www.bsc.es

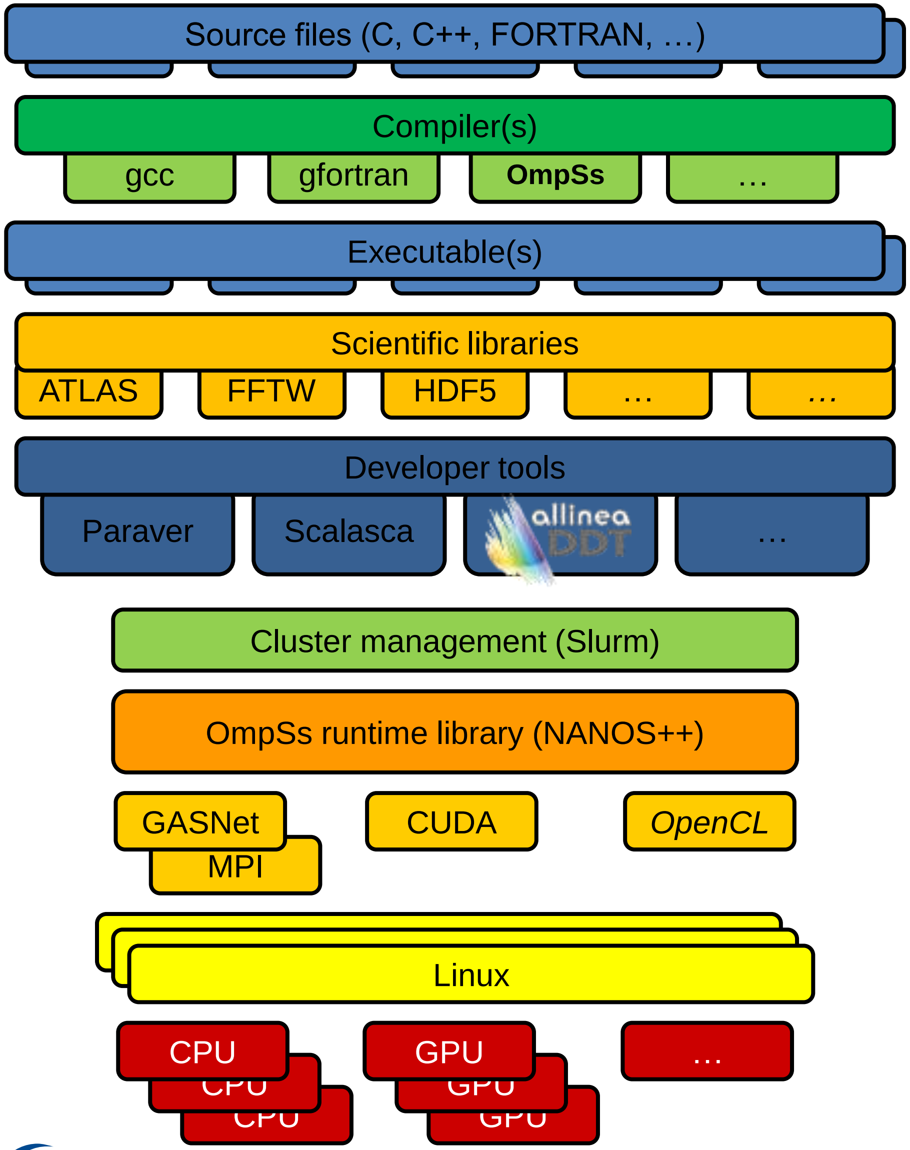
BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tutorial: ARM HPC software stack

PRACE Spring School 2013
New and Emerging Technologies - Programming for Accelerators

Nikola Rajovic, Gabriele Carteni
Barcelona Supercomputing Center

System software stack ready.



- Open source system software stack
  - Ubuntu/Debian Linux OS
  - GNU compilers
    - gcc, g++, gfortran
  - Scientific libraries
    - ATLAS, FFTW, HDF5,...
  - Slurm cluster management
- Runtime libraries
  - MPICH2, CUDA, …
  - OmpSs toolchain
- Developer tools
  - Paraver, Scalasca
  - Allinea DDT debugger

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

ARM HPC SOFTWARE STACK COMPILERS

Compilers (1)

- Our ARM systems utilize GNU compiler suite
  - gcc
  - gfortan
  - g++

- Compilers are installed from source
  - We want to tune everything to get maximum performance
  - Reduce compilation time from the ones from default repositories

- Compilers available in different Linux distributions (repositories) usually have some of ARM specific options enabled by default
  - can badly influence the performance tuning if specific platform flags are not passed
  - Even worse if the entire Linux distribution and kernel are not properly built – performance issues

Compilers (2) – architecture and processor specific

- GCC ARM specific options
  - **-march=arm*** - tells the compiler what kind of instructions can emit when generating assembly code
    - Used mainly for binary portability across different ARM platforms
    - **-march=armv7-a** for Cortex-A9 based mobile SoCs
  - **-mcpu=name** – target ARM processor
    - more optimized binary, reduced binary portability
    - **-mcpu=cortex-a9**
  - -**mtune=name** – target ARM processor
    - Produces even more optimized binary
    - **-mtune=cortex-a9**
    - Often used together with –mcpu

Compilers(3) – floating point – ABI

- **-mfloat-abi**={soft,softfp,hard}
  - **soft** – generates binary with library calls for floating point emulation
    - lots of ARM based SoC did not use to include dedicated hardware for floating-point operations
  - **softfp** – allows the generation of code using the hardware floating-point instructions, but still uses soft-float calling convention
    - Binaries compiled against soft ABI can be executed and will benefit from dedicated hardware.
    - Not back compatible
  - **hardfp** – allows generation of floating-point instructions and uses FPU-specific calling convention
    - Noticeable improvement in floating-point performance compared to softfp
    - Not back compatible
  - Tegra2 (*hands-on*) uses **softfp**

Compilers(4) – floating-point hardware

- **-mfpu=**{*specific_hardware_implementation*}

- **neon**
  - SIMD engine
  - single precision (announced double precision in ARMv8)
  - not fully IEEE754 compliant

- **vfpv3-d16**
  - true double precision floating point unit
  - available in all our prototypes (*hands-on*)

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

ARM HPC SOFTWARE STACK
RUNTIME AND SCIENTIFIC LIBRARIES

Runtime libraries

- Message-passing libraries
  - Available on all prototypes (/gpfs/LIBS/BIN)
    - OpenMPI
    - MPICH2
- Accelerator runtimes
  - CUDA on ARM (available on small ARM cluster)
    - no native ARM compilation support yet
  - OpenCL (recently available for MontBlanc project)

- NANOS++ runtime
  - OmpSS programming model support (/gpfs/LIBS/BIN)

Scientific libraries

- ATLAS
  - auto-tuned linear algebra library
  - It took a month to make it compile and optimize it for our first platform
  - DGEMM routine achieves 65% efficiency (compared to 80-95% on other platforms and with vendor provided libraries)
    - no ARM provided library, so we have to live with this
- FFTW
  - Auto-tune fft library
  - Easy to port (configure; make; make install)
  - Not fully tuned due to missing cycle accurate timer during porting (limited to optimizations using 1uS timer)
- HDF5
  - large numerical data management library
  - Easy to port (configure; make; make install)

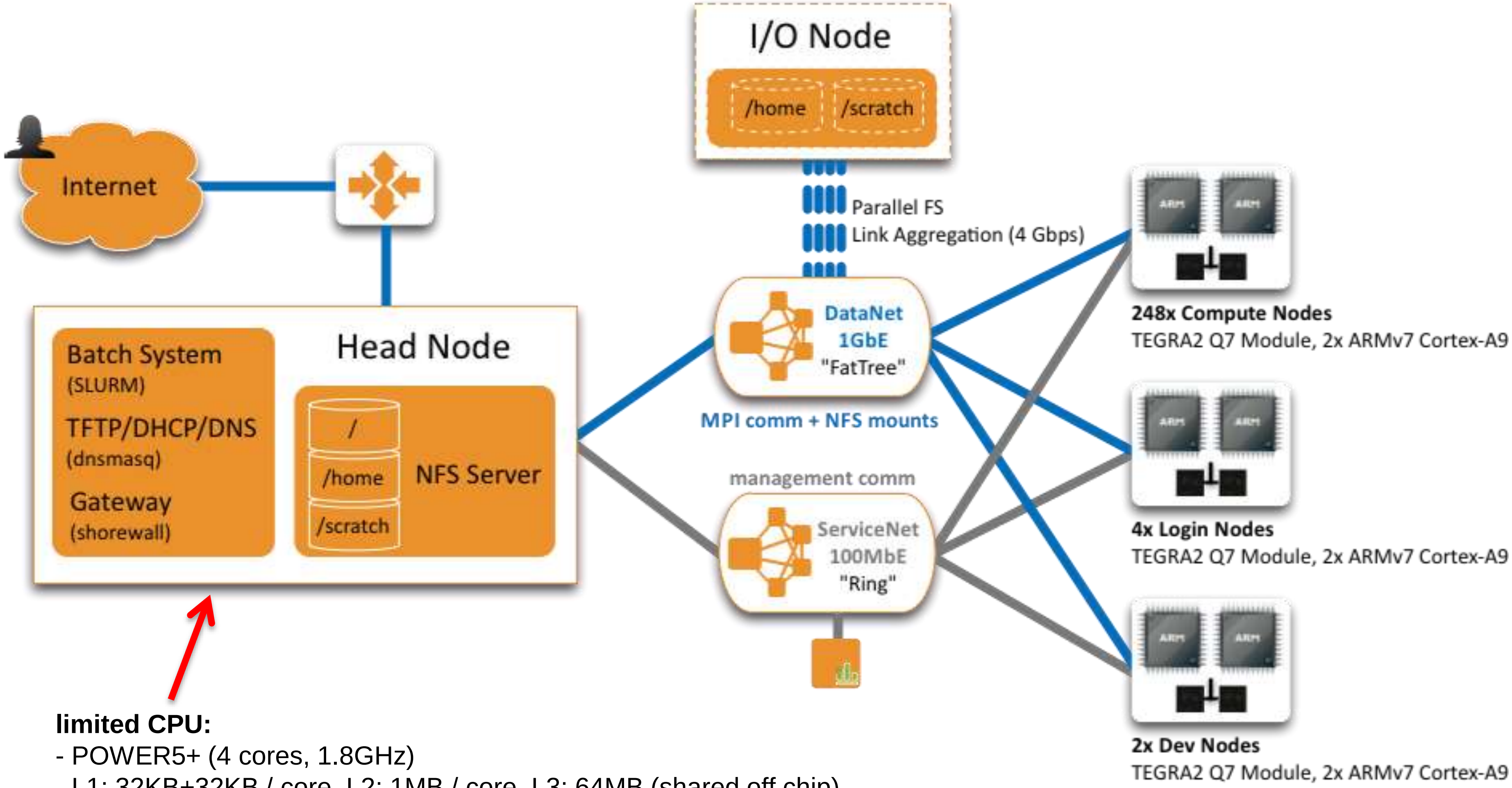
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

ARM HPC SOFTWARE STACK
SYSTEM SOFTWARE, SYSTEM ARCHITECTURE, JOB SCHEDULER, SOFTWARE ENV MANAGEMENT

System Software Stack

- Operating System (GNU/Linux)
  - Head Node: Debian 6.0.4 “squeeze”, release 2012
  - Compute Nodes: Ubuntu Server 10.10
    - Old release (5 new versions were released in the meantime)
    - First one with support for ARM processors
    - netboot from the HeadNode through TFTP (image) and NFS (/, /home, /scratch)
    - OS Image is managed on the headnode with the debootstrap tool
- Cluster Management
  - Set of scripts (script automation) developed by BSC (mainly in bash) for:
    - Account Management, NFS, sanity checks
    - “pdsh” (multithreaded remote shell) is widely used

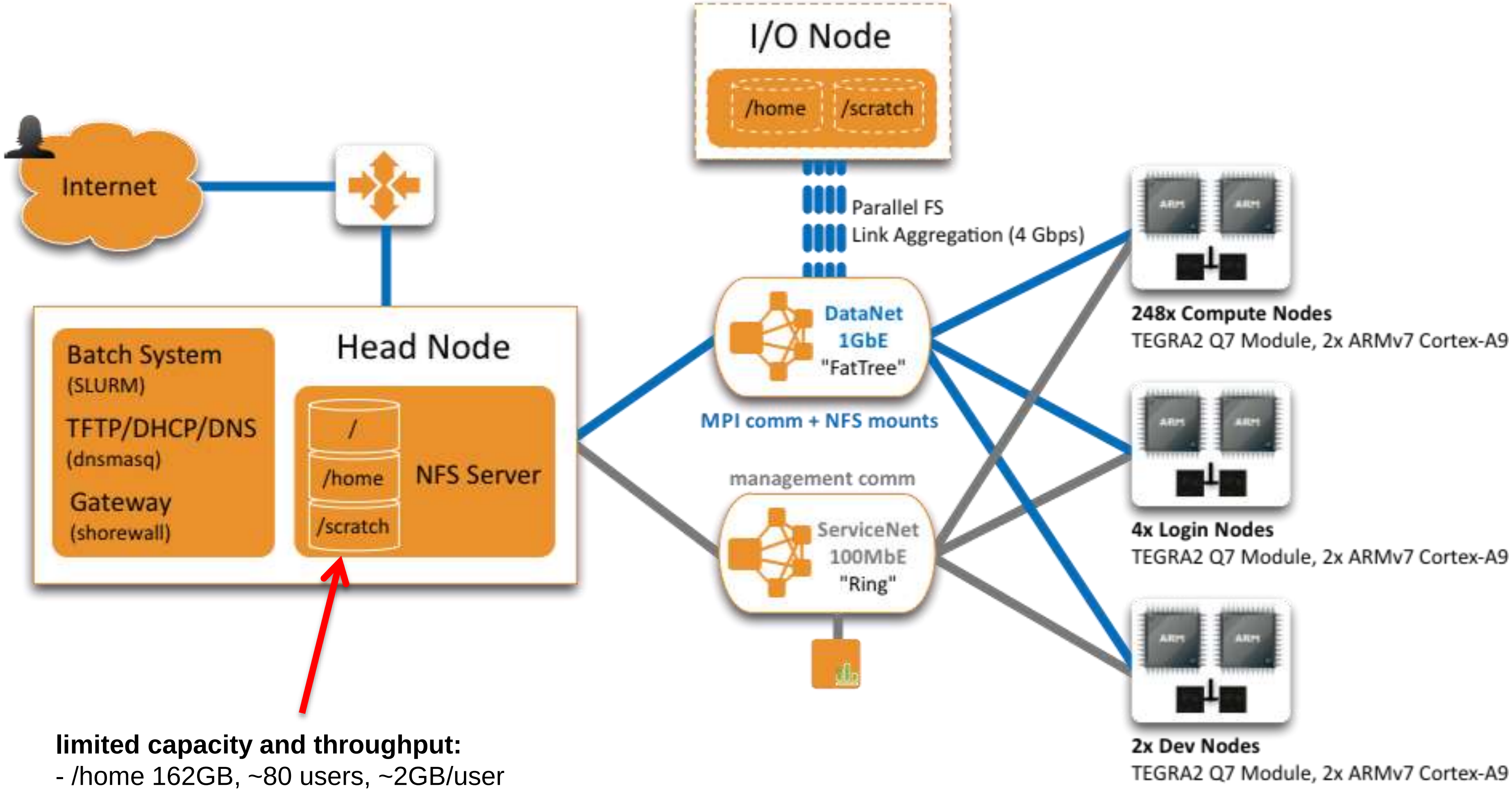
System Architecture (bottlenecks)



limited CPU:

- POWER5+ (4 cores, 1.8GHz)
- L1: 32KB+32KB / core, L2: 1MB / core, L3: 64MB (shared off chip)
- Y2005 (8 years old)

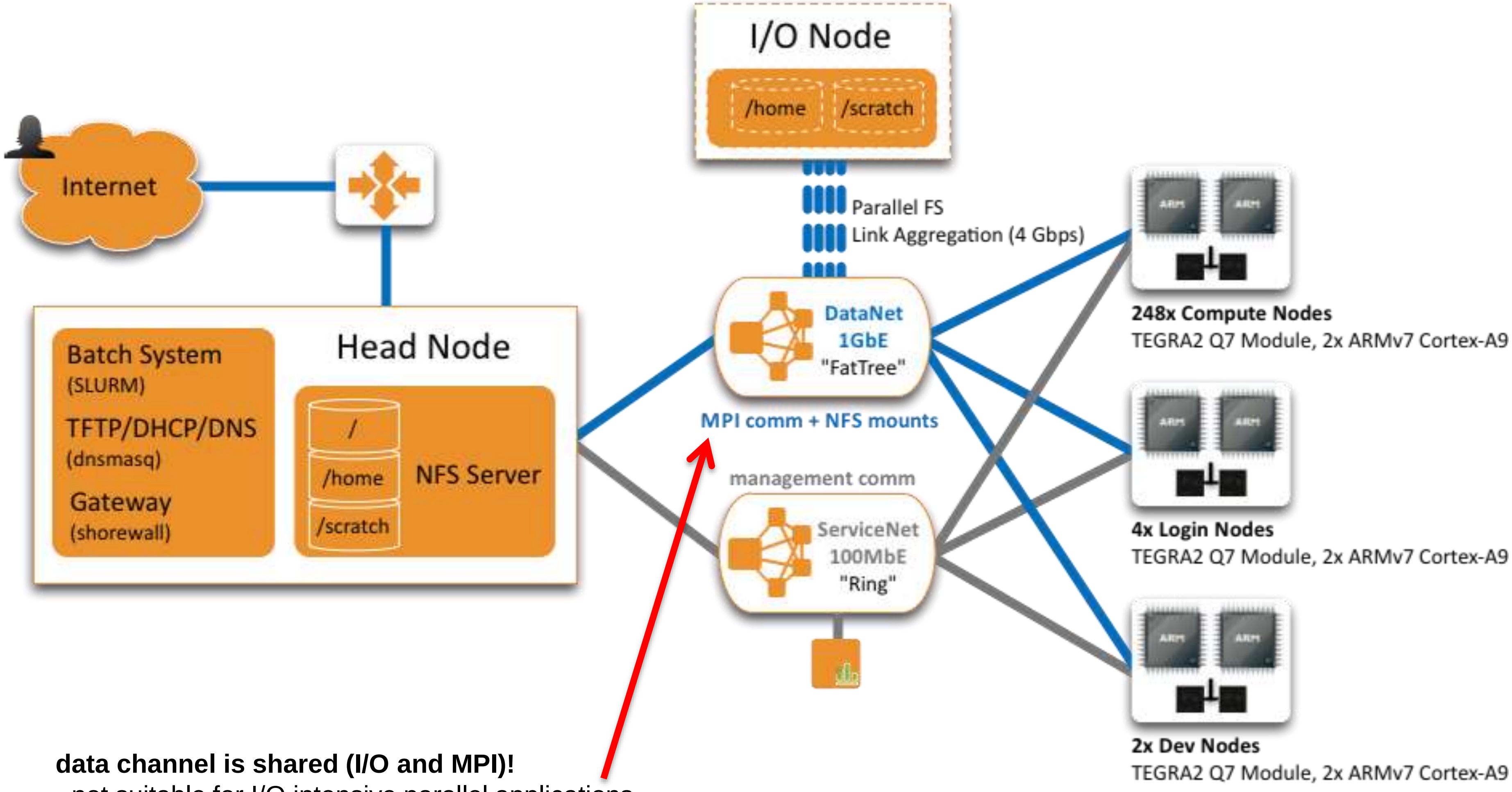
System Architecture (bottlenecks)



limited capacity and throughput:

- /home 162GB, ~80 users, ~2GB/user
- /scratch 196GB
- SCSI Disks (~ 80MB/s read, 40MB/s write)

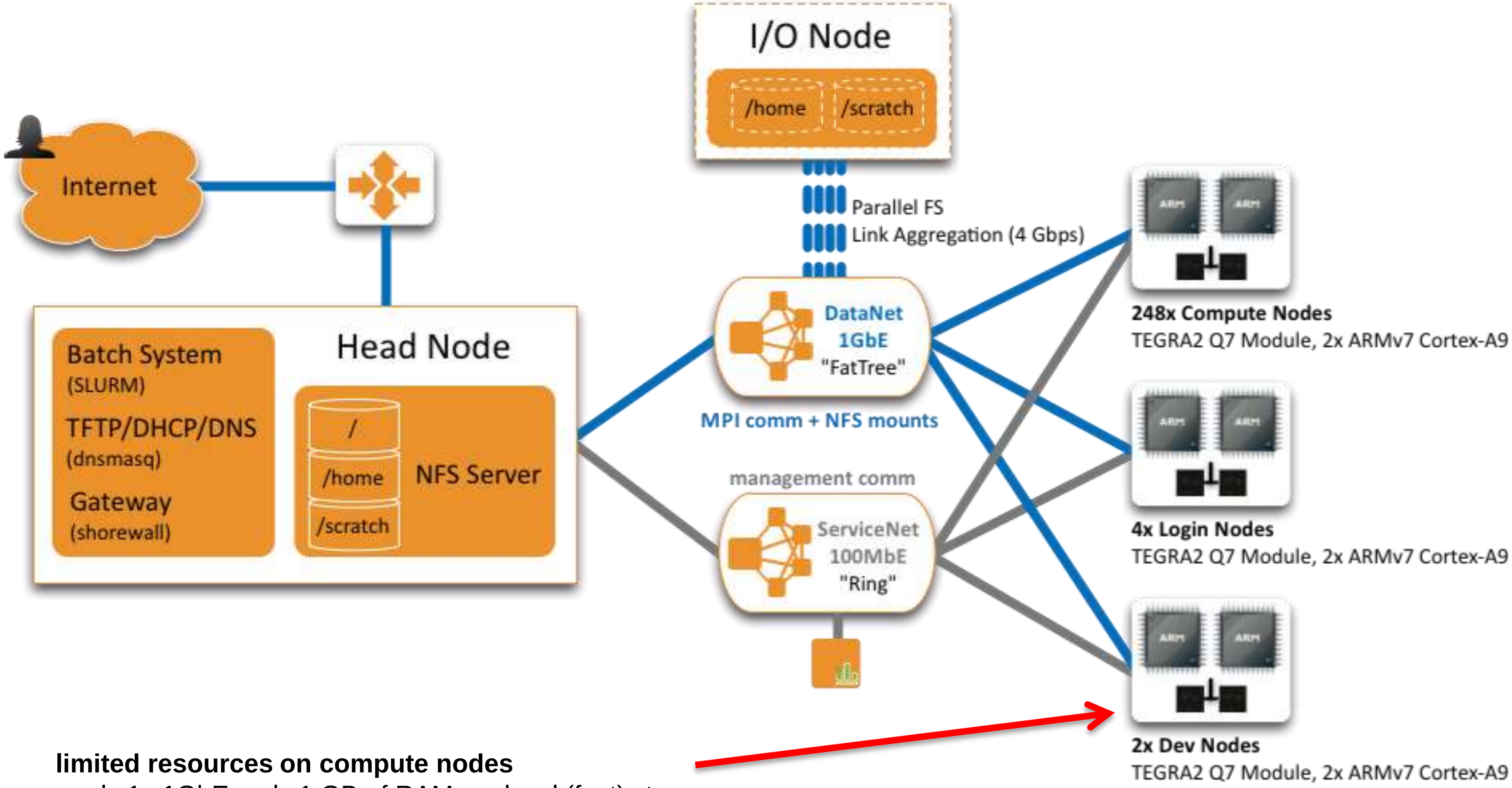
System Architecture (bottlenecks)



data channel is shared (I/O and MPI)!

- not suitable for I/O intensive parallel applications

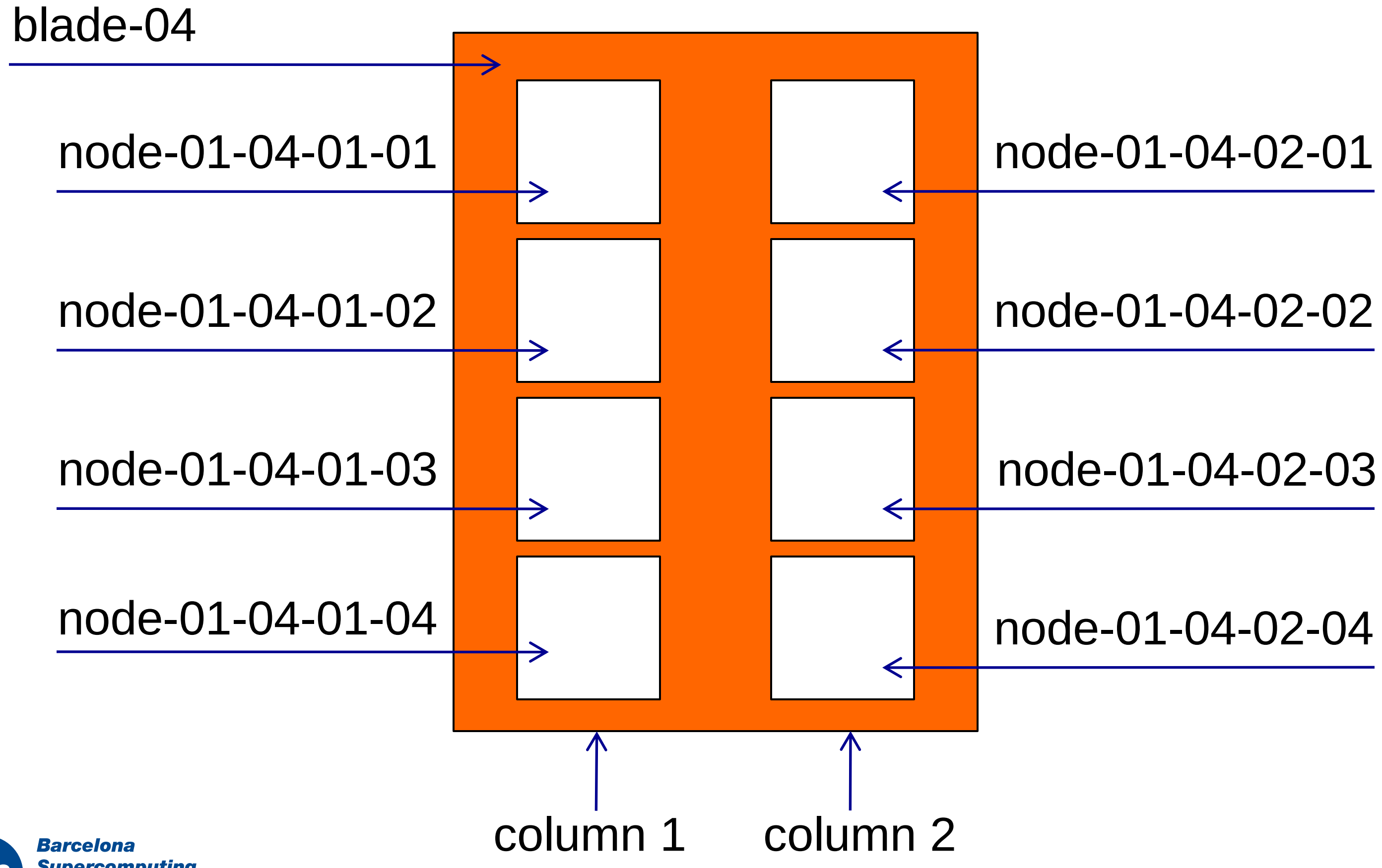
System Architecture (bottlenecks)



limited resources on compute nodes
- only 1x 1GbE, only 1 GB of RAM, no local (fast) storage

System Architecture (naming schema)

- Naming schema for compute nodes
  - node-${rr}-${bb}-${cc}-${nn} rr: rack
    - rr: rack, bb: blade, cc: column, nn: node



System Architecture (naming schema)

- Naming schema for compute nodes
  - node-${rr}-${bb}-${cc}-${nn} rr: rack
    - rr: rack, bb: blade, cc: column, nn: node
- Small exception (as usual)
  - For the 2nd rack, numeration of blades doesn’t start again:
    - node-01-16-01-01
    - node-02-17-01-01
    - node-02-18-01-01
    - …
    - node-02-31-01-01

SLURM as the Scheduler Batch System

- SLURM is opensource job scheduler and resource manager
  - designed to operate in heterogeneous clusters with up to 64k nodes and >100k of processors
  - Developed by Lawrence Livermore National Laboratory (LLNL)
  - Since 2010, maintained by SchedMD LLC
- SLURM is also a scheduler (FIFO, backfilling, GANG)
  - Uses priorities, limits (queues) and shares (users/accounts)
  - Support for Generic Resources (GPU)
  - Support for external schedulers (LSF, MOAB/MAUI)
- SLURM DB (MySQL) for accounting management
- https://computing.llnl.gov/linux/slurm/
- http://slurm.schedmd.com/

Running jobs with SLURM

- sbatch, squeue, scancel have been wrapped by:
  - **mnsubmit**, **mnq**, **mncancel** (BSC customizations for MN)
  - syntax is unchanged
- mnsubmit <myscript.job>
  - myscript.job is a bash script with directives (resources, application, etc…)
  - Syntax for directives:

```
#@directive = value
```

```
gcarteni@node-01-01-01-02:~/$ mnsubmit myscript.job
Submitted batch job 13427
```

Running jobs with SLURM

mnq

```
gcarteni@node-01-01-01-03:~$ mnq

JOBID NAME     USER      STATE    TIME  TIMELIMIT CPUS  NODES NODELIST(REASON)

1926  MyJob-1  gcarteni  RUNNING  0:03  1:00:00   16    8     node-01-02-02-[03-04],
                                                              node-01-03-01-[01-04],
                                                              node-01-03-02-01,
                                                              node-01-05-01-01

1925  MyJob-2  gcarteni  RUNNING  1:56  1:00:00    2    1     node-01-02-01-02
```

mncancel <JobId>

Running jobs with SLURM

Example of a jobscript (allocation of 8 nodes)

```
gcarteni@node-01-01-01-03:~$ cat myslurm.job
#!/bin/bash
#@ initialdir = ./
#@ job_name = MyJob
#@ class = normal
#@ output = myjob_%j.out
#@ error = myjob_%j.err
#@ wall_clock_limit = 01:00:00
#@ total_tasks = 8
#@ cpus_per_task = 2
#@ tasks_per_node = 1
module purge
module load openmpi
srun /home/gcarteni/myjobs/ompi/myopenmpi-app
```

Resources allocation
and distribution.
remember: each node has 2 CPU

Running jobs with SLURM

Example of a jobscript (allocation of 8 nodes)

```
gcarteni@node-01-01-01-03:~$ cat myslurm.job
#!/bin/bash
#@ initialdir = ./
#@ job_name = MyJob
#@ class = normal
#@ output = myjob_%j.out
#@ error = myjob_%j.err
#@ wall_clock_limit = 01:00:00
#@ total_tasks = 8
#@ cpus_per_task = 1
#@ tasks_per_node = 1
module purge
module load openmpi
srun /home/gcarteni/myjobs/ompi/myopenmpi-app
```

Resources allocation
and distribution.
remember: each node has 2 CPU

Running jobs with SLURM

Example of a jobscript (allocation of 4 nodes)

```
gcarteni@node-01-01-01-03:~$ cat myslurm.job
#!/bin/bash
#@ initialdir = ./
#@ job_name = MyJob
#@ class = normal
#@ output = myjob_%j.out
#@ error = myjob_%j.err
#@ wall_clock_limit = 01:00:00
#@ total_tasks = 8
#@ cpus_per_task = 1
#@ tasks_per_node = 2
module purge
module load openmpi
srun /home/gcarteni/myjobs/ompi/myopenmpi-app
```

Resources allocation
and distribution.
remember: each node has 2 CPU

Modules: Software Environment Management

- Tool to help users dynamically manage their Unix/Linux shell environment from switching between compilers, programs, versions, MPI implementations ...
- It usually affects:
  - PATH, LD_LIBRARY_PATH, MANPATH, FLAGS
- Available since 1990 (>20 years) it is largely used in HPC
- http://modules.sourceforge.net/

Modules: Software Environment Management

```
gcarteni@node-01-01-01-02:~$ module

    + add|load          modulefile [modulefile ...]
    + rm|unload                 modulefile [modulefile ...]
    + switch|swap               [modulefile1] modulefile2
    + display|show      modulefile [modulefile ...]
    + avail                     [modulefile [modulefile ...]]
    + purge
    + list
```

Modules: Software Environment Management

```
gcarteni@node-01-01-01-02:~$ module avail

--------- /gpfs/APPS/modules/modulefiles/compilers/ ---------

gcc/4.6.2(default)  gcc/4.6.3  gcc/4.7.0  gcc/4.7.2  gcc/4.8.0

--------- /gpfs/APPS/modules/modulefiles/environment/ ---------

mpich2/1.4.1(default)  openmpi/1.5.4
```

Modules: Software Environment Management

```
gcarteni@node-01-01-01-02:~$ module list
Currently Loaded Modulefiles:
  1) /gcc/4.6.2      2) /mpich2/1.4.1
```

Modules: Software Environment Management

```
gcarteni@node-01-01-01-02:~$ module switch mpich2 openmpi
switch1 mpich2/1.4.1 (PATH, MANPATH, LD_LIBRARY_PATH)
switch2 openmpi/1.5.4 (PATH, MANPATH, LD_LIBRARY_PATH)
ModuleCmd_Switch.c(278):VERB:4: done
gcarteni@node-01-01-01-02:~$ module list
Currently Loaded Modulefiles:
  1) /gcc/4.6.2       2) /openmpi/1.5.4
gcarteni@node-01-01-01-02:~$ module purge
remove openmpi/1.5.4 (PATH, MANPATH, LD_LIBRARY_PATH)
remove gcc/4.6.2 (PATH, MANPATH, LD_LIBRARY_PATH)
gcarteni@node-01-01-01-02:~$ module list
No Modulefiles Currently Loaded.
```

Modules: Software Environment Management

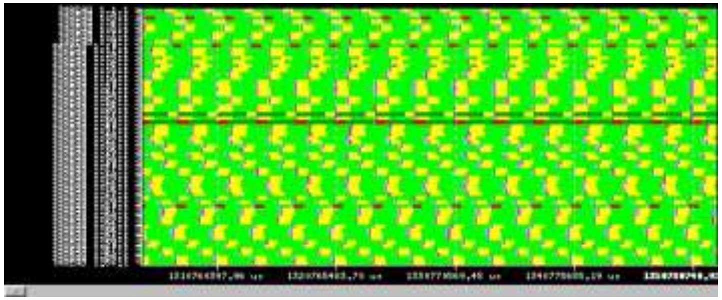
```
gcarteni@node-01-01-01-02:~$ module load openmpi
load openmpi/1.5.4 (PATH, MANPATH, LD_LIBRARY_PATH)
gcarteni@node-01-01-01-02:~$ module list
Currently Loaded Modulefiles:
  1) /openmpi/1.5.4
```

Remember, modules environment is also accessible within the job scripts.

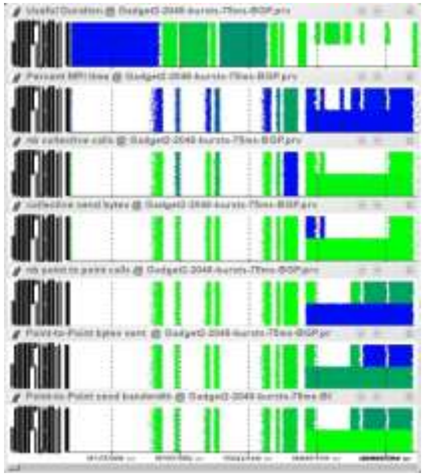
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

BSC PERFORMANCE TOOLS

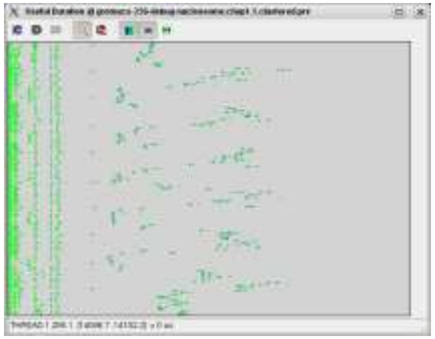
Our Tools



- Since 1991
- Based on traces
- Open Source
  - http://www.bsc.es/paraver



- Core tools:
  - Paraver (paramedir) – offline trace analysis
  - Dimemas – message passing simulator
  - Extrae – instrumentation

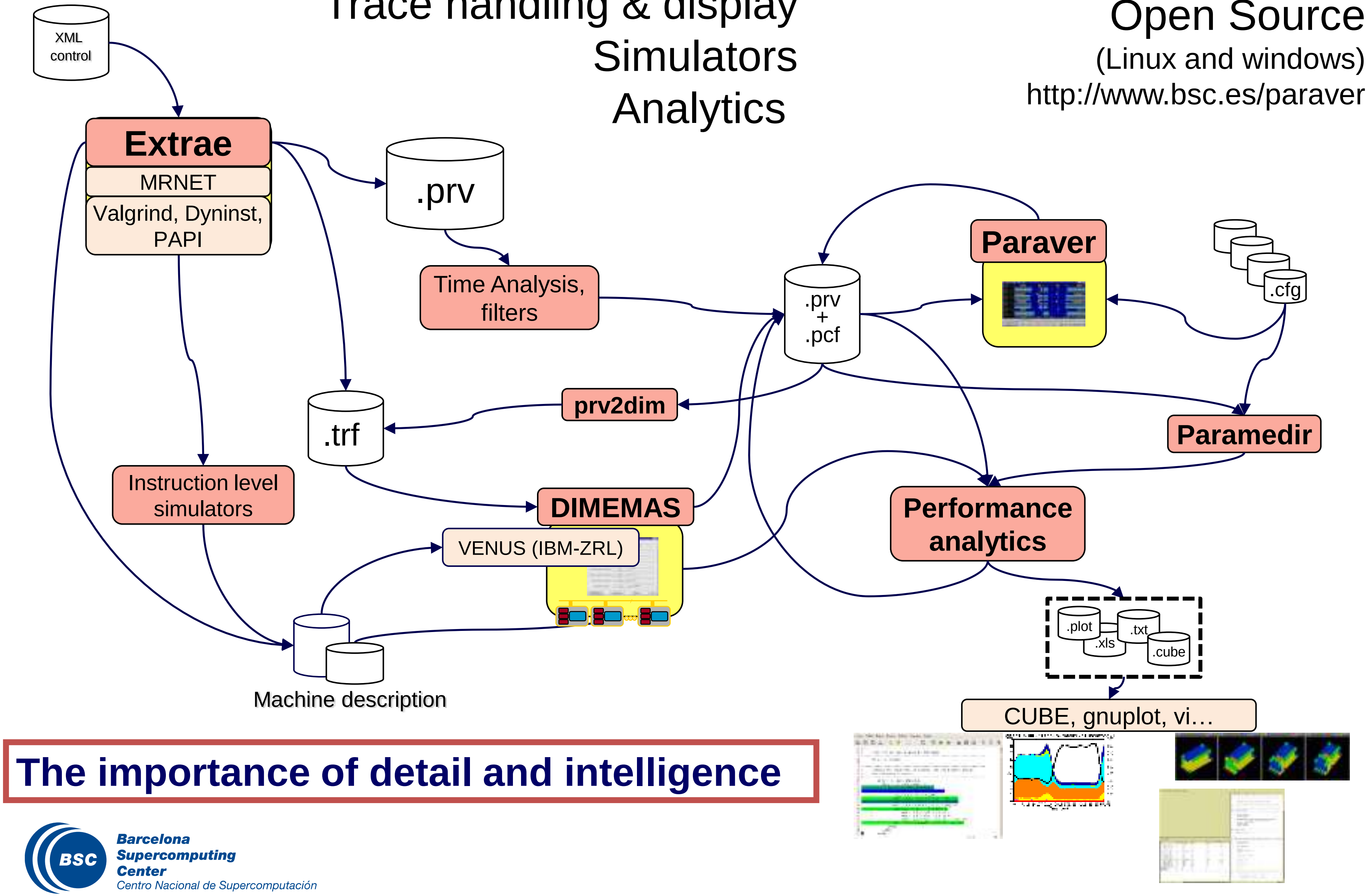


- Focus
  - Detail, flexibility, intelligence

BSC – tools framework

Trace handling & display
Simulators
Analytics

Open Source
(Linux and windows)
http://www.bsc.es/paraver



The importance of detail and intelligence

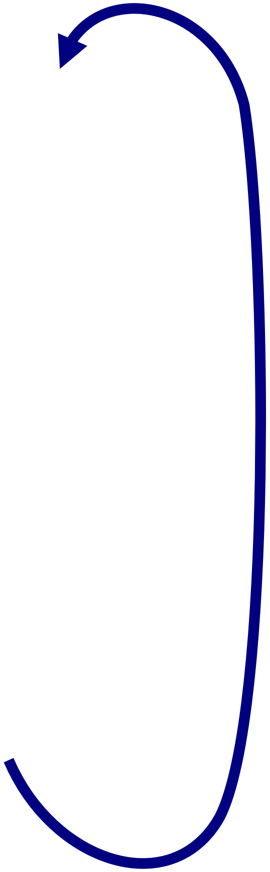
Performance analysis tools objective

Help generate hypotheses

Help validate hypotheses

Qualitatively

Quantitatively



Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

BSC PERFORMANCE TOOLS
EXTRAE

Extrae

- Parallel programming model runtime
  - MPI, OpenMP, pthreads, OmpSs, CUDA, MIC…
- Counters
  - CPU counters
    - Using PAPI and PMAPI interfaces
  - Network counters
  - OS counters
- Link to source code
  - Callstack at MPI
  - OpenMP outlined routines and their containers
  - User functions selected
- Periodic samples
- User events

How does Extrae intercepts your app?

- LD_PRELOAD
  - Specific libraries for each combination of runtimes
    - MPI
    - OpenMP
    - OpenMP+MPI
    - …

- Dynamic instrumentation
  - Based on DynInst (developed by U.Wisconsin/U.Maryland)
    - Instrumentation in memory
    - Binary rewriting

- Other possibilities
  - Link instrumentation library statically (i.e., PMPI @ BG/Q, …)
  - OmpSs (instrumentation calls injected by compiler + linked to library)

Adapt job submission script (an example)

appl.job

```
#!/bin/bash
#@total_tasks = 8
#@tasks_per_node = 2
#@cpus_per_task = 1
… … … …

./trace.sh srun parallel_app
```

trace.sh

```
#!/bin/bash

export EXTRAE_HOME=/gpfs/CEPBATOOLS/extrae/latest/openmpi/32
export EXTRAE_CONFIG_FILE=extrae.xml
export LD_PRELOAD=$EXTRAE_HOME/lib/libmpitrace.so
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${EXTRAE_HOME}/lib

$@
```

Trace control .xml

extrae.xml

```
<?xml version='1.0'?>

  <trace enabled="yes“
    home="/home/judit/tools/extrae-2.3"
    initial-mode="detail"
    type="paraver"
    xml-parser-id="Id: xml-parse.c 799 2011-10-20 16:02:03Z harald $"
  >

  <mpi enabled="yes">
    <counters enabled="yes" />
  </mpi>

  <openmp enabled="no">
    <locks enabled="no" />
    <counters enabled="yes" />
  </openmp>

  <callers enabled="yes">
    <mpi enabled="yes">1-3</mpi>
    <sampling enabled="no">1-5</sampling>
  </callers>

…
```

Activate MPI tracing and emit hardware counters at MPI calls

Do not activate OpenMP tracing

Emit call stack information (number of levels) at acquisition points

Details in $EXTRAE_HOME/share/example/MPI/extrae_explained.xml

Trace control .xml (cont)

extrae.xml (cont)

```
…

  <counters enabled=“no">

    <cpu enabled="yes" starting-set-distribution="1">
      <set enabled="yes" domain="all" changeat-globalops="5">
        PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L2_DCM
        <sampling enabled="no" frequency="100000000">PAPI_TOT_CYC
      </set>
      <set enabled="yes" domain="user" changeat-globalops="5">
        PAPI_TOT_INS,PAPI_FP_INS,PAPI_TOT_CYC
      </set>
    </cpu>

    <network enabled=“no" />

    <resource-usage enabled=“no" />

    <memory-usage enabled="no" />

  </counters>

…
```

Emit counters or not

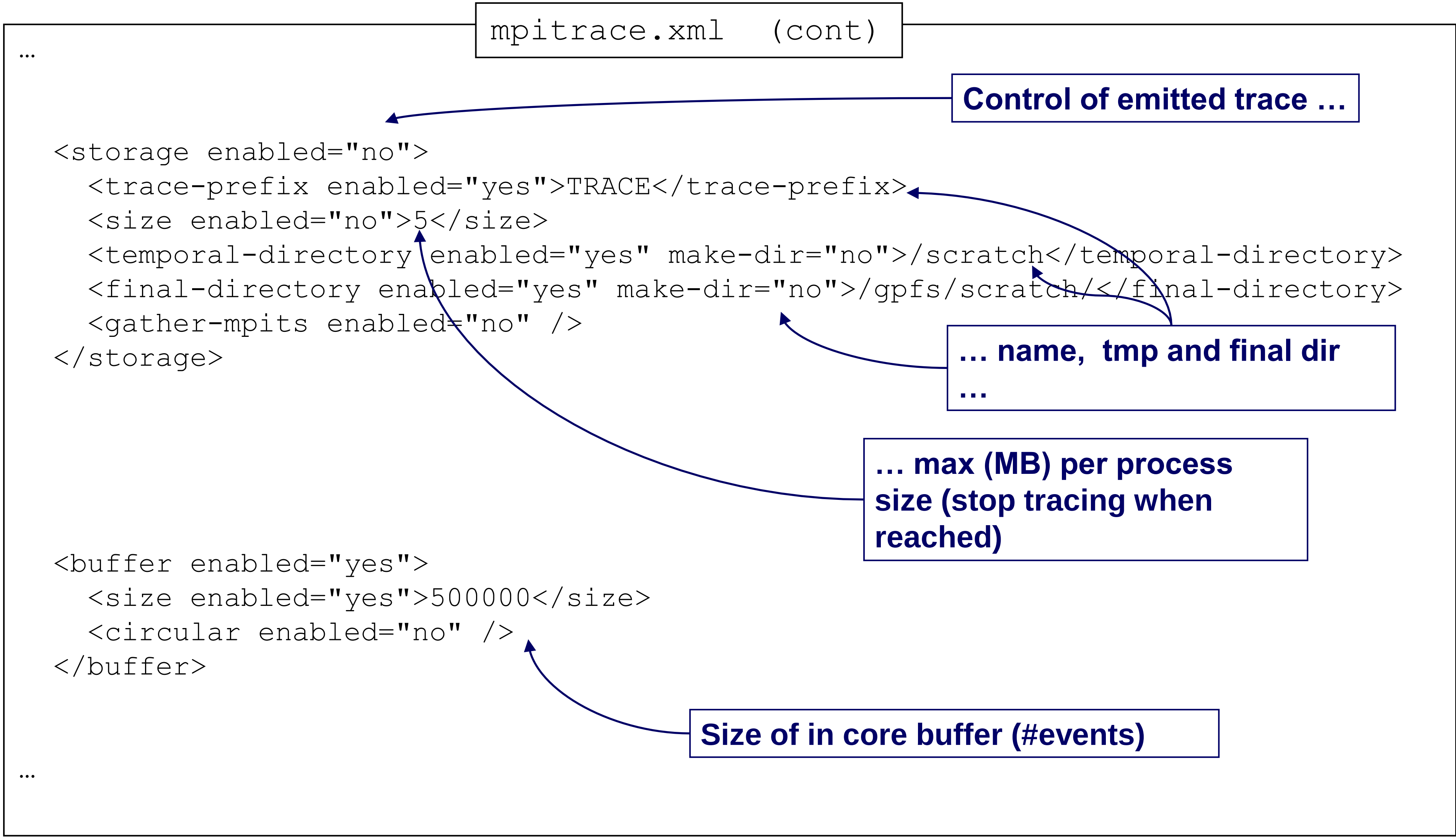
When to rotate between groups

Groups

Interconnection network counters
Just at end of trace because of large acquisition overhead

OS info (context switches,….)

Trace control .xml (cont)



mpitrace.xml (cont)

```
…


  <storage enabled="no">
    <trace-prefix enabled="yes">TRACE</trace-prefix>
    <size enabled="no">5</size>
    <temporal-directory enabled="yes" make-dir="no">/scratch</temporal-directory>
    <final-directory enabled="yes" make-dir="no">/gpfs/scratch/</final-directory>
    <gather-mpits enabled="no" />
  </storage>


  <buffer enabled="yes">
    <size enabled="yes">500000</size>
    <circular enabled="no" />
  </buffer>

…
```

Control of emitted trace …

… name, tmp and final dir …

… max (MB) per process size (stop tracing when reached)

Size of in core buffer (#events)

Trace control .xml (cont)

mpitrace.xml   (cont)

```
…

  <trace-control enabled=“no">
    <file enabled="no" frequency="5M">/gpfs/scratch/bsc41/bsc41273/control</file>
    <global-ops enabled="no"></global-ops>
     <remote-control enabled="no">
      <signal enabled="no" which="USR1"/>
    </remote-control>
  </trace-control>



  <others enabled=“no">
    <minimum-time enabled="no">10M</minimum-time>
    <terminate-on-signal enabled="no">USR2</terminate-on-signal>
  </others>

…
```

External activation of tracing (creation of file will start tracing)

Stop tracing after elapsed time …

… or when signal received

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Trace control .xml (cont)

mpitrace.xml (cont)

```
…

  <merge enabled="yes"
    synchronization="default"
    binary="$EXE$"
    tree-fan-out="16"
    max-memory="512"
    joint-states="yes"
    keep-mpits="yes"
    sort-addresses="yes"
  >
    $TRACENAME$
  </merge>


</trace>
```

Merge individual traces into global application trace at end of run …

… into this trace name

LD_PRELOAD library selection

- Library depends on programming model

Programming model	Library
Serial	libseqtrace
Pure MPI	libmpitrace[f][1]
Pure OpenMP	libomptrace
Pure Pthreads	libpttrace
CUDA	libcudatrace
MPI + OpenMP	libompitrace[f] [1]
MPI + Pthreads	libptmpitrace[f] [1]
Mpi + CUDA	libcudampitrace[f] [1]

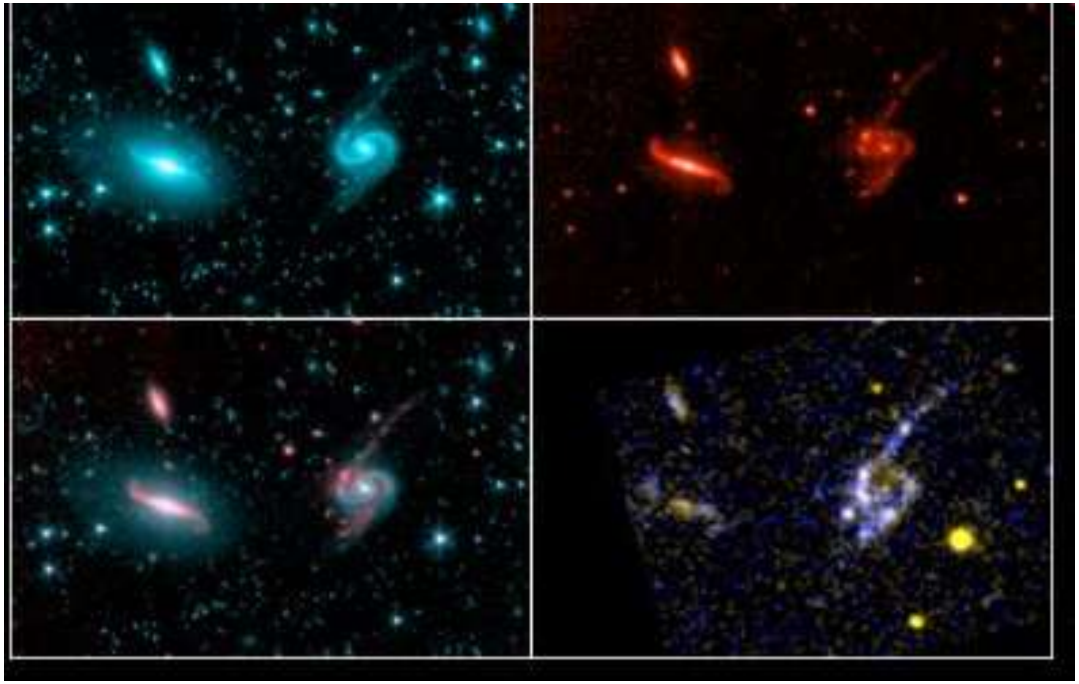
[1] for Fortran codes

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

BSC PERFORMANCE TOOLS
PARAVER

Multispectral imaging

- Different looks at one reality
  - Different spectral bands (light sources and filters)
- Highlight different aspects
  - Can combine into false colored but highly informative images



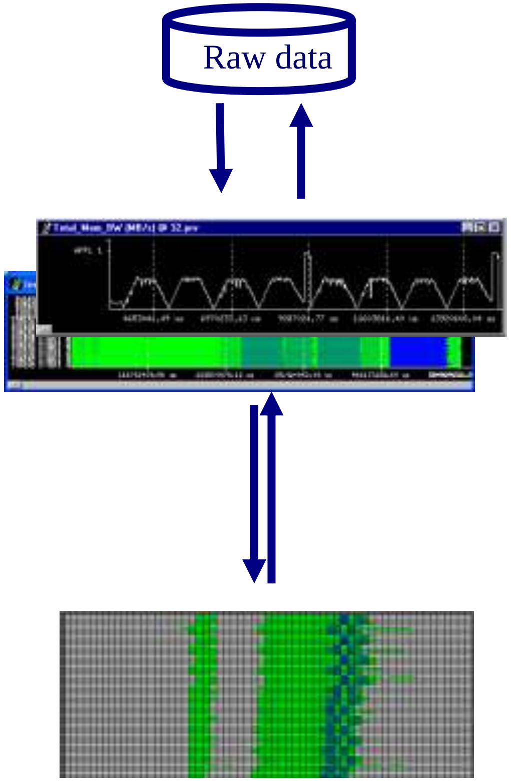
Instruments

- One experiment
  - “Expensive” resources
- Lots of analysis
- To obtain sufficient information/insight
  - Avoid flying blind
  - Identification of productive next steps

What is Paraver

- A browser …
- …to manipulate (visualize, filter, cut, combine, …) ….
- … sequences of time-stamped events …
- … with a multispectral philosophy …
- … and a mathematical foundation …
- … that happens to be mainly used for **performance analysis**

Paraver – Performance data browser



Timelines

2/3D tables (Statistics)

Trace visualization/analysis

+ trace manipulation

Goal = Flexibility

No semantics

Programmable

Configuration files

Distribution

Your own

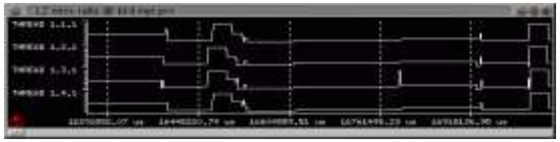
Comparative analyses

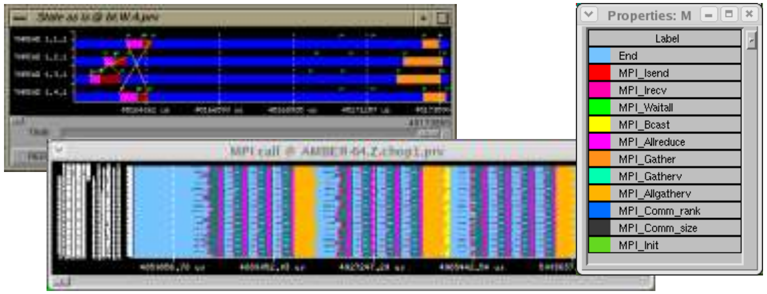
Multiple traces

Synchronize scales

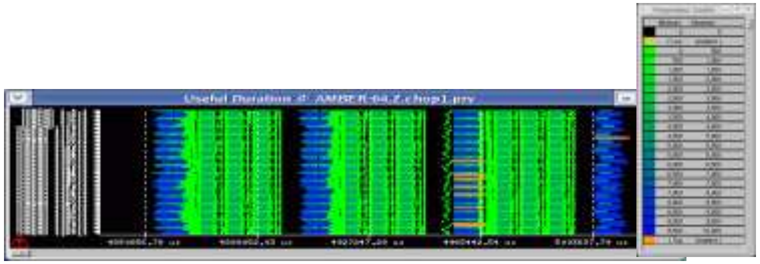
Timelines

- Representation
  - Function of time
  - Colour encoding
  - Not null gradient
    - Black for zero value
    - Light green → Dark blue



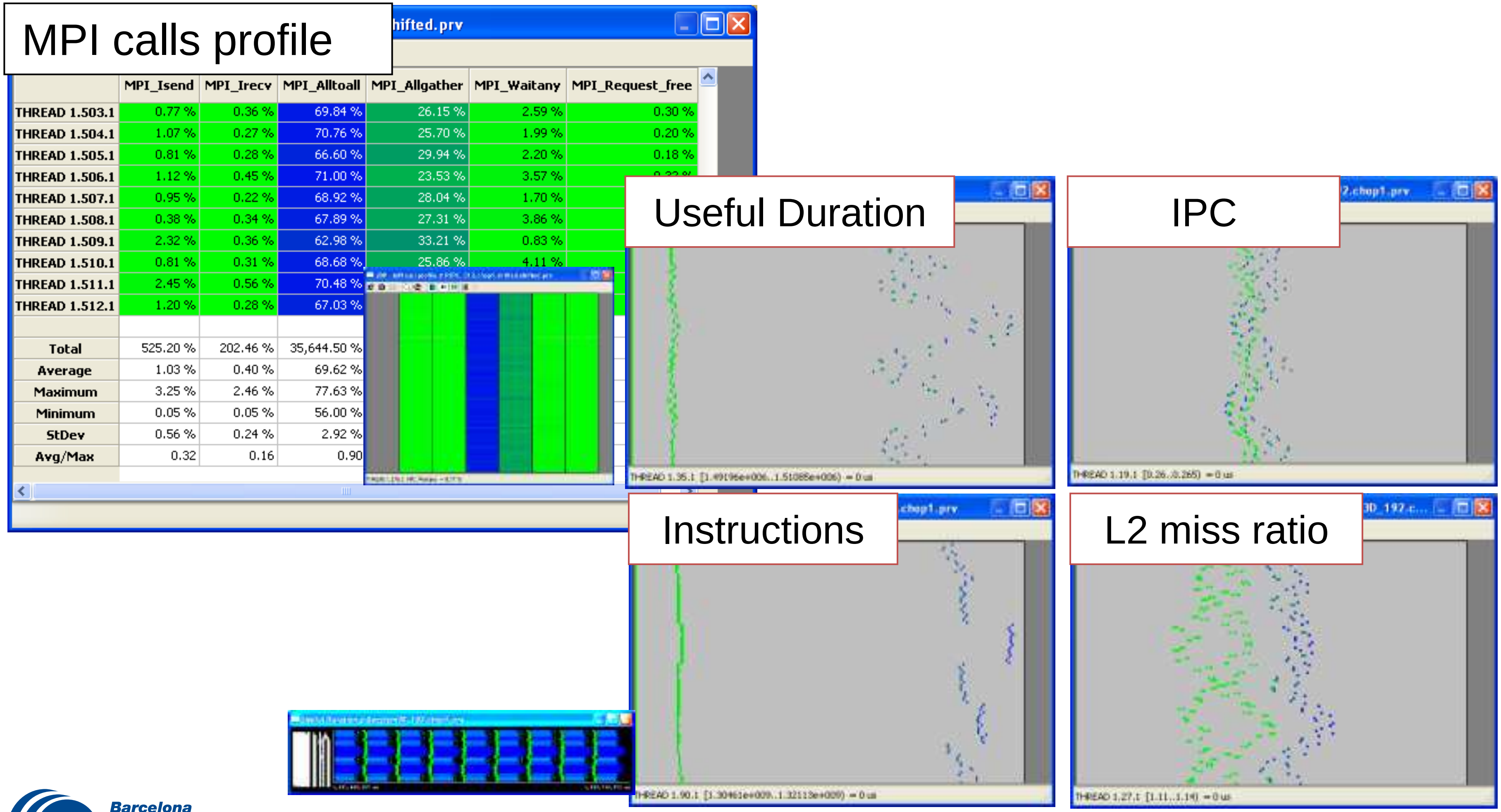


Label
End
MPI_Isend
MPI_Irecv
MPI_Waitall
MPI_Bcast
MPI_Allreduce
MPI_Gather
MPI_Gatherv
MPI_Allgatherv
MPI_Comm_rank
MPI_Comm_size
MPI_Init



Tables: Profiles, histograms

- Huge number of statistics computed from timelines



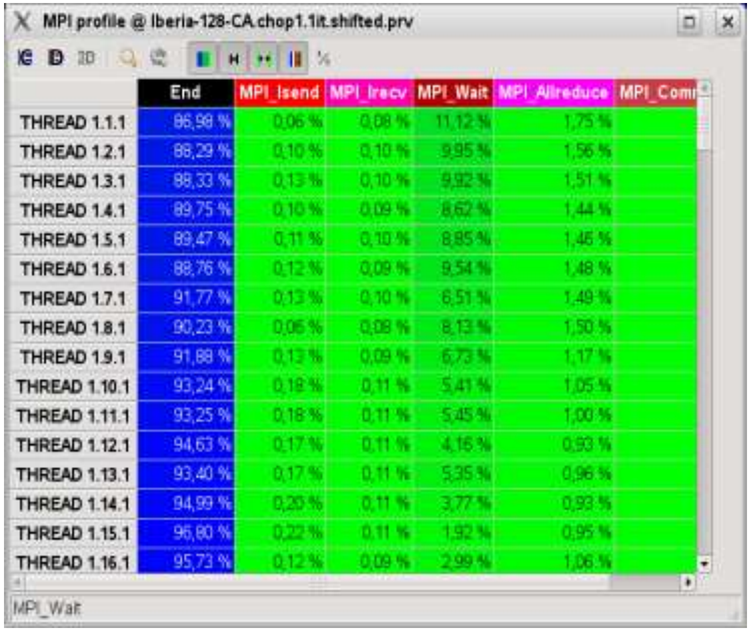
MPI calls profile
...hifted.prv
Useful Duration
IPC
...2.chop1.prv
Instructions
...chop1.prv
L2 miss ratio
...3D_192.c...
THREAD 1.35.1 [1.49196e+006..1.51088e+006) → 0 us
THREAD 1.19.1 [0.26..0.265) → 0 us
THREAD 1.90.1 [1.30461e+009..1.32113e+009) → 0 us
THREAD 1.27.1 [1.11..1.14) → 0 us

	MPI_Isend	MPI_Irecv	MPI_Alltoall	MPI_Allgather	MPI_Waitany	MPI_Request_free
THREAD 1.503.1	0.77 %	0.36 %	69.84 %	26.15 %	2.59 %	0.30 %
THREAD 1.504.1	1.07 %	0.27 %	70.76 %	25.70 %	1.99 %	0.20 %
THREAD 1.505.1	0.81 %	0.28 %	66.60 %	29.94 %	2.20 %	0.18 %
THREAD 1.506.1	1.12 %	0.45 %	71.00 %	23.53 %	3.57 %	
THREAD 1.507.1	0.95 %	0.22 %	68.92 %	28.04 %	1.70 %	
THREAD 1.508.1	0.38 %	0.34 %	67.89 %	27.31 %	3.86 %	
THREAD 1.509.1	2.32 %	0.36 %	62.98 %	33.21 %	0.83 %	
THREAD 1.510.1	0.81 %	0.31 %	68.68 %	25.86 %	4.11 %	
THREAD 1.511.1	2.45 %	0.56 %	70.48 %			
THREAD 1.512.1	1.20 %	0.28 %	67.03 %			
Total	525.20 %	202.46 %	35,644.50 %			
Average	1.03 %	0.40 %	69.62 %			
Maximum	3.25 %	2.46 %	77.63 %			
Minimum	0.05 %	0.05 %	56.00 %			
StDev	0.56 %	0.24 %	2.92 %			
Avg/Max	0.32	0.16	0.90			

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

How to read profiles

One columns per specific value of categorical **Control window**

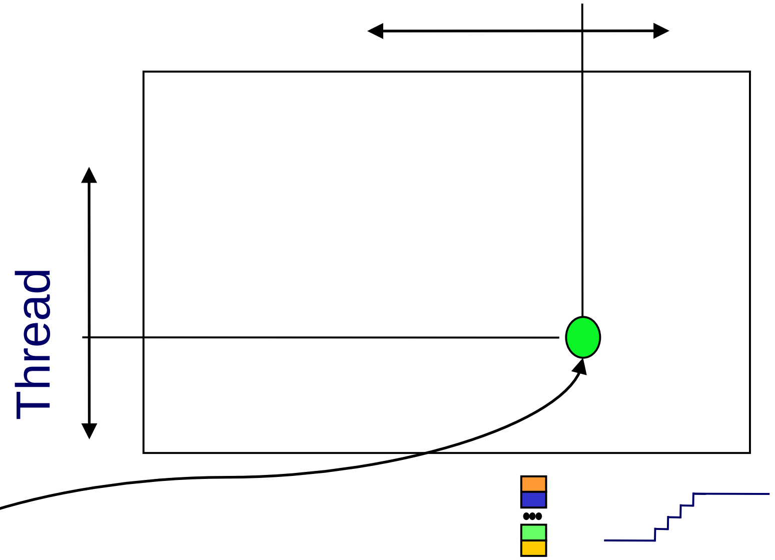


MPI profile @ Iberia-128-CA.chop1.1it.shifted.prv

	End	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Allreduce	MPI_Comm
THREAD 1.1.1	86,98 %	0,06 %	0,08 %	11,12 %	1,75 %	
THREAD 1.2.1	88,29 %	0,10 %	0,10 %	9,95 %	1,56 %	
THREAD 1.3.1	88,33 %	0,13 %	0,10 %	9,92 %	1,51 %	
THREAD 1.4.1	89,75 %	0,10 %	0,09 %	8,62 %	1,44 %	
THREAD 1.5.1	89,47 %	0,11 %	0,10 %	8,85 %	1,46 %	
THREAD 1.6.1	88,76 %	0,12 %	0,09 %	9,54 %	1,48 %	
THREAD 1.7.1	91,77 %	0,13 %	0,10 %	6,51 %	1,49 %	
THREAD 1.8.1	90,23 %	0,06 %	0,08 %	8,13 %	1,50 %	
THREAD 1.9.1	91,88 %	0,13 %	0,09 %	6,73 %	1,17 %	
THREAD 1.10.1	93,24 %	0,18 %	0,11 %	5,41 %	1,05 %	
THREAD 1.11.1	93,25 %	0,18 %	0,11 %	5,45 %	1,00 %	
THREAD 1.12.1	94,63 %	0,17 %	0,11 %	4,16 %	0,93 %	
THREAD 1.13.1	93,40 %	0,17 %	0,11 %	5,35 %	0,96 %	
THREAD 1.14.1	94,99 %	0,20 %	0,11 %	3,77 %	0,93 %	
THREAD 1.15.1	96,80 %	0,22 %	0,11 %	1,92 %	0,95 %	
THREAD 1.16.1	95,73 %	0,12 %	0,09 %	2,99 %	1,06 %	

MPI_Wait

MPI call, user function,…

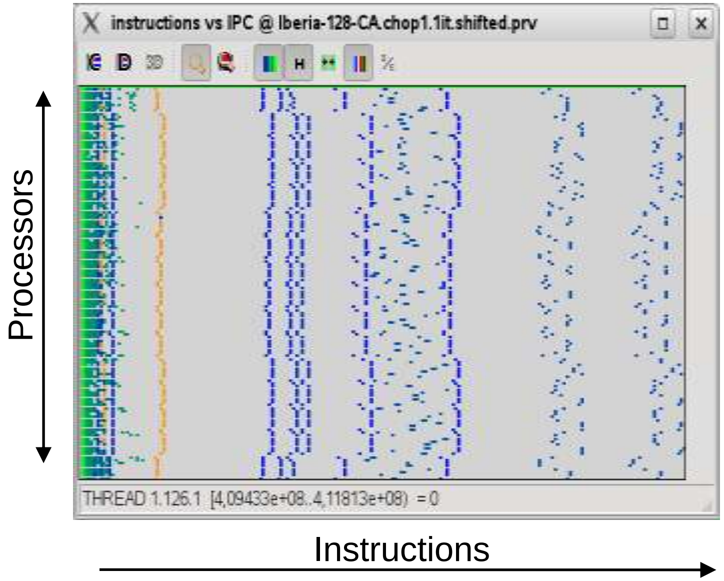


Value/color is a statistic computed for the specific thread
when control window had the value corresponding to the column

Relevant statistics:
Time, %time, #bursts, Avg. burst time
Average of **Data window**

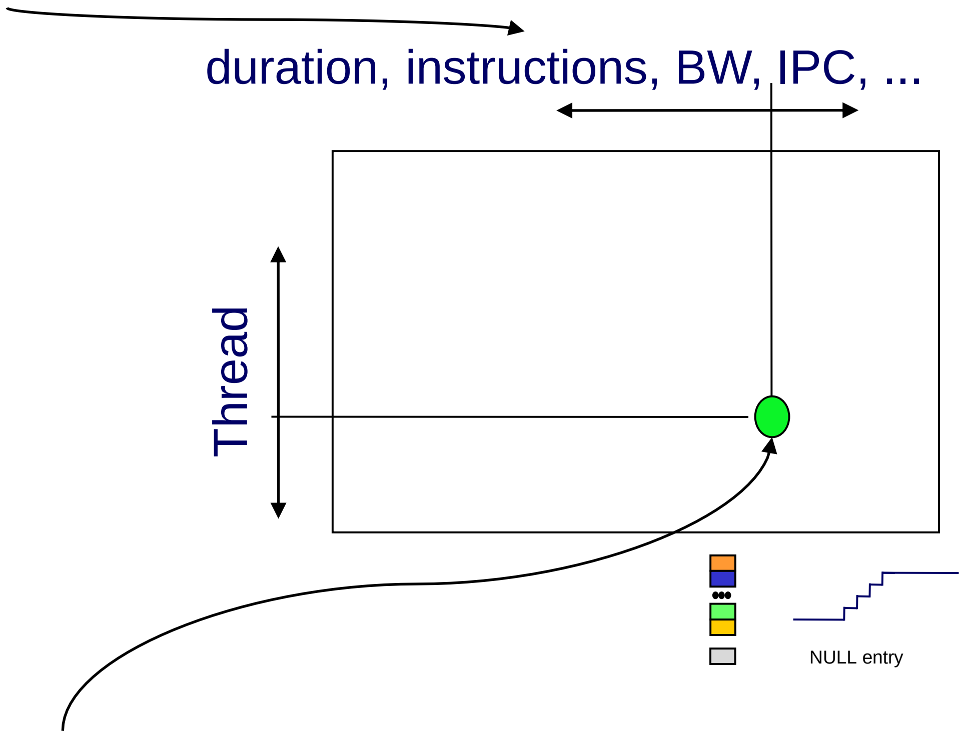
How to read histograms

Columns correspond to bins of values of a numeric **Control window**



instructions vs IPC @ Iberia-128-CA.chop1.1it.shifted.prv

THREAD 1.126.1 [4,09433e+08..4,11813e+08) = 0



duration, instructions, BW, IPC, ...

Value/color is a statistic computed for the specific thread
when control window had the value corresponding to the column

Relevant statistics:
Time, %time, #bursts, Avg. burst time
Average of **Data window**

How to learn PARAVER?

- Get a very well documented beginner tutorial with included sample trace from:
  - http://www.bsc.es/ssl/apps/performanceTools/files/docs/intro2paraver_MPI.tar.gz
  - Follow the instructions