LHCb note 2001-088, Computing

First Implementation of a Diskless Computing Farm for the LHCb Italian Tier-1 Prototype

G. Avoni[1], M. Bargiotti[2], A. Bertin[2], M. Bruschi[1], M. Capponi[2], S. De Castro[2], L. Fabbri[2], P. Faccioli[2], D. Galli[2], B. Giacobbe[2], U. Marconi[1], I. Massa[2], M. Piccinini[1], M. Poli[3], N. Semprini Cesari[2], R. Spighi[1], V. Vagnoni[2], S. Vecchi[1], M. Villa[1], A. Vitale[2], A. Zoccoli[2]

[1] *Istituto Nazionale di Fisica Nucleare, Sezione di Bologna*

[2] *Dipartimento di Fisica dell'Università di Bologna and Ist. Naz. di Fisica Nucleare, Sezione di Bologna*

[3] *Dipartimento di Energetica dell'Università di Firenze and Ist. Naz. di Fisica Nucleare, Sezione di Bologna*

A. Frenkel, R. Santacesaria

Istituto Nazionale di Fisica Nucleare, Sezione di Roma

M. Paganoni

Dipartimento di Fisica dell'Università di Milano and Ist. Naz. di Fisica Nucleare, Sezione di Milano

2nd June 2001

Abstract

Aim of this paper is to outline the designing criteria which guided us in the design and the realization of a computing farm for the LHCb Italian Tier-1 prototype. The hardware architecture, with particular attention paid to the disk data storage and to the disk-less implementation of the farm nodes, is described. The disk-less network boot procedure is also described in detail. Other important issues, concerning system debugging, hardware health monitoring and remote power control, are also discussed.

1 Introduction

LHC experiments at CERN have a very large and challenging computing need, involving hundreds of thousands of SPECint95 of CPU requirements and hundreds of TeraBytes of disk storage. The computing infrastructure has to be designed in order to make use of geographically distributed and hierarchically organized computing resources. It needs to be optimized for the task it must accomplish and requires an accurate plan of investments.

The need of computing power, in off-line systems, is concentrated in Monte Carlo simulation and analysis activities. Here we will focus on Monte Carlo simulation, which is an excellent task for loosely coupled computing systems.

As a matter of fact, Monte Carlo jobs consist of a large number of independent computing units (the events) which are executed sequentially and are uncorrelated each other. Although both fine-grain and coarse-grain parallelism could be performed on the single event generation, however the most efficient parallelization of Monte Carlo code consists in running different jobs on different machines.

A group of PCs, connected together via network, which share the disk space on which all the output are written, is therefore a perfect platform for Monte Carlo event production.

2 Guidelines for the Choice of the Components

We decided to make use of commodity components as far as possible for several reasons:

- *Prices.* The cost of computer components nowadays does not depend on the technology content but only on the market volume. Therefore, to optimize price over performance ratio, it is necessary to choose, as far as possible, commodity components, which now have high enough performance to be used for particle physics needs.
- *Lifetime.* The mean-life of computers is currently estimated to be about three years. Keeping systems for a longer time is not convenient: it is cheaper to replace components than to repair them, but due to the fast changes in hardware components available on the market it is generally very difficult to find the same components after one year or even six months. This possibility is as more possible as the spread on the market is low.
- *Software support.* Wide spread components are better supported by OS (Operating System) distributions and usually have better debugged drivers.

As an example, for these reasons we preferred commodity motherboards and commodity power supplies, with respect to custom motherboard and custom power supplies expressly designed for slim rack-mount boxes. In this way (rather expensive) rack-mount boxes and power supplies can be kept and reused following a motherboard replacement.

3 The Hardware Setup

The hardware setup should be as simple as possible, to make the system management easier, to minimize the number of parts potentially subject to failures and to increase reliability. The most straightforward computer farm is built of standard PC boxes (desktop, tower, mini-tower, etc.), each one having its own hard disk, its own floppy disk drive, its own CD-ROM drive and a network interface. This kind of system, however, has the following limits:

- it is rather space-consuming, because boxes are commonly sized to host many peripherals;
- the system installation and administration tasks are multiplied by the number of nodes;
- data retrieval is difficult because the disk storage is fragmented on the LAN (Local Area Network);
- the reliability of data storage is quite low, unless daily backups of large amount of data are performed across the LAN.

These limits can be overcome by specializing some components and packing the others.

The first specialization can be performed on disk data storage, which can be concentrated on one or more nodes. In this way data retrieval becomes simpler and disk failures can be reduced by means of RAID (Redundant Arrays of Inexpensive Disks) architecture. Such node, which is a computer with a high-bandwidth network connection and a very large disk storage capabilities, is known as **NAS** (Network Attached Storage).

A second specialization can be performed by moving the OS file-systems to the NAS. The Linux OS foresees this capability in a very natural way, by mounting the root file-system over NFS (Network File System). In this way the system installation and administration becomes simpler, because all the root directories of the farm nodes are concentrated on the NAS, hence being accessible at the same time under one unique file system. Moreover some directories containing packages that need to be common to all the farm nodes, can be exported to all the nodes without replications. In this way the system administrator needs to manage only one package installation.

The boot process, however, must be modified in order to be performed across the LAN. The farm nodes must have a Boot-PROM in order to be able to send a boot request through the network and load the files required by the boot procedure.

A further specialization can be performed by organizing the farm nodes into login nodes and job processing nodes. Login nodes are PCs to which users can connect to submit and monitor jobs, and to access data. The batch queue master processes, which start the job execution on the job processing nodes, run on login nodes. GRID authentication, control and monitoring software runs on login nodes as well. Job processing nodes are not accessible by the users. They run only jobs spawned by the batch queue system, one per CPU at a time.

Job processing nodes need no local disk at all (can be completely disk-less), because their memory consumption is predictable (they only run the OS kernel, few system daemons and one job per CPU at a time), while login nodes need a disk swap area because users may want to open many interactive processes at the same time, which consume memory while usually few of them are in running state at the same time.

Job processing nodes can be "naked" PC motherboards with a NIC (Network Interface Card), a VGA (Visual Graphics Adapter) card for on-site debugging purpose and initial BIOS (Basic Input/Output System) configuration, void of any other peripheral. These motherboards can be easily packed in a rack to save floor-space.

A schematic representation of the overall farm architecture is shown in Fig. 1, while Fig. 2 shows a photograph of the current installation of the farm prototype.

3.1 The Disk Storage

A NAS is basically a computer equipped with a large disk array and connected to the LAN, which acts as a network disk server, e.g. a NFS (Network File System) server for a UNIX environment or a CIFS/SMB (Common Internet File System / Server Message Block) server for a Microsoft Windows environment.

Limitations to the NAS performance can arise from the network connection, CPU performance, bus speed, disk interface speed and disk performance.

3.1.1 The Disk Interface

Several interfaces are used to connect hard disks to PCs. Among them are SCSI (Small Computer System Interface), EIDE-ATA (Extended Integrated Drive Electronics - AT Attachment), FC-AL (Fiber Channel - Arbitrated Loop), Fire Wire (IEEE 1394), USB (Universal Serial Bus).

The most widely used interface for large Disk Arrays is **SCSI** [1]. The fastest parallel SCSI bus currently available is ULTRA-160, which is also known as ULTRA-3. It uses a 16 bits wide data path and LVD (Low Voltage Differential) signaling to reach the maximum synchronous bandwidth to 160 MB/s. Asynchronous transfers rate is limited at 5 MB/s. SCSI commands are sent at the asynchronous rate. SCSI data is transferred either at the asynchronous rate (worst case) or at a negotiated synchronous rate (best case)[1]. The maximum number of devices connected to a ULTRA-160 SCSI chain is 15. Speeds of 320 MB/s and 640 MB/s using the same or similar LVD signaling over a 16 bits wide data path may soon be standardized.

The most widely used interface on commodity PCs is **EIDE-ATA** which is very cheap. The fastest EIDE interface currently available is ULTRA-ATA/100, which supports burst rates of 100

[1] In real life applications, SCSI ULTRA-160 based disk arrays can currently provide a sustained throughput of about 60-80 MB/sec.

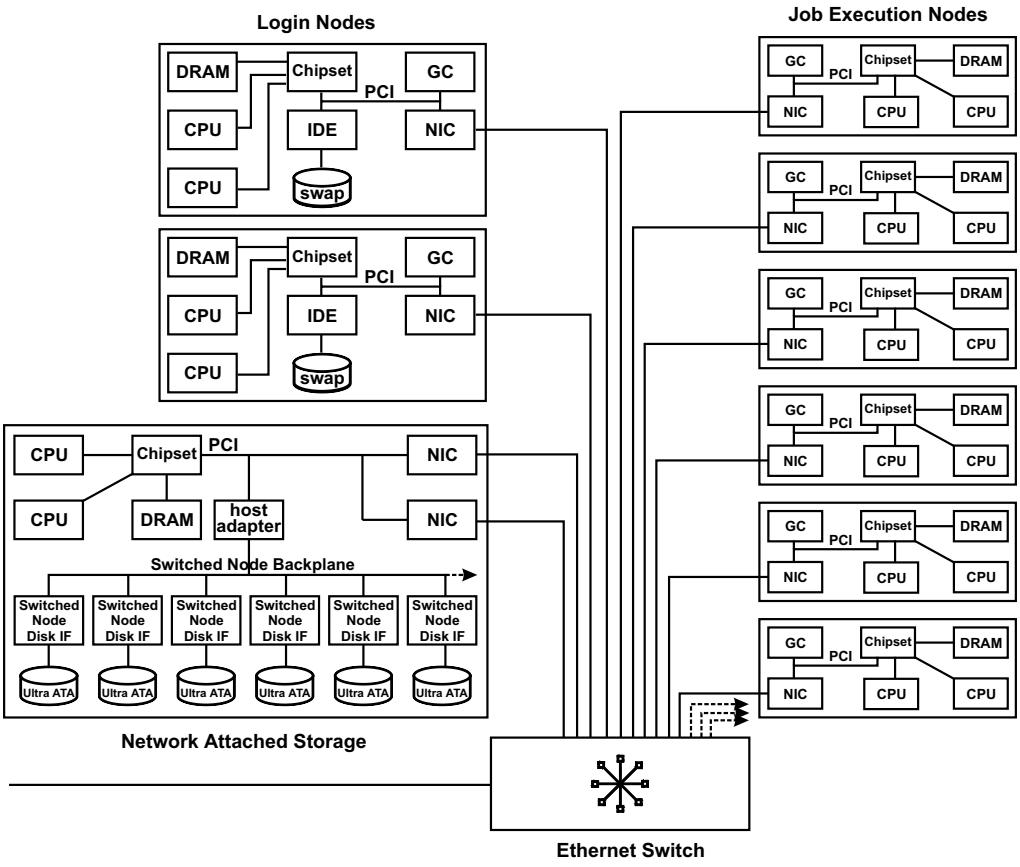

Figure 1: Schematic representation of the farm architecture.

Figure 2: Front view of the farm prototype.

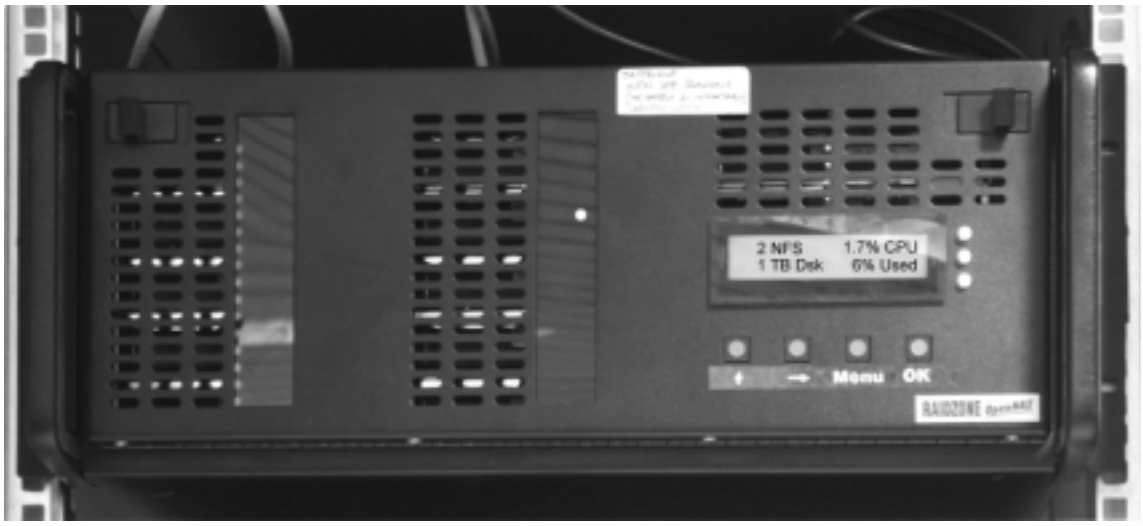

Figure 3: The Network Attacked Storage RaidZone RS15-R1200.

MB/s and up to 2 devices per bus. One of the devices is defined as master, and has a higher priority than the other defined as slave. Arbitration is very simple: when the master device is accessed, the slave device is inaccessible.

NAS units can be equipped with either SCSI or EIDE disks. Performance are nowadays comparable, while prices are lower for EIDE-NAS units.

SCSI-NAS units, based on ULTRA-160, have a PCI-SCSI interface to which the disks are connected through a SCSI chain. Data transfer speed is limited by disk performance until the maximum SCSI chain throughput is reached.

To achieve the needed performance, EIDE-NAS units must connect only one disk to the ATA interface, and need additional hardware (a switched backplane) to connect the ATA interfaces to the PCI bus. In this way, the operating system "sees" every disk connected directly to the PCI bus, and can perform Direct Memory Access transfer directly from/to the main memory at full speed without the intervention by the CPU. DMA transfer is controlled by the disk interface chips on the backplane (one per drive).

The PCI bus which equips the standard PCs has a burst rate of 132 MB/s, and a sustained throughput of about 100 MB/s. Hence the sustained rate increases almost linearly with the number of disks (typically 20 MB/s for each currently available EIDE disk), up to the saturation point of the PCI bus (about 100 MB/s).

High performance Ultra ATA/100 drives are used in tens of millions of PCs, workstations and servers every year. Because of high production volumes and intense competition among manufacturers, fast and high capacity Ultra ATA/100 drives cost much less than SCSI drives of similar size and performance features. When Ultra ATA/100 drives are used in a high performance disk array, the price of the disk array can be similarly reduced.

We have chosen therefore a NAS, namely a RaidZone RS15-R1200 (see Fig. 3), equipped with a disk array of fifteen 80 GB EIDE hard disks, for a total of 1200 GB in JBOD (Just a Bunch of Disks) configuration. We have set the NAS in order to have one hot-spare disk and the remaining 14 configured in RAID-5 (see next subsection), for a total of 1040 GB available.

3.1.2 The RAID Configuration

The term RAID originates from a paper by Patterson, Gibson and Katz of the University of California at Berkeley, published in 1988 [2].

The basic idea of RAID (Redundant Arrays of Inexpensive Disks) was to combine multiple small, inexpensive disk drives into an array of disk drives working in parallel, hence yielding

performance exceeding those of a SLED (Single Large Expensive Drive). Additionally, the array of drives appears to the OS as a single logical storage unit.

The MTBF (Mean Time Between Failure) of the array is equal to the MTBF of an individual drive, divided by the number of drives in the array. Thus the MTBF of an array of drives would be lower than that of a single drive, and consequently the risk of loosing data would be higher. However, disk arrays can be made fault-tolerant by redundantly storing information in various ways.

Five (six, including the not truly redundant RAID-0) types of array architectures, RAID-1 through RAID-5, were defined by the above mentioned Berkeley paper, each providing disk fault-tolerance and each offering different trade-offs in features and performance.

- RAID-0 is not redundant at all, hence does not truly fit the "RAID" acronym. In level 0, data is split across drives, maximising parallel disk access, resulting in higher data throughput. Since there is no overhead due to the storage of redundant information, performance is very good, but the failure of any disk in the array results in complete data loss. This level is commonly referred to as "**striping**".

- RAID-1 provides complete redundancy by writing identical copies of all the data to two different drives. The performance of a level 1 array tends to be faster on reads and slower on writes compared to a single drive, but if either drive fails, no data is lost. This is a good entry-level redundant system, since only two drives are required; however, since one drive is used to store a duplicate of the data, the cost per MegaByte is high. This level is commonly referred to as "**mirroring**".

- RAID-2 performs data striping with a block size of one bit, so that all disks in the array must be read to perform any read operation. This implies that the disk spindles must be synchronised. A RAID-2 system would normally have as many data disks as the word size of the computer, typically 32. In addition, it requires the use of extra disks to store a Hamming error-correcting code for redundancy. With 32 data disks, a RAID-2 would require 7 additional disks for a Hamming code ECC. RAID-2 is intended to be used with drives which do not have built-in error detection. For a number of reasons, including the fact that modern disk drives contains their own ECC, RAID-2 is not a practical disk array scheme.

- RAID-3 stripes data at a byte level across several drives, with parity stored on one drive. RAID-3 is very similar to RAID-2, except that the Hamming-code ECC is replaced by a single parity disk. The parity information allows recovery from the failure of any single drive. RAID-3 can be used in data intensive or single-user environments which access long sequential records to speed up data transfer. However, RAID-3 does not allow multiple I/O operations to be overlapped and requires synchronized-spindle drives in order to avoid performance degradation with short records. Because only one drive in the array stores redundant data, the cost per MegaByte of a level 3 array can be fairly low.

- RAID-4 is very similar to RAID-3, except that the stripe width is greater than the disk sector size. This means that individual read requests can be filled from a single disk. This will give improved performance on multi-tasking or multi-user operating systems, or in any application where read requests are not completely sequential. The performance of a level 4 array is very good for reads (the same as level 0). Writes, however, require parity data to be updated each time on the same disk, so that the parity disk becomes a real bottleneck. This slows down small random writes, in particular, but large or sequential writes are fairly fast. The cost per MegaByte is the same as for level 3.

- RAID-5 is similar to level 4, but distributes parity among the drives (see fig. 4) instead of storing them in only one drive. This can speed up small writes in multiprocessing systems, since the parity disk does not become a bottleneck as in RAID-4. Anyway writes incur a performance penalty on RAID-5: to update the parity block requires two extra reads (of

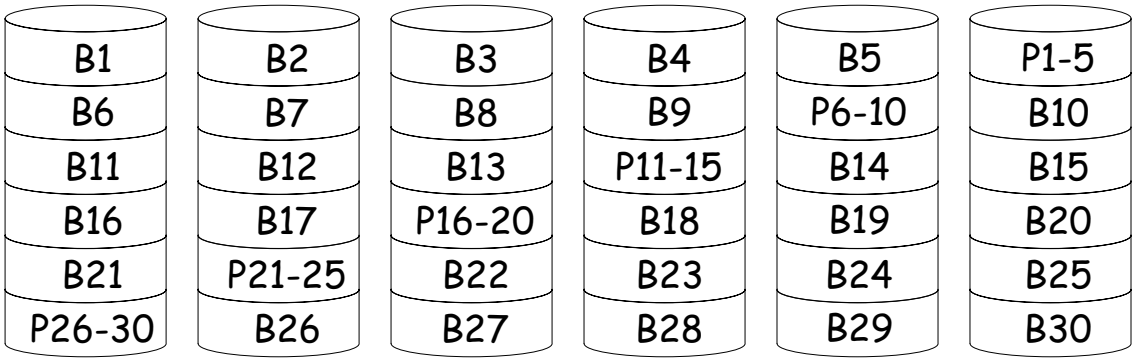

Figure 4: Block Layout of a 6 disks array configured as RAID-5. Bn are data blocks, while Pn-m is the parity block corresponding to data blocks Bn, B(n+1), ..., Bm.

> the old data block and the old parity block) and one extra write (of the parity block). This penalty reduces to one extra read for sequential writes, because the parity block will already be in the cache. Because parity data must be skipped on each drive during reads, the performance for reads tends to be lower than a level 4 array. The cost per MegaByte is the same as for level 4 and level 3.

Current disk arrays usually can be configured as JBOD (Just a Bunch of Disks, no striping, no redundancy), RAID-0, RAID-1 or RAID-5, or even, for particular performance requirements, as a combination of level 0 with levels 1 and 5, commonly referred to as RAID-10 and RAID-50. These two levels can reach higher performance by providing striping across fault tolerant disk arrays, hence combining the features of RAID-0 and RAID-1/RAID-5.

We have chosen a RAID-5 configuration for the Tier-1 disk data storage, which provides good fault tolerance, enough read and write performance, and whose cost is quite low (the space required for the parity storage is almost equal to the size of one disk of the array).

3.1.3 The NAS Operating System

The NAS OS can be proprietary or open source, and can be written on disk or in a Flash PROM (Embedded OS).

Open source OSs are cheaper and sometimes (e.g. Linux) more reliable than proprietary ones. Moreover, OS update and upgrade are easier, timely, and cost-less for open source OSs. Finally on open source OSs is easier to add additional services and server daemons (e.g. Bootp, DHCP, PXE, etc.).

An OS embedded in a PROM should be more reliable than those stored on disks. However update and upgrade are more difficult and sometimes not allowed to the customer.

For these reasons we preferred a disk stored OS. The RaidZone RS15-R1200 stores the OS in slivers on the first 5 drives. Slivers from the first two drives are configured in mirror (RAID-1) and contain the current version of the OS; the third drive contains a 2 day old version of the OS, the fourth drive contains a 2 week old version of the OS and the fifth drive contains the original configuration. In the case of corruption of the OS file-system, the NAS can be booted from any of these drives. This feature, referred to as **EFT** (Extreme Fault Tolerance) bootstrap, protects not only against disk failure, but also against corruption of the root file system, corruption of one or more critical operating system files, or even human error in operating system configuration (e.g. the accidental deletion of files).

3.1.4 Fault Tolerance

To increase fault tolerance, a NAS usually duplicates critical components. For example the Raidzone RS15-R1200 has two power supplies, two Pentium III processors and two 100Base-T network

interfaces which can work in port trunking (bound together with the same IP address).

Another critical aspect of NAS is its behavior in case of unexpected power failures. A "dirty shutdown" (i.e. a shutdown not performed using operating system "shutdown" command) can damage both RAID-5 parity integrity and file system integrity.

After a dirty shutdown usually parity information is left in an unknown state. When the system is rebooted all the RAID-5 parity information must be reconstructed from scratch. Even though this process is carried out in the background, and the server can perform all its normal functions, this can have significant impact on system performance. On a large disk array, it takes many hours to rebuild the RAID-5 parity. The Raidzone RS15-R1200 addresses this problem by using $\mathbf{R}^3$ (Redundancy with Rapid Recovery) technique, which is a sort of journaling RAID-5. A journal is kept in an EEPROM of the location of each write operation. Periodically, at certain checkpoints, it can be guaranteed that the RAID-5 parity has been written up to a certain point in the journal which is then truncated to that point. After a dirty shutdown the RAID-5 module only need to rebuild the parity of the disk sector contained in its journal (the last few seconds of disk activity before shutdown). In general, the parity rebuild operation will complete, in this way, within a minute.

Concerning file system integrity after a dirty shutdown, the Raidzone RS15-R1200 allows to use, in place of the standard Linux file system (ext2), the **ReiserFS** [6], which is a journaling file system based on balanced tree algorithms (with very good performance). A journaling filesystem borrows ideas from realtime transaction processing systems and uses special dedicated logging disk areas where all filesystem changes are stored in realtime. In the event of a disk crash, the logs can then be used to complete the failed transaction and avoid the fsck (file system check), which is very time-consuming on large disks.

The Raidzone RS15-R1200 also allows to configure one disk as **auto-hot-spare**. This disk is usually excluded by the array, but in case of failure of another disk, the failed disk is automatically replaced by the hot-spare, and the data lost on the failed disk is rebuilt on the hot-spare without manual intervention. In this way, the period during which the disk array is not fault tolerant is minimised.

3.2 Disk-less and Swap-less Farm Nodes

A common PC configuration usually requires at least three types of mass storage: I/O data storage, OS storage and a Swap Area for paging and swapping operations.

Concentrating I/O data storage on one or few nodes (the NAS units) is required by data retrieval needs (data spread on many disks is more difficult to be accessed) and reliability (the RAID architecture of the NAS protects against disk failures). The drawback is the network load, which is however not so heavy (a few kB/s) for the Monte Carlo output, being the NFS capabilities about 5-10 MB/s per Ethernet link on a Fast Ethernet LAN. For analysis jobs the expected data throughput across the network will be considerably higher, and different solutions for the network connections should be investigated.

Concentrating also the OS storage on the NAS makes system installation and administration easier, because:

- the OS installation is more reliable due to the redundant architecture of the NAS;
- software packages must be installed only once;
- local variations from standard are not possible;
- the root file-systems of every node are accessible under one unique file-system (the NAS one);
- OS backups can be easily centralized and realized automatically;
- it is simpler to make the system physically tamper-proof.

Another drawback can be the network load during bootstrap (during normal operation the network load due to OS accesses is negligible), especially for simultaneous bootstraps of a lot of machines. These problems, however, can be addressed by using a fast LAN (100Base Ethernet is enough), MTFTP (Multicast Trivial File Transfer Protocol), and multiple server configurations (which increase reliability and allow load balancing).

The suppression of the swap area, that is possible only on job execution nodes, means to avoid completely mass storage devices. Disk-less systems are:

- cheaper both in the initial purchase and in the subsequent maintenance;
- more robust for demanding computing environments (there are no moving parts in the PC, except the fans);
- more robust in coping with improper operations (no file-system check is required after a hard reboot of a hung-up system).

The drawback of swap-less systems is the abrupt job termination if the memory demand of the system exceeds the RAM size. It should be noted, however, that on job execution nodes the memory demand is almost stable, because they only run a fixed number of jobs (usually one per CPU), the OS kernel and few system daemons. On a disk-provided PC this means that either the system never pages (if the RAM size is larger than the system memory demand) or it pages continuously also the memory allocated by the Monte Carlo or analysis jobs, thus unacceptably slowing down the computing. Therefore on both disk-less and disk-provided systems, the RAM must anyway be sized in order to avoid completely swapping and paging. On the contrary, swap space is needed on login nodes, on which many user processes could be started, most of them usually in sleeping state.

Therefore, as job processing nodes, we have chosen naked motherboards equipped simply with an Ethernet adapter and a graphics card, for on-site debugging purpose and initial BIOS configuration, void of any other peripheral. To save space dual processors motherboards have been hosted in 2U boxes[2] mounted on standard 19" rack. A photograph of one motherboard mounted inside its box is shown in Fig. 5.

3.3 RAM Debugging

Memory modules can fail due to production defects (specks of dust that sit down on the chip while enlightening or developing one of its layers) or due to static discharges which damage internal connections.

The memory we use is not composed by ECC RAM modules (which include additional bits to store Hamming Error Correcting Code [3] to detect errors, and correct them if they are not too huge), because ECC modules are more expensive. It should be therefore accurately debugged in order to have a stable and reliable system (memory faults can cause a system hung-up). The memory tests performed by the system BIOS during the startup works fine for hard memory failures, but turns out to be useless for finding intermittent memory errors and sneaky cross-talks, which are the most frequent on memory banks.

To test the memory modules we make use of the Memtest86 program [4].

3.3.1 Memtest86

Memory chips consist of a large array of tightly packed 1-bit memory cells. The vast majority of the intermittent failures are caused by the cross-talk between these memory cells: writing a memory cell can cause one of the adjacent cells to be written with the same data. An effective memory test should attempt to test for this condition.

This strategy requires an exact knowledge of how the memory cells are laid out on the chip. In addition there is a never ending number of possible chip layouts for different chip types and

[2] 1U = 1.75" = 4.45 cm (DIN 41491, IEC 60297)

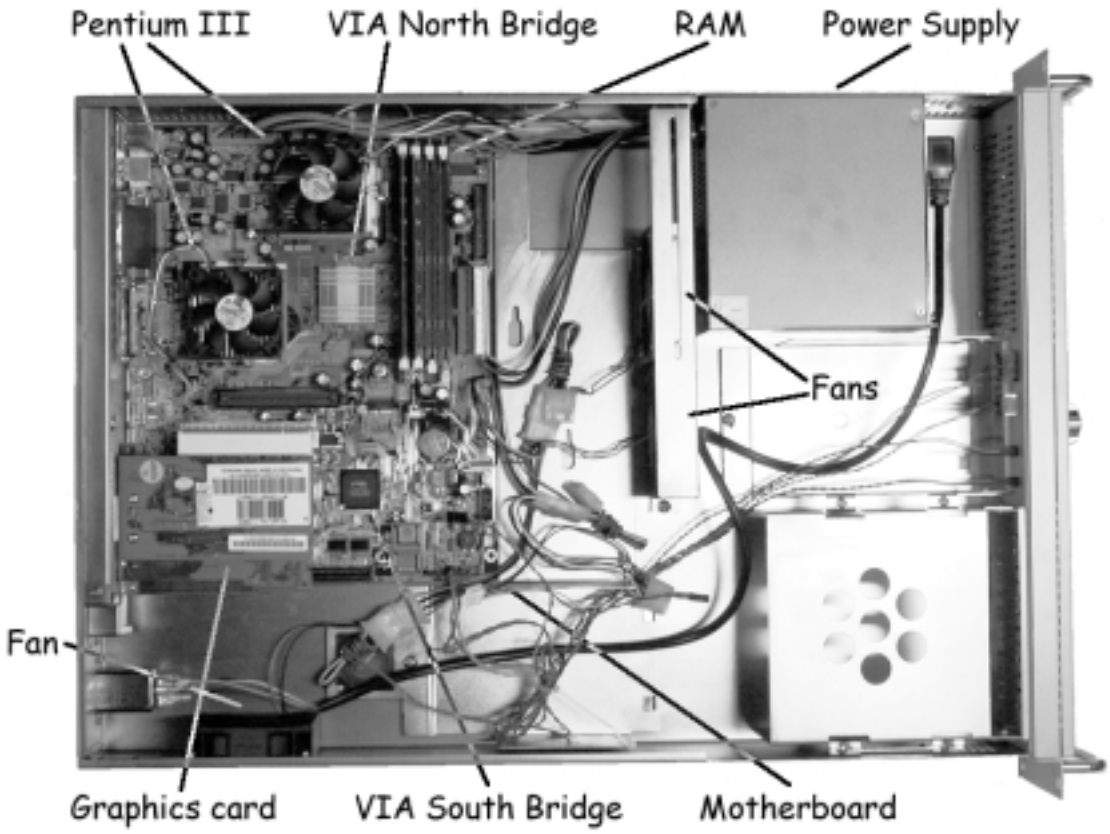

Figure 5: Farm node mounted inside its rack-mount box. The motherboard is a GygaByte 6VXDC7, equipped with 2 Socket 370, 800 MHz Pentium III processors, the VIA Apollo Pro 133A chipset (VT82C694X North Bridge + VT82C686A South Bridge) and 512 MB RAM. Graphics card plugged into PCI bus extender is also visible.

manufacturers making this strategy impractical. However, there are testing algorithms that can approximate this ideal strategy.

Memtest86 uses different algorithms and different tests that provide a reasonable approximation of the ideal testing strategy above mentioned. Details on Memtest86 can be found in [4].

3.3.2 Fixing RAM Errors: the BadRAM Linux Kernel Patch

Once an error is found on the RAM, the damaged RAM module should be obviously replaced. Otherwise, a different method to address the problem consists in excluding the bad memory location from the memory assigned to processes by the kernel, provided that the error is not located in low memory address range, where the kernel itself must reside.

The BadRAM kernel patch [5] can do it, by locking given memory areas (previously identified by means of Memtest86) during the boot up phase of the Linux kernel.

The idea behind relies on the memory management approach inside the Linux kernel, which is able to allocate (and never swap) fixed RAM physical addresses required by the kernel core itself. This feature is exploited by BadRAM to allocate precisely the defective parts of RAM before they are made available to anyone else. By allocating them for the kernel, and never freeing them, those parts of the memory are effectively disabled.

3.4 Temperature, Voltages and Fan Speed Monitoring

Temperature and fan monitoring is an important issue in building large farms. Fans are moving parts of a computer and are subject to failures which can cause system damage or at least system switch off (driven by motherboard protection).

Most of the PCs, built since late 1997, come with a hardware health monitoring chip. This chip may be accessed via the **ISA** bus or the **SMBus** (System Management Bus), depending on the motherboard. Many modern motherboards have a SMBus. There are a lot of devices which can be connected to a SMBus; the most notable are modern memory chips and chips for hardware monitoring.

SMBus is a specific implementation of the more general **I2C** (Inter Integrated Circuits) bus which originates from either the Intel PIIX4 or the Via south-bridge. It is a two-wire, low-speed serial communication bus, used for basic health monitoring and hardware management. The signal on one wire is the clock while on the other is the data. The clock line tells devices when to latch or send bits on the data line.

Using the SMBus one can communicate with many tens of devices, in theory close to a thousand, even if in our case we are interested in information coming from few devices. The throughput speed of the SMBus is somewhat slow (clock is about 100kHz), but is more than fast enough for system control/monitoring.

The monitoring chips, read out via the SMBus, read things like CPU temperatures, fan rotation speeds and voltage levels. There are quite a few different chips which can be used by motherboard builders for having approximately the same results. Most new processors contain a thermal diode on the die itself which is very accurate.

We tested successfully the software package "lm_sensors" [7], which includes general libraries and hardware-specific software, to access the SMBus to get hardware health informations.

3.5 Remote Power Control

As is well known, sometimes a PC can hang-up, due to OS bugs or hardware failures. Even if this event is very rare on a single Linux system, when the number of PCs becomes larger and larger, this possibility should be taken into account.

To restore a hung-up PC, the only way is to reset it, or to switch it off and then on again, because standard shutdown/reboot procedure cannot be started.

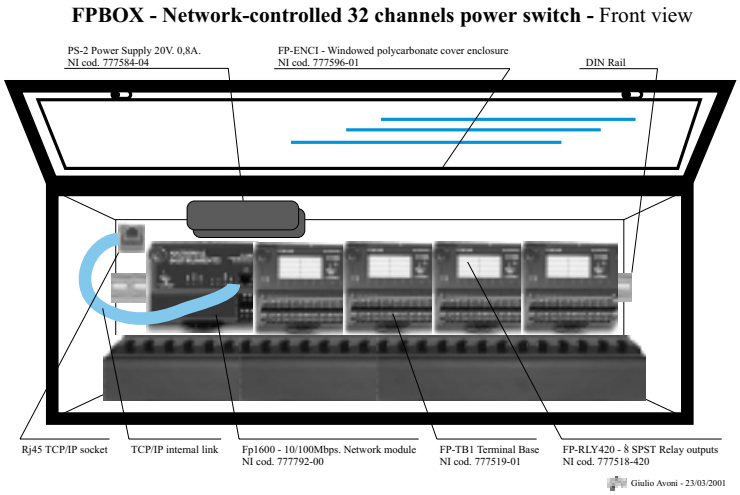

Figure 6: Front and rear views of the power switch control box, hosting the National Instruments FieldPoint distributed I/O modules. In the front view (the upper plot) the FieldPoint modules are visible: one FP-1600 module for the network control and four FP-RLY-420 modules each providing 8 independent relays. In the rear view one can see the 32 IEC female sockets power outputs (grouped four by four), the power input plug and the Ethernet link.

The **Wake-on-LAN** [15] protocol, from **Wired-for-Management** Intel specification [14], does not help for this purpose, because it only allows to switch on a PC in a standby state, but it does not allow to switch it off.

We have decided therefore to use a remote power switch control, driven through the network. Several firms provide this kind of device, among them BayTech [8], National Instruments [9] and Western Telematic [10].

FieldPoint modular distributed I/O system [9], by National Instruments, allows to connect a 100Base-T network interface module (FP-1600) to up to 9 relays modules (FP-RLY-420), each having 8 independent normally open relays. Commands to switch on and off the relays are sent to the system through IP protocol. A picture showing our arrangement of the FieldPoint modules can be found in Fig. 6.

4 The Network Boot

The bootstrap process on a computer is basically a matter of one program starting another, each more intelligent than the one before.

The very first program, executed when the computer is switched on, immediately after the POST (Power On Self Test), is stored in the ROM of the system BIOS. At this point all the PCI cards are given the chance to execute their, eventually present, own ROM code.

During disk boots, the disk controller (EIDE or SCSI) passes the control to a boot routine stored in the master boot record of the hard disk. In network boots, the NIC (Network Interface Card) passes the control to a code stored in a 32k or 64k PROM chip (**Boot-PROM**) plugged on the NIC itself. Boot-PROM contains enough code to contact a server and install a boot image in memory, which is in turn enough to start a full operating system. Common Boot-PROM codes usually use the standard TFTP (Trivial File Transfer Protocol) to download a file from the server.

A network boot starts with a standard **DHCP** (Dynamic Host Configuration Protocol) broadcast (DHCP-Discover) [11, 12]. This is a request to any DHCP server on the network. Every DHCP server can answer with a DHCP-Offer message, containing the PROM basic configuration information (such as its IP address, the subnet mask, the address of a TFTP server to download the boot code from, and so on). The Boot-PROM then chooses the configuration among the offered configurations and sends a DHCP-Request message. The chosen server, in turn, sends back a DHCP-Acknowledge message.

Using the information provided by the DHCP server, the PROM can now set up its own TCP/IP stack and start up the Trivial File Transfer Protocol (TFTP) to fetch the specified file (the boot image) from the TFTP server.

4.1 PXE (Pre-boot Execution Environment) Specifications

The main drawback of the network boot is the possibility of network load saturation during simultaneous boots. When many systems boot simultaneously, they try at the same time to download the kernel image through a TFTP connection causing a high network load (and possibly congestion, because TFTP uses UDP protocol), which can lead to timeouts of the ongoing TFTP transactions and hence to bootstrap failures.

A method to address this problem consists in using IP Multicast, which allows to send the same IP packet to more than one node at the same time. Using TFTP with IP Multicast, allows multiple clients to receive the same file concurrently through the use of Multicast packets. However Multicast TFTP (**MTFTP**) is not yet a standard.

PXE specifications [13], introduced in 1998 by Intel as a part of **Wired-for-Management** specifications [14], define a proprietary implementation of MTFTP with wide spread usage.

PXE embodies three technologies that establish a common and consistent set of pre-boot services within the boot firmware of Intel Architecture systems:

- A uniform protocol for the client to request the allocation of a network address and subsequently request the download of a NBP (Network Bootstrap Program) from a boot server.

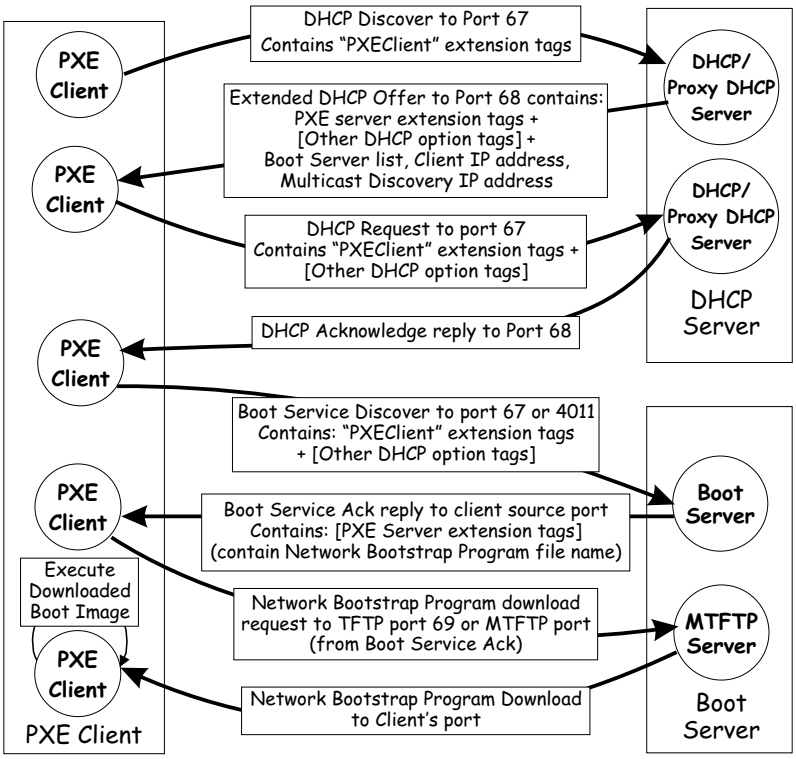

Figure 7: Block-like representation of Intel Pre-boot Execution Environment bootstrap phases.

- A set of APIs (Application Program Interface) available in the machine pre-boot firmware environment that constitutes a consistent set of services that can be employed by the NBP or the BIOS.
- A standard method of initiating the pre-boot firmware to execute the PXE protocol on the client machine.

A block-like representation of each step the PXE boot procedure is shown if Fig. 7.

4.2 Multicast TFTP

Multicast is a standard TCP/IP feature which is described in the Internet Request for Comments document RFC 1054 [16].

"IP multicasting is defined as the transmission of an IP datagram to a 'host group', a set of zero or more hosts identified by a single IP destination address. The membership of a host group is dynamic; that is, hosts may join and leave groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time. A host group may be permanent or transient. A permanent group has a well-known, administratively assigned IP address. It is the address, not the membership of the group, that is permanent; at any time a permanent group may have any number of members, even zero."

The key difference is that all packets go to a specific IP address rather than the IP address of the requesting client. This multicast IP address is determined by the system administrator, and there must be one address per file that is required to be able to be sent. This IP address must be first transmitted to the client so that it knows what to listen for, and this is done through the DHCP response packet, in the vendor-specific tag section. In the absence of RFC support, at present the Boot-PROM code gets all its setup information via DHCP reply. These information consist of:

- A UDP (User Datagram Protocol) port number on which the MTFTP server receives packets from the client.
- A UDP port number on which the MTFTP client receives packets from the server.
- An IP address, to which the server sends data to the client. Any multicast IP address starting at 224.0.0.0 can be used. Due to reservations only addresses starting at 225.0.0.0 should be used.

MTFTP is described in RFC 2090 [17] but it is not yet an Internet standard. MTFTP is an extension of TFTP, which is a simple lock-step, file transfer protocol which allows a client to get a file from a remote host.

TFTP has been implemented on top of UDP. Any transfer starts with a request to read (or write) a file, which also acts as a request of connection. If the server grants the request, the connection is opened and the file is sent in fixed length blocks of 512 bytes.

Each data packet contains one block of data, and must be acknowledged by an ACK (ACKnowledgment) packet before the next packet can be sent. An ACK packet confirms the reception of the n-th block of data and requests the $(n+1)$-th block. A data packet of less than 512 bytes signals termination of the transfer. If a packet gets lost in the network, the intended recipient issues a timeout and may retransmit its last packet (which may be data or an acknowledgment), thus triggering the sender of the lost packet to retransmit it. The sender has to keep just one packet on hand for retransmission, since the lock-step acknowledgment guarantees that all older packets have been received.

By using IP Multicast, a MTFTP server can transfer data to more than one client. The "oldest" client (the first which connected to the server) acts as a "master client" and is responsible for sending back the ACK message to the server.

Any subsequent client can start receiving blocks ("listening") during an already started data transfer. At the end of the master client transfer, the server sends an OACK (Option Acknowledgment) packet to the next oldest client. The second oldest client, in turn, becomes the master client and sends the server the ACK packet corresponding to the packet immediately before the first block required to complete the download.

5 Costs

The total costs (evaluated during March and April, 2001) of the farm prototype, with 8 biprocessor nodes and 1 TB of disk is about 52 kEuros (excluding VAT):

- 3.1 kEuros for the rack,
- 3.4 kEuros for the Ethernet switch (40 channels, expansible up to 80 channels),
- 31.0 kEuros for the NAS,
- 1.6 kEuros for each farm node (12.8 kEuros total),
- 2.1 kEuros for the remote controlled power switch (16 channels, expansible up to 72 channels).

The rack is 47U height[3], and therefore can store the Ethernet switch (4U), the NAS (4U) and up to 19 PCs (2U).

The Ethernet switch (HP ProCurve 4000m) has a backplane speed of 3.8 Gbps, and can be expanded by plugging additional modules in the 5 open slots. HP J4111 module (0.62 kEuro) provides eight 100Base-T sockets, J4113 module (1.24 kEuro) provides one 1000Base-SX, and J4114 module provides one 1000Base-LX socket.

Farm nodes are based on GygaByte 6VXDC7 motherboard, equipped with 2 Socket 370, 800 MHz Pentium III processors, the VIA Apollo Pro 133A chipset and 512 MB RAM.

[3] 1U = 1.75" = 4.45 cm (DIN 41491, IEC 60297)

6 Conclusions

The leading concepts which inspired the design and the realization of the computing farm for the LHCb Italy Tier-1 prototype have been presented.

The current farm prototype is hosted in a standard 19" rack, where each farm node is packed in a 2U rack-mounted box. The basic components of the farm, each of which have been described in this paper, are:

- Job processing nodes, based on a disk-less architecture, with the root file-system mounted over the network through NFS (Network File System). These nodes cannot be accessed directly by the users, and only the farm master batch system can spawn processes on them.
- Login nodes, with an architecture analogous to the job processing nodes, but with a swap area mounted on a local disk. These nodes are designed to run interactive sessions, the master batch queuing system, the GRID software and so on. The users can connect to login nodes to perform program compilations, start jobs on processing nodes, transfer data to/from the outside world and for any interactive-like need.
- Centralized disk data storage served by a NAS (Network Attached Storage), on which both Input/Output files of Monte Carlo jobs and Operating System file-systems reside.

The adopted network boot procedure, based on Intel PXE (Pre-boot Execution Environment) specifications, necessary for the realization of the disk-less nodes, has been described in detail.

The selected solutions for other important issues, like RAM modules debug, hardware health monitoring and remote power control, have been also addressed and discussed.

References

[1] American National Standards for Information System, *Small Computer System Interface*, *SCSI-1*, X3.131-1986.

[2] David A. Patterson, Garth Gibson, Randy H. Katz, *A Case for Redundant Arrays of Inexpensive Disks (RAID)*, Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, June 1-3, 1988. SIGM, September 1988, 109-116.

[3] R. W. Hamming, *Error-detecting and error-correcting codes*, Bell Sys. Tech. J. **29** (1950), 147-160.

[4] Chris Brady, *Memtest86 - A Stand-alone Memory Diagnostic*, `http://reality.sgi.com/cbrady/memtest86/`.

[5] Rick van Rein, *BadRAM kernel extension*, `http://home.zonnet.nl/vanrein/badram`, `http://rick.vanrein.org/linux/badram`, `http://www.badram.com/`.

[6] *ReiserFS file system*, `http://www.reiserfs.com/`.

[7] *Hardware monitoring by lm_sensors*, `http://www.netroedge.com/~lm78/`.

[8] BayTech, *Remote Power Control series*, `http://www.baytechdcd.com/products/rpcseries.shtml`.

[9] National Instruments, Distributed I/O, FieldPoint, `http://www.ni.com`, `http://sine.ni.com/apps/we/nioc.vp?lang=US&pc=bypsc&cid=1206`.

[10] Western Telematic inc., *Remote Power Control and Remote Reboot Products*, `http://www.wti.com/power.htm`.

[11] R. Droms, *Dynamic Host Configuration Protocol*, Request for Comments 2131 (1997), ftp://ftp.isi.edu/in-notes/rfc2131.txt, http://www.faqs.org/rfcs/rfc2131.html, http://www.rfc-editor.org/rfcsearch.html, http://www.faqs.org/rfcs/rfcsearch.html.

[12] S. Alexander, *DHCP Options and BOOTP Vendor Extensions*, Request for Comments 2132 (1997), ftp://ftp.isi.edu/in-notes/rfc2132.txt, http://www.faqs.org/rfcs/rfc2132.html, http://www.rfc-editor.org/rfcsearch.html, http://www.faqs.org/rfcs/rfcsearch.html.

[13] Intel, *Preboot Execution Environment (PXE) Specification Version 2.1*, ftp://download.intel.com/ial/wfm/pxespec.pdf.

[14] Intel, *Wired for Management (WfM) Developer site*, http://developer.intel.com/ial/WfM/index.htm.

[15] Intel, *Wake-on-LAN, Wired for Management System Elements*, http://developer.intel.com/ial/managedpc/elements.htm.

[16] S. Deering, *Host Extensions for IP Multicasting*, Request For Comments 1112 (1989), ftp://ftp.isi.edu/in-notes/rfc1112.txt, http://www.faqs.org/rfcs/rfc1112.html, http://www.rfc-editor.org/rfcsearch.html, http://www.faqs.org/rfcs/rfcsearch.html.

[17] A. Emberson, *TFTP Multicast Option*, Request For Comments 2090 (1997), ftp://ftp.isi.edu/in-notes/rfc2090.txt, http://www.faqs.org/rfcs/rfc2090.html, http://www.rfc-editor.org/rfcsearch.html, http://www.faqs.org/rfcs/rfcsearch.html.