dbApp.startUpdate();
     // update mode transaction
     dbApp.db("analysis");   // switch to db "analysis"

  // create a new container
     ContRef histCont = dbApp.container("histos");
  // create a histogram in this container
     HepRef(Histo1D) h = new(histCont) Histo1D(10,0,5);

     dbApp.commit();      // Commit all changes
     }
```

Figure 9 Setting up a DB session using the HepDbApplication class

5.2.2 Object Clustering

The "new" operator generated by Objectivity for each persistent class accepts an additional parameter – the so-called clustering hint described above. Any other persistent object, container or database may serve as a clustering hint. The ODBMS will attempt to place the new object as close to the hint object as possible. In case the hint is a container or a database, the new object will be created in the container or database.

HepODBMS contains clustering classes that allow clustering objects according to different algorithms. The **HepContainerHint** class is used to store objects in a series of containers or even databases, creating a logical container of unlimited size. Special iterators allow access to all of the objects later as if they were in one container.

5.2.3 Event Collections

LHC++ users will require both "normal" size and very large (10^9) event collections. HepODBMS provides the **h_seq<T>** class that presents the programmer with a single STL-like API for all types of collections. The actual implementation of the collection depends on a strategy object that can be supplied by a user. Currently implemented strategies include:

- Vector of object references
- Paged vector of references
- Single container
- Vector of container references

The **EventCollection** class is defined as below:

```
typedef h_seq<Event> EventCollection;
```

Figure 10 shows an example of how to iterate over a collection of events using an STL-like iterator.

```
EventCollection evtCol();              // Event collection
EventCollection::const_iterator it;    // STL like iterator

For( it = evtCol.begin();  it !=  evtCol.end(); ++it )
  Cout << "Event: " << (*it)->getEventNo() << endl;
```

Figure 10 Iterating over an event collection

6. CONCLUSION

HEP data stores based on Object Database Management Systems (ODBMS) provide a number of important advantages in comparison with traditional systems. The database approach provides the user with in a coherent logical view of complex HEP object models and allows a tight integration with multiple of OO languages such as C++ and Java.

The clear separation of logical and physical data model introduced by object databases allows for transparent support of physical clustering and re-clustering of data which is expected to be an important tool to optimise the overall system performance.

The ODBMS implementation of Objectivity/DB shows scaling up to multi-PB distributed data stores and provides integration with Mass Storage Systems. Already today a significant number of HEP experiments in or close to production have adopted an ODBMS based approach.

REFERENCES

[1] The Object Database Standard, ODMG 2.0, Edited by R.G.G.Cattell, ISBN 1-55860-463-4, Morgan Kaufmann, 1997

[2] C++ Object Databases, Programming with the ODMG Standard, David Jordan, Addison Wesley, ISBN 0-201-63488-0, 1997

BIBLIOGRAPHY

Object Data Management: Object Oriented and Extended Relational Database Systems R.G.G.Catell, Revised Edition, Addison-Wesley, ISBN 0-201-54748-1, 1994

The Object-Oriented Database System Manifesto M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik. In Proceedings of the First International Conference on Deductive and Object-Oriented Databases, pages 223-40, Kyoto, Japan, December 1989.

Object Databases As Data Stores for High Energy Physics, Proceedings of the CERN School of Computing, 1998, CERN 98-08

Object Oriented Databases in Hi High Energy Physics, Proceedings of the CERN School of Computing, 1997, CERN 97-08

The RD45 collaboration reports and papers,
http://wwwinfo.cern.ch/asd/cernlib/rd45/reports.htm

RD45 - A Persistent Object Manager for HEP, LCB Status Report, March 1998, CERN/LHCC 98-x

Using an Object Database and Mass Storage System for Physics Production, March 1998, CERN/LHCC 98-x

RD45 - A Persistent Object Manager for HEP, LCB Status Report, March 1997, CERN/LHCC 97-6

Object Database Features and HEP Data Management, the RD45 collaboration

Using and Object Database and Mass Storage System for Physics Analysis, the RD45 collaboration

GLOSSARY:

ACID Transactions – Atomic, Consistent, Isolated, Durable
MB – Megabyte, 1 000 000 bytes
GB – Gigabyte, 1000 MB
PB – Petabyte, 1 000 000 GB
HEP – High Energy Physics
MSS – Mass Storage System
ODBMS – Object Database Management System
ODMG – Object Database Management Group (standards committee)
LHC - Large Hadron Collider
LHC++ - project aiming to replace the current CERNLIB libraries with a suite of OO software
DDL – Data Definition Language used by Objectivity/DB
OID – Object Identifier
MROW - multiple reader, one writer transaction where the old contents of a database region that is being modified by a writer is still accessible to other database clients in read-only mode

TRACK : INTERNET SOFTWARE TECHNOLOGIES

F.Fluckiger, track co-ordinator

1. INTRODUCTION TO THE TRACK

This track explored the general field of the Software-based technologies in use or planned over Internets and Intranets. Indeed, over the past 10 years, the field underwent considerable changes. In the late 80's, the Internet software -we understand by this the software that runs on end-systems, where the final services are delivered, or on associated server systems- was composed, from an architectural point of view, of essentially three layers.

They were:

- the transport software (also known as the TCP/IP suite)
- what could be called the middleware software, that is the intermediate software, not directly visible to the end-user, lying somewhere between the transport and the application software (example being the Domain Name Service -DNS- software)
- and finally the application software.

The latter was restricted to essentially three applications: remote login (Telnet), file transfer (FTP) and e-mail.

Then, came the web … With the introduction of this new application, the landscape changed drastically. First, a new application: web browsing rapidly became the main Internet application.

This was enriched by the deployment of a series of techniques -available from the inception of the web- to allow not only passive consultation of information, but also transaction between client and server systems. This included the handling of forms as specified by the initial standard for describing web pages (HTML) and later on the specification of a standard way for a server to recuperate the data input on an electronic form (Common-Gateway Interface: CGI).

The next stage in the development of Internet software technologies was the release by SUN of JAVA (and its script version, JavaScript) a new language to write in particular small programs to be associated with web pages. These programs could be carried by web pages travelling from the server to the client system, to be interpreted and executed by the client system -that is, in practice the web browser. This opened a new class of application were part of the usual transaction process (for example, checking the validity of certain data entered by the user in the form field) could be directly performed by the client without triggering multiple exchanges between the two parties.

Then, a new wave of application arrived, significantly distinct from the previous one, as not targeted to the computer-literate users: audio and audio-applications. Though the initial quality was extremely poor, the first implementations gave a clear indication of the wide scope of this field: ranging from two-way communication between pairs (audio, video-phony) to one-way broadcast applications (the Internet audio and "TV channels").

More recently, the Internet applications tackled another major field of computing activities, that of distributed computer. The technique called mobile agents allows pieces of programs to travel through an itinerary of computers, execute functions, possibly meet at certain points or return to their origin after completion of their circuit. The areas of potential use are vast, and many are still to be discovered.

1.1 Overview of the Track

The Internet Software Technologies track was composed of three distinct though connected topics: Distributed Computing using Software Topics Agents, Transaction Technologies, and Advanced Web.

The first course focused on a promising technique for supporting distributed computing: the use of agents written in Java. This method is applied to the specific field of Distributed Physics Analysis. The class comprised 3 lectures where the agent technology were introduced and the Java programming presented. Then, students moved to exercises where they wrote physics analysis algorithms in Java, and agent-based job submission systems, then finally merged their outcome into a global system.

The second course "Web-based Transaction Technologies" described the mechanisms and techniques for supporting client-server interactions based on web forms. It started with a presentation of the HTML language, then carried on with scripting languages (JavaScript) and the CGI interface. Two hours of exercises where students developed a simple transition system based on forms and associated CGI programs complemented the 4 hours of lectures.

The third course was devoted to a selection of more advanced web-based software topics. This included a presentation of the XML language as well as the SMIL language for the support of synchronised multimedia documents.

1.2 Introduction to XML and SMIL

(Based on view material from M. Podgorny at CSC99)

1.3 When the web was invented at CERN in the late eighties, the inventor, Tim Berners Lee designed at the same time to technique by which a client system can dialogue with a remote server (the HTTP protocol) and the way in which pages can be described. The language by which web pages are described was called HTML. HTML is a mark up language. This is a methodology to encrypt data with information about itself.

Like HTML, XML relies on rules to specify tags and the use of tag-processing applications that knows how to deal with the tags. XML is in practice a subset or a more general language called SGML. The specifications are being developed by the World Wide Web Consortium (W3C), supervised by the XM working group.

The most important difference between HTML and XML is that while HTMPO is a well defined and closed set of tags, XML is a meta-language for defining other mark-up languages: it specifies the standards with which you can define your own mark-up language. Therefore, XML may allow each specific industry to develop its own tag set to meet its unique needs. As a side result, XML may be used to describe documents intended to be mainly displayed on a screen (such a "web documents" to be displayed by a web browser) but also documents primarily intended to be printed.

XML is gaining momentum in the Internet software community and well as with major application software manufacturers.

Various other languages are XML-derived languages. An example is RDF, the Resource Description Framework, a standard for exchange what is called "meta-data" and enable better searching on the web.

Another example is SVG, the Scalable Vector Graphics language which allows the description of two-dimensional graphics in SGML. When available, browsers will no longer have to load and display byte-consuming images when simple schematics and figures are to be represented.

The Synchronised Multimedia Integration Language, SMIL, enables simple authoring of TV-like multimedia presentations such a training courses on the web. When you use a CD-ROM, it is frequent that you display sequences of different media which are synchronised together (such as a piece of text, synchronised when displayed with a sound to be played out, followed a few second later by an animation,...). To author such sequences, CD-ROM authors use specific languages, which allows to specify syntonisation between the various media components of the document. SMIL is a similar type of language, but designed to author documents which are aimed at being accessed over the Internet.

AGENTS - MOBILE AGENTS IN JAVA

M. Dönszelmann
CERN, Geneva, Switzerland

(Revised reprint of the CSC98 version)

Abstract
The CERN School of Computing provided an excellent opportunity to try out mobile agents on physics analysis. The course explained the general concepts of mobile agents and their applicability to the field of High Energy Physics. In particular the students took a look at the merits of using mobile agents for physics analysis.
The course consisted mainly of lab works which provided both inexperienced and experienced students in the fields of Object-Oriented programming and Java an interesting introduction into working with Mobile Agent Systems. A full Distributed Analysis System was put together.

1. MOBILE AGENTS

Mobile agents[1] are programs that can move around on the network, while performing their duty, which may be a calculation, a database lookup or some other service. They keep their state as they move along from one machine to the next, thereby taking with them the result they have obtained so far. Machines not only provide a place for agents to work, but also a place for agents to meet and exchange information. A mobile agent may change its itinerary depending on the information it receives from another agent. Mobile software agents very much resemble the way people act, work and meet in the real world.

The fact that mobile agents can move themselves with their information across the network can reduce network traffic. For example, to discover information on another node, we would normally interrogate a server on that node via a mechanism such as remote procedure calls (RPC). We may have a set of questions, where later questions depend on the result of earlier ones. Using RPC every question (and its answer) would be separately transferred across the network. For mobile agents we use the term remote programming (RP). We instruct an agent with a set of questions and send it (with its questions) over to the remote node. The agent asks its questions locally and comes up with the final answer. It then travels back, only taking the final answer with it. Using RP, the network bandwidth consumed is the size of the agent and its final result, rather than all intermediate information of all the questions. We use the locality of information, to reduce the network bandwidth.

TeleScript[1], an older mobile agent system, defines the concept of agent and place. An agent stays in a place and can do some work there. It may travel to another place, either by instructing itself, or being instructed by some other agent. Agents can meet other agents and communicate across the network with other agents. A place provides an environment for a agents to stay, to meet and to communicate. It provides security to protect the host system from hostile agents and to protect agents from each other. A set of collaborating agents, which can safely travel from one place to another and communicate with each other, may provide for a higher level service then the sum of the services provided by each of the agents individually.

Mobile agents have been around for some time, and many systems have been built to deploy them, such as TeleScript. The Java language[2] and its virtual machine seem to be very appropriate to implement an agent system. Java is platform independent, allows for serialization and persistency and comes with security built into the virtual machine. These are just some of the features needed to create an agent system. Several Java implementations exist today, one being the Aglet system[3] from IBM, which follows largely the concepts of TeleScript.

2. COURSE SUMMARY

The course consisted of four lectures: an introduction into agent systems, agent system implementations, examples of agent systems, and an explanation of the exercises.

The introduction into agent systems mainly followed the book by W.T. Cockayne and M. Zyda[1]. The concept of mobile agents was explained using 21 students, who played a small game on stage. In this game 7 students were persons with some characteristics. 7 other students were tasks which had to calculate some number based on the characteristics of the persons. The final 7 volunteers were agents, which moved around on stage, gathering information from the persons, doing the calculation and bringing the result back to their task.

The implementations of agent systems using the Java platform was discussed in depth, including some smaller examples such as a distributed web-search, which uses mobile agents for its search strategy and its results. In the area of High Energy Physics three examples were given: a smart e-mail system, in which e-mail is in fact a mobile agent gathering updates on a physics paper, a slow controls system in which alarms of different severity are sent around as mobile agents, who then meet and decide if a real alarm should be triggered, and a distributed physics analysis system, which the students were to build during the lab works.

The distributed physics analysis system uses mobile agents for physics jobs. This enables us to make the job travel from one data set to another, without moving the actual data. The data sets are assumed to be in different places. The job travels with its set of histograms (results), and fills them with information from each data set. Note that this is different from a job submission system, in which jobs cannot pack up and move to the next data set, and results have to be merged afterwards. An optimization could take place having agent jobs run in parallel on different data sets. Results would still have to be merged, but that responsibility is now with the agent job.

3. LAB WORK SUMMARY

The exercises consisted of three parts. The student could choose between Part-A or Part-B, followed at all times by Part-C. The documents which describe the exercises are available from the agents web site [4].

3.1 Part - A

Part-A was meant for the non-java and non-object-oriented programmers. No programming knowledge was required. Some explanation on Object Oriented Programming was provided. The description in the document for part A was fairly explicit. A step by step description on what to do, using examples, would tell the student how to reach his end goal.

As a first exercise one created a set of three simple classes, and made a small calculation. As a second exercise these classes were extended to create some agents which travelled around and did the same calculation, but in a distributed fashion.The third part of the exercise consisted of programming a small Analysis Job, thereby making use of pre-fabricated classes from an Analysis library. It was this Job which was handed to the group doing Part-B, who would run it in their Distributed Analysis System.

3.2 Part - B

Part-B was meant for the java or c++ experienced. The goal of part B was to write a distributed analysis system, which would take a job of part A and move it around on the network. An Agent System (Aglets) was provided, but some Agent classes had to be written. The description in the document for part B was in the form of requirements for classes and hints on how to implement these. There was no step by step description provided, assuming the students would know how to go about.

3.3 Part - C

At the end of the course, the whole system was exercised on the machines in the lab.

4. CONCLUSIONS

The exercises, the most important part of the course, were a big success. Most people started doing part-A, some with little knowledge on object-oriented programming. They learned some basics of Java and started understanding Agents. All groups finished it well within time.

Part-B was done by a smaller group, including all groups who finished with part-A. Part-B was not finished by everybody, mainly due to time limitations.

Only a few groups managed to get as far as Part-C with their own code. The others were given the default code. We ran the distributed data analysis system over several machines analyzing data by moving agents across the network, rather than moving data. Results were made visible on the screens of the students as well as on a central screen.

In a feedback/wrap-up session, held at the end of the course, two students presented their initial ideas on using agent systems in their work. It was also thought it would be a nice idea to try the same exercise as we had been running in the lab on the machines of the students at their work-place, thereby running the agents world-wide. A date in the near future will be picked to do such an exercise. Information can be found on the agents web site [4].

5. ACKNOWLEDGMENTS

I would like to thank those people who helped making the course on Agent System a success. In particular I thank Paolo Palazzi, my former boss, for giving me the idea to teach about agent systems at the school and Gottfried Kellner for letting me spend time on the preparation of it. The excellent team of Klaus Ackerstaff and Jaroslaw Polok, which provided the webcast (http://webcast.cern.ch) of both the lectures and the exercises. Klaus and Bob Jones are also thanked for their help as assistants during the exercises. Andreu Pacheco and Fabien Collin provided a stable set of machines on which we could do our exercises. Thanks to all of them. However, I should not forget to thank all the students of the school for their active participation, including the summer students at CERN of 1998 and 1999, who tried out the course in a mini version. You all made this course a great success.

6. REFERENCES

[1] W.T. Cockayne and M. Zyda, *Mobile Agents,* Manning Publications Co., 1998.

[2] M. Campione and K. Walrath, *The Java Tutorial: Object-Oriented Programming for the Internet,* Addison-Wesley, 1998, http://www.javasoft.com/docs/books/tutorial

[3] D.B. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets,* Addison-Wesley Publishing Co., 1998.

[4] CSC'98 - Mobile Agents Web Site, http://iptnt.cern.ch/csc99agents

RICHARD FEYNMAN AND COMPUTATION[1]

Tony Hey
Department of Electronics and Computer Science
University of Southampton, England

1. INTRODUCTION

The *Feynman Lectures on Computation* were finally published in September 1996, some eight years after his death. How did an English Professor of Computer Science come to be editing Feynman's lectures given at Caltech which he did not even attend? In November 1987, I received a phone call in Southampton from Helen Tuck, Feynman's secretary for many years, saying that Feynman wanted me to help write up his lecture notes on computation. Sixteen years earlier, as a post-doc at Caltech, I had declined the opportunity to work with Finn Ravndal on editing Feynman's 'Parton' lectures on the grounds that it would be a distraction from my research. I had often regretted my decision so I did not take much persuading this time around. At Caltech in the early 1970's, I had been a theoretical particle physicist, but ten years later, on a sabbatical visit to Caltech in 1981, I became interested in computational physics – playing with Monte Carlo and variational methods that I later found out were similar to techniques Feynman had used years before at Los Alamos. While I was there in 1981, Carver Mead gave a memorable lecture about the future of VLSI – Very Large Scale Integration – and the semiconductor industry. I returned to Southampton inspired by Mead's vision of the future and set about exploring the potential of parallel computing for computational science. Four years later, I completed my move from physics to computer science, when I moved to the Department of Electronics and Computer Science at Southampton. Two years after that, I received the call from Helen Tuck.

The official record at Caltech lists Feynman as joining with John Hopfield and Carver Mead in the fall of 1981 to give an interdisciplinary course entitled 'The Physics of Computation'. The course was given for two years although Feynman was ill with cancer during the first year and Mead on sabbatical for much of the second. A handout from the course of 1982/83 reveals the flavour of the course: A basic primer on computation, computability and information theory followed by a section titled 'Limits on computation arising in the physical world and "fundamental" limits on computation.' The lectures that year were mainly given by Feynman and Hopfield with guest lectures from experts such as Marvin Minsky, Charles Bennett and John Cocke.

In the fall of 1983, Feynman first gave a course on computing by himself, listed in the Caltech record as being called 'Potentialities and Limitations of Computing Machines'. In the following years 1984/85 and 1985/86, the lectures were taped and it was from those tapes and Feynman's notebooks that the lectures were reconstructed. In reply to Helen Tuck, I told her I was visiting Caltech in January 1988 to talk at the 'Hypercube Conference'. This was a parallel computing conference that originated from the pioneering work at Caltech by Geoffrey Fox and Chuck Seitz on the 'Cosmic Cube' parallel computer. I talked with Feynman in January and he was very keen that his lectures on computation should see the light of day. I agreed to take on the project and we agreed to keep in touch. Alas, Feynman died a month later and there was no chance for a more detailed dialogue about the proposed content of the published lectures.

---

[1] Reprinted by permission from *Contemporary Physics*, 1999, Vol.40, no 4, pp 257-267, Taylor & Francis Ltd, Eds, http://www.tandf.co.uk/journals.

As advertised, Feynman's lecture course set out to explore the limitations and potentialities of computers. Although the lectures were given over ten years ago, much of the material is relatively 'timeless' and represents a Feynmanesque overview of some fairly standard topics in computer science. Taken as a whole, however, the course was unusual and definitely interdisciplinary in content and analysis. Besides giving the 'Feynman treatment' to subjects such as computability, Turing machines (or as Feynman said 'Mr. Turing's machines'), Shannon's theorem and information theory, Feynman also discussed reversible computation, thermodynamics of computation and quantum computation. Such a wide-ranging discussion of the fundamental basis of computers was undoubtedly unique for its time and a 'sideways' Feynman-type view of the whole of computing. Not all aspects of computing are discussed in the lectures and there are many omissions, programming languages and operating systems to name but two. Nevertheless, the lectures did represent a survey of the fundamental limitations of digital computers.

Feynman was not a professional computer scientist and he covered a large amount of material very rapidly, emphasising essentials rather than exploring all the details. Having said this, his approach to the subject was resolutely practical and this is underlined in his treatment of computability theory with his decision to approach the subject via a detailed discussion of Turing machines. Feynman takes obvious pleasure in explaining how something apparently as simple as a Turing machine can arrive at such momentous conclusions. His philosophy of learning and discovery also comes through very strongly in the lectures. Feynman constantly emphasised the importance of working things out for yourself, trying things out and playing around before looking in the book to see how the 'experts' have done things. The lectures constitute a fascinating insight into Feynman's way of working.

In at least one respect the published lectures do not do justice to Feynman's course. Included along with the topics discussed above were lectures by invited speakers on a variety of what Feynman called 'advanced applications' of computers. The choice of speaker not only reflected topics that Feynman thought important but also the figures in the computational community with whom he had interacted over the years. The purpose of this article is to put on record these relationships and shed light on Feynman's contribution to the field of computation.

2. FEYNMAN'S COURSE ON COMPUTATION

We begin with a look at the evolution of the Feynman computation lectures from the viewpoint of the three colleagues who participated in their construction. As we have seen, in 1981/82 and 1982/83, Feynman, John Hopfield and Carver Mead gave an interdisciplinary course at Caltech entitled 'The Physics of Computation'. The different memories that John Hopfield and Carver Mead have of the course are an interesting contrast. Feynman was hospitalized with cancer during the first year and Hopfield remembers this year of the course as 'a disaster', with him and Mead wandering 'over an immense continent of intellectual terrain without a map'. Mead is more charitable in his remembrances but both agreed that the course left many students mystified. After a second year of the course, in which Feynman was able to play an active role, the three concluded that there was enough material for three courses and that each would go his own way.

The next year, 1983/84, Gerry Sussman was visiting Caltech on sabbatical leave from MIT intending to work on astrophysics. Back at MIT, Sussman supervised Feynman's son, Carl Feynman, as a student in Computer Science, and at Caltech, Feynman had enjoyed Abelson and Sussman's famous 'Yellow Wizard Book' on 'The Structure and Interpretation of Computer Programs.' Feynman therefore invited Sussman to lunch at the Athenaeum, the Caltech Faculty Club, and agreed a characteristic 'deal' with Sussman – Sussman would help Feynman develop his course on the 'Potentialities and Limitations of Computing Machines' in return for Feynman having lunch with him after the lectures. As Sussman says, 'that was one of the best deals I ever made in my life'.

The topics on which Feynman interacted with these three are an indication of the breadth of his interests. With Hopfield, Feynman discussed the problem of how to implement Hopfield's neural networks in parallel, on the Connection Machine. Hopfield found it curious that Feynman was not himself interested in building models of the brain – although there are many stories testifying to Feynman's interest in the way the brain worked.

From Mead, Feynman learnt about the physics of VLSI and the reasons for the silicon scaling behavior underlying Moore's Law. In 1968, Gordon Moore had asked Mead 'whether [quantum] tunneling would be a major limitation on how small we could make transistors in an integrated circuit.' This question and its answer took Mead 'on a detour that lasted thirty years.' Mead and Feynman also had many arguments about the right way to present electrodynamics and in particular about the role of the vector potential. Mead always thought Feynman evaded the issue in his famous red *Feynman Lectures on Physics*.

While Sussman was at Caltech, he initiated the building of a 'Digital Orrery', a special-purpose computer designed to do high-precision numerical integrations of planetary motions. Much to Sussman's surprise, relatively little was known about the numerical analysis for this classical problem. A serious problem with such very long integrations – Sussman set a new record of 845 million years with Jack Wisdom – is the build-up of numerical errors. Feynman spent a considerable amount of time during that year helping Sussman understand this problem.

3. REDUCING THE SIZE

Feynman had a long-standing interest in the limitations due to size, beginning with his famous 1959 lecture 'There's Plenty of Room at the Bottom', subtitled 'an invitation to enter a new field of physics'. In this astonishing lecture, given as an after-dinner speech at a meeting of the American Physical Society, Feynman talked about 'the problem of manipulating and controlling things on a small scale', by which he meant the 'staggeringly small world that is below'. He went on to speculate that 'in the year 2000, when they look back at this age, they will wonder why it was not until the year 1960 that anybody began seriously to move in this direction'. In his talk Feynman also offered two prizes of $1000 – one 'to the first guy who makes an operating electric motor ... [which] is only 1/64 inch cube', and a second 'to the first guy who can take the information on the page of a book and put it on an area 1/25,000 smaller in linear scale in such a manner that it can be read by an electron microscope'. He paid out on both – the first, less than a year later, to Bill McLellan, a Caltech alumnus, for a miniature motor which satisfied the specifications but which was a disappointment to Feynman in that it required no new technical advances. Feynman gave an updated version of his talk in 1983 to the Jet Propulsion Laboratory. He then predicted 'that with today's technology we can easily ... construct motors a fortieth of that size in each dimension, 64,000 times smaller than... McLellan's motor, and we can make thousands of them at a time.'

It was not for another 26 years that he had to pay out on the second prize, this time to a Stanford graduate student named Tom Newman. The scale of Feynman's challenge was equivalent to writing all twenty-four volumes of the *Encyclopedia Brittanica* on the head of a pin. Newman calculated that each individual letter would be only about fifty atoms wide and, using electron-beam lithography, he was eventually able to write the first page of Charles Dickens *A Tale of Two Cities* at 1/25,000 reduction in scale. Feynman's paper is often credited with starting the field of nanotechnology and there are now regular 'Feynman Nanotechnology Prize' competitions.

Rolf Landauer, who himself has made major contributions to our understanding of computational and informational limits, has contrasted the reception given to Feynman's paper with that given to a seminal paper by his late IBM colleague John Swanson, which addressed the question of 'how much memory could be obtained from a given quantity of storage material'. Swanson's paper appeared in 1960, around the same time as Feynman's 'Room at the Bottom'

paper. In Landauer's opinion, 'Feynman's paper, with its proposal of small machines making still smaller machines, was that of a supremely gifted visionary and amateur; Swanson's that of a professional in the field'. Landauer also deplores the impact of fashions in science – while acknowledging that Feynman 'was very far from a follower of fashions'. Nevertheless, such was Feynman's influence that he could very often *start* fashions, and an unfortunate side-effect of his somewhat cavalier attitude to referencing relevant prior work – that he himself had not needed to read – was that scientists such as Rolf Landauer and Paul Benioff did not always get the credit they deserved. This was an unfortunate and unintended side-effect of Feynman's way of working and in the published *Lectures on Computation* I used my editorial prerogative to set the record a bit straighter with regard to references.

Marvin Minsky was a long term friend of Feynman who also participated in the lecture course. He recalls Feynman's suspicions of continuous functions and how he liked the idea that space-time might in fact be discrete. Feynman was fascinated by the question 'How could there possibly be an infinite amount of information in any finite volume?'

4. QUANTUM LIMITS

The study of the computational limitations due to quantum mechanics really became respectable as an academic field after Feynman attended a 1981 conference at MIT on the 'Physics of Computation', organized by Ed Fredkin, Rolf Landauer and Tom Toffoli. As Landauer has remarked 'Feynman's mere participation, together with his willingness to accept an occasional lecture invitation in this area, have helped to emphasize that this is an interesting subject.' Feynman's keynote speech – 'Simulating Physics with Computers' – contained the suggestion of the possibility of constructing a 'quantum computer'. At the conference, after claiming not to 'know what a keynote speech is', Feynman proceeded to give a masterful keynote speech. In the talk he credited his entire interest in the subject to Ed Fredkin and thanked him for 'wonderful, intense and interminable arguments'. Feynman begins by discussing the question of whether a universal computer can simulate physics *exactly* and then goes on to consider whether a 'classical' computer can efficiently simulate quantum mechanics with its quantum probabilities. Only Feynman could discuss 'hidden variables', the Einstein-Podolsky-Rosen paradox and produce a proof of Bell's Theorem, without mentioning John Bell. In fact, the paper contains no references at all – but it does contain the idea of simulating a quantum system using a new type of non-Turing, quantum computer. Feynman had the insight that a quantum computer would be able to simulate quantum systems more efficiently than a classical computer. It is also interesting to see Feynman confessing that he's 'not sure there's no real problem' with quantum mechanics.

Feynman learnt much of his initial knowledge of reversible computation from Charles Bennett. A colleague of Rolf Landauer at IBM Research in Yorktown Heights, Bennett is famous for his resolution of the problem of Maxwell's Demon and for his demonstration of the feasibility of reversible computation. He has also made important contributions both to the theory of quantum cryptography and quantum teleportation. In a wonderful advertisement, shown to me gleefully by Rolf Landauer, IBM Marketing Department went overboard on Bennett's work on teleportation. Invoking images of 'Star Trek', the ad proclaimed "An IBM scientist and his colleagues have discovered a way to make an object disintegrate in one place and reappear intact in another". An elderly lady pictured in the ad talking on the telephone to a friend says "Stand by. I'll teleport you some goulash." Her promise may be 'a little premature,' the ad says, but 'IBM is working on it.' Charles Bennett was embarrassed by these claims and was later quoted as saying "In any organisation there's a certain tension between the research end and the advertising end. I struggled hard with them over it, but perhaps I didn't struggle hard enough." Bennett has recently been actively involved in exciting developments of quantum information theory, including applications of 'quantum entanglement' – a term used by Schroedinger as long ago as 1935 – and possible 'entanglement purification' techniques.

5. PARALLEL COMPUTATION

Feynman's first involvement with parallel computing probably dates back to his time at Los Alamos during the Manhattan Project. There was a problem with the 'IBM group', who were performing calculations of the energy released for different designs of the implosion bomb. At this date in 1944, the IBM machines used by the IBM group were not computers but multipliers, adders, sorters and collators. The problem was that the group had only managed to complete three calculations in nine months prior to Feynman taking charge. After he assumed control, there was a complete transformation and the group were able to complete nine calculations in three months, three times as many in a third of the time. How was this done? As Feynman explains in *Surely You're Joking, Mr. Feynman,* his team used parallel processing to allow them to work on two or three problems at the same time. Unfortunately, this spectacular increase in productivity resulted in management assuming that a single job took only two weeks or so – and that a month was plenty of time to do the calculation for the final test configuration. Feynman and his team then had to do the much more difficult task of figuring out how to parallelise a single problem.

During the 1980's, Feynman became familiar with two pioneering parallel computing systems – the Connection Machine, made by Thinking Machines Corporation in Boston, and the Cosmic Cube, built by Geoffrey Fox and Chuck Seitz at Caltech. Parallel computing was one of the 'advanced topics' discussed in the lecture course and both types of parallel architecture exemplified by the Connection Machine and the Cosmic Cube were analysed in some detail. Parallel computing was in its infancy and in 1985 Feynman talked optimistically of the future for parallel computing. In a little-known talk he gave in Japan as the 1985 Nishina Memorial Lecture, besides discussing the perceived problems of energy consumption and size limitations for future computers, Feynman also takes a position on the place of parallel computing in the future. However, as Geoffrey Fox has said, the problem is *not* that parallel computing cannot be made to work effectively for many types of scientific problems. The outstanding problem is that the size of the market for parallel computers has been insufficient to allow the development of high quality, high-level parallel programming environments that are easy to use. In addition, there is no straightforward migration path for users with large quantities of 'legacy' sequential software. Feynman's optimistic suggestion that 'programmers will just have to learn how to do it', while true for the 'Grand Challenge' type of scientific problems, has not yet come true in a commercial sense.

Over a decade on from the heady days of the Cosmic Cube, Fox has reflected on the failure of parallel computing and computational science to become a major focus for growth over the last ten years. Instead, he argues that parallel computing and computational science have evolved into the new field of 'Internetics'. This term, first coined by Fox's colleague Xiaoming Li, embodies both the technology and the expertise required to build large-scale distributed computing systems, together the exploding number of applications engendered by the Internet and the World Wide Web.

Feynman's first-hand involvement with parallel computing has been chronicled by Danny Hillis. Feynman's son Carl, then an undergraduate at MIT, was helping Hillis with his ambitious thesis project to design a new type of parallel computer powerful enough to solve common sense reasoning problems. Over lunch, one day in the spring of 1983, Hillis told Feynman he was founding a company to build this machine. After saying that this was 'positively the dopiest idea I ever heard', Feynman agreed to work as a consultant for the new company. As Hillis recounts, when Feynman was told the name of the company 'Thinking Machines Corporation' he was delighted. "That's good. Now I don't have to explain to people that I work with a bunch of loonies. I can just tell them the name of the company." What shines through the article by Hillis is Feynman's need to be involved with the details – with the implementation of Hopfield's neural networks, with a clever algorithm for computing a logarithm, and with Quantum Chromo-Dynamics using a parallel-processing version of BASIC he had devised. Feynman's triumph came

with the design of the message router that enabled the 64,000 processors of the machine to communicate. Using an unconventional method of analysis based on differential equations, he had come up with a more efficient design than that of the engineers who had used conventional discrete methods in their analysis. Hillis describes how engineering constraints on chip size forced them to set aside their initial distrust of Feynman's solution and use it in anger.

One of the earliest applications on the Connection Machine was John Conway's 'Game of Life'. This is an example of a simple cellular automaton model. Feynman was always interested in the idea that down at the bottom, space and time might actually be discrete. What we observe as continuous physics might be merely the large-scale average of the behaviour of vast numbers of tiny cells. In one of the original lecture schedules, Norman Margolus, one of leaders of current research into cellular automata, gave a lecture on 'billiard ball computers'.

6. FUNDAMENTALS

As Rolf Landauer has said of John Archibald Wheeler, '[his] impact on quantum computation has been substantial – through his papers, his involvement in meetings, and particularly through his students and associates.' Feynman was an early student of Wheeler, of course, and so was Wojciech Zurek, now a major figure in the field. In Zurek's view, Wheeler's 1989 paper, entitled 'Information, Physics, Quantum – The Search for Links', is 'still a great, forward-looking call to arms'. The credo of the paper is summarised by the slogan *It from Bit* – the hypothesis that every item of the physical world, be it particle or field of force, ultimately derives its very existence from apparatus-solicited answers to binary, yes/no questions.

Another influential figure in the computational community is Ed Fredkin, who first met Feynman in 1962. Fredkin and Marvin Minsky were in Pasadena with nothing to do one evening and they 'sort of invited themselves' to Feynman's house. The three discussed many things until the early hours of the morning and, in particular, the problem of whether a computer could perform algebraic manipulations. Fredkin credits the origin of MIT's MACSYMA algebraic computing project to that discussion in Pasadena.

Fredkin later visited Caltech as a Fairchild Scholar in 1974. The deal this time was that Feynman would teach Fredkin quantum mechanics and Fredkin would teach Feynman computer science. Fredkin believes he got the better of the deal: 'It was very hard to teach Feynman something because he didn't want to let anyone teach him anything. What Feynman always wanted was to be told a few hints as to what the problem was and then to figure it out for himself. When you tried to save him time by just telling him what he needed to know, he got angry because you would be depriving him of the satisfaction of discovering it for himself.' Besides learning quantum mechanics, Fredkin's other assignment to himself during this year was to understand the problem of reversible computation. They had a wonderful year of creative arguments and Fredkin invented Conservative Logic and the 'Fredkin Gate' – which led to Fredkin's billiard ball computer. During one of their arguments Feynman got so exasperated that he broke off the argument and started to quiz Fredkin about quantum mechanics. After a while he stopped the quiz and said "The trouble with you is not that you don't understand quantum mechanics."

7. FEYNMAN STORIES

Murray Gell-Mann, Feynman's long-time colleague at Caltech, always deplored the way Feynman 'surrounded himself with a cloud of myth' and the fact that 'he spent a great deal of time and energy generating anecdotes about himself'. In fact, I think the stories generate themselves. For example, in 1997 Ed Fredkin came to Southampton to help us celebrate the 50th anniversary of our Department of Electronics and Computer Science – as far as we know the first, specifically 'electronics' department in the world. Ed gave a talk which included an amusing Feynman story. With apologies to Ed, I would like to tell it here.

The story concerns the so-called 'twin paradox' in relativity. In his book, Feynman had written "You can't make a spaceship clock, by any means whatsoever, that keeps time with the clocks at home." Now Fredkin happened to be teaching a course and this subject came up. In thinking about the paradox, Fredkin came up with a trivial way to make a spaceship clock that *did* keep time with the clock at home. Before making a fool of himself in front of his students, Fredkin thought he'd check with Feynman first. There was, of course, an ulterior motive for doing this and that was to 'sandbag' Feynman – a thing that Fredkin loved to do but rarely succeeded. The telephone conversation went something like this. Fredkin said "It says in your book that it is impossible for a clock on the spaceship to keep time with a clock at home. Is that correct?" Feynman replied "What it says in the book is absolutely correct." Having set him up, Fredkin countered "OK, but what if I made a clock this way ...." and then proceeded to describe how his proposed clock had knowledge of the whole trajectory and could be programmed to put the 'back home' time on the face of the clock. "Wouldn't that keep time with the clocks back home?" Feynman said "That is absolutely correct." Fredkin replied "Then what does that mean about what's in your book?" Feynman's instant response was "What it says in the book is absolutely wrong!"

Anyone who has had any long-term contact with Feynman will have a fund of stories such as this one. In all the things he did, Feynman was never afraid to admit he was mistaken and he constantly surprised his audience with his direct and unconventional responses. In this way, the Feynman stories generated themselves without any overt act of creation by Feynman himself.

8. RESEARCH AND TEACHING

What these anecdotes, and what the lectures illustrate, is how intimately research and teaching were blended in Feynman's approach to any subject. Danny Hillis remembers how Feynman worked on problems at Thinking Machines. While he was engaged in solving a problem he hated to be interrupted, but once he had found a solution 'he spent the next day or two explaining it to anyone who would listen.' Explanation and communication of his understanding were an essential part of Feynman's methodology. He also had no problem about the fact that he was sometimes re-creating things that other people already knew – in fact I don't think he could learn a subject any other way than by finding out for himself.

Carver Mead remembers another, more combative side to Feynman. Besides improving his skills on integrals in duels with Hans Bethe, the hot-house atmosphere of Los Alamos during the war had honed Feynman's skills in argument: 'The one who blinked first lost the argument.' As Mead says, 'Feynman learned the game well – he never blinked.' For this reason, Feynman would never say what he was working on: He preferred 'to spring it, preferably in front of an audience, after he had it all worked out.' Mead learnt to tell what problems Feynman cared about by noticing which topics made him mad when they were brought up. Furthermore, Mead goes on to say, if Feynman was stuck about something, 'he had a wonderful way of throwing up a smoke screen' which Mead calls Feynman's "proof by intimidation."

Feynman's grasp of the big picture, coupled with his love for knowing first-hand of practical details – from low-level programming to lock-picking – gave him an almost unique perspective on any subject he chose to study. It was this mastery, both of the minutiae of a subject and of its overall intellectual framework, that gave him the seemingly effortless ability to move back and forth between the two levels at will, without getting lost in the detail or losing the overall plot.

9. HOW TO BE AN EDITOR

Feynman also declined the 'easy' option of giving the same course every year: He chose to spend a large part of the last decade of his life thinking about the fundamentals of computation. Stan Williams, who works at Hewlett-Packard on nanostructures, quotes me as saying that the *Feynman Lectures on Computation* were the most important thing I have done in my career. Now I am not sure that I quite said that, but it *is* true that I am glad his lectures have seen the light of day. Furthermore, with the publication of a companion volume, the links and connections with the people in the computational community that he was inspired by, or who were inspired by him, are recorded.

When I took on the job of putting together a companion volume, I fondly imagined it would be easier than constructing the first. I little knew what skills an 'editor' requires. Getting agreement in principle for a contribution is easy: Getting the contribution in reality is much more difficult. Some examples will make the point. Marvin Minsky was wonderfully encouraging about the project initially – but I felt bad about telephoning Marvin at his home at regular intervals, badgering him for his paper. Gerry Sussman daily demonstrates an incredible breadth and depth of knowledge, on subjects ranging from programming in SCHEME to the foundations of classical mechanics. On talking with him and Tom Knight at MIT, he described their current research project by holding up his hand and saying "I want to know how to program this." It is therefore not surprising that I found it difficult to intrude on his manifold activities and persuade him to set them aside for the time required to complete his brief contribution to this volume. I'm glad he did, since his contribution to Feynman's course was worthy of acknowledgement.

A special note of thanks is owing to Rolf Landauer: He not only was first to deliver his text but he was also wise enough to apply subtle pressure on me to complete the task. This he did by telling me he had no doubts about my skills to put together an exciting volume. There certainly were times when I doubted whether I would be able to persuade Charles Bennett to devote enough time to writing up his talk, that he had given at our Southampton Electronics celebrations, for his contribution. Since Charles was one of those who had been responsible for educating Feynman about the field, and had participated in the original lecture course, I felt it was important to persevere. Finally, I hit on the idea of telling him that his colleague, Rolf Landauer, did not think he would make my final, final deadline .....

List of Students

Gayane AGHUZUMTSYAN, Yerevan Physics Institute, Yerevan, Armenia
Aneta BARAN, Institute of Nuclear Physics, Krakow, Poland
Thomas BARON, CERN, Geneva, Switzerland
Alexei BAZAVOV, Bogolyubov Institute for Theoretical Physics, Kiev, Ukraine
Claus BEIER, Institut für Hochenergiephysik (IHEP), Heidelberg, Germany
Tito BELLUNATO, Università degli Studi di Milano, Milan, Italy
Driss Benchekroun, Université Hassan II, Casablanca, Morocco
Cristina BORDEANU, National Institute of Physics and Nuclear Engineering, Bucharest-Magurele, Romania
Holger BRAND, Gesellschaft für Schwerionenforschung GSI, Darmstadt, Germany
Ozlem CELIK, Bogazici University, Istanbul, Turkey
Sofia CHOURIDOU, Ludwig-Maximilians-Universität Muenchen, Garching, Germany
Tuba CONKA-NURDAN, Bogazici University, Istanbul, Turkey
Mario DAVID, LIP, Lisbon, Portugal
Pardeep Kumar DHIMAN, Kurukshetra University, Kurukshetra, India
Hakima ERRIDI, Faculty of Sciences, Casablanca, Morocco
Jonathan FULCHER, Imperial College London, London, UK
Gonzalo GRACIA ABRIL, CERN, Geneva, Switzerland
Philippe GRAS, CERN, Geneva, Switzerland
Gerald GUGLIELMO, Fermi National Accelerator Laboratory, Batavia, USA
Sebastian HAUSMANN, Universität Heidelberg, Heidelberg, Germany
Jukka HEINONEN, Helsinki Institute of Physics, Helsinki, Finland
Francesco IANNONE, ENEA Frascati Research Center, Frascati (Roma), Italy
Venkata Ramana Rao KAZA, University of Rajasthan, Jaipur, India
Romuald KNAP, CERN, Geneva, Switzerland
Veronique LEFEBURE, CERN, Geneva, Suisse
Michael LEVTCHENKO, PNPI, Gatchina, Russia
Serghei MALKOV, Academy of Sciences of Moldova, Kishinev, Moldova
Bruno MANSOUX, Laboratoire de l'Accelerateur Lineaire, Orsay, France
Richard MARACEK, Max-Planck-Institut, München, Germany
Renate MOHR, MPI für Physik Muenchen, München, Germany
Maria NASSIAKOU, National Research Center for Physical Sciences "Demokritos", Greece
Wolfgang OTT, Gesellschaft fuer SchwerionenforschungDarmstadt GSI, Germany
Jose Ramon PADILLA CARRENO, Universidad Politecnica de Valencia, Valencia, SPAIN
James Rezende PITON, LNLS-Laboratorio Nacional de Luz Sincrotron, Campinas, SP, Brazil
David PITZ, CEA Saclay, Gif sur Yvette cedex, France
Alessandro RAZETO, University of Genoa, Genova, Italy
Marcella SCARPA, Università degli Studi di Milano, Milano, Italy
Karsten SCHMIDT, Gesellschaft für Schwerionenforschung GSI, Darmstadt, Germany
Ullrich SCHWANKE, DESY Zeuthen, Zeuthen, Germany
Igor SFILIGOI, INFN, Frascati (Roma), Italy

Armin SIMMA, University of Linz, Linz, Austria
Venktesh SINGH, Banaras Hindu University, Varanasi, India
Kurt STOCKINGER, Department of Data Engineering, Vienna, Austria
Heinz STOCKINGER, Institute for Applied Computer Science and Information Systems, Vienna, Austria
Stephen THOMPSON, CLRC, Daresbury Laboratory, Cheshire, UK