JHEP PROCEEDINGS

International Europhysics Conference on HEP

Data handling techniques, evolutions or revolutions?

Etienne Augé*

Laboratoire de l'Accélérateur Linéaire
Université Paris-Sud
F91898 Orsay FRANCE
E-mail: auge@lal.in2p3.fr

ABSTRACT: The gigantic size of their data sets forces all present and future collaborations in experimental High Energy Physics to use efficient tools to access these data, and to carefully design the structure of the data sets and the organization of their processing. The corresponding tools may either be specifically developed or bought from industry. The present situation is controversial. It is superficially reviewed in the present contribution.

Despite its title, the matter of the present contribution is very closely related to Physics: the efficiency to process experimental data, ultimately including the efficiency of the analysis work by individual physicists, strongly depends on the organization of the data and on the tools available to access it. This has always been true, but is becoming even more crucial with those experiments producing huge quantities of data, like BABAR, D0, CDF, STAR, PHENIX, COMPASS, and the future experiments at the LHC collider.

At the moment, the computing industry is very active and various tools recently appeared on the market, but none of them exactly fits HEP experiments' needs. This is in particular the case with Object Oriented Databases which offer features considered as unnecessary by HEP experiments while we ask for much higher data volumes (typically by factors above 1000) than other customers.

This triggers hot debates in all the experiments in preparation or starting to take data. The present contribution very superficially reviews the various strategies, showing that small differences in the estimated requirements result in very different solutions adopted by the collaborations. The interested reader should direct himself to the proceedings of the recent CHEP2001 conference in Beijing[1].

1. Requirements

Table 1 shows that the data volume of some present and future experiments has nothing to do with that of the LEP experiments. The increase mainly comes from the trigger rate.

*Speaker.

[1]Computing in High Energy Physics, September 3rd to 7th 2001, http://www.hep.ac.cn/ chep01/

Some experiments search for very rare processes and ultimately select a reduced event sample while some others aim at precision measurements and make full use of all recorded physics events. In both cases, the amount of needed CPU power, disk space and tapes is also much larger than previously, as the associated cost (see Table 2). Because of the very rapid progress of the CPU and Disk hardware, one is confident that the challenge will be met without any revolution for these components. However, such a statement is not at all obvious for the overall organization of the infrastructure, and is certainly not true for the connections between CPU and Disks, even within the same box. Despite real progress, the increase in the bandwidth that a single process uses has been less than 10 in 10 years, part of it coming from the parallel access to several disks in RAID systems [2], whereas the disk capacity and the CPU power would increase by factors above 20, and the amount of data by a factor 1000. As a result, even with an infrastructure of thousands of PC boxes linked by a very high bandwidth network in several big computer centers, a single user will not be able to access more than one terabyte of data in three hours, which represents 0.05% of one year of CMS data. In the past few years, the same user was able to access typically 10 gigabytes of data, 1% of one year of Aleph data. As a result, every user has to define very precisely the kind of data that he or she wants, according to criteria defined in advance. Furthermore, data transfers between disk and CPU deserve severe optimization. This is one of the reasons why Databases are being considered: a specific language allows the user to precisely define the data he requests, and the database system is supposed to provide him with optimized ways to get those data.

	ALEPH	BABAR(2002)	CDF(2002)	CMS
Raw Event Size (MB)	0.25	0.05	0.25	1.
Event Rate (Hz)	1.	100.	75.	100.
Data volume (TB/year)	1.	900.	450.	2000.
Disk space (TB)	0.5	60.	100.	1000.
Collaborators	300	500	500	2000

Table 1: Some typical numbers from recent experiments

	CERN	Regional Centers	Total	At CERN today
CPU (kSI95)	2560	4970	7530	20
Disk (PB)	2.4	8.7	11.1	0.025
Tapes (PB)	17.6	20.3	37.9	
Cost (MEuros)	150.			

Table 2: Computing resources required by the LHC experiments [1]

Usual optimization of the access to the data consists first in separating different kinds of data. Although the vocabulary is not standard, in most experiments, a reconstruction

[2] very high bandwidth networks are necessary because of the large number of simulatneous users, or because of very special applications like video-conferencing, but a single user's job cannot make use of more than O(100 Mb/s)

program produces *Event Summary Data* (ESD) [3], including charged tracks and calorimeter clusters, from the event *Raw Data*. A second processing step allows building more sophisticated objects like electron candidates, jets, variables describing the energy flow... Physics analysis is mainly based on these *Analysis Object Data* (AOD), which definition has to evolve with the understanding of the physics and of the detector behavior. An additional very small data set called *Event Tags* (TAGS) allows a convenient selection of the events of interest for a given physics analysis. Finally, the calibration and alignements constants, together with the many parameters describing how the detector evolves with time form a specific dataset hereafter named *Data taking conditions*. As an example, table 3 shows the relevant numbers for the H1 [2] and CMS [3] experiments.

CMS		H1	
Raw data	1 MB/event	P.O.T.	200 kB/event
ESD	500 kB/event	ODS	13 kB/event
AOD	10 kB/event	μODS	1 kB/event
TAGS	0.1 to 1 kB/event	HAT	400 B/event

Table 3: Size of the different data sets for the H1 and CMS experiments

The first computing requirement concerns the massive processing of the Raw Data, producing Event Summary Data. This work is organized and scheduled. Events are processed sequentially. The bookkeeping work, although very important should not be considered as a technical challenge. At the other end of the chain, physics analysis work requires access to Analysis Objects and Event Tags, producing *Derived Physics Data* (DPD) [4] in order to interactively produce plots. Many physics channels are studied in parallel by many physicists, trying at the same time to use the maximum available statistics and to have the fastest possible turn-around of their computing jobs. In this case, the optimization process strongly relates the computing resources and the intellectual performances of the individuals. One of the challenges, specific for large worldwide collaborations is to allow easy exchanges between physicists working on similar analysis on one hand, and to avoid unnecessary duplication of the work of the other hand. This is not easily achieved when two physicists use completely separated computing resources, and this pushes the collaborations towards "distributed computing", i.e. a worldwide organization of their computing resources. One may classify into an intermediate category many kinds of systematic studies needed for a precise understanding of the detector behavior and of the performances (efficiency, biases) of the various reconstruction and selection algorithms. These require access to ESD and often to Raw data.

From the above, one might too quickly conclude that, as it was the case in the past, AOD and TAGS, representing relatively small data sets for which frequent and fast access is required, should be well separated from the rest of the data. However, a given physics

[3] called Data Summary Tapes (DST) until recently

[4] the equivalent of PAW Ntuples

analysis also requires some systematic studies to guarantee that the detector performances are understood, and the event selection procedure is unbiased, specifically for the events of interest in this analysis. At the LHC for example, physicists will desire to check the electron identification criteria specifically for those events selected with 4 lepton candidates. Having selected and studied those events using AODs and TAGs, one will therefore desire to access ESDs. Furthermore, one might want to check the reconstruction program, still for those events, implying access not only to the Raw data but also to Calibrations and to Data taking conditions. Experiments studying either very rare processes like at the LHC or performing precision measurements are confronted with the necessity of a huge number of such systematic checks, which should therefore be as easy to perform as possible, even if it is not possible to anticipate them exhaustively. The probability of fancy systematic effects such as e.g. a dependence of the rate of Higgs candidates with the temperature in the experimental cavern (following temperature dependence of the electronics gains) or a dependence of the track multiplicity with the phase of the 50Hz AC power, should be compared to the extremely low signal to background ratio. This clearly implies the possibility to easily *navigate* in the data sets, jumping from AODs to ESDs, to Raw data and Data taking conditions for a given event, with minimal restrictions (if any). Whatever the used tools, pointers allowing this navigation are stored, thus substantially increasing the disk space requirements. This makes the question of restricting navigation, particularly to Raw Data especially pertinent.

2. Tools

Programs are now written according to OO rules, usually using the C++ language. The data manipulated by these programs are structured into objects containing pointers to other related objects, allowing an easy navigation (jump from one object -a track- to another related object -a hit in a tracker detector). These objects have to be written to/read from a disk file, managed by the operating system of some computer (usually *Linux*). Furthermore, the size of the objects may change when they are modified (e.g. because the number of hits associated to a track may change when a refined version of the track reconstruction is applied). Translating the objects and their interdependencies into a normal ("flat") file and vice-versa therefore requires a *persistency system*, which may or may not be an Object Oriented Database system.

Despite rapid progress in disk technology, the large data sets do not and will probably not entirely stay on disk. Users will have to read or write tapes from time to time. With *Hierarchical Mass Storage systems*, users access data files as if they were on disk. Only the system knows that this file has been written to a tape, chosen by itself.

For many good reasons, the computing resources available to one collaboration are scattered around the world. The result is that also some data are scattered around the world. A good network and elaborate resource management procedures should allow any user to access specific data independently of where it is stored, and process it conveniently. This avoids duplication and central gathering of these data and is the main motivation to develop the GRID infrastructure.

2.1 Hierarchical Mass Storage systems

All major HEP computing centers are now equipped with a Hierarchical Mass Storage system, in charge of automatic migration of disk files from/to tapes. The user sees his files conveniently organized in a tree structure of directories and sub-directories. He is allowed to list, move and delete them. In case of read or write access, the system automatically performs a copy from tape to disk, and the user indeed uses a real disk file. The optimization of such a system, serving a large number of simultaneous users is very tricky. For example, one has to keep the number of tape mounts limited, and simultaneously to achieve a decent filling of the tapes. The problem is even more complicated if one uses parallel access to several tapes to improve the tape/disk bandwidth. Because of all navigation possibilities in the various datasets, performing resource reservation when a job starts is essentially impossible.

The HPSS system, produced by a consortium of American partners [4], is in operation in the SLAC and CCIN2P3 centers. This commercial product is expensive. CERN has developed its own system, named CASTOR [5], now in production. A sustained input bandwidth of 100 MB/s has been obtained within a Data Challenge exercise of the ALICE experiment. This is more than enough for on-line recording of the COMPASS data [6]. DESY, CERN and the QSW company have also developed the EUROSTORE system [7] in use at DESY, as an EC-funded ESPRIT project. For this purpose, they have developed the *Parallel File System* (PFS), which has been also used in ENSTORE [8], developed and operational in Fermilab.

2.2 Persistency Object Managers

Reading from/Writing to a disk file inter-related objects requires a translation ("flattening process"), which in turn requires a precise knowledge of the structure of the concerned objects. Up to recently, performing this description of the object's structure required the expertise of a computer scientist. As a result, user specific information, stored in separate files, is presently limited to very simple structures: lists of events or lists of pointers, PAW Ntuples, or ROOT Trees. Therefore, navigation in the user specific data is not really possible. Pioneer experiments like HARP might change this in the near future [9].

Most experiments use relational databases, mostly mySQL [10] or Oracle [11] systems. These are adequate tools to manage catalogs, lists of pointers, lists of event numbers and, more generally, all kinds of data which fit in tables, like calibration and alignment constants and Event Tags. However such systems are not appropriate to Objects, unless a very sophisticated interface is developed [12]. As a result, present and future experiments have either developed their own persistency system, used a commercial OO database (in practice from Objectivity inc. [13]) or the ROOT [14] system developed at CERN.

The D0 collaboration has chosen to write Raw Data, ESD and AOD in different files. A large Oracle database (1TB) contains a catalog of all events with "thumbnail information" (the equivalent of TAGs) on each of them. It also contains a list of event files, together with various run and physics information. A user provides an a priori definition of the

data he wants to process (using thumbnail-based criteria). The *Sequential Access through Metadata system* (SAM)[15], using the database, first translates this request into a list of files that need to be accessed. An optimized relation with the ENSTORE mass storage system, and an optimized use of caches give a method to successively move the relevant files from tape to disk. Once a file is available, the system provides a list of events to be processed. This list is treated sequentially, the system giving a pointer to directly access each event record in the file.

This system is optimum in the case of sequential processing of the events by the user. Since events selected for a specific physics channel analysis are physically copied in different files, most users indeed perform sequential processing of (almost) all events in the files of interest. However, navigation between Raw data, ESD, AOD (and user specific data) is impossible. The user does not have to decode and unpack the entire event record into objects. Only those objects of interest are built from the flat data. However, the user gets the entire event record, even if he is interested only in a very specific part (e.g. a specific subdetector).

In principle, Object Oriented Databases provide exactly the required data, exactly when required by the user. They allow navigation almost without restriction within the database and allow storing *collections* (like non-OO databases), i.e. lists of pointers. Instead of physically copying the events of interest for a specific physics analysis into a separate files, one may copy the pointer to any selected object of the database into a list (collection), also stored into the database. This has obvious advantages in the case of Babar that each event, being selected in average for three different physics analysis, would otherwise have to be copied in three different files, thus tripling the amount of necessary storage. On the contrary, any single user in the collaboration may define as many collections as desired. 30,000 of them are presently stored in the collaboration's database.

OODB systems have appealing features like the management of the read/write operations: the user starts a *transaction*, performs all his I/O operations, which become effective in the database only when the user ends the transaction by issuing a *commit* command. On the contrary, the user may choose to leave the database unchanged by issuing an *abort* command. In case of machine crash during a transaction, a journalling system allows a safe recovery. Safety against corruption of the data is of utmost importance when one relies on pointers to jump from an object to another: the corruption of a single pointer may make the entire database unusable.

Commercial OODB systems are relatively new on the market(less than 8 years). They have much less experience and many fewer customers than Oracle or mySQL systems. Furthermore, they have put emphasis on multiple read and write accesses to ensure maximum availability of the stored data. This results in a complicated centralized lock system to prevent a user to access data while they are being modified by someone else. This is certainly important for a plane ticket reservation system, but is of little use in HEP because most data are read-only for most users. Situation has however recently evolved, since read-only database systems are available from Objectivity and used in BABAR.

Optimization of the transfers between the database and the user's job is not at all automatic. On the contrary, careful design of the database structure (*placement*) is mandatory to benefit from the advertised I/O performances. Usual operating systems perform the disk to memory transfers in blocks of the order of a kB (*page*). They use their idle time to anticipate the next user command and prepare for its execution, e.g. by pre-fetching into memory the next (contiguous) page. In the same spirit, objects stored next to each other in an OODB are transferred together to or from the disk: one can consider the objects as stored in *pages* (Objectivity terminology), of size 50 kB in the case of ATLAS. Someone systematically using all objects in some pages will make a much more efficient use of the (critical) disk to memory bandwidth than someone using a single object in each of the accessed pages (by factors that may exceed 100).

Designing the database for an experiment requires deciding how objects will be clustered into pages, and the approximate size of those pages and sets of pages. This in turn requires guessing which access patterns will be more frequently used by the physicists (i.e. which objects will frequently be used together in a job), which has a very strong connection with the way physics analysis is performed. The unfortunate is that, for very large data sets, it is impossible in practice to modify the objects clustering: one has to get the organization correct from the beginning, or to suffer from overheads in the I/O operations. This is one of the reasons why large scale analysis exercises (Data Challenges) are of utmost importance for the preparation of the LHC experiments.

The OODB from Objectivity has been chosen first by the BABAR collaboration[16], while being also studied by the RD45 [17] group at CERN. It has now been adopted by several experiments like HARP [9], COMPASS[18], CLEO [19] and PHENIX (for calibrations and TAGs) [20]. It has the interesting feature that a given database (a *federation* in the Objectivity terminology) spans over many files[5]. In principle, this allows the development of large databases. It also allows transferring data sets between two federations by transferring files.

To minimize the performance loss due to the lock system (each user has to ask a central server whether he is authorized to read a given container or if it is being modified by someone else), the BABAR collaboration has chosen to have specific federation for the reconstruction step only, allowing the desired processing rate of 100 events/s, and other ones for the users. Data are progressively transferred from the first system to the other, which now is read-only (except for the domain where users store their collections). Objectivity is also used to store the data taking conditions and all constants. Within the same federation, an event is split into 8 different files, each with a size recently changed to 500 MB. As a result, a user reading one single event would have to load 4 GB from tape to disk.

Transition from the R&D phase to the production phase was painful, despite the huge amount of research done on large scale test benches at SLAC, in particular on the lock service. Furthermore, installing Objectivity and transferring database data from SLAC to remote institutes or computer centers remains a complicated operation. As a result, most

[5] up to 64k, but this is not enough in practice

institutes participating in BABAR use ROOT to import the data and analyse it. The analysis programs may read the data either from Objectivity or from ROOT.

The ROOT system has been designed specifically for the analysis of HEP data. It is supposed to be a complete data analysis framework in which the user inserts his own C++ code, and which includes a persistency system. No lock system is implemented (besides the one from the operating system, at the file level). The interactive part of the system is a follow-up of the very popular PAW tool. Furthermore, ROOT is a free product.

ROOT has three interesting features:

- A *tree* structure may be given to the data in the database. Objects corresponding e.g. to a given event form a *tree* with several *branches*. Objects belonging to the same branch for successive events are stored contiguously. I/O performances when sequentially accessing all objects in a branch are optimal. A branch may contain a single variable, which is then very quickly histogramed over a large number of events. This generalizes the column- wise Ntuple system in PAW. Assigning a separate branch to each subdetector of an experiment, allows fast access to the data for systematic studies involving only one subdetector.

- A gzip-like compression is built-in, allowing reducing the volume of the stored data.

- An "abstract streaming method" allows the flattening of any object into a root file directly using the relevant header files in the user code (provided the C++ standard is the same for ROOT and for the user), or using a Data Dictionary when implemented. As a result, the user may store his own objects into ROOT with very little expertise.

One of the ROOT drawbacks as compared to Objectivity is that one cannot navigate from one ROOT file to another, whereas navigation is possible between the many files in the same Objectivity federation. However, the H1Tree system [2] developed by the H1 collaboration on top of ROOT circumvents this drawback. H1 uses a mySQL relational database to store a catalog of the runs containing, for each event, the corresponding file name and a pointer for direct access to the event data in that file. A user declares a range of events to be processed, together with conditions based on the TAG information. The H1Tree system provides access to ESD, AOD and TAG (sitting in different ROOT files) and the H1PointerManager allows navigation between those files, allowing jumping from e.g. AOD to ESD.

The full ROOT system has been adopted by the ALICE [22] experiment at the LHC. ROOT persistency system is used by CDF [23], STAR [24] and H1 [2]. The ROOT interactive analysis system is used in all collaborations, more and more physicists switching from PAW to ROOT despite problems with the C++ interpreter (still slightly different from the standard C++).

ATLAS [21] and LHCb [25] experiments at the LHC develop the concept of *transient store*: in each job, a transient store is created, from which the user program requests objects. If the requested object is not already there, then it is extracted into it by the

persistency system. With this solution, the user only sees the (small, relatively simple and well defined) transient store. The piece of software in charge of the relations between the transient store and the persistency system is independant from the rest of the application software. This makes the eventual switching from one persistency system to another one relatively simple. It also allows manipulations such that the stored object is not exactly identical to the one seen by the user (obviously without loosing information). For example, those variables which may be recalculated from others in the same object, convenient for the user, need not be made persistent and are only recalculated when the object is copied into the transient store. The storage of different versions of the same object can also been made possible (although ROOT already allows this).

3. Performance estimates

Not only are the design of a data model and of a database structure critical tasks, but one also has to take into account that a data storage system will use complex mass storage systems, will be running in different sites around the world, each with its local management and local constraints, will be accessed by remote users (sometimes with fancy requests, and with whom it will be difficult to interact) and will have to behave coherently together with the other sites in the collaboration to execute common tasks. Furthermore, the cost for duplicating the large data sets (tape units and tapes) is often prohibitive. In the case of LHC experiments, Raw data cannot be duplicated at all while only a few (typically 5) ESD copies can be performed. Designing of a computing model is therefore also a difficult task, because of the many interactions of complex systems. Because of the lack of flexibility in such large organizations, one has to find the right solution at once, and converge quickly on the correct tuning of the many parameters. For example, the overall behavior of the infrastructure when one of its components approaches saturation (*bottlenecks*)is essentially unpredictable from first principles.

Year	2001	2002	2003	2004
Number of boxes	200	400	800	1200
CPU power (kSI95)	13	33	85	160
Number of disks	200	400	1200	1600
Disk capacity (TB)	16	44	120	320
Tape capacity (PB)	30	120	350	600

Table 4: LHC prototype at CERN [1]

One of the necessities in order to progress is to develop prototypes, starting from small ones, eventually devoted to a specific component, up to relatively large-scale prototypes. Prototype infrastructures must be tested by the future users in as realistic a way as possible. This is one motivation for the Data Challenges of increasing complexity planned by the LHC collaborations. However, theses exercises are very costly, not only in work, but also in money because the prices of computing hardware decrease with time: buying 10% of the total infrastructure in 2003 costs as much as buying 50% of it in 2005. For this reason, the 4 LHC experiments will work together on a single prototype (see Table 4).

To reach a detailed understanding of a prototype, to tune the parameters and optimize its performances, one needs tools to simulate the overall behavior of the system. Furthermore, with detailed comparisons between the model and the actual behavior, one can guarantee that those simulation tools are realistic and allow reliable extrapolations to more complex infrastructures, with more users, in various load conditions. Reliable simulation tools allow clear comparisons between different computing models, database structures and even database systems, leading ultimately to safe decisions and a correct sizing of the system. It might eventually show the best way to adapt to unforeseen (positive or negative) evolutions, e.g. in the network bandwidth. However, a reliable simulation of the physicists' behaviour, e.g. in case of unforeseen discoveries, especially if unprepared to saturation of the analysis facilities, will remain an important uncertainty.

The task of developing simulation (modeling) of the large, world-wide, computing infrastructures and of the way they are used by the LHC experiments is carried out by the MONARC [26] collaboration. For the sake of clarity, MONARC has classified the various sites according to the computing resources locally available:

- The main processing center, located on the experiments' site at CERN, is mainly devoted to the reconstruction of the raw data, and provides (limited) access to them by the users. It is called "Tier-0 center".
- ESD are copied into a few (5 or 6) large computer centers ("Tier-1 centers"), one of them being at CERN, providing users with access to this large data set. This probably requires Hierarchical Mass Storage systems. The number of such centers and the access policy to them should be such that all users have (controlled) access to the ESD.
- Smaller Tier-2 computer centers may be used for event simulations, which are CPU intensive operations, or serve a relatively small community of physicists.
- The model then considers the smaller machine clusters in laboratories or institutes (Tier-3) and the huge computing power represented by the PCs on almost each physicist's desk (Tier-4), particularly difficult to federate into a coherent facility, even when not used by the physicists (e.g. at night).

Each collaboration has to be imaginative on the possible role of all these components.

4. GRIDs

The aim of this development is to allow in practice a world-wide infrastructure, made of heterogeneous pieces of hardware to be used as a coherent single facility, making the whole greater than the sum of its parts. Taking into account the huge progress in the available bandwidth on wide area networks (including transatlantic links), evolving from 2 Mb/s in 1996 to 155 Mb/s in 2001 with perspectives towards 10 Gb/s in the near future, also taking into account the fact that such a facility cannot be centrally managed, even on the relatively simple question of the registration of authorized users, the GRID [27] idea has

emerged. According to this idea, the user does not care at all where the data he requires is sitting, and on which machine his job will run: he submits his job to the GRID, which will automatically determine the location of the needed data and send the job to a machine with the easiest access to those data, unless such machines are too busy. In this latter case, the required data are automatically copied elsewhere, close to an idle computer. In all cases, the output is sent back to the user at the end of the job.

The GRID has to know where all data sets are sitting, manage their replication in caches (keeping replicated files up to date when the original file is modified). It has to know at any time the load on the various computers and the status of the various network links. In the end, it has to identify a GRID user, check the operations that are authorized for him, perform proper accounting and monitor its own performance.

Several GRID projects have started in the US (PPDG [28], GriPhyN [29]). The European Community is funding the DataGrid project [30], which involves not only particle physics, but also earth observation and biology. All of them use the GLOBUS tool-kit [31]. A first complete DataGrid prototype, including a job scheduler and resource reservation system, a data management system, monitoring tools, tools to manage the computing fabrics, mass storage management and networking monitoring tools will be operational by the end of 2001 [32].

5. Conclusions

Designing an organization of the data processing (computing model) and the structure of the data storage (data model) are crucial steps for an experiment. Access to the data must be as fast and simple as possible for the most frequent access patterns. Furthermore, limitations in the possibilities to navigate within the data translates into difficulties to perform some systematic checks; but navigation makes the tools to access the data more complicated, especially for the largest data sets. To achieve a satisfactory design, one first has to imagine how data will be analyzed, which is clearly the physicists' responsibility. Because of the large data sets it manages and of its worldwide extension, the infrastructure will be complex and of limited flexibility, despite many parameters to tune. In order to make reliable comparisons between the various possible technical choices, and to reach the optimal sizing and organization of the resources, both the building of prototypes and the development of modeling tools are of crucial importance. The prototypes have to be operated under realistic load conditions, provided by Data Challenge exercises, which must involve a decent fraction of the collaborators from each experiment. Despite the rapid evolution of the software tools in general, and the limited life time of most of the companies in the field, Object Oriented data bases (either commercial or HEP specific like ROOT) will certainly remain an open possibility for part of the HEP data in the next 10 years. The GRID development is a challenge of an extraordinary complexity, in particular when the number of nodes and their geographical dispersion is large. However, progress is visible, and the first step towards the revolution of distributed computing might be achieved very soon, allowing HEP to once more play a leading role in the evolution of some computing techniques.

Acknowledgments

I would like to thank A.M. Lutz, C. Helft and RD Schaffer for their comments on the manuscript, and the organizers for this very interesting and perfectly organized conference.

References

[1] LHC Computing Review, CERN/LHCC/2001-004, february 2001

[2] A new analysis framework for H1, Andreas Meyer, these proceedings

[3] An ODBMS approach to persistency in CMS, V. Innocente and L. Silvestris, Proceedings of CHEP2000, pp 423-430

[4] http://www.sdsc.edu/hpss/

[5] http://wwwinfo.cern.ch/pdp/castor/

[6] Castor project status, J.P. Baud et al., Proceedings of CHEP2000, pp 365-369

[7] http://eurostore.web.cern.ch/eurostore/

[8] http://www-isd.fnal.gov/enstore/

[9] Data Handling in HARP, Ioannis Papadopoulos, these proceedings

[10] http://mysql.org/

[11] http://www.oracle.com/

[12] ORACLE9i products

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

[14] http://root.cern.ch // R. Brun and F. Rademakers, Nucl. Inst. and Meth. A 389 (1997) 81-86

[15] The D0 data handling system, V. White et al., CHEP2001 conference in Beijing, september 2001

[16] The BABAR database : challenges, trends and projections, I. Gaponenko et al., CHEP2001 conference in Beijing, september 2001.

[17] http://wwwinfo.cern.ch/asd/rd45/ , RD45 status report, CERN/LHCC 99-28, sept. 1999.

[18] The COMPASS offline system, A. Martin, Proceedings of CHEP2000

[19] The CLEO III data storage, M. Lohner et al., Proceedings of CHEP2000, pp 473-477

[20] The PHENIX Offline computing system, D. Morrison, Proceedings of CHEP2000

[21] The ATLAS Data Management Architecture, D. Malon, CHEP2001 conference in Beijing, september 2001

[22] The ALICE Off-line strategy, a successful migration to OO, F. Carminati, Proceedings of CHEP2000

[23] The CDF computing and analysis system, first experience, R. Colombo et al., CHEP2001 conference in Beijing, september 2001.

[24] Event data storage and management in STAR, V. Perevoztchikov, Proceedings of CHEP2000, pp 458-461

[25] Data persistency solutions for LHCb, G. Barrand et al., Proceedings of CHEP2000, pp 431-435

[26] http://www.cern.ch/MONARC/, Validation of the MONARC simulation tools, Y. Morita, Proceedings of CHEP2000, pp 418-422

[27] http://www.gridforum.org/

[28] http://www.ppdg.net/

[29] http://www.griphyn.org/

[30] http://www.eu-datagrid.org/

[31] http://www.globus.org/

[32] http://marianne.in2p3.fr/datagrid/documentation/