The IFISC computing network

Pere Colet

Rubén Tolosa
Eduardo Herraiz
Juanjo Enseñat
Manuel A. Matías

IFISC

Outline

Overview of previous configuration

LANs & VLANs

Public and private IPs

IFISC VLAN configuration

disk server: dolsa

Mail and web server: galiota

Desktops & Laptops

Backup: molinar

WIFI

HPC/HTC

Nuredduna. Configuration & Use

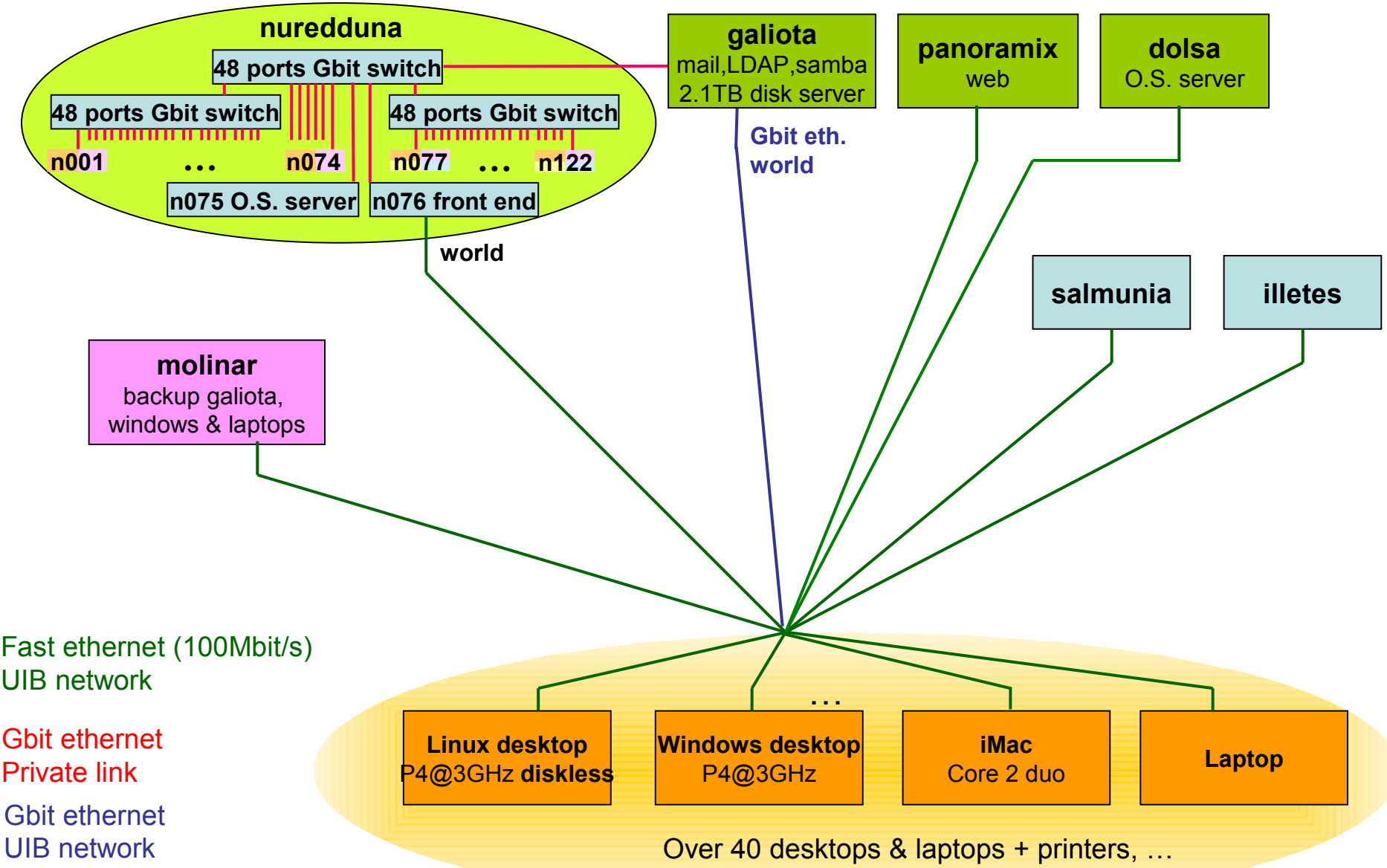
Other computational servers: salmunia & illetes

CPUs

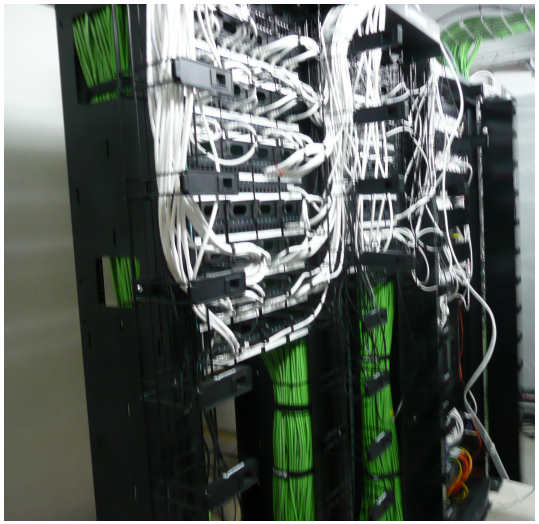
Efficient programming

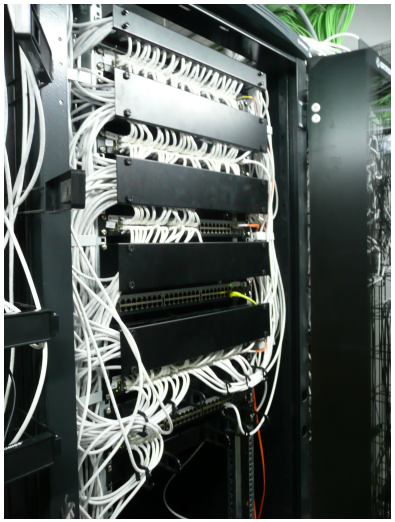
Still to be done …

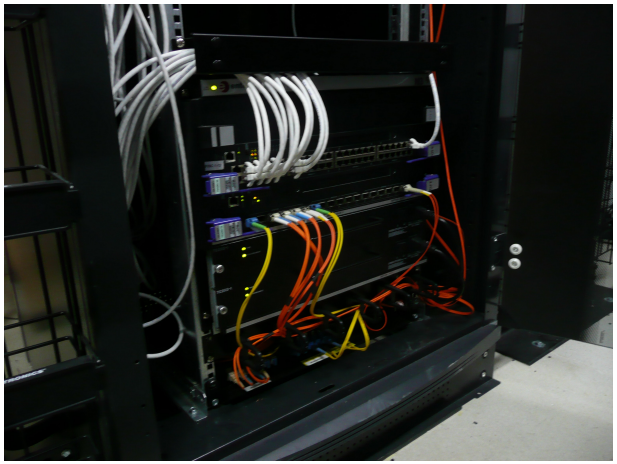
Previous network configuration

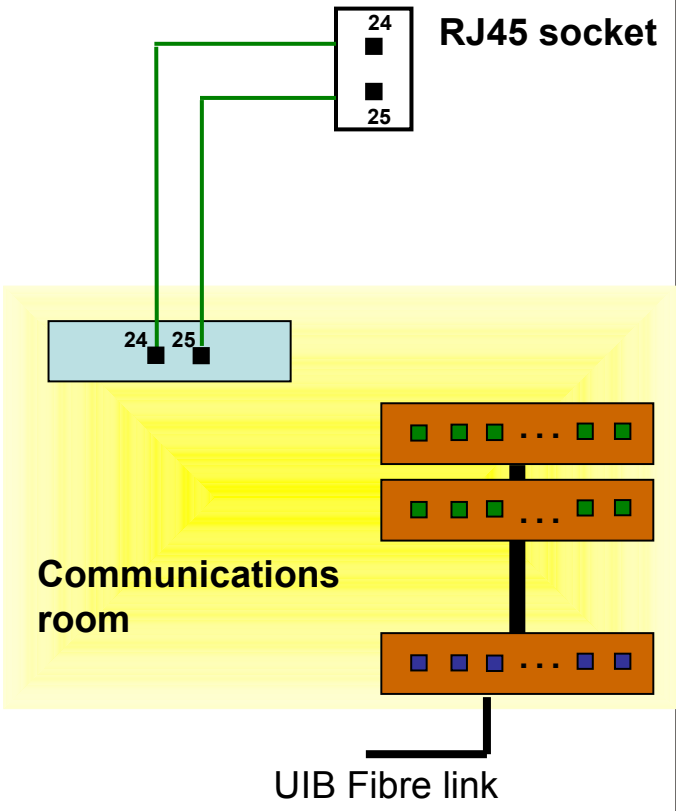


Ethernet layout







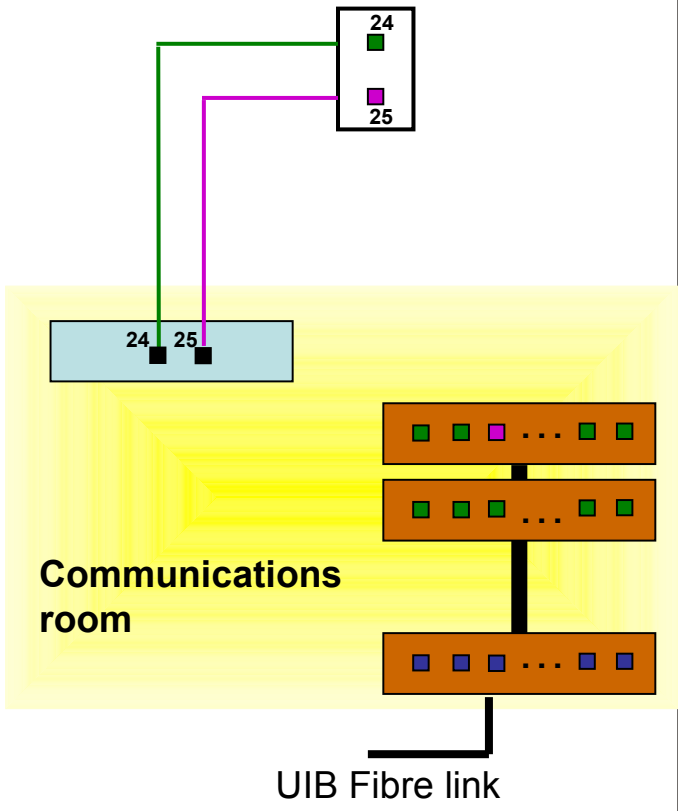


Fast ethernet (100Mbit/s)

Gbit ethernet

VLAN: Virtual Local Area Network:

- Each switch port can be configured (via software) to be part of a given VLAN.
- Only the ethernet traffic of that VLAN goes through the port (and through the associated RJ45 socket).
- Separates ethernet traffic among IFISC computers from the of the rest of the UIB traffic so that communications between IFISC computers should be more efficient.



Fast ethernet
IFISC VLAN

Fast ethernet
Pinky VLAN

IP addresses

Each computer has an Internet Protocol (IP) address.
Within IFISC VLAN there are:

- **Public addresses**: are directly visible from any computer in the world. For example galiota.uib.es is 130.206.32.132

- **Private addresses**: 10.34.Y.X (Class B subnet) **computername.ifisc.lan**
  - Addresses managed by us, quick set up of new computers (visitors).
  - Only directly visible inside IFISC VLAN. Security: direct external attacks not possible.
  - Transparent access to external computers through a router managed by UIB-CTI: ssh –X hostname, rdesktop hostname to address remote unix/windows.
  - All ifisc.lan computers go out as IP 130.206.32.131. Easy to configure access (for example at CSIC servers) to all IFISC computers.

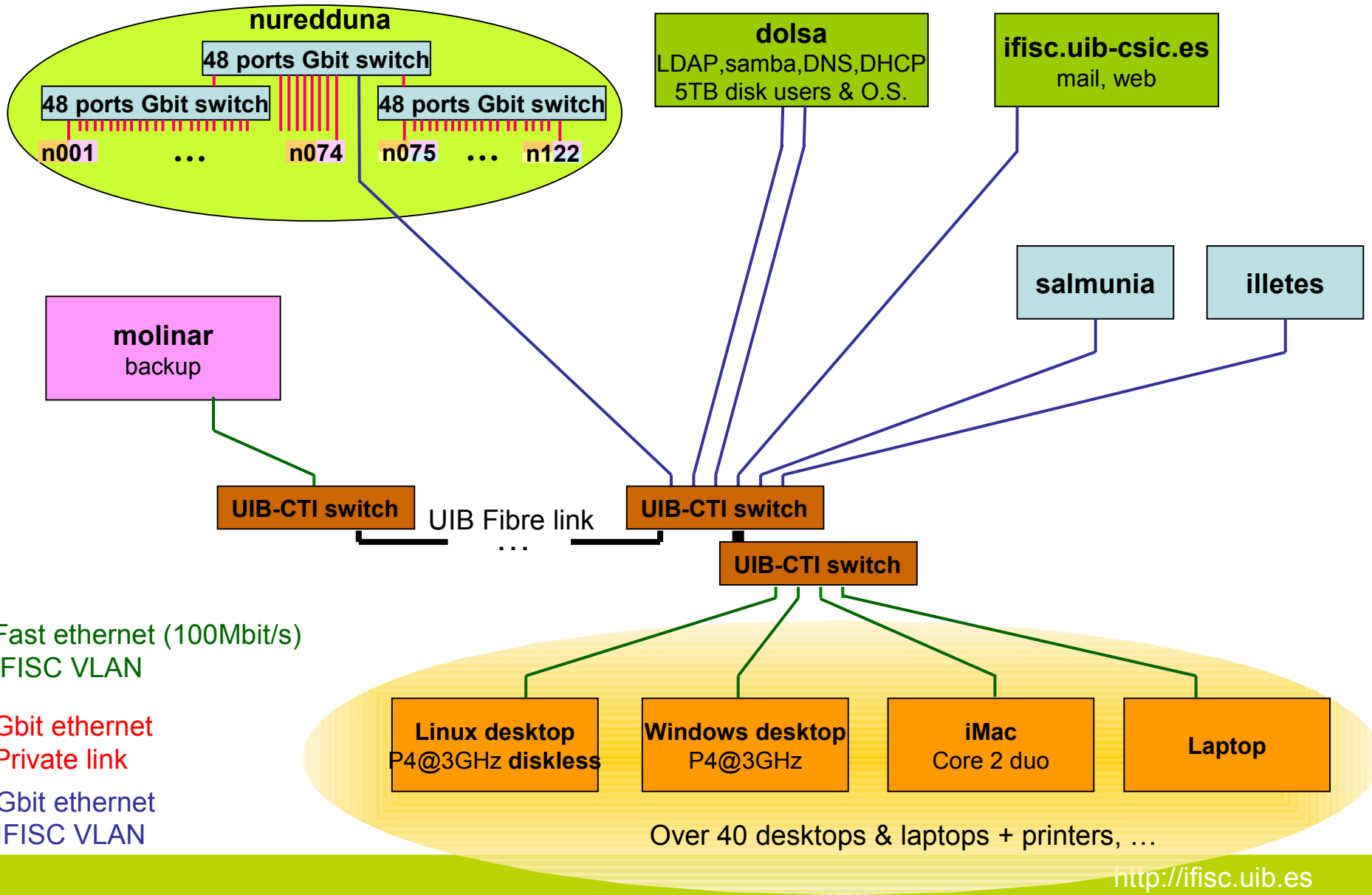
10.34.0.X - Reserved for CTI @ UIB
10.34.1.X - Servers
10.34.2.X - Desktops
10.34.3.X - Laptops
10.34.4.X - Printers
10.34.10.X - Nuredduna
10.34.254.X - Tests

Domain Name Server

Domain Name Servers (DNS) translate computer names to IP addresses.

- Up to now we had a Slave DNS which replicated information of the UIB DNS.
- Now our DNS is the Master for the **ifisc.lan** subnet. It will also cache external IP addresses. It works as follows:
  - 1. Searches inside the ifisc.lan domain (for example, to access nuredduna.ifisc.lan one could simply write ssh -X nuredduna).
  - 2. If not found looks at the local cache.
  - 3. If not cached forwards the request to UIB DNS which resolves the name or forwards the request to other DNSs.
  - 4. Once the name is resolved its IP is stored in the local cache.
- We can define several aliases for computer names if any one would like to give a particular name to its computer e.g. beca07.ifisc.lan could also be JamesBond.ifisc.lan

IFISC VLAN configuration



Fast ethernet (100Mbit/s)
IFISC VLAN

Gbit ethernet
Private link

Gbit ethernet
IFISC VLAN

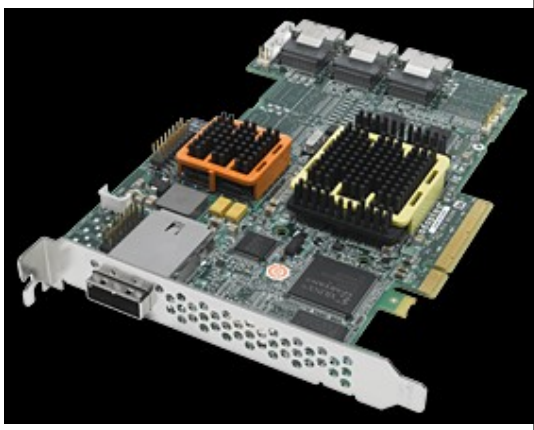
The most important server for day to day operations:

- **Lightweight Directory Access Protocol (LDAP)** for user authentication in servers, desktops and in the intranet.
- **Dinamic Host Configration Protocol (DHCP)** which assigns names and IPs to all IFISC desktops and laptops when they boot.
- **Master DNS of ifisc.lan**.
- **Disk Server:**
  - users home directory. 5 TB. XFS file system.
  - operating system for all diskless desktops and nuredduna nodes.
- **Samba Server:** file and print services for windows computers.
- **Common Unix Printing System (CUPS) Server:** Allowing easy printers configuration in linux computers.

Operating System: Ubuntu Server 8.04 LTS considered particularly stable so that Long Term Support (5 years) is provided. Only few selected Ubuntu server distributions are LTS.

dolsa.ifisc.lan. The hardware

- 2 quad-core Opteron 2350 processors @ 2GHz.
- 8 GB of registered Error Correcting Code (ECC) memory.
- State-of-the-art Adaptec RAID 51245 card:
  - up to 16 SAS or SATA disks (256 with expanders).
  - online capacity expansion.
  - dedicated dual core processor. 512 MB cache memory.
  - Dedicated battery to protect cache contents for three days in case of power failure
  - Self-Monitoring Analysis and Reporting Technology (SMART) alert of problems before critical failure.
  - Raid 5: if one disk breaks continues working without interruption.
- Six 1TB disks Seagate Barracuda ES.2 (for 24 hours 7 days a week operation).
- Dual Gbit ethernet channel bonding: Up to 2Gbit/s. If one channel fails, the other takes over.



It should not fail, but …

We have spare power supply, motherboard and disk.

Only one processor needed. Still, we have a spare one.

The main IFISC public server.

- Aliases: galiota.uib.es, www.ifisc.uib-csic.es, ifisc.uib.es, www.ifisc.uib.es.
- Web server (accesible with any of the aliases, e.g, http://ifisc.uib.es)
  - statistics http://ifisc.uib-csic.es/cgi-bin/awstats.pl
- Mail server: handles e-mail to ifisc.uib-csic.es, ifisc.uib.es, imedea.uib.es
  - Client configuration: server: ifisc.uib-csic.es
    imap: you may have to re-subscribe folders
  - Some aliases (e.g. manuel.matias) may not work.

Hardware

- Quad-core Q6700 Intel Core 2 processor.
- 2 GB DDR2-800 memory.
- two 500 GB hard disks in Raid1 configuration for redundancy

Software:

- OS Ubuntu Server 8.04 LTS
- apache 2.2.8
- php 5.2.4 warnings/errors in few pages, being fixed.
- mysql 5.0.5

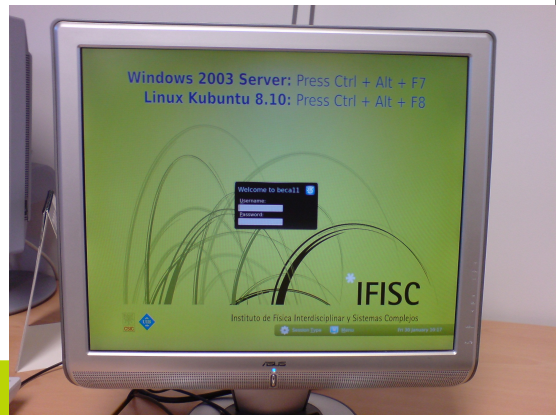
Desktops & Laptops

All desktops and laptops have private IP addresses and assigned by dolsa via DHCP. Names are of the form **computername.ifisc.lan**

IPs are not provided outside the IFISC VLAN, even in the UIB campus (use WIFI)

All linux desktops are diskless.

- Operating system: Latest Ubuntu 8.10 (released October 2008).
- New boot procedure:
  - **Previously:** the ethernet driver was integrated in the Kernel. Linux distributions come with a Kernel that loads dynamically the ethernet driver as a module. It was necessary to recompile the Kernel.
  - **Now:** initally a ram disk is created were modules needed by Kernel are be stored. This allows the use of any standard Kernel.
- Login screen allows to choose between the standard Linux environment or a windows remote desktop system (this is a light X-server that starts a windows remote desktop client on the windows server calabona).



Server for data backup.

A new version of molinar will be assembled using the old galiota hardware.

- Dual core Intel Core2 E6600 processor at 2.4 GHz.
- 2 GB DDR2 ram memory
- Gbit ethernet: Marvell 88E8001
- Adaptec RAID 51245 card
- Eight 750GB SATA2 disks (the disks that galiota and molinar have now)
- Raid 5 configuration.
- Operating System: Ubuntu Server 8.04 LTS
- It will be configured so that it can perform dolsa functions should it be necessary.

For security reasons will not be located in the IFISC building.

Virtual machine running windows server 2003.

- Used to run windows remote desktop to be displayed in desktops (linux, windows or mac), providing access to windows software.
- calabona was runing in salmunia. It worked but when salmunia was running several intensive computational programs the response of calabona was slow.
- calabona is going to be an independent computer to be assembled soon recycling a computer used for tests.
  - Quad-core Intel Q6600 processor @ 2.4 GHz.
  - 8 GB of DDR2 ram memory.

WIFI

- The WIFI network in the new building will be installed by the CTI soon.
- The WIFI will be managed by the CTI (as in the rest of the campus).
- The CTI has given us several usernames so that we can quickly configure some laptops for visitors.
- The WIFI network will not be part of the IFISC VLAN. In fact, the UIB WIFI is in itself a VLAN.
- The CTI will configure the router so that for example our printers (which have a private IP address) can be accessed from the WIFI.

The IFISC computing network II

Pere Colet

Rubén Tolosa
Eduardo Herraiz
Juanjo Enseñat
Manuel A. Matías

IFISC

Outline

Overview of previous configuration

LANs & VLANs

Public and private IPs

IFISC VLAN configuration

disk server: dolsa

Mail and web server: galiota

Desktops & Laptops

Backup: molinar

WIFI

HPC/HTC

Nuredduna. Configuration & Use

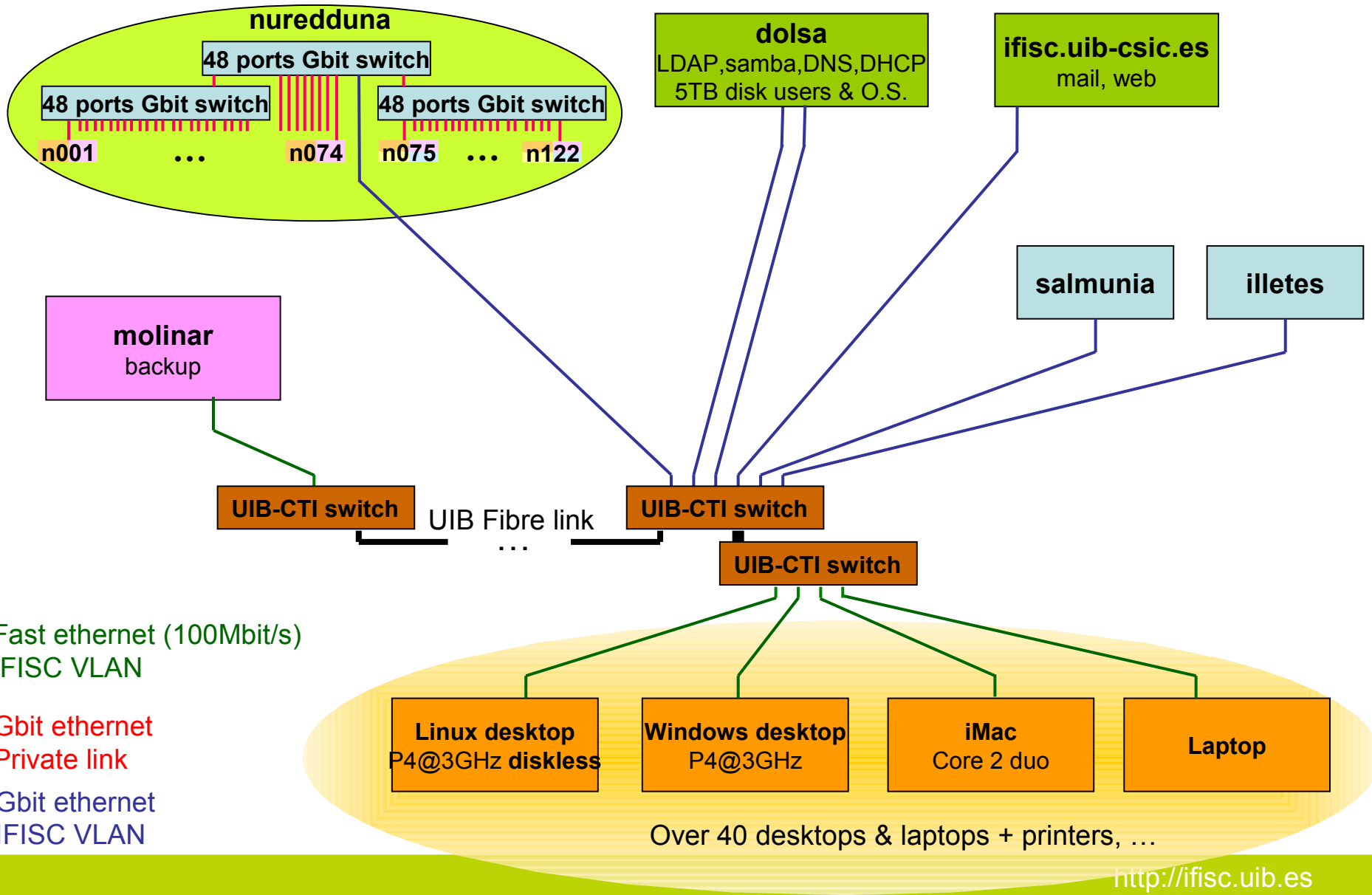
Other computational servers: salmunia & illetes

CPUs

Efficient programming

Still to be done …

IFISC VLAN configuration



Fast ethernet (100Mbit/s)
IFISC VLAN

Gbit ethernet
Private link

Gbit ethernet
IFISC VLAN

HPC/HTC context

High Performance Computing:

Using a big computer to solve big programs efficiently.

One (a few) programs are solved at a time.

High Throughput Computing:

Using a big computer to solve many (small) programs efficiently.

Communication speed between different computer parts is critical for HPC, not for HTC

Computer Architecture

- **Shared memory:** All processors see all the memory.
  - Easy with 2 processors: illetes, salmunia
  - 100 - 1000 processors. Very Expensive. SGI Altix, Finis Terrae

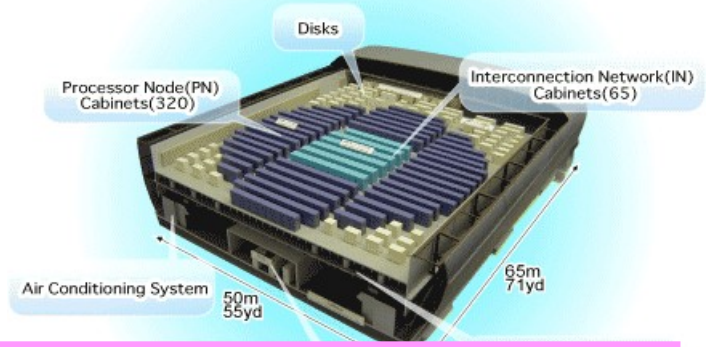
- **Distributed memory (Clusters):**
  - Each node sees only the local memory and runs its OS
  - Communication between nodes:
    - Gbit ethernet, 1 Gbit/s; Latency 30 μ s
    - Myrinet 2 or 10 Gbit/s; Latency 2.2 μ s
    - Infiniband 2.5, 5, 10, … Gbit/s; Latency 1.2 μ s
  - hundreds of processors cheap: nuredduna
  - >10.000 processors: Mare Nostrum

The fastest computers solving a system of linear equations: http://www.top500.org/

June 2004:

#1: The earth simulator (NEC) 5120 vector proc.: 35.9 Tflops
#2: Thunder (CDC) 4096 Itanium2 - Quadrics: 19.9 Tflops
#3 Alphaserver SC45 (HP) 8192 alpha – Quadrics: 13.9 Tflops
#4: BlueGene/L (IBM) 8192 PPC 440: 11.7 Tflops
#5: PowerEdge1750 (Dell) 1250 dual xeon - Myrinet: 9.8 Tflops

#500 delivers 673 Gflops.



June 2005:

#1 BlueGene (IBM), 65536 cell: 136.8 Tflops
#2 BlueGene (IBM), 40960 cell: 91.3 Tflops
#3 SGI Altix, 10160 Itanium2: 51.8 Tflops
#4: The Earth Simulator (NEC): 35.9 Tflops
#5 MareNostrum (IBM) 4800 PPC970: 27.9 Tf

#500 delivers 1.2 Tflops.

November 2006:

#1 BlueGene (IBM)131072 cell: 280.6 Tflops
#2 Red Storm (Cray) 26544 opteron dualcore: 101.4
#3 BlueGene (IBM) 40960 cell: 91.3 Tflops
#4 ASC Purple (IBM) 12208 power5: 75.8 Tflops
#5 MareNostrum (IBM) 10260 PPC970: 63.6 Tflops

#500 delivers 2.7 Tflops.

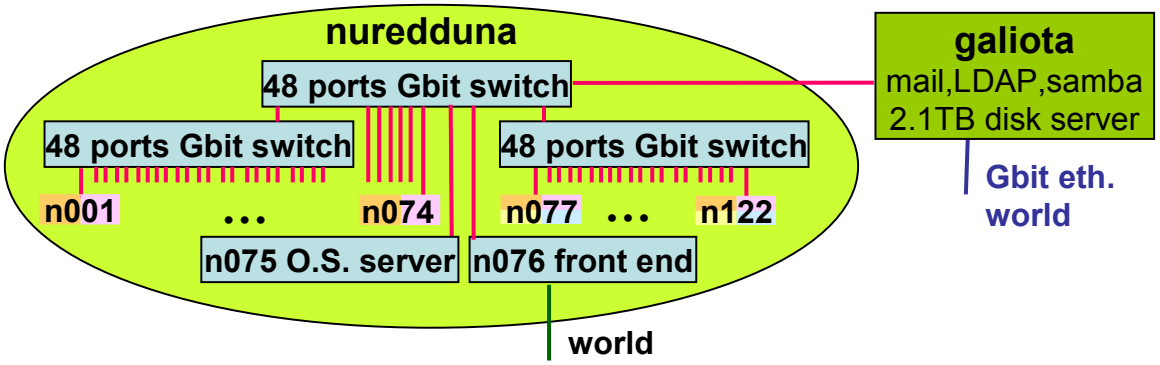
November 2008:

#1 Roadrunner (IBM) 129600 PowerXcell/opteron:	1105 Tflops	2483KW
#2 Jaguar (CrayXT5) 37538 opteron quadcore:	1059 Tflops	6950KW
#3 Pleiades (SGI), 12800 xeon quadcore:	487 Tflops	2090KW
#4 BlueGene/L (IBM), 212992 cell proc:	478 Tflops	2329KW
#40 MareNostrum (IBM) 10240 PPC970:	63.8 Tflops	
#73 The earth simulator: 5120 proc.	35.9	3200KW
#427 Finis Terrae, 2528 Itanium2 dualcore	14.1 Tflops	

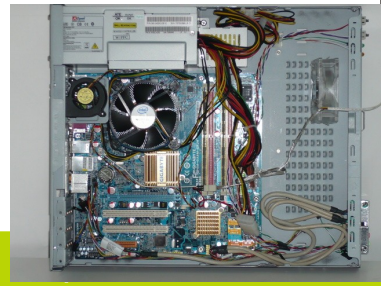
#500 delivers 12.6 Tflops. **Max. production Baleares: 2046KW**

nuredduna. Old configuration

IFISC's computing needs: Solve many (small) scientific calculations efficiently (HTC).
nuredduna: HTC cluster designed and assembled at IFISC. Periodically updated since Aug 2000.

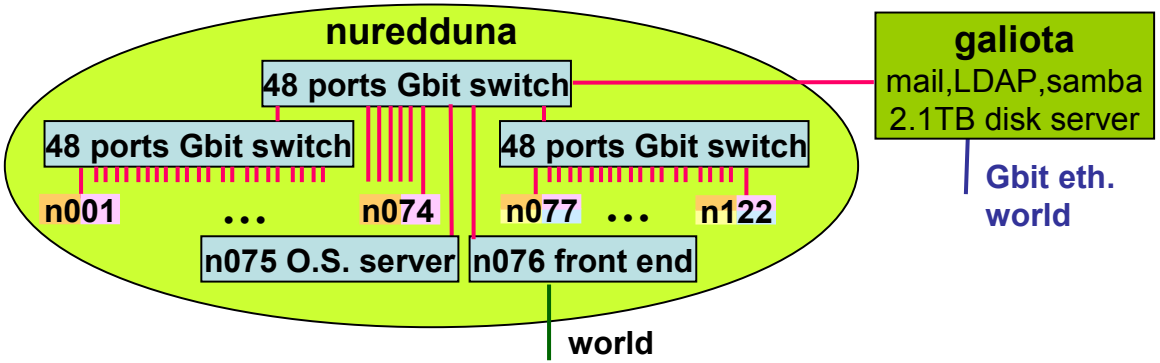


- Performance/Cost: PC components selected at IFISC after tests.
- 120 computing nodes: diskless: less consumption/heat, more reliable, cheaper.
- 2 special nodes: Front end (2 ethernets) and O.S. server.
- CPUs: Intel Core 2 Duo E6600 & Core2 Quad Q6700 processors.
- 340 computational cores, 256 GB of memory. Largest cluster at UIB.
- Three dedicated Gbit switches.
- Users see the cluster as a single linux computer.
- Mosix (a kernel patch) distributes load among nodes dynamically: executables migrate from one node to another without interrupting the calculation.



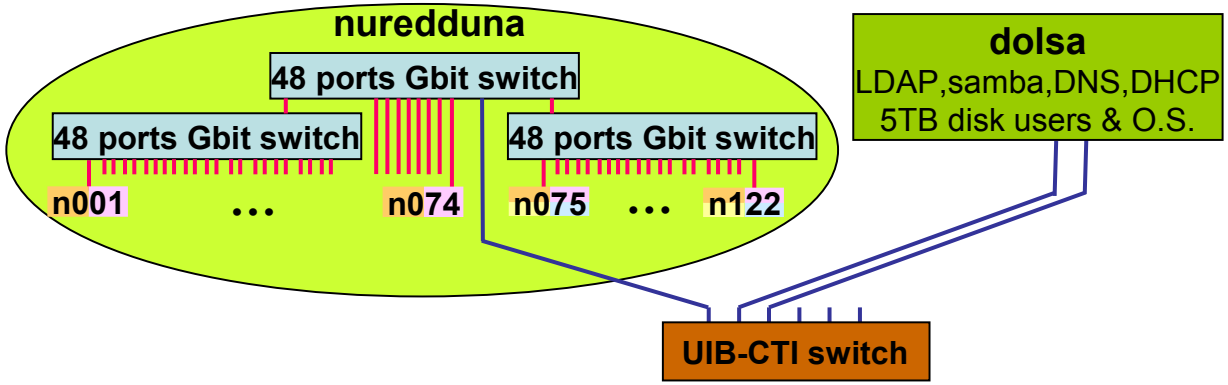
Recycling: previous nodes become desktop computers.

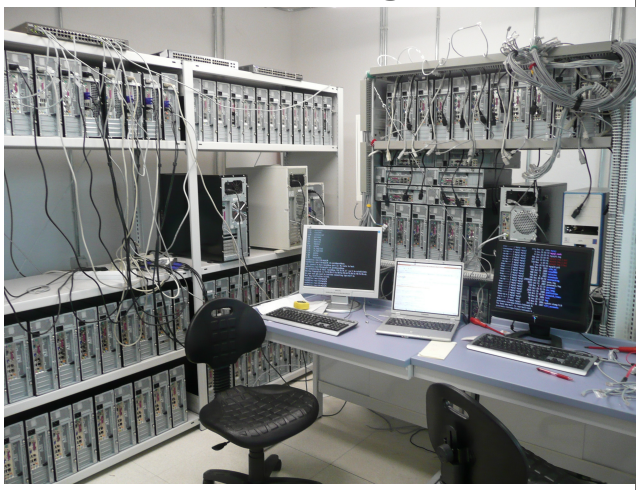
nuredduna.ifisc.lan



Old configuration

New configuration





- Same computational hardware as before.
- All nodes diskless. OS served by dolsa. New boot procedure as for linux desktops.
- All nodes can be accessed directly from any computer inside IFISC VLAN.
- No “special” front end: nuredduna.ifisc.lan is just an alias of n001.ifisc.lan. **Elegant & flexible**.
- OS: Ubuntu Server 8.10. Kernel 2.6.27.7, Mosix 2.24.2.4

Mosix

- Software layer that allows applications to run in remote nodes, away from its home-node, as if they run locally:
  - Mosix Kernel patch intercepts all system-calls.
  - Mosix can migrate a process to another node without interrupting the calculation.
  - For migrated processes, most of the system-calls are forwarded to home-node, performed there and results are sent back to the process.
  - A few system-calls can be performed directly on the remote node.
- Load balancing:
  - Each node calculates its load: # programs / (#cores*speed).
  - Every few seconds each node compares its load with another one randomly.
  - The load of these two nodes is balanced by migrating one or several programs.
  - Over the time this averages the load automatically without any “central authority”.
- If one node is shut down, mosix first migrates to another node the programs that have been sent there.
- The original home-node has to remain active until the program finishes, if not the program dies.
- Only programs launched with the command mosrun can be migrated.

Use of nuredduna

Login: **>ssh -X username@nuredduna.**

Also possible to log directly in a node **>ssh –X username@n005**

User directories: Same as in desktops. You can edit a program in desktop, compile and execute in nuredduna and analyze the results in your desktop.

Compilers: ifort & icc; pgf77, pgf90, pgcc & pgCC, gcc, g77 & g90.

Executing a program: **>run myprogram.x &**

Only programs launched with **run** or **mosrun** will be migrated.

Batch queue: Mosix scheduled to run one program per core simultaneously.

Other programs wait. Jobs of different users are interleaved.

Monitoring the system: **>mon –d** shows the load of every node.

>mosps aux similar to ps with indication of the node where program runs.

>mosq listall shows all queued jobs, including previous queued now running.

>estado shows a combination of the two above.

Checkpointing: saves an image of a running process: **>migrate pid checkpoint**

The process can be recovered from that point. **>mosrun -R filename**

More info: **http://ifisc.uib.es/intraweb/infowiki/doku.php?id=doku:nureduna**

Use of nuredduna. Be aware!

Be aware of the disk space you are using: **>du –sh .** Shows the directory size.

Be aware of how much time your programs are going to take.

Avoid launching hundreds of very short programs. Combine them so that the program takes at least a few minutes.

No restriction on number of programs you can run but you are not the only user!

Nuredduna is intended for programs with lots of calculations and few input / output:

- **Be aware of the amount of data your programs are going to generate.**
- **Use desktops for programs that handle a lot of data and do few calculations.**
- **Avoid printing data on the screen:** Slows down tremendously the programs because data is transfered in very small packages.
- **Avoid using the standard input (stdin), standard output (stdout) and standard error (stderr).** Instead data should be read from a file and written on a file since data it is transfered in larger blocks.
- If there is no way to change the program, redirect the standard output to a file

  >run myprogram.x > file_output.log &

- **Store large amounts of data in unformatted form.** Formatted data requires 5 to 10 times more space and time.

Computational server for programs requiring large memory. Also for software as mathematica, matlab or maple.

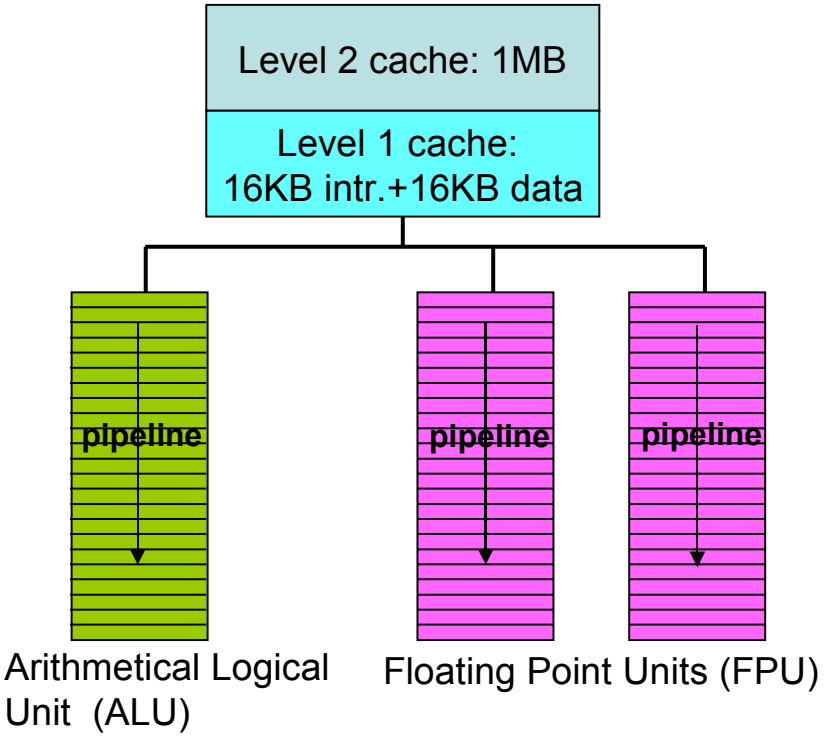
- Two quad-core CPUs AMD Opteron 2354 @ 2.2 GHz
- Memory: 32 GB DDR2 800 ECC registered.
- Gbit Ethernet: 2
- 250 GB SATA disk for operating system and applications.
- Nvidia GTX8800 graphics card for GPGPU (General Purpose computing on Graphics Processing Units). CUDA Toolkit.
- Operating System Ubuntu 8.04 LTS
- Software:
  - Mathematica 6.0
  - Matlab
  - Maple 12
  - Intel Compilers
  - Portland Group Compilers
  - Public domain GNU compilers
  - Intel Math Kernel Library (MKL)
  - AMD Math Core Library (ACML)

Computational server for using software such as **idl**, mathematica or matlab.

- Two dual core CPUs AMD Opteron 275 @ 2.2 GHz
- Memory: 8 GB DDR 400 ECC registered.
- Gbit Ethernet**:** 2 Broadcom NetXtreme BCM5704
- 250 GB SATA disk for Operating System and applications.
- Operating System: Ubuntu Server 7.04 (x86-64)
- Software:
  - **IDL 6.2**
  - Mathematica 6.0
  - Matlab
  - Maple 11
  - Intel Compilers
  - Portland Group Compilers
  - Public domain GNU compilers
  - Intel Math Kernel Library (MKL)
  - AMD Math Core Library (ACML)
  - FFTW

IFISC

Pentium 4



- Floating point units do a sum or a multiplication.
- Flux inside the CPU is divided in many stages (20 Nortwood, 31 Prescott).
- Every clock cycle data is moved from one stage to the next. New data can feed the previous stage.
- The first operation takes as much clock cycles as stages.
- Subsequent operations are released one every clock cycle.

Theoretical Peak: 3GHz x 2 Floating point op. = 6 GFlops.

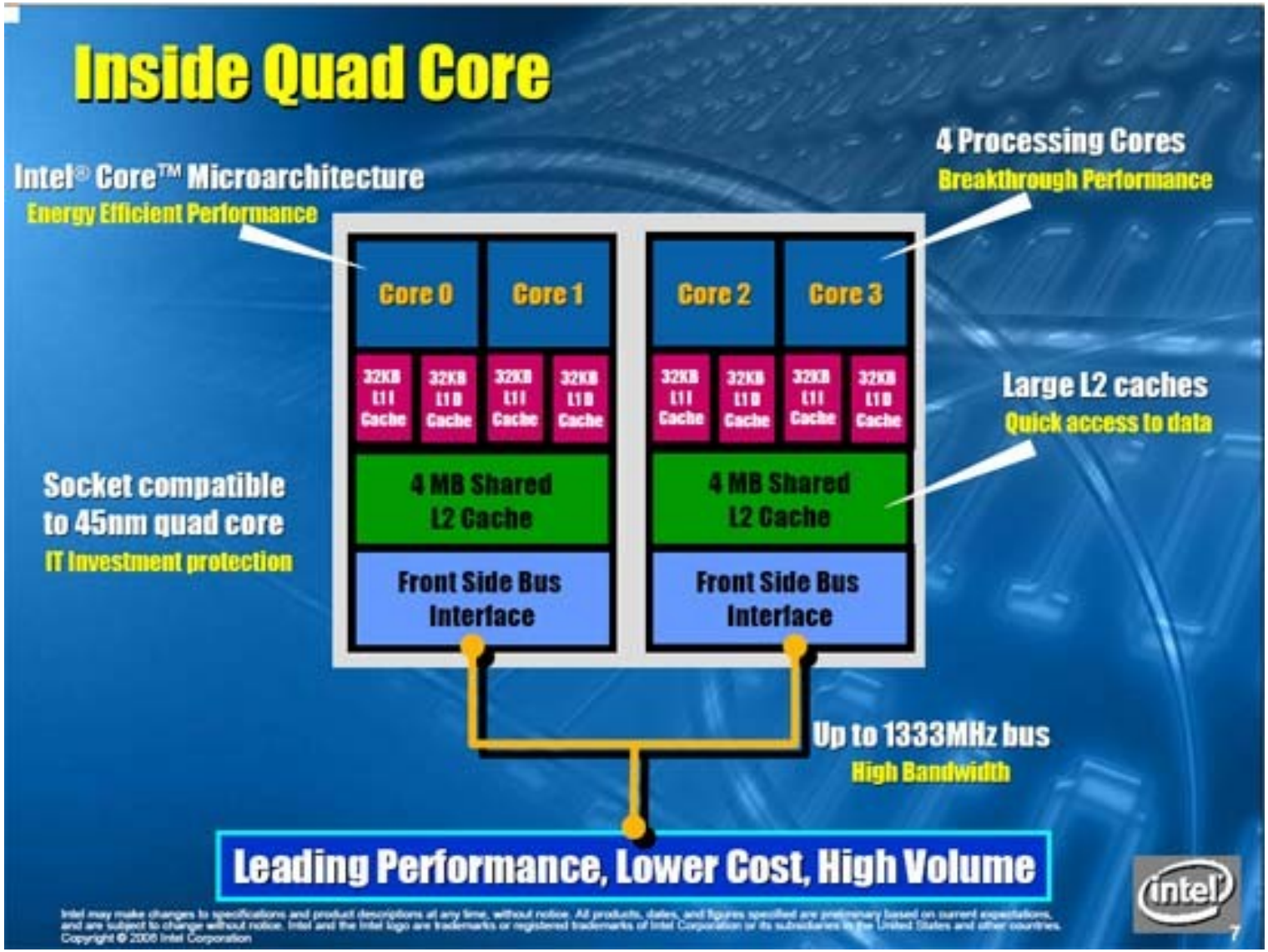
But:

- Conditionals break the flux. The pipeline has to be emptied and started again.
- If data is not in L1 cache, it has to be feed from L2 cache, which induces delays.
- If data is not in L2 cache, it has to be feed from main memory. Further delays.

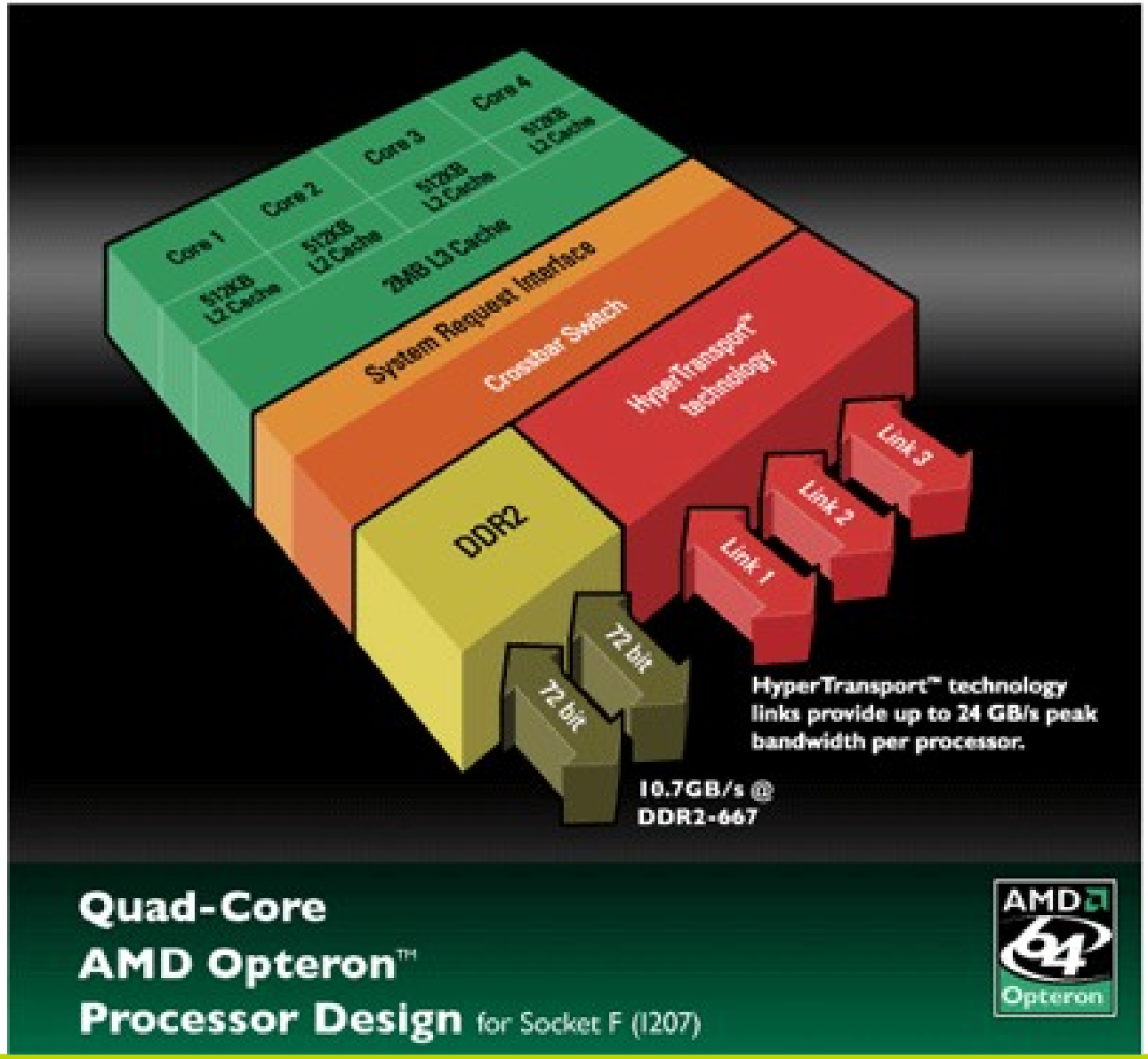
Beware also that:

- A division takes many clock cycles. Consecutive divisions are NOT released one every cycle.
- Sqrt takes of the order of 50-100 clock cycles.
- Trigonometric functions take about 300 clock cycles.

Intel Quad Core Processors



AMD Quad Core Processors



Programming: Look for possible mistakes!!

- When preparing a program make sure there are not common mistakes like:
  - dividing by zero.
  - accessing an element of a vector or an array which is out of the range.

- **Modern compilers** to optimize execution time, generate an executable that does **neither check out of limits memory access nor divisions by zero.**

- To perform these checks during execution compile with the option -C.
  >ifort -C -o myprogram_slow_verifies_all.x myprogram.f
  It runs slowly but at execution time everything is check and in case of a illegal access or a division by zero the program stops and reports the error.

- **Once you are sure the program is right**, then it is convenient to compile with optimization options for better performance.

- More information is avaliable on http://ifisc.uib.es/intraweb/infowiki/doku.php?id=doku:checking_a_program

Efficient programming: Eliminating the clutter (1)

Avoid conditionals inside loops:

```
do i=1,10000
  if (i.lt.500) then
    ...
  else
    ...
  endif
enddo
```

```
do i=1,500
  ...
enddo
do i=501,10000
  ...
enddo
```

```
do i=1,10000
  if mod(i,3) then
    a(i)=2.d0*i+28.d0
  else
    a(i)=5.d0*i*i
  endif
enddo
```

```
do i=1,10000,3
  a(i)=5.d0*i*I
  a(i+1)=5.d0*(i+1)*(i+1)
  a(i+2)=2.d0*(i+2)+28.d0
enddo
```

Replace slow operations by faster ones:

```
do i=1,10000
  a(i)=i**3/4.d0
enddo
```

```
do i=1,10000
  a(i)=0.25d0*i*i*i
enddo
```

```
real*8 a
complex*16 e
   a=3.d0*abs(e)^2
```

```
real*8 a
complex*16 e
   a=3.d0*e*conjg(e)
```

Efficient programming: Eliminatting the clutter (2)

It works.

```
a=0.d0
b=7.69
do i=1,10000
  a=sqrt(b)*i*i+a
enddo
```

It works faster

Compiler may take care of this…

```
a=0.d0
b=7.69
aux=sqrt(b)
do i=1,10000
  a=aux*i*i+a
enddo
```

It works even faster

But not of this.

```
a=0.d0
b=7.69
do i=1,10000
  a=i*i+a
enddo
a=a*sqrt(b)
```

```
do i=1,10000
  a=sqrt(b*i)*i+a
  c=sqrt(b)/sqrt(i)+c

enddo
```

Compiler will not do it!

```
do i=1,10000
  a=sqrt(b*i)*i+a
  c=sqrt(b/i)+c
enddo
```

```
do i=1,10000
  aux1=sqrt(i)
  a=aux1*i+a
` c=1.d0/aux1+c
enddo
a=a*sqrt(b)
c=c*sqrt(b)
```

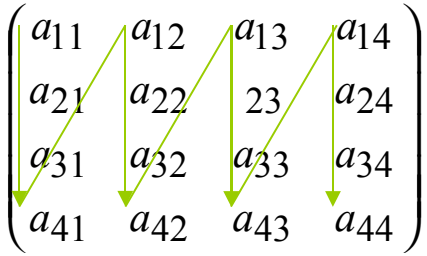
In loops, avoid repeated to simple subroutines

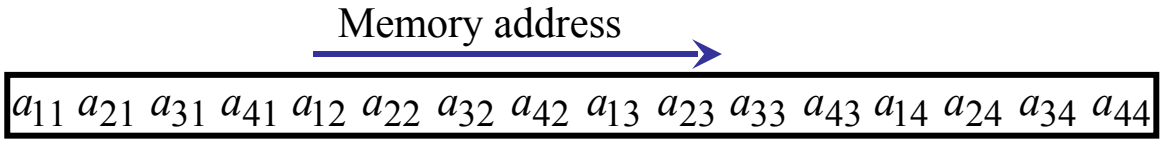
```
do i=1,10000
  call sub1 (a(i),b(i),c)
enddo
...
Subroutine sub1 (x,y,z)
x=x+y*z
return
end
```

```
do i=1,10000
  a(i)=a(i)+b(i)*c
enddo
...
```

Efficient programming: Memory access

Array storage in Fortran:





Keypoint: It is much more efficient to access data whose memory addresses are contiguous

In double loops, the inner loop should correspond to the last index of the array

It works.

```
do i=1,1000
  do j=1,1000
    a(i,j)=3.d0*i*j+2.d0/j
  enddo
enddo
```

It works much faster

```
do j=1,1000
  do i=1,1000
    a(i,j)=3.d0*i*j+2.d0/j
  enddo
enddo
```

Note: In C arrays are stored by rows!!

Matrix multiplication

It runs… very slowly

```
do i=1,n
  do j=1,n
    do k=1,n
      c(i,j)=c(i,j)+a(i,k)*b(k,j)
    enddo
  enddo
enddo
```

It runs fast

```
do j=1,n
  do k=1,n
    do i=1,n
      c(i,j)=c(i,j)+a(i,k)*b(k,j)
    enddo
  enddo
enddo
```

Multiplication of two 512x512 complex matrices on Pentium 4 Prescott @ 3GHz: 1.5 Gflops

It flyes …

```
call zgemm('N','N',512,512,512,dcmplx(1.d0,0.d0),
a,512,b,512,dcmplx(1.d0,0.d0),c,512)
```

4.2 Gflops

Using one of the Basic Linear Algebra Subroutines (BLAS) which have standard calls. Those libraries are included in Intel's Math Kernel Library.

Last but not least: Use the best compiler options

Intel compilers: Extremely efficient in Pentium 4 & Intel Core processors.

ifort: Intel Fortran 77 and Fortran 90 compiler.
icc: Intel C compiler.
idb: intel debugger.

Full documentation available locally in the intranet.

Options in **desktops**: **> ifort –xP –O3 –o myprogram.x myprogram.f**
Options in **nuredduna**: **> ifort –fast –o myprogram.x myprogram.f**

Portland Group compilers:

pgf77 (Fortran 77) pgf90 (Fortran 90) pghpf (High Performance Fortran)
pgcc (C), pgCC (C++).
pgdbg: debugger, pgprof: Performance analysis tool.

Full documentation availble locally in the intranet.

Options in **illetes/salmunia** (opteron) '-fast'

> pgf90 -fast -o myprogram.x myprogram.f

Options in **nuredduna**:

-fastsse –Mfprelaxed=rsqrt –Mipa=fast –Mipa=inline –Msmartalloc –tp core2-64

gnu compilers (gcc,g++,g77) are also installed

Still to be done …

- Only a few nuredduna nodes are working. It is necessary to rewire nuredduna shelves.
- Air conditioning in cluster room does not work.
- Printers have not been fully configured.
- Update intranet information with the new configuration.