NOTE: This copy has been prepared for external distribution. Detailed pricing information and proposals for financing have been removed from chapter 6 and appendix H.

shift

Scalable Heterogeneous Integrated Computing Facility Testbed Design Study and Implementation Proposal

Jean-Philippe Baud, Julian Bunn, David Foster

Frédéric Hemmer, Erik Jagel, Joop Joosten, Olivier Martin

Les Robertson, Ben Segal, Rainer Többicke

CN Division, CERN

CH-1211 Geneva 23

version 1 - July 25, 1990

Contents

1. Introduction

The estimated computing resources needed for data analysis by LEP experiments have consistently exceeded the planned capacity of the CERN Computer Centre. To reduce this shortfall, the LEP experiments have each set up private computing facilities. While these vary in design from mainframes to workstation farms, they share the same fundamental problems: too little cpu power; too little disk space; too little network bandwidth.

With more than six months of LEP physics data-taking behind them, considerable offline computing experience has been gained by the LEP collaborations, especially in the use of workstations as compute servers for "pass1" reconstruction, limited DST analysis and Monte Carlo event generation. This experience, together with CN Division's expertise in creating and operating large services, allows for the first time a proposal to be put forward for a large scale cpu and file service based on workstation technology.

Earlier this year we prepared a note describing a scalable model for the provision of networked cpu and file services for physics in the early 90s,[1] aiming to demonstrate that commercially available products and technology can cheaply and reliably be networked together to provide an integrated computing service for the demanding HEP physics programme. The CN Division management strongly encouraged us to undertake a detailed design study. The present paper is the result.

The paper begins with an analysis of LEP offline computing requirements in the period 1990/91 and identifies the scale of the facility which would be required to satisfy them. A set of goals for a project to create such a facility is defined, and the paper then describes in some detail the architecture of the proposed model and discusses the major technical issues. The work done on simulating the model is described and some of the results of the simulation are presented. Finally, a detailed cost estimate is provided, based on cpu, disk and network benchmarks,[2] together with estimates of the manpower requirements. The appendices contain detailed specifications and material supporting the design decisions which have been taken.

2. Lep Requirements in 1990/91

This section presents an up-to-date determination of the computing needs of the four LEP experiments. These needs were thoroughly studied by the authors of *Computing at CERN in the 1990's*, (hereafter referred to as **GB2** for Green Book II). The **GB2** estimates of computing needs are now 18 months old. This section updates the **GB2** numbers in light of the actual resources which the four LEP experiments have consumed and will need to consume[3] as LEP luminosity increases and data sets grow larger. The section is subdivided in three parts: overall CERN computing needs, LEP cpu needs, and LEP data handling needs. Predictions of needs based on current LEP experience and future LEP performance are projected two years into the future. Due to the scalability of the proposed model, hardware additions necessary to meet the needs beyond the next two years could readily be integrated

---

[1] *Network File and CPU Services for Physics, an Implementation Proposal*, CN/MCM/90/81.

[2] The cpu benchmark data was provided by Eric McIntosh, CN Division.

[3] see, e.g., Richard Mount's *Recent Experiences and Future Needs at CERN and HERA* from the Santa Fe CHEP conference, April, 1990.

into the proposed framework. Tasks which only the CERN Central Computing Service can effectively handle are separated from jobs which can be assumed by the experiments themselves.

2.1 CERN Central Computing: a Resource Overview

The central computing services offered by CERN are utilized by three main consumers: interactive users, system overhead and physics batch. The bulk of these services go to physics batch jobs. The recent trend however has been that more and more central resources are dedicated to supporting interactive users and system services (ORACLE data base enquiries, networking, staging, etc.). Currently, the fraction of cpu power consumed by physics batch is less than 30-% on the central VAX cluster, less than 50% on the central IBM and about 80% of the CRAY. During 1989, the fraction of cpu taken by interactive use on the central IBM went from 20% to 30%, and this trend should continue in the future.

In addition to an increasing load from interactive users, the next few years will see a significant increase in system overhead from the high volume of LEP data handling and network communication. According to **GB2** (p. 125) simply handling LEP DST data will take about 40 CERN units[4] of cpu power. On the networking side, measurements of cpu overhead for TCP/IP over FDDI have shown that FDDI hosts can expend 2-6 CERN units of cpu for every MB/sec of traffic (depending on vendor). Because the estimated aggregate I/O need for the LEP experiments is in excess of 10 MB/sec by 1992, networking cpu support must be included in any balanced computing services model.

Of the physics batch jobs, significant resources go to non-LEP experiments. The overall non-LEP physics batch needs were estimated by **GB2** (p. 102, 108) to be equivalent to that of the LEP experiments themselves (50% UA experiments and 50% other). This estimate did not include the large amount of simulation time currently being requested by LHC accelerator and detector physicists (90 CRAY days in 1990 for example).

Overall, the Central Computing Service provides LEP experiments with 20 CERN units on the IBM and 15 CERN units on the CRAY – that is, about 10 CERN units per LEP experiment. For the moment, there are no firm commitments to upgrade the central servers.

Each LEP experiment has been given approximately 5 GB of IBM disk space, 2 GB of CRAY space and 2 GB on the central VAX cluster. **GB2** (p. 113) recommends a minimum disk space of about 1.5% of the total data volume. Based on past and current LEP experience, each LEP experiment will have produced an average of over 20,000 tape cartridges by the end of 1990, or 4 TB of data. 1.5% of 4 TB is 60 GB. With virtually no ability to hold data on disk, tapes staging rates will rise. This already produces a heavy overhead load on the staging cpu while at the same time slowing the turn around time for physics batch jobs.

2.2 LEP CPU Needs

It was estimated in **GB2** (p. 98) that each LEP experiment would need 25 CERN units of cpu power by the end of 1990 (one million Z's)[5]. This included the cpu needed to generate and analyze Monte Carlo events. It turns out that this estimate is too low by about a factor of 3. 1990/1 cpu needs are broken down here into three categories: Pass1 reconstruction, Monte Carlo generation, and Pass2,3

---

[4] 1 CERN cpu unit is equivalent to the power of a VAX 8600.

[5] It now looks as if each experiment will have to wait until the end of 1991 before they have one million Z's

analysis jobs.

1. **Pass1 Reconstruction:** Reconstruction of raw data must be done within hours of data taking in order to ensure adequate monitoring of the detectors. This means that Pass1 must be done at CERN. To compute the cpu needs for Pass1, we use the following averages from the four LEP experiments:

   a. trigger rate of 2 Hz during collisions
   b. 10 CERN-unit-seconds per raw event analysis time

   The average LEP experiment therefore needs 20 CERN units dedicated to Pass1. The experiments are currently meeting about two thirds of this need on their own facilities, with the remainder consuming the majority of their central service allocations. All experiments hope to be able to analyze Pass1 completely independently of the CERN central service as soon as they can obtain more hardware. This then leaves their allocations free for the more I/O intensive jobs of DST analysis on large data sets, which only the large central mainframes can do effectively. In this proposal, it is assumed that few Pass1 jobs are run on the central servers. This is not a proposed restriction, just a reasonable assumption. The related job of running Pass1 on Monte Carlo generated events consumes an average of 3 CERN units per experiment. The number of generated Monte Carlo events is about 3% of the number of real triggers written to tape. Pass1 analysis time per event averages four times higher for Monte Carlo than for real (background dominated) raw data.

2. **Monte Carlo generation:** The generation of Monte Carlo events is one of the heaviest consumers of cpu for the LEP experiments. Monte Carlo needs for the LEP experiments are calculated using the following averages:

   a. generation rate of .03 Hz (for one million Z's generated per year)
   b. 500 CERN-unit-seconds per generated event

   The average LEP experiment therefore needs 15 CERN units for Monte Carlo generation. Some Monte Carlo work is currently being done by LEP experiments on the CERN central computers, but it is expected that most of this work will move off to either private computers here at CERN or to outside institutes. As with Pass1, it is assumed in this proposal that few Monte Carlo generation jobs are submitted to the central batch service. Again this is not a restriction – submission of Monte Carlo jobs to the central cpu servers would be reasonable if adequate cpu power were available.

3. **Pass2,3 Physics Analysis jobs:** The cpu needs for Pass2 and subsequent passes through the data are the most difficult to assess because heavy use of the large accumulating data sets is only just beginning. A conservative estimate is possible however, using the following averages and assumptions:

   a. Pass2 (defined to be a full reconstruction on 10% of the raw data using more accurate calibration constants and/or code) is done twice
   b. Pass3 jobs (defined to do little or no reconstruction, but to analyze tens of thousands of events per job – both real and Monte Carlo) consume one CERN-unit-second per event and are submitted at the rate of one million events analyzed daily (e.g., by ten physicists each running one job per day on 100,000 events)

The average Pass2 need is then 4 CERN units and the average Pass3 need is 12 CERN units per LEP experiment. All of this cpu need is assumed here to be provided by the CERN central computers due to the heavy dependence on cartridge tape handling. It does not include the inumerable analysis jobs which run on micro DSTs – assumed here to be run on personal workstations.

So far no mention has been made of computer downtime, job crashes, or peak *vs.* average computer use. To supply a yearly productive average of 16 CERN units of cpu to each experiment requires an online resource of *at least* 20 CERN units. We therefore state the following requirement: **a minimum central service of 20 CERN units of cpu per LEP experiment must be in place by the end of 1990.**

Because Pass2 and Pass3 cpu requirements scale approximately linearly with the amount of data, then in 1991, when the data volume is expected to double, CERN should be prepared to provide LEP experiments with 40 CERN units each of processing capability. To be more specific, CERN should provide the infrastructure for experiments to have 40 CERN units connected to the central file and cpu servers. The overhead entailed in networking and data handling *must* be CN's responsibility. We therefore state the requirement: **an additional 10 CERN units of cpu capacity must be installed to support the central file service by the end of 1990.**

The total central cpu support which CERN should provide to the LEP programme is then 90 CERN units. (This assumes that the experiments are supplying a total of 250 CERN units of power with their own resources.) 40 units are already supplied by CERNVM and the CRAY. It must be stated here that any cpu capacity needed for the rest of the CERN physics programme is in addition to these 90 units. We therefore set the following goal: **50 CERN units of cpu capacity should be installed in 1990, expandable to 100 CERN units by the end of 1991.**

2.3 LEP Data Handling Needs

LEP experiments are producing more data than can be efficiently handled by CERN's current hardware. In the distributed computing environment of this proposal, a networking backbone must be in place in order that the analyzing machines have access to data fast enough to keep their powerful cpus busy. The choice of backbone must be founded upon an estimate of the traffic this backbone will eventually have to manage. This subsection gives such an estimate, according to the model described above, that is, essentially no Pass1 and no Monte Carlo generation jobs are submitted to the central cpu and file servers. An estimate of network bandwidth necessary to support the data rates of one LEP experiment is then given by using the following facts and assumptions:

1. trigger rate of 2 Hz during collisions
2. 10% of Pass1 events go to Pass2
3. average size of Pass2 input event is 250 kB
4. average size of Pass2 output event (= Pass3 input size) is 20 kB
5. Pass2 is performed twice, on both real and Monte Carlo data
6. Pass3 jobs run an average of one million events per day
7. 60% of submitted jobs require that the data be staged in and out to a disk not local to the analyzing cpu

These assumptions lead to an aggregate I/O need per LEP experiment of 0.15 MB/sec for Pass2 and of 0.35 MB/sec for Pass3, that is, an overall bandwidth in 1990 of 2 MB/sec for the four LEP experiments. Again, because this figure represents an average over the course of a year, the actual bandwidth needed to meet the fluctuating network demands should be two to three times this amount. We therefore set the goal that **for 1990, the cpu and file server backplane bandwidth should be 6 MB/sec.**

In 1991, the network bandwidth capacity should double for the central cpu and file service jobs, in direct proportion to the cpu power present on the network. In addition, many jobs will be run on remote cpus which will force an incremental increase in bandwidth on the backplane. We have the additional goal then that **for 1991, the backplane bandwidth should increase to 15 MB/sec.**

In the above discussion, the expected average staging rate is more than 1 MB/sec. In order to support a mean lifetime of twelve hours per staged event, **a minimum of 40 GB disk staging space is necessary.** In 1991, as stage requests approach the combined IBM and CRAY staging bandwidth of 2-4 MB/sec, it will become necessary to expand the disk space in order to allow some heavily used data sets to reside for longer periods of time on disk. This forces the requirement that **at least 100 GB of disk space should be provided in the central file server for both staging and long term storage of selected data sets by 1991.**

2.4 Summary of Requirements

Assuming a minimal service (very few Pass1 or Monte Carlo simulation jobs) for the four LEP experiments only, we conclude that the central cpu and file server must be supported with at least,

1. **50 CERN units of cpu power by the end of 1990; 100 CERN units by the end of 1991.**

2. **a 6 MB/sec backplane bandwidth by the end of 1990; 15 MB/sec by the end of 1991.**

3. **40 GB of file server disk space by the end of 1990; 100 GB by the end of 1991.**

3. Goals of the Project

The following are the basic goals of the *shift* project.

- Provide a system of integrated cpu, disk and tape servers capable of satisfying the requirements specified above and providing a high quality batch service, with an overall cost-effectiveness which is at least a factor of four better than mainframe-based solutions of similar capacity. The architecture should also be capable of supporting applications other than batch, such as large-scale cpu services for interactive workstation users, and general physics-data network file services.

- The high quality batch service should be at least as good as mainframe batch.

  — availability – system in production > 95% of time

  — high reliability and resilience – MTBF > 100 hours

  — utilisation – > 80% of potential cpu delivered to physics

  — operability – < 4 operator interventions per week

  These figures must be sustained over 12 months.

- The facility must support 3480-compatible magnetic tapes and the CERN automatic cartridge tape mounting facility.

- Automated file space control, including a tape staging service, must be provided.

- The batch system must operate in a distributed environment, using a single set of queues for each cluster of cpu servers, and implementing class scheduling and low-priority background queues.

- The facility must be capable of scaling to twice the cpu capacity of the mainframes installed in the CERN Computer Centre. Today this implies up to 160 CERN cpu units, but we take as design aim a figure of 300 CERN units to cover possible growth of the mainframe configuration.

- The system must be constructed from heterogeneous components and use a Unix operating environment, in order to enforce portability of applications and software and thereby:

  — protect investment in cpu and disk hardware as the capacity grows;

  — retain flexibility towards new technologies and products;

  — encourage competitive financial and support conditions from suppliers.

- The architecture must be able to be replicated for use within individual experiments or at other institutes.

- Exploit new storage technologies.

4. Architecture

4.1 Introduction

Figure 1 on page 8 illustrates the proposed *shift* architecture. The system consists of sets of *cpu servers, disk servers and tape staging servers*, together with a single *file control server* which is responsible for managing the disk space, moving data between tape and disk, and locating staged files. The main motivations in choosing this design, which stresses separation of function, are:

- simplicity;
- ability to optimise the various components for their specific functions;
- flexibility.

The servers are interconnected by two networks : the *backplane*, a very fast network medium used for optimized special purpose data transfer (see section 4.6 on page 11), and the *secondary network*, used for control, operations and general purpose file access. The backplane is connected to the site's general purpose network infrastructure by means of an IP router, providing file services also to workstations distributed throughout the organisation with the same functionality as, though with lower performance than, the services available to systems connected directly to the backplane.

4.2 The File Base

This section describes how files are seen by the user. The distinction is made between *user files* (programs, scripts, source code, physics databases, etc.) and *physics data files*. The collection of all user files is stored in the so-called *Home File Base* while the physics data files are staged between their permanent magnetic tape homes and the *Data File Base*.

The Home File Base

User files are accessible from the cpu servers as conventional Unix files, and will thus be referenced as leaves in a tree structure (e.g. /a88k/u/c3/userx/production/test). These files could be local to the cpu servers if they are of general interest (e.g. paw, cernlib ...) or reside on a remote file system. In general, space will be allocated for the home file base for each different type of cpu server. The cpu servers will automatically mount a number of remote file systems constituting the home file base, with the Network File System (NFS) being used over the secondary network to access the files. File quotas will be implemented using standard Unix facilities wherever available.

The Data File Base

The Data File Base supports two modes of operations:

- transparent access to staged files using NFS;
- very high performance access (though with reduced functionality) using *fast file access* routines operating across the backplane.

The backplane communication layer is entirely hidden from the user by the fast file access library, which must be used explicitly by applications wishing to benefit from this high performance, low overhead facility. More information on the fast file access primitives is provided below.

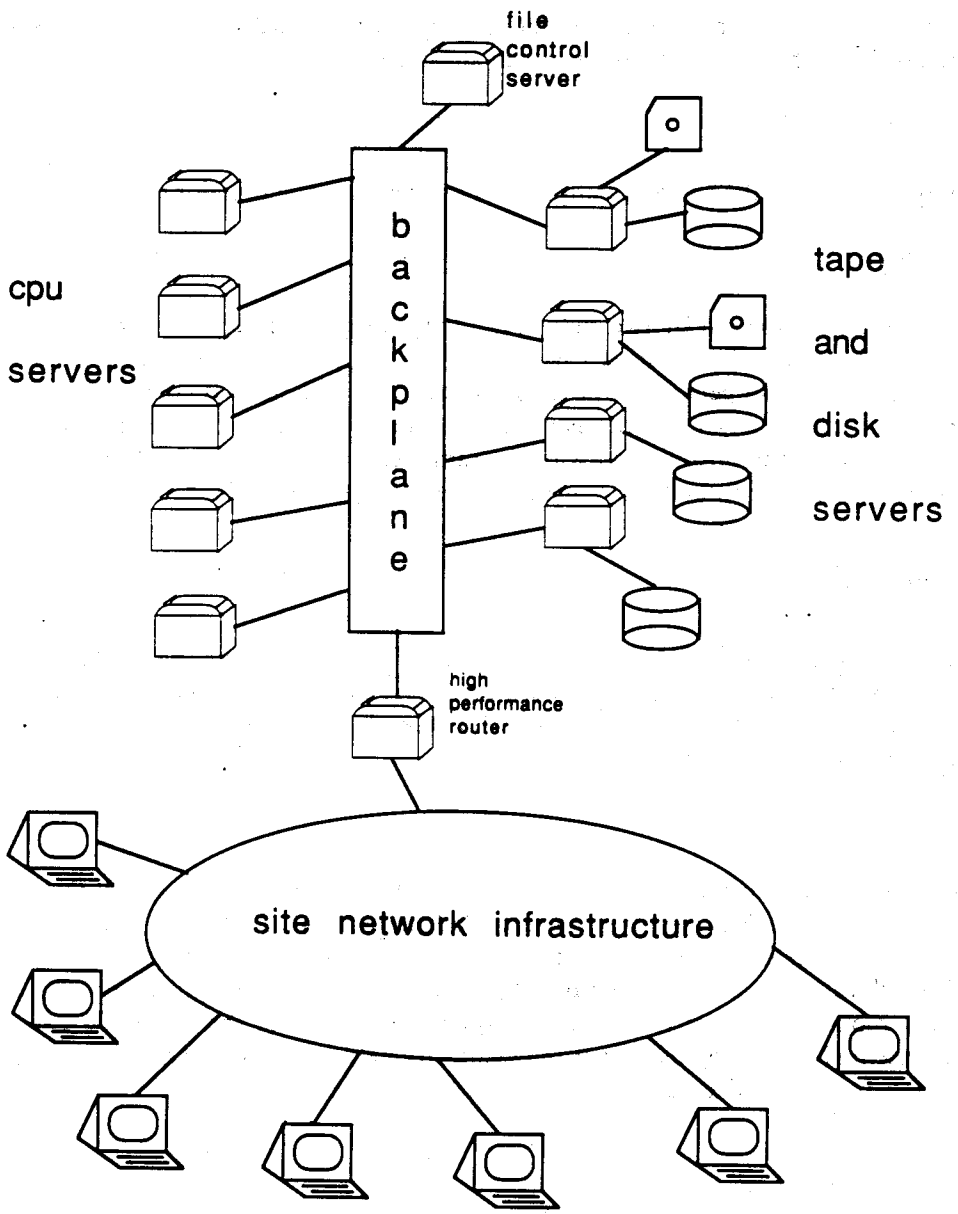

Figure 1: Overall Architecture

Prior to using files, either through NFS or the fast file access facility, the application must request access to the file through the *file control system* (fcs). The next section describes the fcs.

4.3 The File Control System

The *shift* file space is organised as a series of logical *pools*. A pool consists of one or more Unix file systems, each of which resides on a *disk server* or, in exceptional cases, on a *cpu server*. The file space is intended principally for *staged* files, which have a permanent home on magnetic tape. The disk file consists of a copy of all or part of an existing tape, or of data which will eventually be written to a pre-defined file on a specific tape.

The *file control system* (fcs) manages the *shift* pools, allocating files within pools, locating previously staged files on behalf of users, moving data between magnetic tape and disk, and performing garbage collection within each pool.

Staged files will eventually be deleted by the garbage collector using a "least-recently-used" algorithm to maintain adequate free space within the system, but a user may specify that the file should be deleted at the end of the job if it is known that it will not be re-used. Some pools may be defined for *permanent* files, subject to space budget controls. These pools are not subject to automatic garbage collection.

The user may interact with the fcs through the *stage* user procedures. These are very similar to those used on the Cray, except that the "pathname" may not be supplied by the user for input staging but will be generated and returned by the system.

stagein copies tape data to disk

stageout creates a file and specifies details of the tape to which it will eventually be copied, by *stageput*.

stageput initiates the immediate copy to tape of a file for which a *stageout* has been issued

stagewrt copies a disk file to tape, changing the characteristics of the tape file specified in a previous *stageout*

stagedel removes a file

All files in the system are identified by an *nfs_pathname*, which is the full pathname of the file as seen from any of the cpu servers. This pathname may be used to refer to the file using NFS or the *shift* fast file access procedures. The nfs_pathname normally fully describes the magnetic tape home of the file, i.e. the tape VSN/VID, label type, file sequence number, and type and location of tape-unit will usually be encoded in the pathname. In general, a user may not assume the location of a file on disk, but must use a stage command procedure to locate the file from its tape attributes.

Permanent files are handled in exactly the same way as staged files, except that they reside in special pools which are subject to space budget control. Initially the complete pool will belong to a single accounting group, which will therefore be guaranteed the use of the space allocated to the pool. A permanent file is created by means of a stageout command, specifying the tape home of the file. The details of the tape may be changed later by means of the stagewrt command.

Two important staging pools are *PUBLIC*, the default pool available for use by all users, and *LOCAL*. *LOCAL* is a pool which resides on a disk connected directly to the cpu server on which the client job is executing. Files staged in this pool will always be removed at the end of the job.

Stagein and stageout optionally issue *assign* commands to map the nfs_pathname returned onto a FORTRAN unit number for use in a subsequent program.

4.4 The Distributed Batch System : NQS

User jobs are submitted to the cpu servers using the "Network Queueing System" (NQS), originally developed for NASA. NQS is the base of the batch system for Cray Unicos and Silicon Graphics Irix.

NQS is a Unix batch and device queueing facility able to work in a networked environment. Batch requests are written as shell scripts which describe the resources needed (cpu, memory, etc.) in a header, and provide the list of UNIX commands to be interpreted by a standard shell (Bourne shell or C shell). Requests are queued either in a batch queue specified by the user or in a pipe queue waiting to be assigned to a target queue according to the resources needed. The requests may be executed on the local machine or submitted to a remote cpu server; in the latter case the job output is sent back to the originating machine. NQS deals only with stdin, stdout and stderr UNIX files. All other data files must be accessed via the fast file access system or NFS. NQS also provides job enquiry and job control over the network, an operator interface to drop a job or to change job limits, and administrative tools to define or modify queues dynamically.

In order to use NQS efficiently in the *shift* project, some enhancements are needed to support a central batch load balancing dispatcher. Jobs will be submitted to central queues (one for each logical cpu cluster), and forwarded to individual cpu servers as required. Appendix B on page 26 provides details of the required NQS enhancements.

4.5 File Access

4.5.1 Transparent File Access : NFS

The stage commands return the location of the files as a Unix (NFS) pathname. All of the file systems comprising the physics data file base are mounted on each of the cpu servers. Therefore, application programs may access transparently the data files (mapping the FORTRAN unit number to the corresponding file using the normal mechanisms provided by the operating system of the cpu server). The source programs do not need to be changed to use this facility. However, some performance penalty may occur, as NFS is not tailored to make efficient use of the backplane.

4.5.2 The Fast File Access Routines

Although NFS provides a transparent file access facility, the Fast File Access routines are an alternate optimized method to access remote files. These primitives are especially tuned towards large block data transfer using the underlying backplane facilities. The routines may be called from user-level applications, but also may be imbedded in more general purpose programs and/or libraries (e.g. paw, fatmen, zebra). Needless to say, all the primitives are available from FORTRAN and C. Note that fast file access and FORTRAN I/O cannot be mixed for the same FORTRAN unit number.

Four basic primitives are provided, in order to open, close, read or write a file. Prior to using these primitives, the file must have been located using one of the stage commands of the file control system. The open statement may be omitted, in which case all defaults will be assumed, and the actual open will take place at the first I/O operation. Similarly, the close statement is optional, the process exit taking care of releasing and closing all resources. FORTRAN programs will refer to a file using a FORTRAN unit number. An ASSIGN facility is provided to map the unit number onto a remote file name (See appendix C on page 28).

A dedicated server process is created on the disk server for each opened file during the file open operation (explicit or implicit). This server then handles any subsequent I/O from the requesting program using optimized I/O transfer over the backplane. On file close, or when the application process terminates, the server closes the file. The client/server model is straight-forward, and should be able to cater for exceptions and abnormal conditions such as process abortion, when it will perform appropriate recovery/cleanup actions.

4.6 The Network

The network consists of a very high-speed "backplane" connecting disk servers, cpu servers, tape servers and the file control server.

There will also be a secondary network consisting of a standard LAN (Ethernet and/or FDDI) to be used for general-purpose access. There will also be a high performance gateway between the backplane and the main CERN networking infrastructure, to permit access to the disk servers from external cpu's, i.e. by workstations not directly connected to the backplane, at CERN and also in external institutes.

4.6.1 File Staging Strategy Summary

File I/O is performed in three steps: Step 1 involves staging input data from tapes on to disk servers; Step 2 involves accessing those disks from the cpu servers during job execution; Step 3 involves staging back any required output files to cassette tape. The choice of this staging approach permits the isolation of the complex operational and software aspects of online tape handling, enables the movement of data between disk and tape to be optimised for high performance and low overhead, and simplifies the management of the distributed servers.

4.6.2 Networking Feasibility

The overall feasibility of *shift* from a networking aspect depends upon the quantity of I/O traffic generated by a given mix of jobs running on a system with given cpu capacity. The Appendix D on page 31 attempts to estimate the network's overall peak and sustained data rates plus the peak and sustained data rates needed per I/O interface, for a nominal configuration of 100 CERN cpu Units delivered to a representative job mix.

The conclusion is that, in order to provide sufficient overall and per-interface I/O capacity, a scalable system of this or greater capacity will require use of a backplane such as UltraNet. FDDI or Ethernet will not enable the system to be built from a reasonably small and manageable number of elements.

4.7 Management, Operations and User Support

In the heavily heterogeneous and distributed environment so far described, one of the major issues is management : user and disk quotas, file space and organization, performance monitoring, accounting, authorization, operation. Athough many standard Unix tools may be used (ac(8), sa(8), quota, rquota, yellow pages, NFS) these may not be available on all the environments. Moreover various statistics must be collected to understand and tune the behaviour of the overall system.

Operation of such a system will be quite challenging, as experience has shown that even homogeneous distributed systems are difficult to manage and operate. Current experience of workstation operation has not yet shown the high degree of reliability usually found on big mainframes. However, the Unix world helps here as it provides tools like remote backup and distributed file access.

A very important factor is the provision of adequate user support to ensure the service quality that users have traditionally found while using mainframe computers. However, as only standard or common facilities are used, it is expected that some of the aspects of user support may be provided by the Unix and Workstation support services. The "resource management" function will be vital to ensure the efficient allocation and use of the resources of the facility.

5. Simulation of the Model

A simulation program has been used to evaluate various combinations of file and cpu servers and network devices. This program takes as input a configuration data file. The data file contains a description of the type of job that will execute on the facility, the number and type of each cpu and disk server, the length of time the simulation should run, and so on. The program takes into account multi-processing cpu servers, multi-stream disk servers, input data tape staging probability, the size of the executable job image, the frequency at which jobs arrive on the queue, and various other parameters that affect performance. Further details may be found in Appendix G on page 36.

The simulation was used, for example, to find the optimum number of jobs that should be allocated per processor in order to minimise the idle time. Results showed that, contrary to expectations, multiple-cpu servers required around two jobs per processor, as did single-cpu servers. This of course means that the decision on whether to equip the facility with many single-cpu servers or fewer multiple-cpu servers becomes one of financial cost only from this point of view. Some of the most useful results obtained from the simulation indicate peak backplane data rates and power delivered by the facility to jobs (or, conversely, the idle time for the facility). Examination of data rates at the backplane-server interfaces also indicate bottle-necks for the proposed configuration. In fact, the staging rate deliverable by the tape staging server (the Cray in our model), proves to be a significant problem. The Figures show some representative plots obtained from simulating a configuration of the size proposed. Figure 1 on page 13 is a plot of the data rate on the backplane, in MBytes per second. Figure 2 on page 13 indicates the staging rate achieved by the Cray, and has a peak at the practical maximum, showing a bottleneck. Figure 3 on page 14, a plot of the power delivered by the cpu servers to jobs, nevertheless shows a satisfying peak close to the full cpu capacity. For comparison, we include Figure 4 on page 14 which shows the delivered cpu for the same configuration using FDDI instead of UltraNet.

The simulation has also been run using a configuration with twice the cpu server power of our proposed configuration, 50% more disk servers, and an additional tape staging server. The results indicate that the model is practical also on this scale, using current technology.

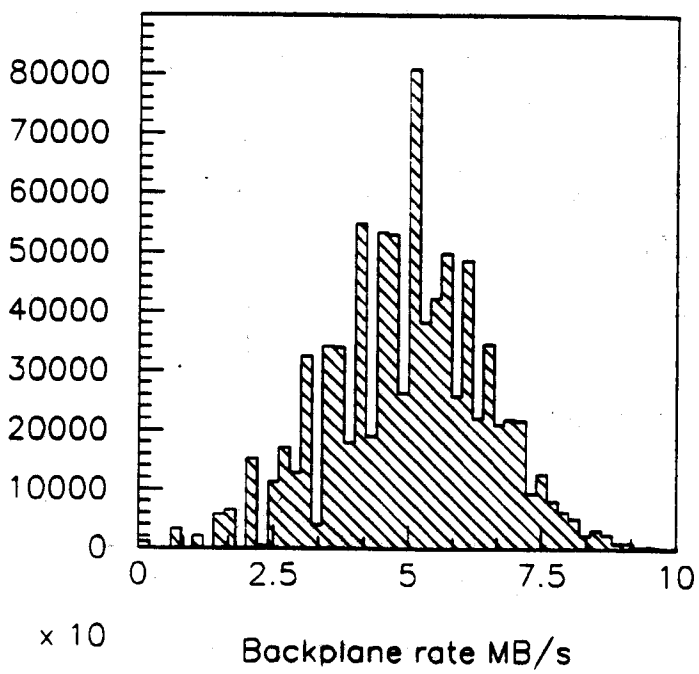

Figure 1: Data Rate on the *shift* Backplane

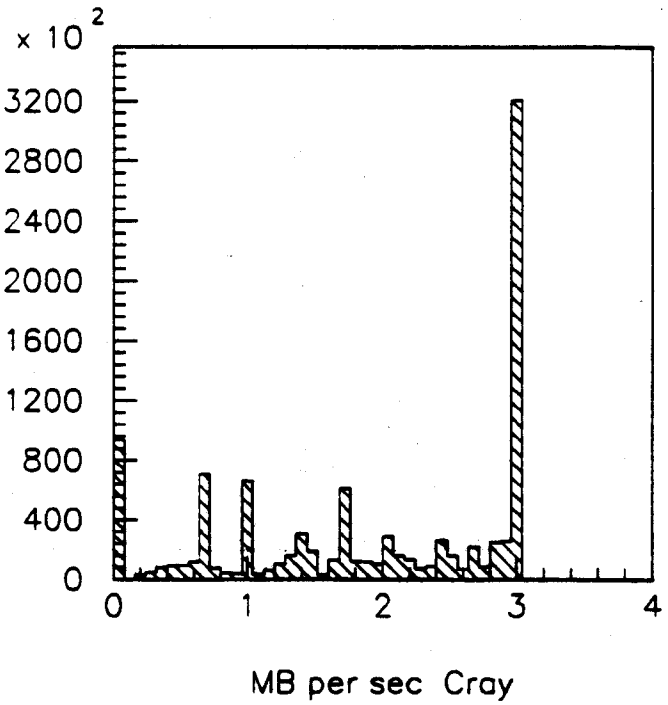

Figure 2: Staging Rate for the *shift* Tape Staging Server

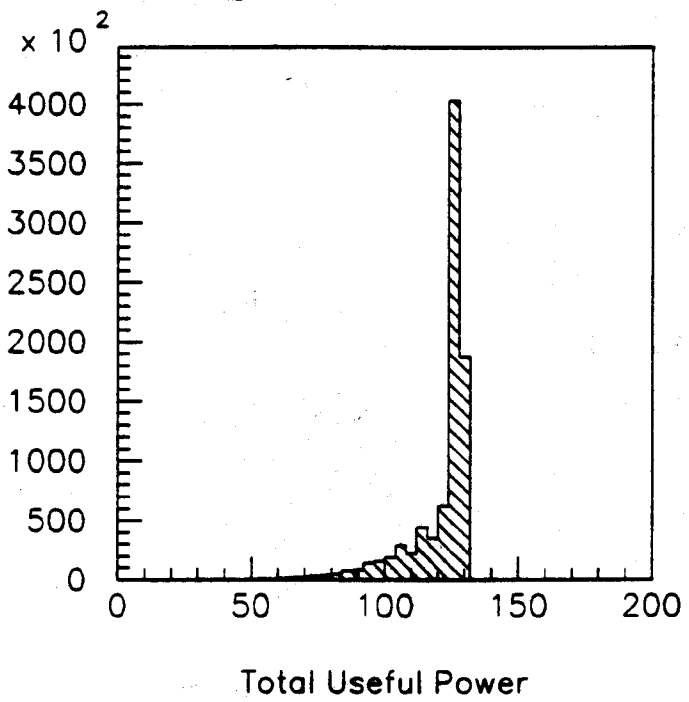

Figure 3: Delivered Power by the *shift* CPU Servers

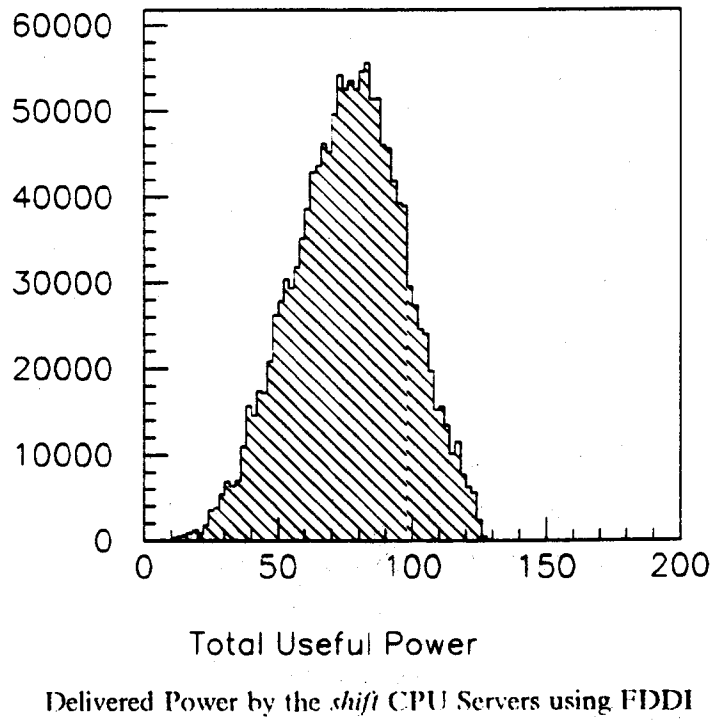

Figure 4: Delivered Power by the *shift* CPU Servers using FDDI

6. Proposed Configurations and Costs

As explained in section 4.6 on page 11 it is proposed to use UltraNet equipment as the backplane for the system.

It is also proposed to use the CERN Computer Centre Cray X-MP/48 as a tape server. Additional tape servers would be included, based on workstations and StorageTEK 4280 drives. The Cray would remain as the single tape server with cartridge robot access. Alternative methods of connecting to the tape robot, without using a mainframe, are under study, but we cannot at present estimate the probability of obtaining such a facility within the the next two years.

No specific costs are included for the file control server, as it will run on one of the other servers, or on the Cray.

There is a wide choice of systems which could be used as cpu and disk servers. The purpose of this part of our study has been to propose a *realistic* configuration for which we have been able to obtain measurements of some of the critical factors affecting performance, and which we believe would satisfy the goals of the project. The systems which have been considered include the H-P/Apollo DN 10.000, the DECsystems based on the MIPS R3.000 processor, Silicon Graphics 3x0S systems and IBM RS/6000 systems. While each of the manufacturers involved has provided us with pricing and performance data, it is clear that this has been done in the context of an informal study, and therefore comparative prices may vary in the event of a formal negotiation. Other suppliers will also have to be considered.

We are confident that the following proposal provides a good estimate of the overall costs of a system which could be constructed during the next twelve months.

The proposal consists of two parts:

a pilot implementation providing a total cpu capacity of over 50 CERN units and 40 GBytes of disk space;

a production configuration to provide 130 CERN units of cpu capacity and 100 GBytes of disk space.

Appendix H on page 38 contains comparative price information which has been used in developing the proposals, and Appendix I on page 39 contains benchmark results for cpu power, and disk performance and overhead costs for the systems considered.

6.1 The Pilot Project

A pilot project is proposed, to be initiated immediately and completed by the end of March 1991. The choice of components is rather pragmatic, and has been influenced by the interest shown by manufacturers, and the possibilities for obtaining equipment on loan. Nevertheless, the configuration is considered to be technically correct.

1. *BACKPLANE*

   a. UltraNet backplane supporting the Cray X-MP, a Cisco high-performance IP Router, and 7 workstations.

b. A Cisco AGS+ IP Router connected to the backplane and the CERN FDDI backbone network. []

2. *TAPE SERVERS*

a. The CERN Computer Centre Cray X/MP-48 acting as tape server, supporting IBM 3480 and successor tape drives and the IBM System Managed Cartridge Tape Facility (SMCF – tape robot). []

b. Additionally, two double StorageTEK 4280 drives, with SCSI interfaces, will be connected to two of the disk servers.

3. *CPU SERVERS*

a. DN 10.040 with 64 MBytes of memory and 2 GBytes of local disk space, providing a nominal 20 CERN units of capacity. []

b. An IBM RS/6000-530 with 32 MBytes memory and 1.2 GBytes of disk space, providing a nominal 6.2 CERN units of cpu capacity. []

c. A Silicon Graphics SGI 340S, with 96 MBytes of memory and 2.3 GBytes of disk space, providing a nominal 25 units of cpu capacity. []

4. *DISK SERVERS*

Four disk servers are proposed, each with 5 GBytes of disk space supplied by the manufacturer. In addition 5 GBytes of less expensive OEM SCSI disk would be attached to each server. This will provide invaluable experience with OEM equipment, which appears today to offer the best price/capacity.

a. 20 GBytes of OEM SCSI disk, including controllers, power supplies and cabinets.

b. One IBM RS/6000-530 with 16 MBytes memory and three SCSI channels. One IBM/RS6000-320 with 16 MBytes memory and two SCSI channels. Six 850 MByte disk drives (3 MB/s nominal transfer rate) and eight 670 MByte disk drives (1.8 MB/s nominal tranfer rate) will be installed, giving a total of 10.5 GBytes between the two systems. []

c. Two Digital DECsystem 5000-200 systems, each with 16 MBytes of memory and five 1.0 GBytes RZ57 disk drives (2.2 MB/s nominal transfer rate).

Table 1 on page 17 summarises the net costs of the pilot proposal.

6.2 Full Implementation Proposal

Subject to the success of the pilot project, we propose a full implementation providing 130 CERN units of cpu capacity and 100 GBytes of disk space. We provide here cost estimates for this system based on prices which are available today for equipment installable during the next six months. Due to the presence of equipment in the pilot proposal which is loaned to the project or shared with other projects, only the backplane equipment, the OEM disks and the STK 4280 tape units are assumed to be available for re-use. We assume that this second stage would be implemented progressively during

Table 1: Net Cost of the Pilot Implementation

Function	Component	Cost KCHF
Backplane	Ultranet	[]
	Cisco router	[]
Tape servers	STK 4280s	[]
cpu servers	Apollo DN 10.040	[]
	IBM RS/6000-530	[]
	SGI 340S	[]
disk servers	20GB OEM SCSI disk	[]
	IBM RS/6000-530, -320, 10.5 GB disk	[]
	2*DEC 5000-200, 10 GB disk	[]
Total		[]

Refer to the text for configuration details, and pricing assumptions.

1991, and so actual costs and equipment will be different from those which can be specified now. The purpose of the present section is therefore to provide information for use in budget planning for 1991.

The configuration proposed is as follows.

1. *BACKPLANE*

   Extension of the UltraNet configuration to support an additional 15 workstation interfaces.

2. *TAPE SERVERS*

   a. The CERN Computer Centre Cray X/MP-48 or its successor. As before, the costs are not accounted to this project.

   b. Two additional double StorageTEK 4280 drives, with SCSI interfaces, to be connected to disk servers.

   c. Four high capacity tape drives, a mixture of Exabyte and DAT units, connected to disk servers. For pricing purposes, the current cost of an Exabyte drive has been used.

3. *CPU SERVERS*

   a. DN 10.040 (new model operating at 36 MHz) with 128 MBytes of memory and 3 GBytes of local disk space, providing a nominal 40 CERN units of capacity.

   b. A Silicon Graphics SGI 380S, with 128 MBytes of memory and 3 GBytes of disk space, providing a nominal 50 units of cpu capacity.

   c. Six IBM RS/6000-530s, each with 32 MBytes memory and 1.2 GBytes of disk space, providing a nominal 37 CERN units of cpu capacity.

4. *DISK SERVERS*

   Eight servers are proposed, each with 5 GBytes of disk space supplied by the manufacturer. In addition a total of 60 GBytes of OEM SCSI disk would be attached to the servers, i.e. 40 Gbytes in addition to the OEM disks included in the pilot implementation.

   a. 40 GBytes of OEM SCSI disk, including controllers, power supplies and cabinets.

   b. Four IBM RS/6000-930s, each with 16 MBytes memory and three SCSI channels. Twelve 850 MByte disk drives (3 MB/s nominal transfer rate) and sixteen 670 MByte disk drives (1.8 MB/s nominal tranfer rate) will be installed, giving a total of 21 GBytes between the four systems.

   c. Four Digital DECsystem 5000-200 systems, each with 16 MBytes of memory, two SCSI channels and five 1.0 GBytes RZ57 disk drives (2.2 MB/s nominal transfer rate).

   d. *Magneto-Optical Disk Juke Box*

      A Hewlett-Packard magneto-optical disk system including 2 drives and capable of holding 32 disks to provide a total capacity of 20 GBytes is included in the cost estimates.

Table 2 on page 19 summarises the costs of the full proposal.

6.3 Manpower Requirements

The manpower requirements are clearly different during the two phases of the project: the development phase and the production phase. These phases are not distinct, but we estimate the manpower level required initially to develop the service, and the level required in a later "stable" production phase. We also estimate the specific costs of software development. It is assumed that the majority of the software development would be carried out by the staff operating the service, so these costs should not be added together.

6.3.1 Manning Levels

This section is not intended to define specific jobs, but rather to indicate the manpower required for the major functions. All figures refer to fractions of full-time staff.

Table 2: Cost Estimates of the Full Implementation

Function	Component	Cost KCHF
Backplane	Ultranet	[]
Tape servers	STK 4280s	[]
	4 EXABYTE/DAT units	[]
cpu servers	Apollo DN 10.040+	[]
	6*RS/6000-530	[]
	SGI 380S	[]
disk servers	40GB OEM SCSI disk	[]
	4*IBM RS/6000-930, 20 GB disk	[]
	4*DEC 5000-200, 20 GB disk	[]
	H-P juke box	[]
Total		[]

Refer to the text for configuration details, and pricing assumptions.

1. *PILOT PROJECT*

   a. NQS: 0.5
   b. File Control System and Fast File Access Routines: 1.0
   c. System management: 1.0
   d. UltraNet: 0.5
   e. User support: 1.0
   f. Operation: 0.5
   g. Network interface: 0.2
   h. Library and package support: 0.3

   This requires a total of 5 full-time equivalents. It is suggested that a core team of six people who spend the majority of their time on the project would be satisfactory, together with assistance from other specialists.

2. *PRODUCTION PHASE*

   a. Operation: 1.0
   b. System management: 1.0
   c. Resource management: 1.0
   d. User support: 1.0

There would be a continuing need for assistance from specialist groups.

6.3.2 Software Development Costs

a. Distributed NQS: 5 man-months.
b. File Control Service: 4 man-months.
c. Fast File Access Routines: 5 man-months.
d. Tape support: 2 man-months per tape server, not including the driver which it is assumed will be provided by the server manufacturer.
e. Management software: 6 man-months.
f. Simulation Program: 3 man-months.

The total cost of new software development is therefore estimated to be equivalent to 27 man-months.

Appendix A

Functional Specification of the File Control System

A.1 nfs_pathname

The format of the nfs_pathname, generated by the system, is as follows.

/disk_servers/node/pool_file_system/tape_type_location/vsn.vid.label_type.fseq

disk_servers is the pathname of the NFS mount point used on all cpu servers for access to the staging pool file systems on the disk servers.

node is the network node name of the disk server on which the file resides.

pool_file_system
defines the name of the file system on the disk server which belongs to the pool to which the file is assigned, and in which the file resides.

tape_type_location
is a name which identifies the type of tape used, and any necessary details concerning its location. For example: 3480.robot1, exabyte.jukeb3.

vsn.vid defines the magnetic and external labels of the tape, or first tape reel in the case of a multi-volume tape (input only).

label_type specifies the type of label processing required for the magnetic tape.

fseq defines the file sequence number on the tape. In the case of multi-file staging, a directory will be created with a name consisting of the *nfs_pathname* with *fseq* replaced by the characters MULT. The file itself will be contained within this directory identified by a unique name, and a file with this name prefixed by ".m" will contain the list of file sequence numbers from which the tape was created.

Example:

The tape staged from the cartridge tape robot, to the pool *gamay* with *vsn 15678X, standard labels, files 4, 7 and 11*, may be created with nfs_pathname:

/disk_servers/gamay/public/15678X..sl.MULT/563291074

The file sequence list would then be specified in the file with nfs_pathname:

/disk_servers/gamay/public/cart.robot1/15678X..sl.MULT/.m563291074

The account data for the owner of the file (the user issuing the stagein or stageout request which created the file) is contained in the directory entry for the file, together with the current user mask (default rw_rw_r__).

In the case of stageout, a special file, with the same pathname as the data file but with ".s" prefixed to the final element, is created. This is used by the *stageput* command to define details of the output staging required. This mechanism may also be used in future versions to stage files out automatically at the end of a job. The file is deleted when the stage out is complete.

Example: A file scheduled for staging to exabyte tape 03400876 with standard labels may be created with the nfs_pathname:

/disk_servers/pinot/public/exabyte/03400876..sl.1

The stage parameters would be stored in the file:

/disk_servers/pinot/public/exabyte/.s03400876..sl.1

A.2 File Block Structure

The *shift* system caters only for binary data files with undefined record format – i.e. equivalent to RECFM = U on the IBM service. Data structure within the block must be handled by the application.

The files are formatted on disk using the Unix convention of encapsulating the data block between two four-byte words, each containing the length of the data in bytes. Thus, from most Unix systems, these files may be read and written using FORTRAN unformatted I/O statements. The encapsulating length fields are generated on stagein and removed on stageout.

A.3 Stage Command Specifications

This section is largely extracted from the Cray service manual pages for stage and tpmnt, since in the initial implementation of *shift* the fcs will use the Cray for staging to and from tape. Other environments will also be supported, for which some modification will be required to the detailed parameters concerned with tape definition.

- *NAME*

  stagein, stageout, stageput, stagewrt, stagedel – stage file from tape to disk or from disk to tape.

- *SYNOPSIS*

  stagein −v vsn [−fun unit] [−pool pool] [−l {sl|nl|al|blp}] [−d {6250|1600}] [−g {CART|TAPE|SMCF}] [−q sequence] [−rm] [−F U] [−N nread] [−R] [−V vid] [−sh]

  stageout −v vsn [−fun unit] [−pool pool] [−l {sl|nl|al|blp}] [−d {6250|1600}] [−g {CART|TAPE|SMCF1}] [−A] [−rm] [−V vid] [−sh]

  stageput nfs_pathname | −fun unit

  stagewrt nfs_pathname [−v vsn] [−l {sl|nl|al|blp}] [−d {6250|1600}] [−g {CART|TAPE|SMCF1}] [−A] [−rm] [−V vid] [−sh]

  stagedel nfs_pathname

- *DESCRIPTION*

stage performs pre or post−staging, depending on the command name:

stagein implies pre−staging (tape to disk). Stagein checks if the file has already been staged to disk in the specified pool, waiting if the same file is currently being staged by another job. Otherwise, space is allocated within the specified pool, an nfs_pathname is assigned, and the requested tape file(s) is copied to disk. Stagein echos the nfs_pathname, and returns it in the environment variable NFS_PATHNAME.

stageout prepares for post−staging. Stageout allocates space within the specified pool, generates an nfs_pathname and creates an empty file with this name. The details of the tape file are preserved for use by a subsequent stageput. Stageout echos the nfs_pathname, and returns it in the environment variable NFS_PATHNAME.

Stageout must be issued before the step which creates the file.

stageput implies post−staging (disk to tape). Stageput schedules the copy to tape of the file specified by nfs_pathname, using the parameters given on a previous stageout. It must be used only after the step which creates the file.

stagewrt copies a disk file to tape. It is used when it is wished to change the tape parameters specified by a previous stageout.

stagedel removes the specified file from the disk.

The stage commands execute on the user's computer, but call the fcs server to perform most of the work. The fcs server locates the file or allocates it to a file system according to the specified pool, and then schedules the movement of data as necessary, using a procedure which executes on the computer to which the tape unit is connected. Stagein and stageout echo the nfs_pathname of the file, and return it in the NFS_PATHNAME environment variable. Optionally, they also *assign* the file to a FORTRAN logical unit.

- *PARAMETERS*

The space between option name and option value is **significant**.

−fun unit defines the FORTRAN unit number to which the staged file is to be assigned.

−pool pool specifies the staging pool to be used.

−d density specifies the density of the tape volume, which may be 1600 or 6250 bpi. This option is valid only for half-inch round tapes.

−F record-format
specifies the record format of the data on the tape. Only U is supported, meaning that each tape block is treated by the system as a single logical record.

−g device-type-location
requests a device from an installation defined set of "device groups". This may define, for example, that a cartridge tape in the Computer Centre tape robot should be used (SMCF).

−l label_type specifies the type of label which should be checked or generated for the tape. The following label types are supported.

al	ANSI label
blp	bypass label processing
nl	no label
sl	standard IBM label

−q sequence specifies the file sequence number of the tape file to be staged in. This parameter may be repeated, in which case files will be staged from the tape and concatenated to form single file on disk.

−A specifies on stageout that the disk file is to be appended to the end of the tape.

−rm specifies that the file should be removed from the disk at the end of the job.

−N nread reads only nread records (partial staging)

−R replaces the existing copy of the file by a fresh one.

−v vsn specifies the volume serial number of the tape. In the case of a multi-volume file this parameter must be repeated, in the order in which the tapes are to be used.

−V vid specifies the tape VID if different from the VSN. In the case of multi-volume staging when −v and −V are both used, each −v must be matched by a −V in the same order.

−sh specifies that the nfs_pathname should not be echoed.

Options may be given in any order, with the exception of multiple ocurrences of −q, −v and −V.

Default values are as follows.

−d 6250	half-inch tape density
−F U	tape record format
−g SMCF	device-type-location
−l sl	label type
−q 1	file sequence number 1

- *STAGING OF MULTIFILES*

stagein

Several files may be staged from one tape and concatenated to form a single file on disk.

−q seq specifies the file sequence number of the tape file.

This parameter may be repeated, and must be specified in the order in which the files are to appear in the disk copy.

Example:

stagein −v XX1234 −q 1 −q 2 −q 5

will stage the first two files and the fifth file from tape XX1234.

stageout

A file will be appended to the end of a tape on stage out if the −A option is specified on the stageout or stagewrt command.

- *ERROR RECOVERY*

In case of a hardware failure during the staging process, the staging operation is automatically retried once. If it fails any incomplete disk file is removed and the stage command terminates execution.

The command returns 0 if successful, otherwise a number specifying the error encountered.

A.4 Special Files .stageout and .stagedel

Stageout and stagedel maintain special files in the batch job working directory (TMPDIR). These contain respectively a list of files for which stageout commands have been issued and a list of staged files which should be deleted at the end of the job.

Appendix B

NQS Enhancements Specification

B.1 Load Balancing

A user wants to get the best turnaround possible: the batch dispatcher should submit the job to the least loaded machine which has enough resources (memory, tape units...) to run the job. However it is difficult to determine the best queue: jobs can execute for an undefined although limited period of time and can wait for memory or a free device... Furthermore if a batch server is unavailable for a certain period of time (hardware fault, development...), all jobs queued in it are blocked and invisible from other nodes.

B.1.1 Polling for Free Initiator

We propose a solution where all nodes send the requests to a central batch dispatcher; this in turn forwards the requests to the batch servers only when an initiator is free. The central NQS server would periodically inquire of the batch servers to find a free initiator which has enough resources (memory, tape units...) to run the job. All batch servers need to be registered in the central configuration database /etc/nmap.

B.1.1.1 Development needed

1. Enhance the queue status enquiry mechanism to get the load for a specific machine.

2. Write a networked version of the extended queue status enquiry to get the limits enforced on a remote machine. The current version finds the limits on the local machine only.

3. For better performance and portability between different versions of NQS (the display layout varies from one version to another), the information should be sent in tabular format rather than in display form (it's done like this on CRAY for USCP requests).

4. Modify the pipe client to submit the job according to the scheme described above.

Note 1: The same modifications would be needed on all versions of NQS existing in the *shift* system (central dispatcher + cpu servers).

Note 2: Such modifications would require 2 to 3 months.

B.1.2 Polling for Work

If the network load is an issue, the NQS protocol could be enhanced so that when an initiator is free, NQS sends a packet requesting work to the local servers. This is not possible with the public domain version of NQS but is seen as a useful enhancement.

B.2 Remote user Validation

The public domain version of NQS does not provide any password checking mechanism; it just checks if the requester is a valid user on the target machine: username and uid are valid and correspond to each other. To improve this situation, a call to a **chkpasswd** program will be forced at the beginning of every job. It is not possible to encrypt the password in such a case. but the method is easy to implement. The protocol could also be changed to send the password (encrypted if necessary) to the target machine and abort the submission in case of failure. This solution is not seen as necessary for the type of service we envisage at this stage.

Appendix C

Fast File Access Routines Reference

C.1 C Interface

Whenever an error occurs, the error code will be stored in the global integer variable **fferrno**.

C.1.1 Opening a File

#include <fio.h>

FFILE *ffopen (const char* file_spec, const char *amode);
The arguments for the **ffopen** function are as follows :

file_spec A character string containing the file name specification. See FFOPEN for more details.
amode An access mode indicator. One of the characters "r", "w".

On failure, **ffopen()** returns the NULL pointer. Failure may be caused by opening a non-existent file, lack of stage space, network communication failures... The file control block may be freed with the **ffclose** function.

C.1.2 Closing a File

#include <fio.h>

int *ffclose (FFILE* file_ptr);
The arguments for the **ffclose** function are :

file_ptr A pointer to the file to be closed.
On success, **ffclose** returns 0. On failure, it returns -1 and the error code is stored in ffermo.

C.1.3 Reading a File

#include <fio.h>

int *ffread (void * ptr, int i_size, int items, FFILE* file_ptr);
The arguments for the **ffread** function are as follows :

ptr A pointer to the memory location in which to place the information.
i_size The size each of the items to be read, in bytes.
items The number of items to be read.
file_ptr A pointer to the file to be read.

On failure, **ffread()** returns -1, the error code being stored in fferro. Otherwise it returns the number of items actually read.

C.1.4 Writing a File

#include <fio.h>

int *ffwrite (void * ptr, int i_size, int items, FFILE* file_ptr);
The arguments for the **ffwrite** function are as follows :

ptr	A pointer to the memory location in which the information is stored.
i_size	The size of each of the items to be written, in bytes.
items	The number of items to be written.
file_ptr	A pointer to the file to be written.

On failure, **ffwrite()** returns -1, the error code being stored in ff errno. Otherwise it returns the number of items actually written.

C.2 FORTRAN Interface

The FORTRAN interface actually uses the C routines previously described. All remarks applying to the C primitives also apply to the FORTRAN subroutines.

C.2.1 Assign

assign unit filename

The **assign** command assigns a FORTRAN unit number onto a remote file name. It is compatible with the local facility provided for this on the cpu server, i.e. when **assign** is called, it actually also calls its local equivalent. The **assign** command stores the appropriate information in environment variables, so that its validity does not survive multiple sessions but is effective for subsequent steps within the same job. The environment variable is named FFASNGxx where xx is the FORTRAN unit number

C.2.2 Opening a File

CALL FFOPEN(UNIT, FORMAT, ACCESS, RECL, STATUS)

The arguments for the **FFOPEN** subroutine are as follows :

UNIT	A FORTRAN unit number.
FORMAT	The formatting specifier. Only 'UNFORMATTED' is currently supported.
ACCESS	The access mode. One of 'SEQUENTIAL' or 'DIRECT'
RECL	The record length. Only valid for 'DIRECT' access.
STATUS	The error status if any, 0 otherwise.

The FFOPEN subroutine opens a file previously assigned with **assign** for subsequent Fast File I/O operations.

C.2.3 Reading a File

CALL FFREAD(UNIT, BUF, COUNT, REC, STATUS)

The arguments for the **FFREAD** subroutine are as follows :

UNIT	A FORTRAN unit number.
BUF	The data area.
COUNT	Integer variable. Number of bytes requested on entry, number of delivered bytes on exit.
REC	Integer variable. Relative record position for direct access files. Ignored otherwise.
STATUS	Integer variable. Zero on exit if success, otherwise non-zero.

C.2.4 Writing a File

CALL FFWRITE(unit, buf, count, status)

The arguments for the **FFWRITE** subroutine are as follows :

UNIT	A FORTRAN unit number.
BUF	The data area.
COUNT	Integer variable. Number of bytes to write.
STATUS	Integer variable. Zero on exit if success. Non-zero otherwise.

C.2.5 Closing a File

CALL FFCLOSE(UNIT, STATUS)

The arguments for the **FFCLOSE** subroutine are as follows :

UNIT	A FORTRAN unit number.
STATUS	The error status if any, 0 otherwise.

The **FFCLOSE** subroutine cleanly closes a file previously opened with **FFOPEN.**

C.2.6 Rewinding a File

CALL FFREWND(UNIT, STATUS)

The arguments for the **FFREWND** subroutine are as follows :

UNIT	A FORTRAN unit number.
STATUS	The error status if any, 0 otherwise.

The **FFREWND** subroutine rewinds a file opened by **FFOPEN.**

Appendix D

Estimates of Network Data Rates

To construct a balanced and scalable system, FOUR network I/O bandwidth parameters must be confirmed to have reasonable values:

- network total I/O rate (Peak & Sustained)
- per-interface I/O rate (Peak & Sustained)

The detailed estimates of these parameters obtained below depend on four basic sets of assumptions:

1. Assumed Job Mix, and the Data Consumption per Delivered cpu Unit.
2. Staging and Peaking Overheads of Data.
3. Configuration: Number of cpu, File and Tape I/O Interfaces.
4. CPU Overhead and Cost of Protocol Processing.

We consider these parameters in turn:

D.1 Assumed Job Mix.

The system will support four main job types: "P1" (Pass 1 Reconstruction), "P2" (Pass2 Analysis), "P3" (Pass3 Analysis), and "M1" (Monte Carlo Generation). The cpu and I/O parameters for these types can be summarized in the following way:

- Pass 1: 50 KB/sec I/O per CERN cpu unit per compute data pass.
- Pass 2: 8-50 KB/sec I/O per CERN cpu unit per compute data pass.
- Pass 3: 2-10 KB/sec I/O per CERN cpu unit per compute data pass.
- MCarlo: < < 1 KB/sec I/O per CERN cpu unit per compute data pass.

A representative data consumption value of 20 KB/sec. per CERN cpu unit for the whole job mix will be assumed.

D.2 Assumed Data Staging and Peak Overheads

It is assumed that all data is staged from tape to a File Server disk (i.e. that cpu Servers have no large local disks), and that a quantity of output data equal to half of the input data is generated and also staged back to tape. This is very conservative, but can only be wrong (high) by a factor of two.

It is also assumed that to prevent congestion developing, the required peak I/O data rates are 3 times the required sustained rates (both overall and per-interface). Simulation studies performed so far with a homogeneous mix of "20KB/sec. jobs" indicate a peaking factor of at least 2; with our factor of 3 we wish to allow for a spread of job types around the 20KB/sec. nominal value.

Thus, per installed cpu capacity of 100 *effective* CERN Units, and using 20KB/sec/Unit for average I/O *consumption*, we arrive at a *sustained* network bandwidth requirement of

2+2+1+1 = 6 MBytes/sec

(accounting for one full stage-in and half a stage-out of all consumed data), and a *peak* overall rate of 18 MBytes/sec (using the peaking factor of 3 above).

D.3 Configuration Considerations

Let C, F and T be respectively the number of cpu Server, File Server and Tape Controller I/O interfaces in a network configuration supporting 100 CERN cpu Units, consuming data at 2 MBytes/second and producing data at 1 MByte/second (sustained rates). Let I be the maximum I/O rate that a network interface can sustain (in MBytes/second). Then:

- Total cpu data rate = CxI = 3 (sustained) = 9 (peak).
- Total File data rate = FxI = 6 (sustained) = 18(peak).
- Total Tape data rate = TxI = 3 (sustained) = 9 (peak).

Note that TWO network interfaces participate in each I/O transfer and we have incorporated both cpu data consumption and output between C and F, and a stage-in and half stage-out between F and T.

Thus: C = T = 9/I; F = 18/I.

- For Ethernet, the value of I would be about 0.3 and we obtain absurd values for C, F and T.
- For FDDI, the value of I would be approximately 1 and we obtain:
  C = T = 9; F = 18.
- For UltraNet, the value of I would be at least 3. yielding worst case values of:
  C = T = 3; F = 6.

We see that to keep C, F and T within reasonable limits (in particular F and T), the nodes must have network interfaces capable of at least 3 MBytes/sec peak to be realistically a scalable and compact system. Only UltraNet achieves this.

D.4 CPU Protocol Overhead and Costs

Taking figures of Table 3 on page 34 as a basis for LAN (Ethernet/FDDI) cpu overheads for protocol processing, we see values lying between 3% and 12% of delivered cpu capacity (depending on the host implementation) for *each end* and *each pass* through the data. The variations are a function of host *and peer host* implementations, which encourage us to be conservative in estimating this quantity. **a value between 5% and 10% is representative.**

After tests of UltraNet attachments on Sun and Cray equipment (Appendix F on page 35) we can estimate that **cpu protocol overheads are reduced by a factor of from 5 to 20 when using UltraNet.** We believe that similar reductions will be obtained with other workstations.

Counting *both* ends of *each data pass*, we multiply the above values by 6, using the same argument as in section D.2 on page 31 above. This explicitly charges all cpu overhead to the cpu's. (In fact the Tape Controller may be a Cray with an even higher cpu price).

We arrive at a substantial figure of the order of **50% cpu overhead per unit of data processing** for Ethernet or (today's) FDDI LAN attachments.

Using UltraNet we obtain an estimated *cpu* overhead of well below 20% in the worst case, and more likely a value between 5% and 10%.

Having estimated the cpu overheads, we now estimate the costs for the nominal 100 cpu Unit system: a 50% cpu overhead amounts to 50 CERN Units, for which we estimate a cost of CHF 450K. This would be the value without UltraNet.

With UltraNet we would make a saving of around CHF 375K, per installed 100 CERN Units, minus the cost of the corresponding UltraNet equipment.

D.5 Conclusion

Both I/O bandwidth requirements and its cpu protocol costs force us to choose a network backplane solution like UltraNet (which addresses both issues at once). Ethernet is clearly too slow to support more than a few CERN cpu Units. If FDDI were to be used, then we would be forced to use multiple segments and the segment switching would anyway have to use faster-than-FDDI technology to avoid becoming a bottleneck. We would also be forced to use only those components with very efficient TCP/IP implementations, to keep us down to acceptable cpu overhead costs.

Finally, we can observe that UltraNet also makes sense financially. We have seen that each installed 100 Units of cpu could incur a CHF 450K overhead without UltraNet, and this in fact represents a major portion of a suitable UltraNet configuration. We conclude that with UltraNet, the proposal has every chance of success, while without such a solution it will never be scalable to an interesting size, at least with no changes to its architectural design.

Appendix E

Estimates of CPU Protocol Processing Overheads

This table shows measurements of cpu consumption for memory-memory transfers using C standard I/O with TCP sockets. The results varied over read and write directions, and also depended on the peer system. All results used Ethernet. The second column gives the cpu power of the system under test, using the figures from the benchmarks of Eric McIntosh. These are provided to enable comparisons to be made between different systems, but note the benchmark takes into account the floating point performance of the system, which is not particularly relevant for protocol processing. CU stands for a CERN UNIT of cpu. The third column gives cpu consumption of the test, normalised to native cpu seconds per MByte of data transferred. The fourth column expresses this in CERN-unit-seconds per MByte transferred. The final column gives the percentage of the cpu of the system under test which would be required to feed it with data via Ethernet and TCP if it were executing a job which had a data consumption rate of 20 KBytes/sec for each CERN unit of capacity.

Table 3: CPU Protocol Processing Overheads

System Type	cpu Power CU	consumed cpu sec/MB	consumed CU-sec/MB	%consumed (20KB/s/CU)
SGI 4D/25G	3.5	0.7	2.4	5%
IBM RS/6000-530	6.0	0.7	4.2	8%
DECstn 5000-200	4.2	0.4	1.7	3%
Apollo DN10K	5.0	1.2	6.0	12%

Appendix F

Results of Ultra/FDDI/Ethernet Tests

Measurements of UltraNet, FDDI and Ethernet bandwith and cpu costs were made at RUS, Stuttgart on May 21-22 1990. The test program performs memory-memory transfers using C standard I/O with TCP sockets. The results are summarised in the following tables. The figure in brackets in the first column is the size of the record presented at the socket interface. Values greater than 20KBytes do not lead to improved performance in the case of Ethernet and FDDI, because the system divides the record into datagrams which are relatively small. In the case of UltraNet all of the transport level processing takes place on the interface hardware, and so larger record sizes lead to lower workstation processing costs and higher overall throughput. The cpu consumption figures in columns 4 and 5 are normalised to native cpu seconds consumed for each Megabyte of data transferred.

Table 4: Sun 3/110 and Cray 2

Medium	Direction	Kilobytes/sec	Sun cpu-sec/ Megabyte	Cray cpu-sec/ Megabyte
Ethernet (20KB)	Cray->Sun	220	0.95	0.05
	Sun->Cray	250	2.40	0.39
Ultra (20KB)	Cray->Sun	1750	0.28	0.02
	Sun->Cray	1850	0.36	0.02
Ultra (200KB)	Cray->Sun	3300	0.14	0.002
	Sun->Cray	3350	0.14	0.002

Table 5: Sun 3/260 and Cray 2

Medium	Direction	Kilobytes/sec	Sun cpu-sec/ Megabyte	Cray cpu-sec/ Megabyte
Ethernet (20KB)	Cray->Sun	305	0.85	0.07
	Sun->Cray	290	1.70	0.33
FDDI (20KB)	Cray->Sun	530	0.50	0.05
	Sun->Cray	435	0.95	0.12

Appendix G

Simulation Program

In this appendix we describe how the simulation program works. The first section describes the input data required for the simulation. These data are supplied to the program in the form of a configuration file, and are thus readily modified when making several simulation runs. The second section then describes the operational assumptions that are made, and which affect how the simulation handles the job load, the staging of tapes, and so on.

G.1 Configuration Data

1. The configuration of machines:

   a. The power of each machine in CERN Units.
   b. The "network board speed" i.e. the maximum speed at which the machine can do I/O to the network. For the High Speed Router this is specified in terms of number of packets per second and packet size.
   c. The type of machine i.e. "epu" server, File Catalogue Server, Disk Server, or High Speed Router.
   d. For Disk Servers, the available disk space in MBytes.
   e. For Disk Servers, the maximum single stream I/O rate, and the maximum *aggregate* rate.
   f. For cpu servers, the number of processors, the maximum number of concurrent jobs that may execute, and the I/O cost factor, in CERN Units per MBytes/s.

2. The maximum speed of the backplane in MBytes/s.

3. The type of batch job being submitted to the Facility. This is taken from real benchmark results:

   a. The power of the machine the benchmark ran on, in CERN Units.
   b. The number of native seconds the benchmark ran.
   c. The cpu time (native) to process one event.
   d. The size of each input event in MBytes.
   e. The amount of data written for each input event in MBytes.
   f. The total amount of data written, in MBytes.
   g. The I/O speed achieved by the machine for the job.
   h. The size of the job (i.e. the executable image size).
   i. The amount of summary data (histograms etc.) written by the job at job end.

4. The rate of submission of such jobs by "the users", per second. (The jobs are assumed to arrive via the High Speed Router.) This is chosen so that there is always a job available to fill an empty "job-slot".

5. A factor which determines what fraction of jobs will find the data they require already staged to disk and available. (Set to be one in three jobs.)

6. The period in real time over which the simulation should run.

G.2 Operational Assumptions

1. The job which has queued the longest is assigned to a server which has a spare job slot. This is done so as to spread jobs over all servers evenly.

2. The executable job image is not resident on the assigned server, and must be sent together with the job request. During this time, the job is said to be in state "INIT".

3. The input data for the job must (in most cases) be staged from the File Catalogue Server to a Disk Server. The File Catalogue Server has its "network board" speed thus set to the staging speed. During this time, the job is said to be in state "STAGEW".

4. There is no delay between the issue of a stage request and the start of staging (i.e. it is assumed that the "operator" mounts the required tape immediately).

5. Once the input data are staged, the job enters a cycle of operations READ-CPU-WRITE. It repeats these operations until the requisite data have been written to the relevant Disk Server. It then enters an END phase, where summary data are assumed written to the Disk Server.

6. The benchmark figures determine the relative times the cpu spends in each part of the cpu-READ-WRITE cycle.

7. It is assumed that the I/O scales linearly with the power of the machine compared with that of the benchmark machine.

8. The virtual clock "ticks" are of a duration much smaller than the time needed for the job to be processed.

9. When the File Catalogue Server or a Disk Server are dealing with more than one request, then their "network board speeds" are decreased proportionally.

10. For transfer of data between two machines, the instantaneous data rate is determined by taking the minimum value of the two "network board speeds" and the backplane speed.

11. For a given cpu server, the available power for a running job in state "cpu" is calculated taking into account the IO cost for other jobs on the server in I/O.

The simulation is written in FORTRAN-77, and has been ported and executed on a variety of platforms. The code is rather general, and may thus be of interest for other applications.

Appendix H

1990 Prices for Hardware under Consideration

This appendix has been omitted from this copy, since it contains confidential price information.

Appendix I

CPU and Disk Performance Data

I.1 CERN CPU Benchmarks

Table 3: CERN Standard CPU Benchmarks

Processor	CERN Units
H-P Apollo DN 10.010 (18MHz)	5.0
SGI 4D/25G	3.5
SGI 4D/320S	6.3
IBM RS/6000-530	6.0
DECstation 5000-200	4.2

The CERN Unit is the performance of the benchmarks on the target system, relative to the performance on a VAX 8600. The benchmark comprises four large FORTRAN programs, and the performance is taken as the geometric mean of these programs.

I.2 Disk Performance

The table gives data rates and cpu consumption as measured for a trivial C program which writes a file and re-reads it. The Unix kernel uses internal data buffering and in some cases mapping of files into memory. In all cases the size of file used was large compared with the buffer or memory available, but the figures are only estimates of the true disk performance. Some measurements have been made of simultaneous I/O to disks sharing the same "channel" and on separate channels. These results are not given here as they are not available for all systems of interest. The cpu consumption figure is normalised to native seconds required to transfer 1 Mbyte of data.

Table 4: Disk Performance

Processor	Disk Model	Write KB/sec	Read KB/sec	cpu consumption sec/MByte
H-P Apollo DN 10.010 (18MHz)	ESDI	750	1,100	0.09-0.1
SGI 4D/25G	____	520	1,200	0.2-0.26
SGI 4D/320S	____	1,200	1,400	____
IBM RS/6000-530	670 MB	1,200	1,400	0.08-0.1
DECstation 5000-200	RZ56	380	1,300	0.16-0.2

Imprimé au CERN. Prix : 2.20 Fr.