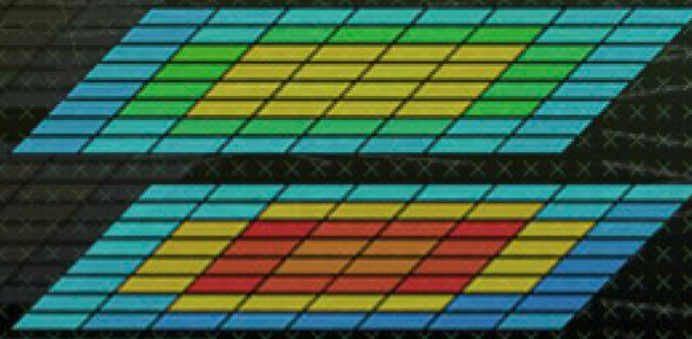


BSC Tools Hands-On

Judit Giménez, Lau Mercadal (lau.mercadal@bsc.es)

Barcelona Supercomputing Center



Extrae

Extrae features

- Parallel programming models
 - MPI, OpenMP^(*), pthreads, OmpSs, CUDA, CUPTI, OpenCL, Java, Python...
- Platforms: Intel, Cray, BlueGene, Fujitsu Sparc, MIC, ARM, Android...
- Performance Counters
 - Using PAPI and PMAPI interfaces
- Link to source code
 - Callstack at MPI routines
 - OpenMP outlined routines and their containers
 - Selected user functions
- And more: Sampling, IO, memory allocation...
- User events (Extrae API)

**No need to
recompile / relink!**

Extrae overheads

	Average values	CLAIX
Event	150-200 ns	140 ns
Event + PAPI	750 ns - 1 us	600 ns
Event + callstack (1 level)	600 ns	690 ns
Event + callstack (6 levels)	1.9 us	2.6 us

How does Extrae work?

- Symbol substitution through 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
- Static link (i.e., PMPI, Extrae API)



Recommended

Using Extrae in 3 steps

1. Adapt the job submission script
 2. [Optional] Tune the Extrae XML configuration file
 - Examples distributed with Extrae at \$EXTRAE_HOME/share/example
 3. Run with instrumentation
 - For further reference check the **Extrae User Guide:**
 - Also distributed with Extrae at \$EXTRAE_HOME/share/doc
- https://tools.bsc.es/tools_manuals

Log in and copy the examples to your home directory

```
> ssh -Y <USER>@login.hpc.itc.rwth-aachen.de  
> cp -r /home/em440321/tools-material $HOME  
> ls -l $HOME/tools-material  
...apps  
...extrae  
...slides  
...traces
```



Here you have a copy of these slides.

Step 1: Adapt the job script to load Extrae with LD_PRELOAD

```
> vi $HOME/tools-material/extrae/job.lsf
```

```
#!/usr/bin/env zsh

#BSUB -J lulesh2.0
#BSUB -o lulesh2.0.out
#BSUB -e lulesh2.0.err
#BSUB -W 00:30
#BSUB -M 512
#BSUB -n 27
#BSUB -a openmpi
#BSUB -U PPCES17
#BSUB -P hpclab

export OMP_NUM_THREADS=1

# run the script
$MPIEXEC $FLAGS_MPI_BATCH
--bind-to core --map-by core ./trace.sh
lulesh2.0 ...
```


Step 1: Adapt the job script to load Extrae with LD_PRELOAD

```
> vi $HOME/tools-material/extrae/job.lsf
```

```
#!/usr/bin/env zsh

#BSUB -J lulesh2.0
#BSUB -o lulesh2.0.out
#BSUB -e lulesh2.0.err
#BSUB -W 00:30
#BSUB -M 512
#BSUB -n 27
#BSUB -a openmpi
#BSUB -U PPCES17
#BSUB -P hpclab

export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh2.0.prv

# run the script
$MPIEXEC $FLAGS_MPI_BATCH
--bind-to core --map-by core /trace.sh
lulesh2.0 -i 10 -p -s 65
```

Step 1: Adapt the job script to load Extrae with LD_PRELOAD

```
> vi $HOME/tools-material/extrae/trace.sh
```

```
#!/usr/bin/env zsh

#BSUB -J lulesh2.0
#BSUB -o lulesh2.0.out
#BSUB -e lulesh2.0.err
#BSUB -W 00:30
#BSUB -M 512
#BSUB -n 27
#BSUB -a openmpi
#BSUB -U PPCES17
#BSUB -P hpclab

export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh2.0.prv

# run the script
$MPIEXEC $FLAGS_MPI_BATCH
--bind-to core --map-by core ./trace.sh
lulesh2.0 -i 10 -p -s 65
```

trace.sh

```
#!/bin/bash

# Configure Extrae
export EXTRAE_HOME=/rwthfs/rz/SW/unite/modules-...
source $EXTRAE_HOME/etc/extrae.sh
export EXTRAE_CONFIG_FILE=./extrae.xml

# Load the tracing library (choose C/Fortran)
export LD_PRELOAD=$EXTRAE_HOME/lib/libmpitrace.so
#export LD_PRELOAD=$EXTRAE_HOME/lib/libmpitracef.so

# Run the program
$*
```

Pick a tracing library

Step 1: LD_PRELOAD library selection

- Choose depending on the application type

Library	Serial	MPI	OpenMP	pthread	CUDA
libseqtrace	✓				
libmpitrace[f] ¹		✓			
libompitrace			✓		
libpttrace				✓	
libcudatrace					✓
libompitrace[f] ¹		✓	✓		
libptmpitrace[f] ¹		✓		✓	
libcudampitrace[f] ¹		✓			✓

¹ include suffix "f" in Fortran codes

Step 3: Run with instrumentation

- Submit your job

@ CLAIX

```
> cd $HOME/tools-material/extrae  
> bsub < job.lsf
```

Step 2: Extrae XML configuration: extrae.xml

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

Trace MPI calls + HW counters

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

```
<pthread enabled="no">  
  <locks enabled="no" />  
  <counters enabled="yes" />  
</pthread>
```

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

Trace call-stack events @ MPI calls

Step 2: Extrae XML configuration: extrae.xml (II)

```
<counters enabled="yes">
  <cpu enabled="yes" starting-set-distribution="cyclic">
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L1_DCM,PAPI_L3_TCM,
      PAPI_BR_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_BR_MSP,RESOURCE_STALLS,
      PAPI_SR_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L2_DCM,PAPI_LD_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_VEC_SP
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      ...
    </set>
  </cpu>
  <network enabled="no" />
  <resource-usage enabled="no" />
  <memory-usage enabled="no" />
</counters>
```

Define which
HW counters
are measured

Step 2: Extrae XML configuration: extrae.xml (III)

```
<buffer enabled="yes">
  <size enabled="yes">5000000</size>
  <circular enabled="no" />
</buffer>

<sampling enabled="no" type="default" period="50m" variability="10m" />

<merge enabled="yes"
  synchronization="default"
  tree-fan-out="27"
  max-memory="512"
  joint-states="yes"
  keep-mpits="yes"
  sort-addresses="yes"
  overwrite="yes"
>
  $TRACE_NAME$
</merge>
```

Trace buffer size

Enable sampling

Merge intermediate files into Paraver trace

Check the resulting trace

- After the execution you will get the trace (3 files):

@ CLAIX

```
> ls -l $HOME/tools-material/extrae
...
lulesh2.0.pcf
lulesh2.0.prv
lulesh2.0.row
```

- Any trouble? Traces already generated here:

@ CLAIX

```
> cd $HOME/tools-material/traces
```


Paraver

Installing Paraver

- Download the Paraver binaries from our website

The screenshot shows the BSC website's Downloads page. The navigation bar includes links for Home, Paraver, Dimemas, Extrae, Research, Documentation, Downloads, and Publications. The main content area is titled "Downloads" and features a "CORE TOOLS" section. Under this section, there are three tool cards: EXTRAE, PARAVAR, and DIMEMAS. Each card includes a description, a "Get" button, version and size information, and icons for supported operating systems and architectures. The PARAVAR card is highlighted, showing its version (4.6.3) and size (1.56 MB), along with icons for Linux (RAW), Windows, and various processor architectures (32-bit and 64-bit). Below the CORE TOOLS section is a "PERFORMANCE ANALYTICS" section with cards for CLUSTERING, TRACKING, and FOLDING.

news@tools:~ > Paraver

Home » Downloads

Downloads

CORE TOOLS

EXTRAE
Instrumentation framework to generate execution traces of the most used parallel runtimes.

Get EXTRAE

Version 3.4.1 • 2.24 MB

101 RAW

+

PARAVAR
Expressive powerful and flexible trace visualizer for post-mortem trace analysis.

Get PARAVAR

Version 4.6.3 • 1.56 MB

101 RAW

+

DIMEMAS
High-abstracted network simulator for message-passing programs.

Get DIMEMAS

Version 5.2.12 • 1.09 MB

101 RAW

+

PERFORMANCE ANALYTICS

CLUSTERING
Automatically expose the main performance trends in applications' computation structure.

TRACKING
Analyze how the behavior of a parallel application evolves through different scenarios.

FOLDING
Combined instrumentation and sampling for instantaneous metric evolution with low overhead.

Installing Paraver

- Or copy them from CLAIX:

@ your computer

```
> scp <USER>@login.hpc.itc.rwth-aachen.de:/home/em440321/  
tools-packages/<VERSION> $HOME
```

Pick your version

Linux 64 bits

```
wxparaver-4.6.3-linux-x86_64.tar.gz
```

Linux 32 bits

```
wxparaver-4.6.3-linux-x86.tar.gz
```

Mac

```
wxparaver-4.6.3-mac.zip
```

Windows

```
wxparaver-4.6.3-win.zip
```

Installing Paraver

- Uncompress the package into your home directory: @ your computer

```
> tar xvfz wxparaver-4.6.3-linux-x86_64.tar.gz  
> ln -s $HOME/wxparaver-4.6.3-linux-x86_64 $HOME/paraver
```

- Download Paraver tutorials and uncompress into the Paraver directory
<https://tools.bsc.es/sites/default/files/documentation/paraver-tutorials-20150526.tar.gz>

@ your computer

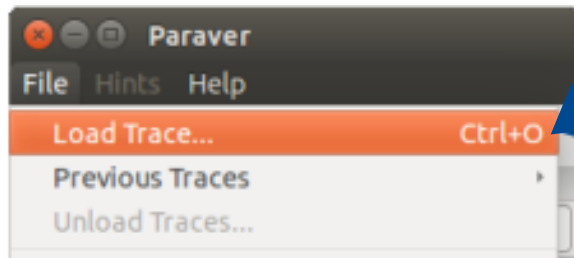
```
> tar xvfz $HOME/paraver-tutorials-20150526.tar.gz  
> mv paraver-tutorials-20150526 $HOME/paraver/tutorials
```

Check that everything works

- Start Paraver

```
> $HOME/paraver/bin/wxparaver
```

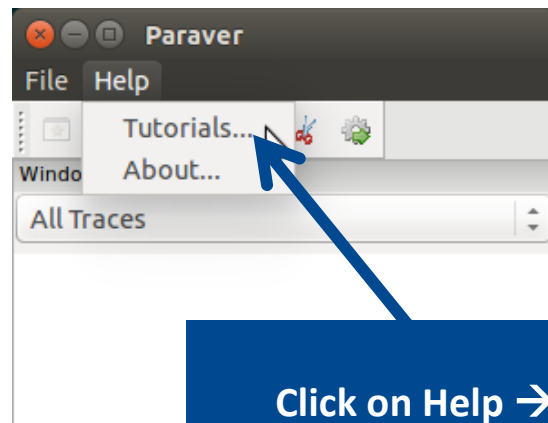
- Load the trace



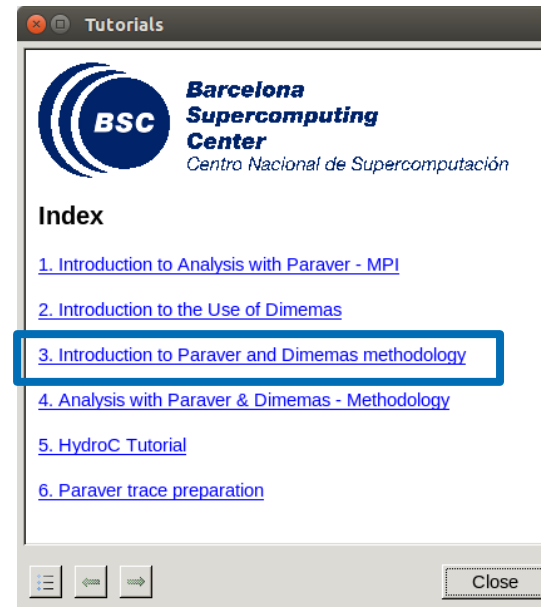
Click on File → Load Trace → Browse to
"lulesh2.0.prv"

Check that everything works

- Check that tutorials are available & follow #3



Click on Help → Tutorials



Measure the parallel efficiency

- Click on the “mpi_stats.cfg”
 - Check the Average for the column labeled “Outside MPI”

Tutorials

To measure the parallel efficiency load the configuration file [cfigs/mpi/mpi_stats.cfg](#). This configuration pops up a table with %time that every thread spends in every MPI call. Look at the global statistics at the bottom of the outside mpi column. Entry *Average* represents the application parallel efficiency, entry *Avg/Max* represents the global load balance and entry *Maximum* represents the communication efficiency. If any of those values are lower than 85% is recommended to look at the corresponding metric in detail. Open the control window to identify the phases and iterations of the code.

- To measure the computation time distribution load the configuration file [cfigs/general/2dh_usefulduration.cfg](#). This configuration pops up a histogram of the duration for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram does not show vertical lines, it indicates the computation time may be not balanced. Open the control window to look at the time distribution and visually correlate both views.
- To measure the computational load balance load the configuration file [cfigs/papi/2dh_u...](#). This configuration pops up a histogram of the instructions for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram doesn't show vertical lines, it indicates the distribution of the instructions may be not balanced. Open the control window to look at the time distribution and correlate both views.
- To measure the serial regions performance look at the IPC timeline loaded with [cfigs/general/2dh_useful...](#). This configuration would depend on the machine used to run the code. If the IPC is lower than 1 identify poor performance in the serial regions. Correlate the computation time modifying the Statistic of the useful duration histogram to use correlate with metric and verify that the selected Metric is Instructions per cycle. Now the cell color corresponds to the IPC value. The distribution of the histogram would allow you to identify different IPC. Change the Metric to Instructions per cycle.

Close

Parallel efficiency

Comm efficiency

Load balance

MPI call profile @ lulesh2.0_0.prv

THREAD 1.16.1	93.06 %	0.10 %	0.03 %	0.07 %	1.46 %	0.01 %	0.00 %
THREAD 1.17.1	92.66 %	0.11 %	0.03 %	0.06 %	1.60 %	0.01 %	0.41 %
THREAD 1.18.1	88.51 %	0.08 %	0.03 %	0.44 %	2.47 %	0.01 %	0.00 %
THREAD 1.19.1	93.19 %	0.05 %	0.02 %	0.11 %	1.43 %	0.00 %	0.00 %
THREAD 1.20.1	88.33 %	0.06 %	0.03 %	0.04 %	1.38 %	0.00 %	0.00 %
THREAD 1.21.1	86.32 %	0.04 %	0.02 %	0.17 %	1.90 %	0.00 %	1.02 %
THREAD 1.22.1	86.95 %	0.08 %	0.03 %	0.25 %	1.52 %	0.01 %	0.00 %
THREAD 1.23.1	97.69 %	0.11 %	0.04 %	0.26 %	0.79 %	0.00 %	0.00 %
THREAD 1.24.1	94.44 %	0.04 %	0.03 %	0.20 %	1.46 %	0.00 %	0.00 %
THREAD 1.25.1	95.35 %	0.03 %	0.02 %	0.06 %	1.21 %	0.01 %	0.00 %
THREAD 1.26.1	94.43 %	0.05 %	0.02 %	0.10 %	1.26 %	0.01 %	0.00 %
THREAD 1.27.1	67.84 %	0.03 %	0.01 %	0.11 %	15.33 %	0.01 %	0.00 %
Total	2,416.82 %	1.85 %	0.76 %	6.33 %	73.38 %	0.17 %	4.05 %
Average	89.51 %	0.07 %	0.03 %	0.23 %	2.72 %	0.01 %	0.15 %
Maximum	97.79 %	0.27 %	0.05 %	0.50 %	15.33 %	0.02 %	1.02 %
Minimum	67.84 %	0.02 %	0.01 %	0.04 %	0.31 %	0.00 %	0.00 %
StDev	7.70 %	0.05 %	0.01 %	0.15 %	4.24 %	0.00 %	0.31 %
Avg/Max	0.92	0.26	0.61	0.47	0.18	0.35	0.15

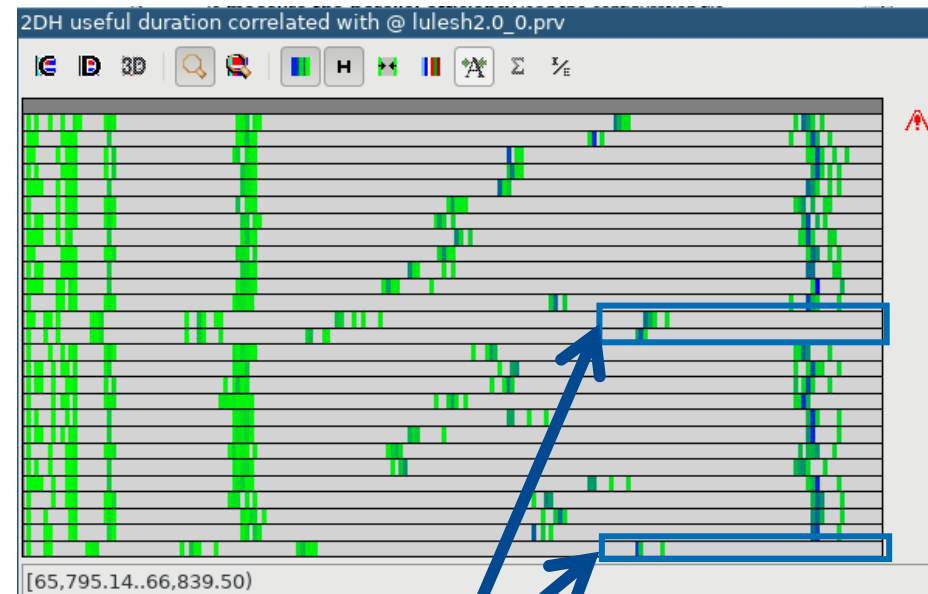
Measure the computation time distribution

- Click on the “2dh_usefulduration.cfg”

Tutorials

- To **measure the parallel efficiency** load the configuration file [cfs/mpi/mpi_stats.cfg](#) This configuration pops up a table with %time that every thread spends in every MPI call. Look at the global statistics at the bottom of the outside mpi column. Entry *Average* represents the application parallel efficiency, entry *Avg/Max* represents the global load balance and entry *Maximum* represents the communication efficiency. If any of those values are lower than 85% is recommended to look at the corresponding metric in detail. Open the control window to identify the phases and iterations of the code.
- To **measure the computation time distribution** load the configuration file [cfs/general/2dh_usefulduration.cfg](#) This configuration pops up a histogram of the duration for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram does not show vertical lines, it indicates the computation time may be not balanced. Open the control window to look at the time distribution and visually correlate both views.
- To **measure the computational load (instructions) distribution** load the configuration file [cfs/papi/2dh_useful_instructions.cfg](#) This configuration pops up a histogram of the instructions for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram doesn't show vertical lines, it indicates the distribution of the instructions may be not balanced. Open the control window to look at the time distribution and correlate both views.
- To **measure the serial regions performance** look at the IPC timeline loaded with [cfs/general/2dh_usefulduration.cfg](#). What it's a reasonable IPC would depend on the machine used to run the application, but typically values lower than 1 identify poor performance sections. You can correlate the IPC with the computation time modifying the Statistic of the useful duration histogram to use correlate with metric and verify that the selected Metric is Instructions per cycle. Now the cell color corresponds to the IPC showing the correlation between duration (position) and IPC (color). Zooming into an unbalanced region of the histogram would allow you to verify if the unbalance is related to a different IPC. Change the Metric to Instructions to correlate the duration with

Close



3 tasks run faster

Measure the computation time distribution

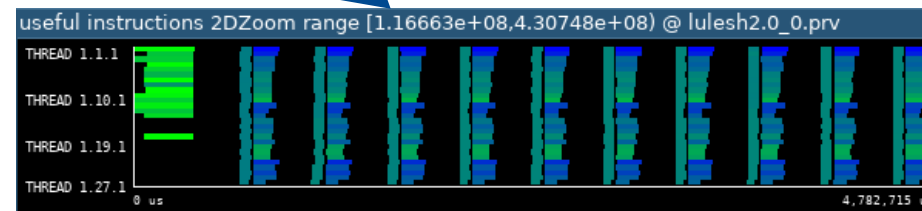
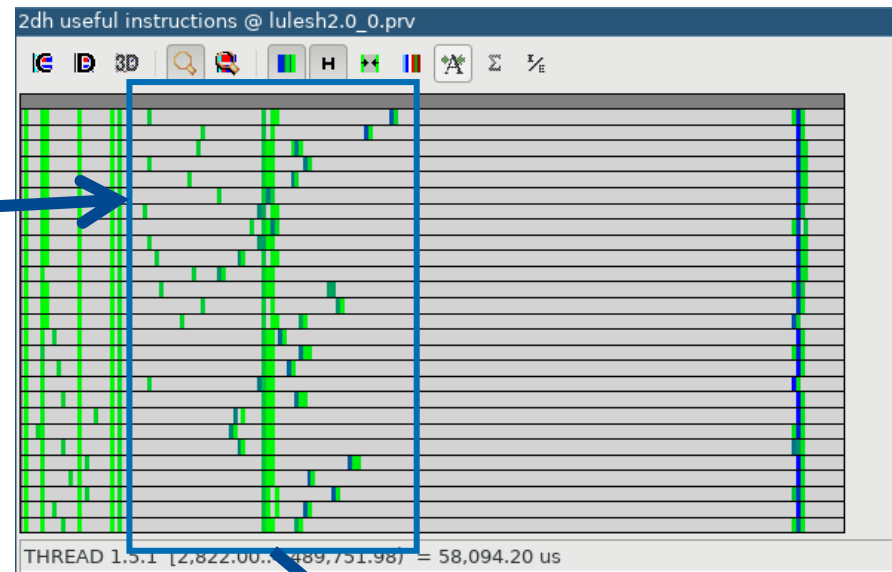
- Click on the “2dh_usefulduration.cfg”

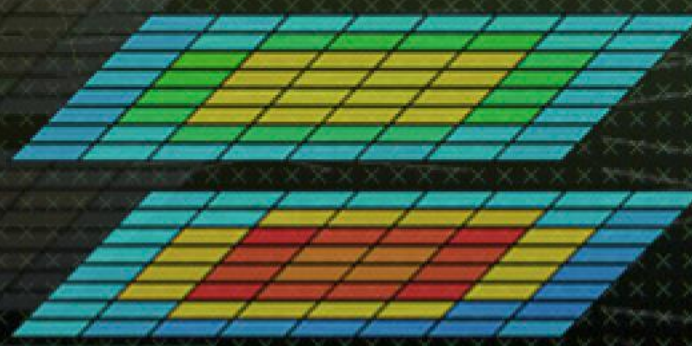
Tutorials

- To measure the parallel efficiency load the configuration file [cfs/mpi/mpi_stats.cfg](#). This configuration pops up a table with %time that every thread spends in every MPI call. Look at the global statistics at the bottom of the outside n parallel efficiency, en try values are Maximum re lower than 85 Open the cor
- To measure the duration file [cfs/general/2dh_usefulduration.cfg](#) of the duration are delimited by the exit from an MPI call and the entry to the next call. If the histogram does not show vertical lines, it indicates the computation time may be not balanced. Open the control window to look at the time distribution and visually correlate both views.
- To measure the computational load (instructions) distribution load the configuration file [cfs/papi/2dh_useful_instructions.cfg](#). This configuration pops up a histogram of the instructions for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram doesn't show vertical lines, it indicates the distribution of the instructions may be not balanced. Open the control window to look at the time distribution and correlate both views.
- To measure the serial regions performance look at the IPC timeline loaded with [cfs/general/2dh_usefulduration.cfg](#). What it's a reasonable IPC would depend on the machine used to run the application, but typically values lower than 1 identify poor performance sections. You can correlate the IPC with the computation time modifying the Statistic of the useful duration histogram to use correlate with metric and verify that the selected Metric is Instructions per cycle. Now the cell color corresponds to the IPC showing the correlation between duration (position) and IPC (color). Zooming into an unbalanced region of the histogram would allow you to verify if the unbalance is related to a different IPC. Change the Metric to Instructions to correlate the duration with

Click on “Open Filtered Control Window” and select this area

Close





Cluster analysis

Cluster-based analysis

- Run clustering

@ CLAIX

```
> cd $HOME/tools-material/clustering  
> ./clusterize.sh ../extrae/lulesh2.0.prv
```

- If you didn't get your own trace, you can use a previously generated one:

@ CLAIX

```
> cd $HOME/tools-material/traces
```

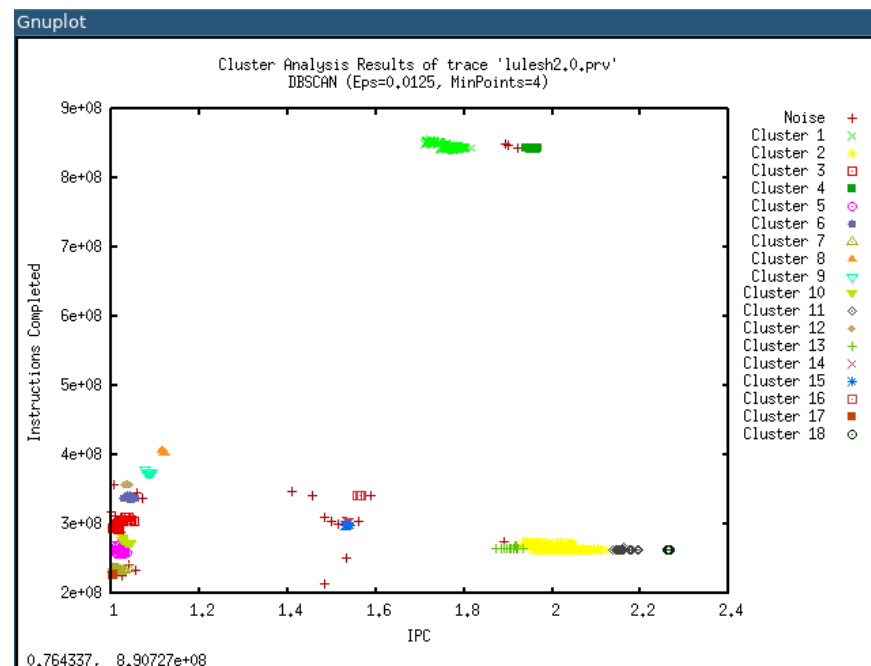
Looking at the clusters

- Check the clustering scatter plot

@ CLAIX

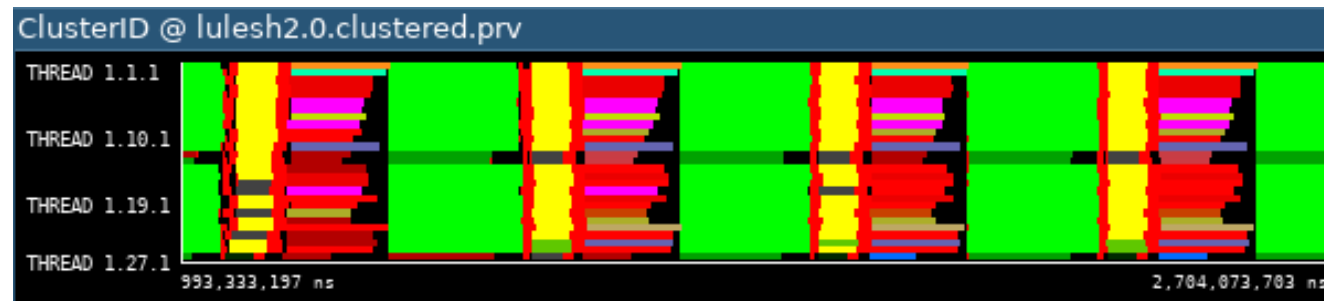
```
> gnuplot lulesh2.0.clustered.IPC.PAPI_TOT_INS.gnuplot
```

- Identify main computing trends
 - Work (Y-axis), Performance (X-axis)
- See the elongated clusters?
 - Large IPC variability
 - Wide range of instructions
 - Indicate potential imbalances



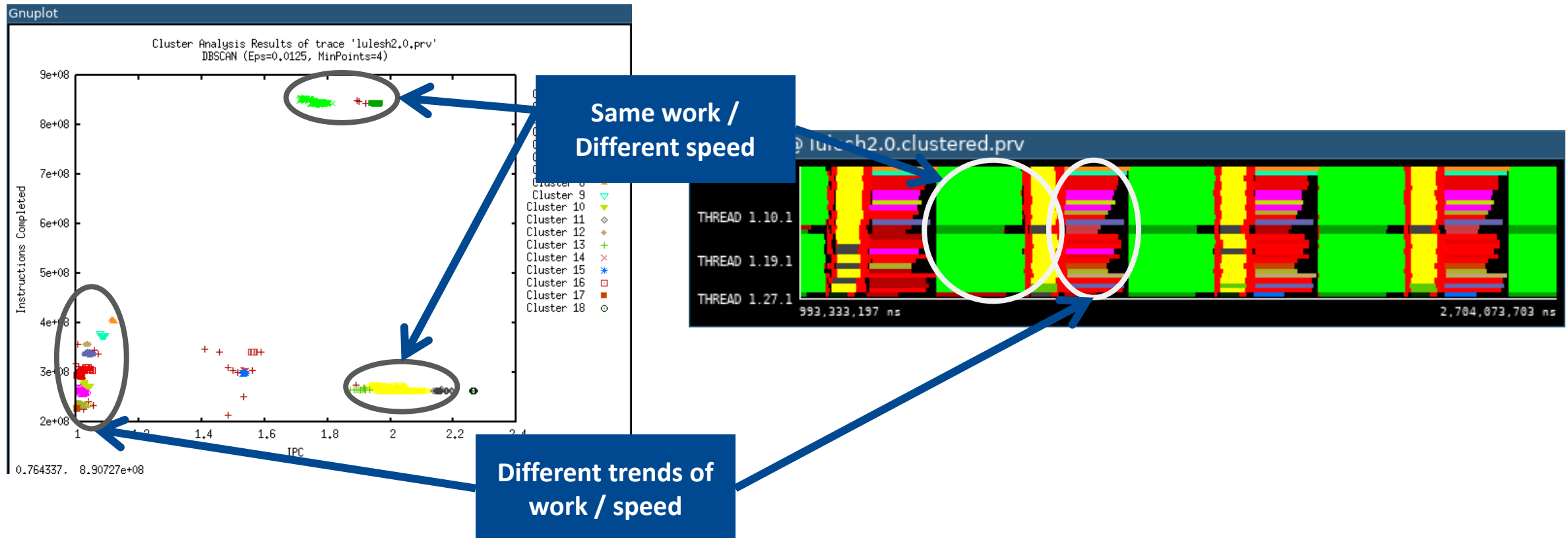
Looking at the clustered trace

- Load the clustered trace with Paraver
 - File → Load Trace → Browse to **\$HOME/tools-material/clustering/lulesh2.0.clustered.prv**
- Display the distribution of clusters over time
 - File → Load configuration → Browse to **\$PARAVER_HOME/cfgs/clustering/clusterID_window.cfg**



Looking at the clustered trace

- Correlate scatter-plots and timelines to detect imbalances



BSC Tools Hands-On

Judit Giménez, Lau Mercadal (lau.mercadal@bsc.es)

Barcelona Supercomputing Center
