GG

CERN/CN/94/9

SW 9425

The Parallel Interactive Analysis Facility

CA

Fons Rademakers
(f.rademakers@cern.ch)

CERN
1211 Geneva 23, Switzerland

Abstract

Using the Parallel Interactive Analysis Facility, PIAF, users can query in real-time multi gigabyte databases. The interactive performance is obtained by executing the queries in parallel on a cluster of high speed workstations. Using five workstations in parallel a near linear speedup is achieved for compute intensive queries and, depending on the database size, even a non-linear speedup can be observed. In this paper we describe our, unique, Ntuple database model, the PAW system used to query, analyze and visualize the data and the hardware and software aspects of the PIAF system.

1 Introduction

As the amount of data from modern high energy physics detectors continues to increase, so does the demand for high performance tools for analyzing these data. In the 1980's the concepts of the *Ntuple* and PAW, the Physics Analysis Workstation, were introduced to the high energy physics community [1]. An Ntuple is a convenient way of storing event variables on an event-by-event basis which, when combined with PAW, provides a powerful method for statistical analysis of physics data. Unfortunately, the rate with which the data sizes increase is not the same as the rate with which the processor speed increases. PIAF is designed to provide a high speed data analysis system that can scale with the increasing amount of data by employing the ideas of parallel processing.

Although PIAF was designed and developed to solve the data analysis problems of the High Energy Physics community it can equally well be employed in any other field where large amounts of data need to be processed in a relatively short amount of time. In general, PIAF is extremely well suited for decision support or statistical database queries ("data mining").

2 Short History

The PAW project was launched at CERN[1] in 1986. The first public release of the system was made at the beginning of 1988. Many extension and improvements in the area of graphics and data visualization were made in the following years. An improved version of the Ntuple database system was designed and implemented in 1991 and 1992. Starting in 1992 experiments in parallel processing were done by individual team-members and, end 1992, the PIAF project was launched. In July 1993 the first sequential version of PIAF was released. The parallel version

1. European Organization for Nuclear Research, Geneva, Switzerland.

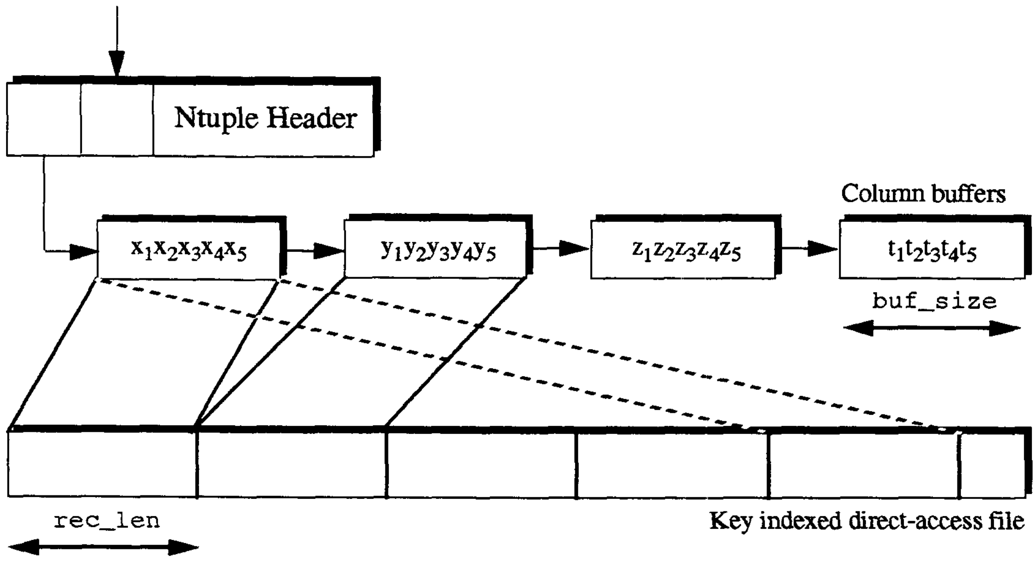

Figure 1 Layout of the Ntuple data structure. Optimal performance is achieved when `buf_size` is equal or larger than the file system record length, `rec_len`. The columns of record `i` are represented by x_i, y_i, z_i, t_i.

arrived by the end of September 1993. Since November 1993 PIAF is in production as a public service.

3 The Ntuple Data Model

The data in the Ntuple model is perceived as a table. The table is presented as an unordered set of rows with the same set of attributes or columns. A row is equivalent to the definition of a record in the more traditional data models. Once the data are stored in this form, it becomes easy, in particular with PAW, to make one- or more-dimensional projections of any of the attributes according to various selection criteria.

3.1 Storage Scheme

The nature of statistical data analysis requires direct access to complete columns, therefore the Ntuple storage mechanism has been optimized for column-wise access. To achieve this, elements of each column are stored sequentially in individual memory buffers (full vertical fragmentation). During filling of the Ntuple full memory buffers are flushed to a key indexed direct-access file and reused for the storage of new records (see Figure 1). Each key is unique and the key containing the element of a specific row can be calculated using a simple algorithm, hence also direct access to each record. Once filling is completed also the Ntuple header structure is written to the file. The direct-access file is written in a machine independent format and can be transferred using binary ftp or shared via NFS between machines of different architecture. Thanks to this storage scheme the column access time is independent of the total number of columns[2] and only an absolute minimum amount of data needs to be read from file during queries. For example accessing one column of a 300 column Ntuple will require only the reading of

1/300th of the complete database. This storage scheme also makes it very easy to add a-posteriori new columns as well as new rows to the Ntuple without having to rewrite the complete database.

Logically columns are grouped into blocks. Blocks of columns can be created, filled and retrieved independently from other blocks. This feature allows different program modules to define their own blocks whereas otherwise an Ntuple would have to be defined in one central place. Physically, however, all columns in all blocks are stored independently in their own buffers.

3.2 Data Types

The Ntuple database supports the following basic data types: 4 and 8 byte floating point numbers, 4 byte integers, 4 byte unsigned integers (bit masks), 4 byte booleans, and character strings of up to 32 characters. Arrays of each of these data types are also supported.

3.3 Data Compression

For the floating point and integer data types a value range can be specified allowing the system to pack a data item into a minimum number of bits. For example, when it is known that an integer column can only have values between 0 and 7, 3 bits are enough to store each instance of the column[3]. Booleans are always packed into 1 bit.

Arrays can be of variable length when they depend on a, so called, *index variable*. An index variable is an integer telling the system how many array elements should be stored per record. For example, if an array column `A(`*m*`)` depends on the index variable `N[0,`*m*`]` (i.e. `N` can only have values in the range of `0` to *m*) only `N` array elements are stored for each record. Thanks to a unique indexing mechanism direct access to any row remains possible without having to store additional information.

3.4 Example Ntuple Definition

An Ntuple can be created and filled in a program using a few simple subroutine calls. Below an example is given using the FORTRAN API[4]. This example shows how to define an Ntuple, in this case containing demographic information, and how to store data in it.

```
*
*-- Common blocks containing column information
*
      INTEGER            SSNUM, AGE, CHILDREN, MSTAT
      LOGICAL            SEX
      CHARACTER*12       CITY
      CHARACTER*4        STATE
      COMMON /GENERAL/   SSNUM, AGE, CHILDREN, MSTAT, SEX
      COMMON /GENERCH/   CITY, STATE

      INTEGER            EDUC_YEAR, WORK_YEAR
      REAL               SALARY(80)
```

2. Strictly speaking this is not completely true since the disk seek time will increase a little if many keys belonging to other columns have to be skipped. However, this effect is quite small and can generally be compensated for by increasing the size of the column buffers.
3. A positive side effect of this is that the system can perform range checks during the filling of the database to ensure data integrity.
4. A C language API is also available.

```
      COMMON /PROF/    EDUC_YEAR, WORK_YEAR, SALARY

      LOGICAL          SMOKING, HEART, LUNG
      COMMON /HEALTH/  SMOKING, HEART, LUNG
      ...
      ...
*
*-- Create the Ntuple
*
      CALL HBNT(10,'Population of the U.S.A.',' ')
*
*-- Define the Ntuple columns
*
      CALL HBNAME(10, 'General', SSNUM, 'SSNUM:I, AGE[0,130]:I,'//
     +            'CHILDREN[0,20]:I, MSTAT[0,3]:I, SEX:L')
      CALL HBNAMC(10, 'General', CITY, 'CITY:C*12, STATE:C*4')

      CALL HBNAME(10, 'Prof', EDUC_YEAR, 'EDUC_YEAR[0,40]:I,'//
     +            'WORK_YEAR[0,80]:I, SALARY(WORK_YEAR):R')

      CALL HBNAME(10, 'Health', SMOKING, 'SMOKING:L, HEART:L,'//
     +            'LUNG:L')
      ...
      ...
*
*-- Loop over the total population: set for each person the values in
*-- the commons and fill the Ntuple
*
      DO 10 I = 1, NPOPUL
         ...
         ...
         CALL HFNT(10)
 10   CONTINUE
```

An Ntuple is created using the routine HBNT. The column definitions are added using calls to the routines HBNAME and HBNAMC (the latter routine is used to define character columns and necessary because of the way FORTRAN passes character arguments). HBNAME takes as arguments the Ntuple ID, the block name, the address of the first column as described in the next argument and the column description. In its simplest form the column description is an exact copy of the common block definition. In the above example the third call to HBNAME shows how an index variable is used to make 'SALARY' a variable length array. Once all columns are defined the Ntuple can be filled using the routine HFNT. HFNT uses the addresses stored via HBNAME to obtain the values of each column.

A complete description of all Ntuple routines (including the different data retrieval routines) can be found in the HBOOK reference manual [2].

4 PAW – The Physics Analysis Workstation

PAW [3] is an interactive data analysis and presentation tool designed to make optimal use of the processing and graphics capabilities of modern workstations and high-end PC's. It performs best when run on a powerful workstation with a substantial amount of memory, disk space and a high resolution graphics display. However, in case such facilities are unavailable, substantial effort has been made to ensure that PAW can also be used in non-interactive (i.e. batch) mode from mainframes or compute servers.

PAW features a *command line* interface that works on any platform and an OSF/Motif based *point-and-click* interface that works on all workstations supporting the X Window system and Motif. All user interface chores are handled by KUIP (Kernel User Interface Package) [4]. KUIP parses the commands, verifies the arguments and calls the relevant PAW action routines. KUIP also supports command aliases, macros (with extensive control-flow logic) and an on-line help system. In Motif mode KUIP creates an object browser and an own terminal emulator.

PAW interfaces to many different graphics output devices via the, machine independent, HIGZ [5] graphics package. HIGZ has drivers for all major graphics systems like, X Window, GKS, PHIGS, GL, Postscript, Macintosh, PC, etc.

4.1 The Ntuple Query Processor

To facilitate the analysis and understanding of the data stored in Ntuples PAW has a wide range of commands that allow the user to query, plot and print variables or functions of variables. The four main commands are:

```
NT/PLOT id.v1[%v2[%v3[%v4]]] [selection] [nrows] [ifirst] [opt] [idh]
NT/SCAN id [selection] [nrows] [ifirst] [nvar] [v1,..., vnvar]
NT/PROJ idh id.v1[%v2] [selection] [nrows] [ifirst]
NT/LOOP id ufunc [nrows] [ifirst]
```

The PLOT command fills and plots a one- or multi-dimensional histogram showing the statistical distributions of the variables. Here, id is the numeric Ntuple identifier, v_1, v_2, v_3 and v_4 are variables or functions of variables. *Selection* is a formula involving logical operators AND, OR and NOT and comparison operators =, <=, < and so on, essentially as in SQL. *Selection* can also contain FORTRAN or C functions of which the return values are used in a comparison or as a weight. *Nrows* is the number of rows to be read from the Ntuple and *ifirst* is the starting row. Special plot options can be specified via *opt*. Optionally a predefined histogram, *idh*, can be filled.

The SCAN command prints, for every row that passes the selection, the variables v_1,. . ., v_{nvar}.

The PROJ command fills a predefined one- or two-dimensional histogram *idh* using the variables v_1 and v_2 of the rows that pass the selection. The PROJ command can be repeated several times to accumulate statistics from different Ntuples in the same histogram. To plot the histogram a separate histogram plotting command has to be issued.

The LOOP command invokes a user function for each row. The user function, *ufunc*, can either be written in FORTRAN or C. In the function the user has access to all Ntuple variables of the current row which can be used to fill one or several predefined histograms. Also all data analysis routines linked in the PAW application can be used in this function. A FORTRAN function can either be interpreted, via the COMIS [6] FORTRAN interpreter, or compiled by the native f77 compiler and dynamically linked in. C functions can only be compiled and dynamically linked in.

Before executing any of the above commands the selection formula and user functions are parsed and analyzed to find all variables used in the query. Only the used Ntuple columns are read from disk and stored in a dedicated column cache.

Single Ntuple files can have a size of up to a few hundred Mbyte. Using the `CHAIN` command Ntuple files can be chained together to form a single logical Ntuple limited in size only by the available disk space.

Query Examples

Using the Ntuple defined in section 3.4 we want to plot the age distribution of all single parents (assuming `MSTAT` is 0 for not married, 1 for married, 2 for divorced and 3 for widowed):

```
NT/PLOT 10.age children.gt.0.and.mstat.ne.1
```

To plot the same distribution for all single mothers (assuming `SEX` is false for male and true for female):

```
NT/PLOT 10.age children>0&&mstat!=1&&sex=true
```

An often used selection can be stored using the `CUT` command and later recalled by using its *cut identifier* in a selection. Cut identifiers can be referenced in the CUT command to create a tree of selections:

```
CUT $1 children>0.and.mstat<>1.and.sex=true
```

To print the social security number, years of education, city and state of all single mothers younger than 18 years:

```
NT/SCAN 10 $1.and.age<18 ! ! 4 ssnum educ_year city state
```

A `!` is a place holder for an unspecified argument of which the default value will be used. A selection can be described using either a FORTRAN or C or mixed syntax.

A more complicated typical physics query looks like this:

```
NT/PLOT 30.mass weight.f.AND.ener>5.AND.chel>-0.8.AND.mass<1.3
```

Here, `weight.f` is a FORTRAN function that is used to normalize the distribution.

4.2 The Ntuple Column Cache

The database engine stores each column in unpacked form in the column cache. The size of the column cache is a function of the available amount of memory and swapping space. Once the cache is full columns are deleted according to a least frequently, least recently used algorithm.

5 PIAF – The Parallel Interactive Analysis Facility

The Parallel Interactive Analysis Facility's main design goals were the creation of a system that would be able to process multi gigabyte databases in real time, that would scale with increasing amounts of data and that would have an affordable price tag. To reach these ambitious goals we decided to run the efficient Ntuple database engine and PAW's query processor in parallel on a dedicated cluster of powerful workstations, each equipped with large memories and fast disks and interconnected via a high bandwidth, low latency network. Other, secondary, design goals

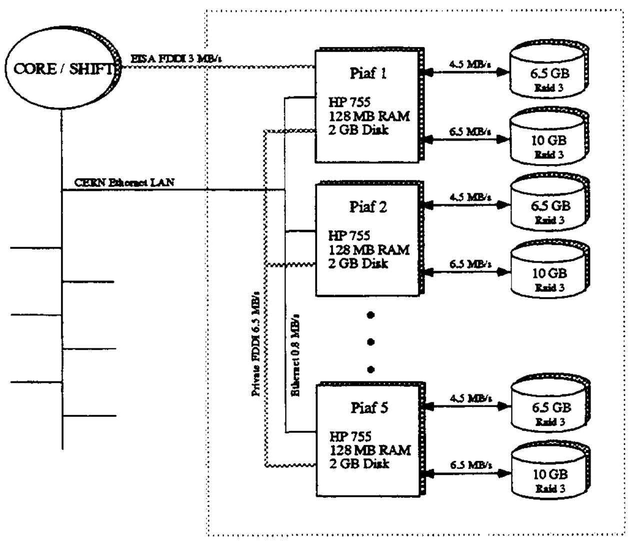

Figure 2 The current PIAF hardware configuration.

were to make the system as transparent as possible for the existing PAW users and to make the system easily clonable and portable.

In the next sections a description of the hardware configuration of the PIAF cluster currently in production at CERN and the organization of the PIAF client-server software is given. The paper concludes with some usage statistics and performance figures.

6 The PIAF Hardware

The PIAF cluster currently installed at CERN consists of five HP 755 workstations each with 128 Mbyte RAM, 2.6 Gbyte internal disk and two RAID disks (one 6.5 Gbyte and one 10 Gbyte unit). The machines are interconnected by a 6.5 Mbyte/s FDDI network and a conventional ethernet.

Of the 2.6 Gbyte internal disk 1.6 Gbyte is configured as device swap, while the remaining 1 Gbyte is used for the operating system and the users' work directories. The RAID disks are used to store the Ntuple database files. Since the Ntuples are staged from tapes or large file servers on to PIAF the Ntuples are considered to be "volatile" data and are, therefore, not backed-up. To provide some form of security the RAID systems (HP-C2425 and HP-C2430) are running in RAID mode 3, which combines high throughput (up to 6.5 Mbyte/s) with data protection. By cross-mounting the RAID systems, using NFS over FDDI, each machine in the cluster has access to a disk pool of 66 Gbyte[5]. An overview of the current hardware configuration is given in Figure 2.

5. Each 6.5 Gbyte unit looses 1.3 Gbyte and each 10 Gbyte unit looses 2 Gbyte in RAID mode 3 for parity storage.

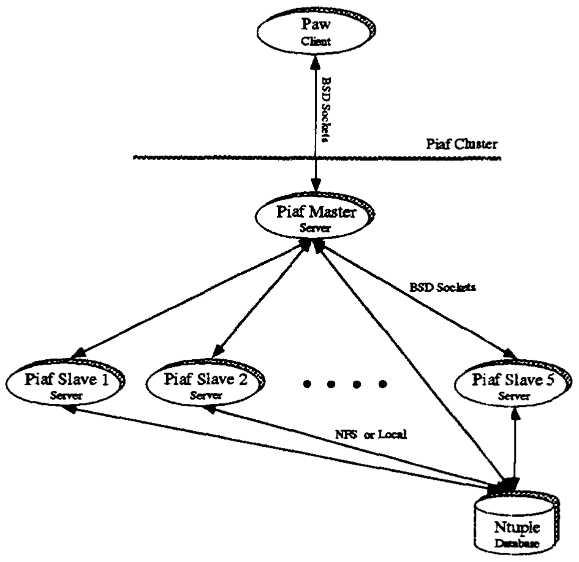

Figure 3 The PIAF processes.

In the near future (Q2-94) the cluster will be expanded with three additional HP 755's (same configuration as the existing nodes). Around the same time the 8 nodes will be upgraded to 256 Mbyte RAM each and in Q3-94 the CPU's will be upgraded to the new 125 MHz model (50% faster than the current CPU's).

6.1 Hardware Robustness

Concerning the stability of the cluster we can be short. The system did not suffer a single hour of down-time between November 1993 and April 1994. One disk of a RAID unit failed and was replaced without loosing any data.

7 The PIAF Software

The core of the PIAF software is based on the PAW query processor and the Ntuple database engine. These core components are packaged together with interprocess communication routines in the `piafserv` program. Depending on its start-up arguments this program can either become a master server or a slave server. Each user connecting to PIAF will get one master server and N slave servers, where N is the number of available nodes[6], see Figure 3.

The user's local PAW session acts as the client which communicates over the CERN-wide ethernet, or any WAN of at least 64Kbit/s, with the PIAF master server. The master server coordinates the database queries: it farms out the query commands to the slave servers, receives and merges the results, in the form of histograms or lists of row numbers, and feeds the final results back to the client.

6. This algorithm is likely to change when the number of nodes increases. Probably, the number of slaves will be limited to a subset of available nodes but can, depending of the database size, be increased for optimal performance.

Each slave knows exactly which part of the total database it has to process thanks to its *group-view*. The group-view contains two numbers: the slave server's unique ID (from 1 to N) and N (the number of active slave servers). If a slave does not respond within a certain time limit it is considered dead and the master server broadcasts a new group-view to the remaining slaves.

Since all slave servers traverse an equally sized part of the total database the system is automatically load-balanced. To maintain this balance there are no interactive sessions allowed on the CERN PIAF cluster.

7.1 Client-Server Communication

The PIAF master server is started by the `inetd` super-daemon when a client connects to a specific port on one of the nodes in the PIAF cluster. The `inetd` daemon forks a small program, called `piafront`, which handles the user authentication. To distribute the master servers evenly over all nodes `piafront` also checks on which node in the cluster the master server should be run. If the authentication was correct and the node is the right one `piafront` overlays itself with the master server. Otherwise it will either send back an authentication failure or reroute message. Once the master server is launched successfully it, in turn, will start the slave servers in a similar way. When all slaves are running the master sends an initial group-view to all slaves. During a session the user can interactively change the number of slaves participating in a query.

After all servers have been started the client will send some initial state information to the master server which, in turn, relays the information to the slaves.

To meet stringent performance, functionality and portability requirements the interprocess communication is done via BSD sockets. Message passing libraries, like PVM, were considered but rejected due to the extra overhead and configuration chores they introduce. Also, using sockets directly enabled finer-grain control over communication behavior (e.g. signal driven asynchronous I/O and TCP buffering).

8 Transparent Usage

The keep the system as transparent as possible for the existing PAW users we limited the set of new commands to access PIAF to a bare minimum. The opening and closing of a PIAF connection is done using the commands `CONNECT` and `DISCONNECT`. When connecting to PIAF the user is asked for his PIAF login and password. Once connected to PIAF the user can stage Ntuple database files using the `PUT` and/or `STAGE` commands.

To access data files on PIAF the user simply precedes the file name with the keyword '`//piaf`'. For example:

```
HISTO/FILE 1 //piaf/staff.hbook
```

tells PAW to connect the file '`staff.hbook`' on PIAF to the logical unit 1. Compare this with the command to connect the local file '`staff.hbook`' to logical unit 2:

```
HISTO/FILE 2 staff.hbook
```

From this moment on Ntuples in both files can be queried in the same way. The only difference is that when an Ntuple from logical unit 1 is queried the query command is transferred to the master server and via the master server to the slave servers. For example the command:

```
    NT/PLOT //LUN1/10.age cost>10000
```

will be executed in parallel on PIAF, whereas the command:

```
    NT/PLOT //LUN2/10.age cost>4000
```

will be executed by the local PAW session.

Since PIAF is a general public service a lot of effort was made to make the system as robust as possible. Especially, PIAF is able to gracefully handle hard interrupts (`Ctrl-C`'s) generated by users who want to break off on-going queries.

9 Usage and Performance Monitoring

To be able to track the usage and performance of the system an extensive monitoring scheme was implemented using the BSD `syslog` facility. Every begin and end of session is monitored as well as every command that is being executed. The information thus gathered tells us, for example, how many users used the system, how much data was processed, what kind of queries were executed, the real and CP times used per query, the Ntuple cache hit rate, etc. Per month about 25 Mbyte of information is logged. After storing this data in Ntuples PIAF is used for the analysis.

A short summary of the March 1994 usage statistics: 1.2 Tbyte was processed by 60 different users, connecting from 11 different countries, enjoying an average speedup of 2.5.

10 Performance Figures

The PIAF performance scales nearly linearly with the number of processors participating in the query. This is due to the small amount of time spend in the interprocess communication compared to the time it takes to traverse the database. The larger the Ntuple the better the speedup since the amount of communication is independent of the number of records processed. See Figure 4.

In some cases a non-linear speedup was observed. This happened when the database was too large to fit into the cache of a single machine. Running in parallel, however, each node could cache its part of the database which resulted in a speedup of more than N, where N is the number of nodes.

Concerning raw performance. Using 5 slaves in parallel, the system can process and histogram, on average, a column of 40 million entries in about 120 seconds (as stated before, this time is independent of the total number of columns in the database). Repeat accesses of the same column, from cache, take about 50 seconds.

The above results are obtained on a empty system with a large database evenly distributed over 10 disk devices. The day-to-day situation is not this ideal since in general users are not

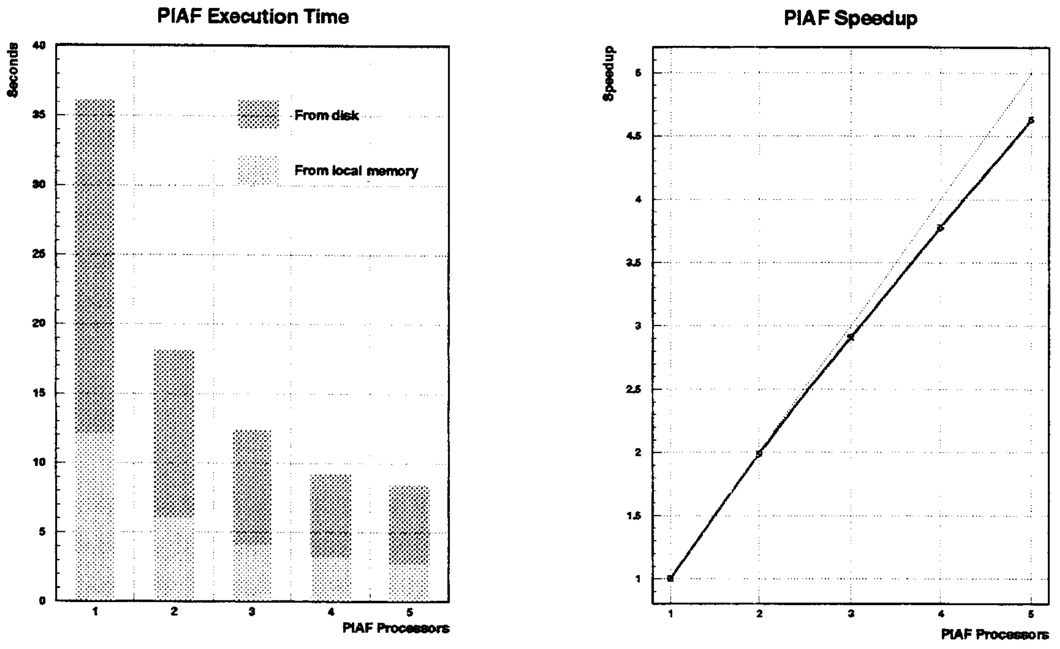

Figure 4 Real time and speedup with PIAF.

alone on the system and they use (still) relatively small databases (< 500000 rows) that are not distributed over multiple disks. Therefore, first time access to a file will not result in much speedup since the file has to be read by five slaves from a single (sequential) disk device. Repeat accesses to the same database show a better speedup but because the files are relatively small the processing time is short compared to the communication overhead, which reduces the speed-up. Figure 5 shows the average speedup obtained during normal, multi-user, operation (March 1994 data).

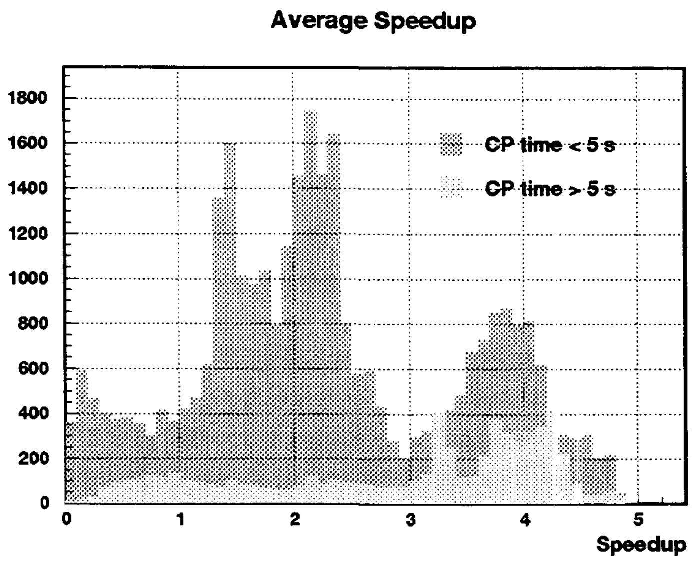

Figure 5 Experimental speedup distribution

11 Problem Areas

Clearly, the main bottleneck is caused by the fact that most databases are stored on a single disk. To solve this problem we are actively investigating the design of a parallel file system. In that case files will be striped, during staging, in N equal parts each stored on a separate device. Making this transparent for the user will be the hardest part. However, once this is working the next bottleneck will become the network throughput. Image the worst case when 5 slaves have to get their part of the database over the network. This will require a network throughput of about 32 Mbyte/s (5 times 6.5 Mbyte/s). To alleviate this problem we plan to interconnect the cluster via a HiPPI network which should give a throughput of about 25 Mbyte/s.

12 Summary & Conclusions

In this paper we presented the Ntuple database which is optimized for column-wise access to facilitate statistical data analysis. Further, we described the interactive analysis and presentation tool PAW, and its Ntuple query language. Finally, the hardware and software aspects of the parallel interactive analysis facility, PIAF, were explained.

The first phase of the PIAF project was quite successful. The system is, since November 1993, in full production and services many users with a remarkable robustness and stability.

However, many areas still need to be researched and clarified, like, the best way to make a parallel file system, a scheme for automatic data migration (hierarchical storage management), scalability with respect to an increasing number of users and nodes, cluster robustness, etc.

13 Acknowledgments

Many people contributed to the PAW and PIAF systems. Many years of ideas, discussions, comments and suggestions have gone by. A paper on PIAF cannot be published by one of the team-members without recognizing the whole team as well. In particular, René Brun, the chief architect, deserves the major credit for steering and motivating the team during this ambitious project. Finally, I wish to thank my colleagues Olivier Couet and Alfred Nathaniel who were also actively involved in the implementation of PIAF.

14 References

[1] R. Brun *et al., PAW: A General Purpose Portable Software Tool for Data Analysis and Presentation,* Comp. Phys. Comm. **57**, 432-437 (1989).

[2] CN/ASD Group, *HBOOK Users Guide (version 4.21),* CERN Program Library **Y250** (1994).

[3] CN/ASD Group, *PAW Users Guide,* CERN Program Library **Q121** (1993).

[4] CN/ASD Group, *KUIP Users Guide,* CERN Program Library **I202** (1994).

[5] CN/ASD Group, *HIGZ Users Guide,* CERN Program Library **Q120** (1993).

[6] CN/ASD Group, *COMIS - Compilation and Interpretation System,* CERN Program Library **L210** (1994).