Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

EXCELENCIA
SEVERO
OCHOA

Understanding applications with Paraver

Judit Gimenez
judit@bsc.es

Jan 31st 2019

CRHPCS19, San José

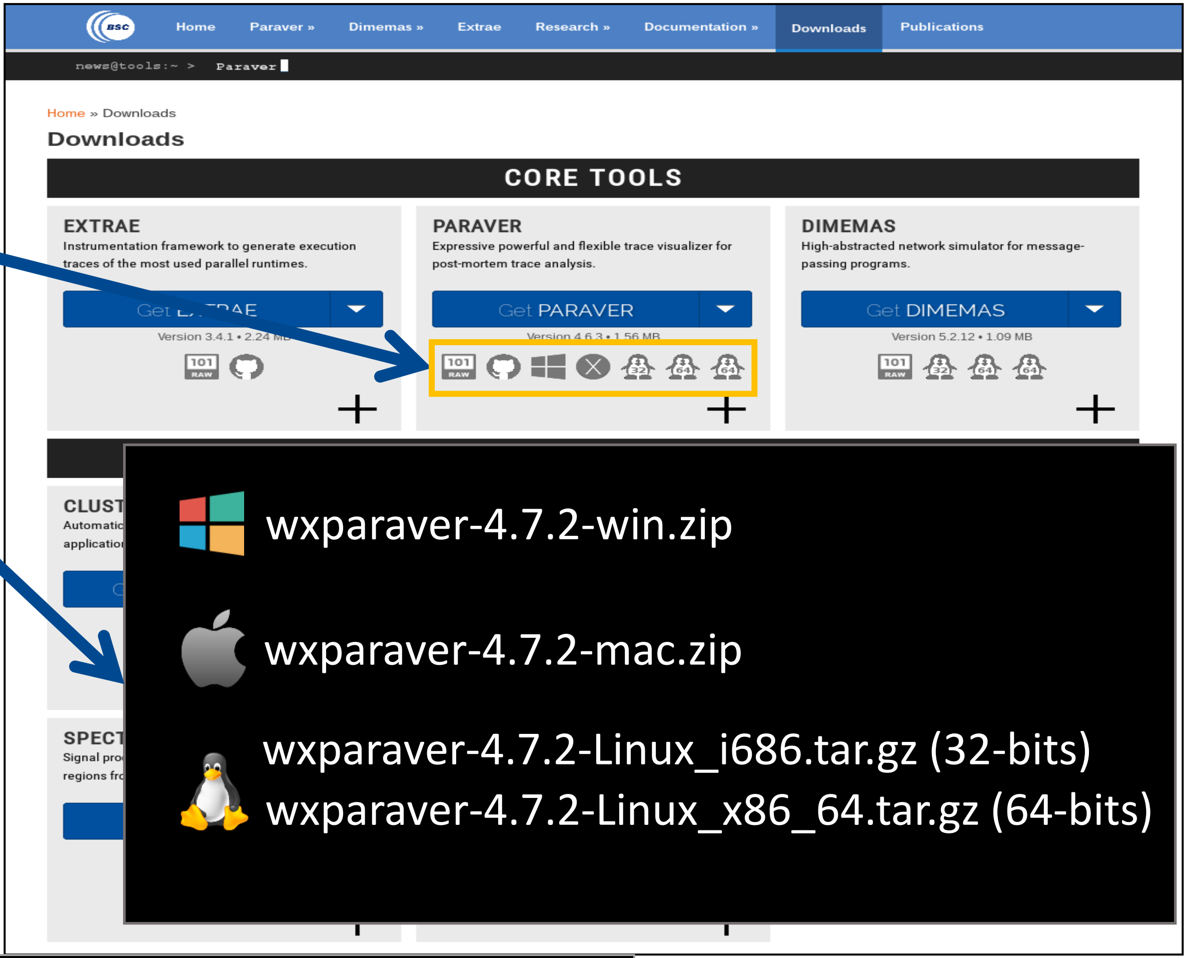
Extrae and Paraver Hands-on

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Install Paraver

- Download from https://tools.bsc.es/downloads

Pick your version



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

PARAVER
Expressive powerful and flexible trace visualizer for post-mortem trace analysis.
Get PARAVER
Version 4.6.3 • 1.56 MB

DIMEMAS
High-abstracted network simulator for message-passing programs.
Get DIMEMAS
Version 5.2.12 • 1.09 MB

CLUST
Automatic
application

SPECT
Signal pro
regions fr

wxparaver-4.7.2-win.zip

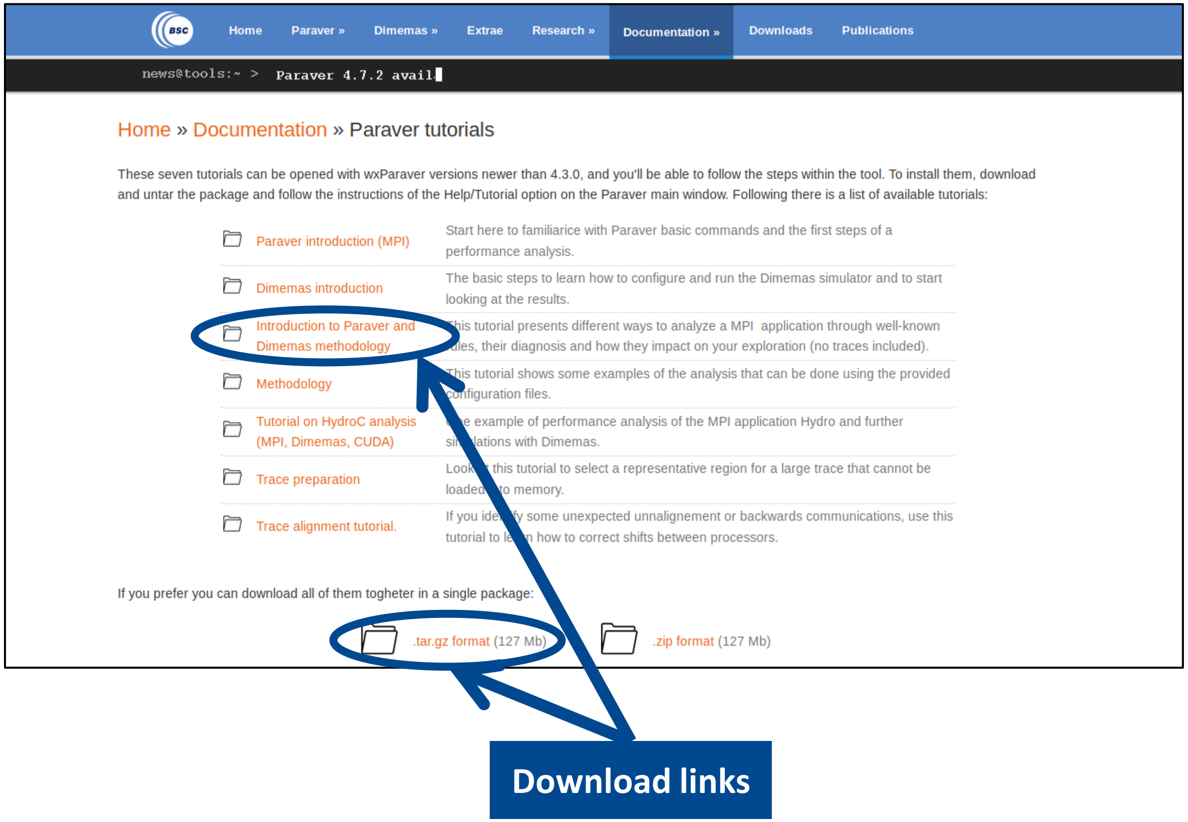
wxparaver-4.7.2-mac.zip

wxparaver-4.7.2-Linux_i686.tar.gz (32-bits)
wxparaver-4.7.2-Linux_x86_64.tar.gz (64-bits)

```
~gimenez1/BSC-paraver
```

Install Paraver (II)

- Download tutorials:
    - Documentation
        - Paraver tutorials



```
news@tools:~ >  Paraver 4.7.2 avail
```

Home » Documentation » Paraver tutorials

These seven tutorials can be opened with wxParaver versions newer than 4.3.0, and you'll be able to follow the steps within the tool. To install them, download and untar the package and follow the instructions of the Help/Tutorial option on the Paraver main window. Following there is a list of available tutorials:

Paraver introduction (MPI)	Start here to familiarice with Paraver basic commands and the first steps of a performance analysis.
Dimemas introduction	The basic steps to learn how to configure and run the Dimemas simulator and to start looking at the results.
Introduction to Paraver and Dimemas methodology	This tutorial presents different ways to analyze a MPI application through well-known rules, their diagnosis and how they impact on your exploration (no traces included).
Methodology	This tutorial shows some examples of the analysis that can be done using the provided configuration files.
Tutorial on HydroC analysis (MPI, Dimemas, CUDA)	One example of performance analysis of the MPI application Hydro and further simulations with Dimemas.
Trace preparation	Look at this tutorial to select a representative region for a large trace that cannot be loaded into memory.
Trace alignment tutorial.	If you identify some unexpected unalignement or backwards communications, use this tutorial to learn how to correct shifts between processors.

If you prefer you can download all of them togheter in a single package:

.tar.gz format (127 Mb)   .zip format (127 Mb)

Download links

~gimenez1/BSC-paraver

Uncompress, rename & move

- Paraver

@ your laptop

```
> tar xf wxparaver-4.8.1-linux-x86_64.tar.gz

> mv wxparaver-4.8.1-linux-x86_64 paraver
```

- Tutorials

@ your laptop

```
> tar xf paraver-tutorials-20150526.tar.gz

> mv paraver-tutorials-20150526 paraver/tutorials
```

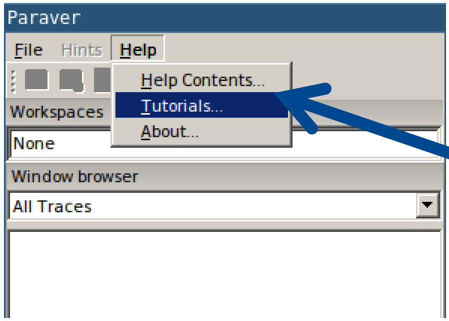
Check that everything works

- Start Paraver

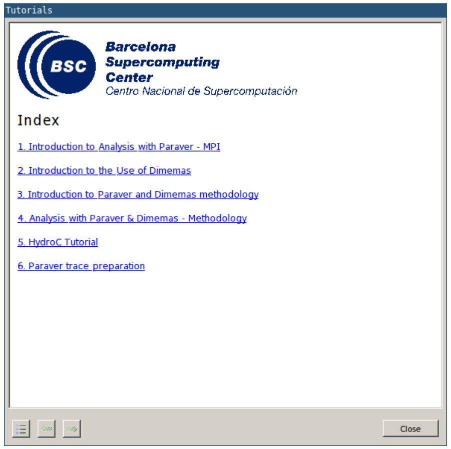
@ your laptop

```
> paraver/bin/wxparaver
```

- Check that tutorials are available



Click on Help → Tutorials


Tutorials
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación
Close

Index

1. Introduction to Analysis with Paraver - MPI
2. Introduction to the Use of Dimemas
3. Introduction to Paraver and Dimemas methodology
4. Analysis with Paraver & Dimemas - Methodology
5. HydroC Tutorial
6. Paraver trace preparation

- Trouble installing locally? Remote open from kabre

@ cluster.cenat.ac.cr

```
> ssh -Y <USER>@cluster.cenat.ac.cr
➢ ln -s ~gimenez1/tools/wxparaver64/bin/wxparaver
  wxparaver
```

Log in to kabré

@ your laptop

```
> ssh -Y <USER>@cluster.cenat.ac.cr
```

- Copy the examples to your home folder:

@ cluster.cenat.ac.cr

```
> cp -r ~jgimenez/BSC-handson/ $HOME

> ls -l $HOME/BSC-handson/
    … apps
    … traces
    … slides
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

OpenMP example: matrix

- Comparing the location of the parallel for pragma

```
#pragma omp parallel for
for(int i=0; i<N; i++){
        for(int k=0; k<N; k++){
                for(int j=0; j<N; j++)
```

matrix.l1

```
for(int i=0; i<N; i++){
#pragma omp parallel for
        for(int k=0; k<N; k++){
                for(int j=0; j<N; j++){
```

matrix.l2

- Check the script content
- Submit the job script

@ cluster.cenat.ac.cr

```
> cd $HOME/BSC-handson/apps/matrix
> qsub matrix.pbs
```

OpenMP example: matrix

- Copy the traces to your laptop

@ your laptop

```
> scp <USER>@cluster.cenat.ac.cr: \
    BSC_handson/apps/matrix/matrix.l?.* $HOME
```

- Load the traces with Paraver

@ your laptop

```
> wxparaver matrix.l?.prv*
```

- Trouble getting in the queues? Already available at ~/BSC-handson/traces/matrix/matrix.l?.*

- Compare the two executions with paraver

MPI example: jacobi

- Submit the job script

@ cluster.cenat.ac.cr

```
> cd $HOME/BSC-handson/apps/jacobi
> qsub job.kabre
```

- Copy the traces to your laptop and load the trace with Paraver

@ your laptop

```
> scp <USER>@cluster.cenat.ac.cr: \
    BSC_handson/apps/jacobi/jacobi.???* $HOME
> wxparaver jacobi.prv*
```

- Trouble getting in the queues? Already available at ~/BSC-handson/traces/jacobi/jacobi.???*
- Look at the execution with paraver

Extrae features

- Platforms
  - Intel, Cray, BlueGene, MIC, ARM, Android, Fujitsu Sparc…
- Parallel programming models
  - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Java, Python…
- Performance Counters
  - Using PAPI interface
- Link to source code
  - Callstack at MPI routines
  - OpenMP outlined routines
  - Selected user functions (Dyninst)
- Periodic sampling
- User events (Extrae API)

No need to recompile / relink!

Extrae overheads

	Average values	Kábre (login)	Kabré (Zárate)
Event	150-200 ns	180 ns	600 ns
Event + PAPI	750 ns – 1 us	580 ns	4.7 us
Event + callstack (1 level)	600 ns	750 ns	3.4 us
Event + callstack (6 levels)	2 us	1.7us	8.2 us

How does Extrae work?

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

Recommended

- Dynamic instrumentation
  - Based on Dyninst (developed by U.Wisconsin / U.Maryland)
    - Instrumentation in memory
    - Binary rewriting
- Alternatives
  - Compiler instrumentation (-finstrument-functions – Intel, GNU)
  - Static link (i.e., PMPI, Extrae API)

Using Extrae in 3 steps

1. **Adapt** your job submission scripts

2. (Optional) **Tune** the Extrae XML configuration file
   - Examples distributed with Extrae at $EXTRAE_HOME/share/example

3. **Run** it!

- For further reference check the **Extrae User Guide:**
  - https://tools.bsc.es/sites/default/files/documentation/html/extrae/index.html
  - Also distributed with Extrae at $EXTRAE_HOME/share/doc

Step 1: Adapt the job script to load Extrae

@ cluster.cenat.ac.cr

```
> vi $HOME/BSC-handson/apps/lulesh/job.kabre
```

job.kabre

```
#PBS -N extrae
#PBS -q phi-n1h72
#PBS -l nodes=1:ppn=27
#PBS -l walltime=00:20:00

cd $PBS_O_WORKDIR

module load mpich/3.2.1




mpirun  ./lulesh2.0 -i 20 -s 64 –p
```

Step 1: Adapt the job script to load Extrae

@ cluster.cenat.ac.cr

```
> vi $HOME/BSC-handson/apps/lulesh/job.kabre
```

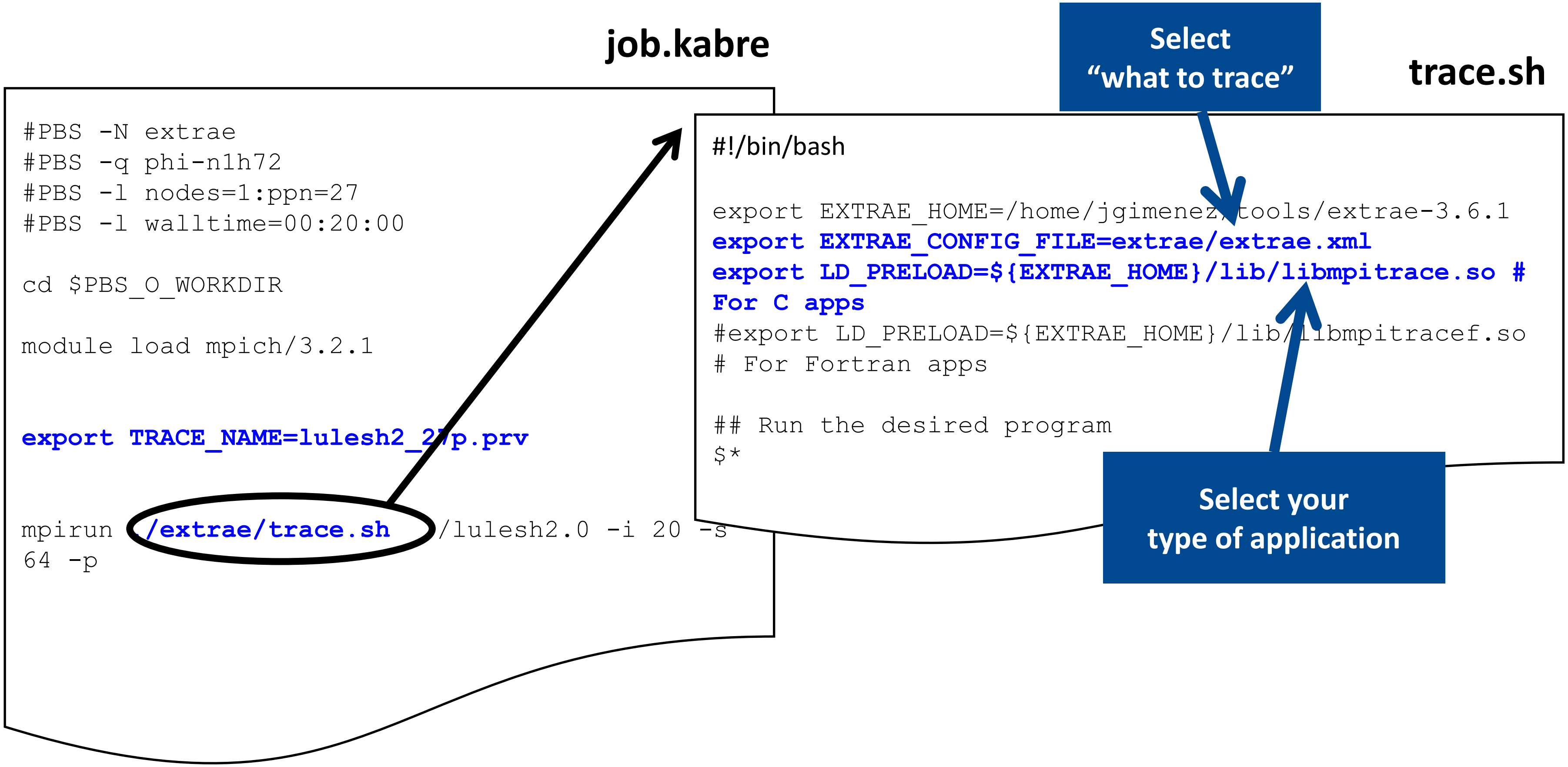
job.kabre

```
#PBS -N extrae
#PBS -q phi-n1h72
#PBS -l nodes=1:ppn=27
#PBS -l walltime=00:20:00

cd $PBS_O_WORKDIR

module load mpich/3.2.1


export TRACE_NAME=lulesh2_27p.prv


mpirun ./extrae/trace.sh  ./lulesh2.0 -i 20 -s
64 -p
```

Step 1: Adapt the job script to load Extrae

@ cluster.cenat.ac.cr

```
> vi $HOME/BSC-handson/apps/lulesh/extrae/trace.sh
```



job.kabre

```
#PBS -N extrae
#PBS -q phi-n1h72
#PBS -l nodes=1:ppn=27
#PBS -l walltime=00:20:00

cd $PBS_O_WORKDIR

module load mpich/3.2.1


export TRACE_NAME=lulesh2_27p.prv


mpirun ./extrae/trace.sh ./lulesh2.0 -i 20 -s
64 -p
```

trace.sh

```
#!/bin/bash

export EXTRAE_HOME=/home/jgimenez/tools/extrae-3.6.1
export EXTRAE_CONFIG_FILE=extrae/extrae.xml
export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitrace.so #
For C apps
#export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitracef.so
# For Fortran apps

## Run the desired program
$*
```

Select “what to trace”

Select your type of application

Step 1: Which tracing library?

- Choose depending on the application type

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

[1] include suffix “f” in Fortran codes

Step 3: Run it!

- Submit your job

@ cluster.cenat.ac.cr

```
> cd $HOME/BSC-handson/apps/lulesh

> qsub job.kabre
```

- Easy! ☺

Step 2: Extrae XML configuration

@ cluster.cenat.ac.cr

```
> vi $HOME/BSC-handson/apps/lulesh/extrae/extrae.xml
```

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

<openmp enabled="no">
  <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 the MPI calls (What's the program doing?)

Trace the call-stack (Where in my code?)

Step 2: Extrae XML configuration (II)

@ cluster.cenat.ac.cr

```
> vi $HOME/BSC-handson/apps/lulesh/extrae/extrae.xml
```

```
<counters enabled="yes">
  <cpu enabled="yes" starting-set-distribution=“1">
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_L1_DCM, PAPI_L2_DCM, PAPI_BR_MSP, RESOURCE_STALLS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_L3_TCM, PAPI_LD_INS, PAPI_SR_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_VEC_DP
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_VEC_SP, PAPI_FP_INS
    </set>
  </cpu>
  <network enabled="no" />
  <resource-usage enabled="no" />
  <memory-usage enabled="no" />
</counters>
```

Select which HW counters are measured

Step 2: Extrae XML configuration (III)

@ cluster.cenat.ac.cr

```
> vi $HOME/BSC-handson/apps/lulesh/extrae/extrae.xml
```

```
<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=“16"
  max-memory="512"
  joint-states="yes"
  keep-mpits="yes"
  sort-addresses="yes"
  overwrite="yes“
>
  $TRACE_NAME$
</merge>
```

Trace buffer size
(Flush/memory trade-off)

Enable sampling
(Want more details?)

Automatic post-processing
to generate the trace

All done! Check your resulting trace

- Once finished (check with `squeue`) you will have the trace (3 files):

@ cluster.cenat.ac.cr

```
➢ ls -l $HOME/BSC-handson/apps/lulesh
   ...
   lulesh2_27p.pcf
   lulesh2_27p.prv
   lulesh2_27p.row
```

- Trouble getting in the queues? Already available at ~/BSC-handson/traces/lulesh/lulesh2_27p.*

- Now let's look into it !

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

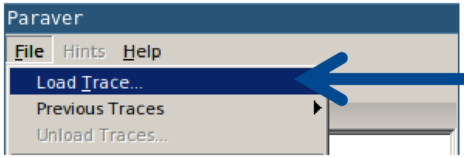
First steps of analysis

- Copy the trace to your laptop

@ your laptop

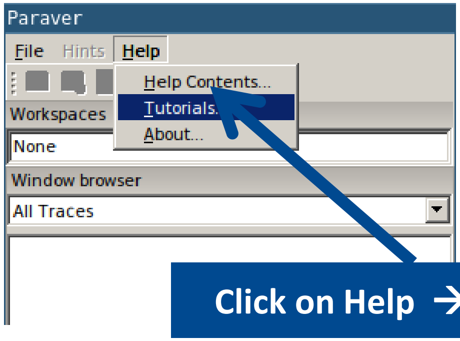
```
> scp <USER>@cluster.cenat.ac.cr: \
    BSC_handson/apps/lulesh/lulesh2_27p.* $HOME
```

- Load the trace with Paraver

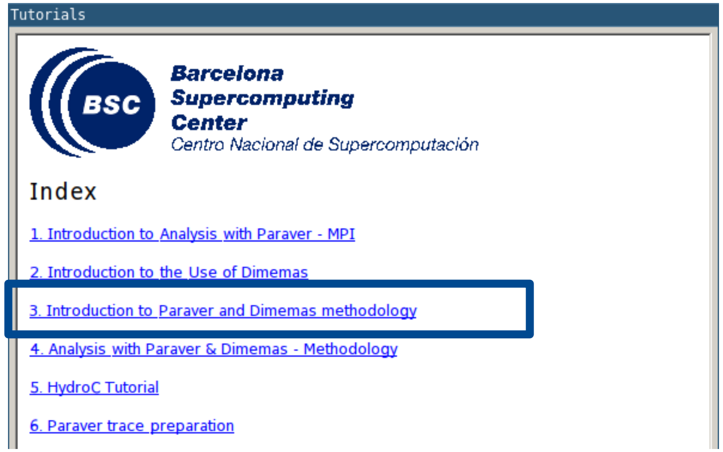


Click on File → Load Trace → Browse to “lulesh2.0_27p.prv”

- Follow Tutorial #3



Click on Help → Tutorials


Tutorials
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

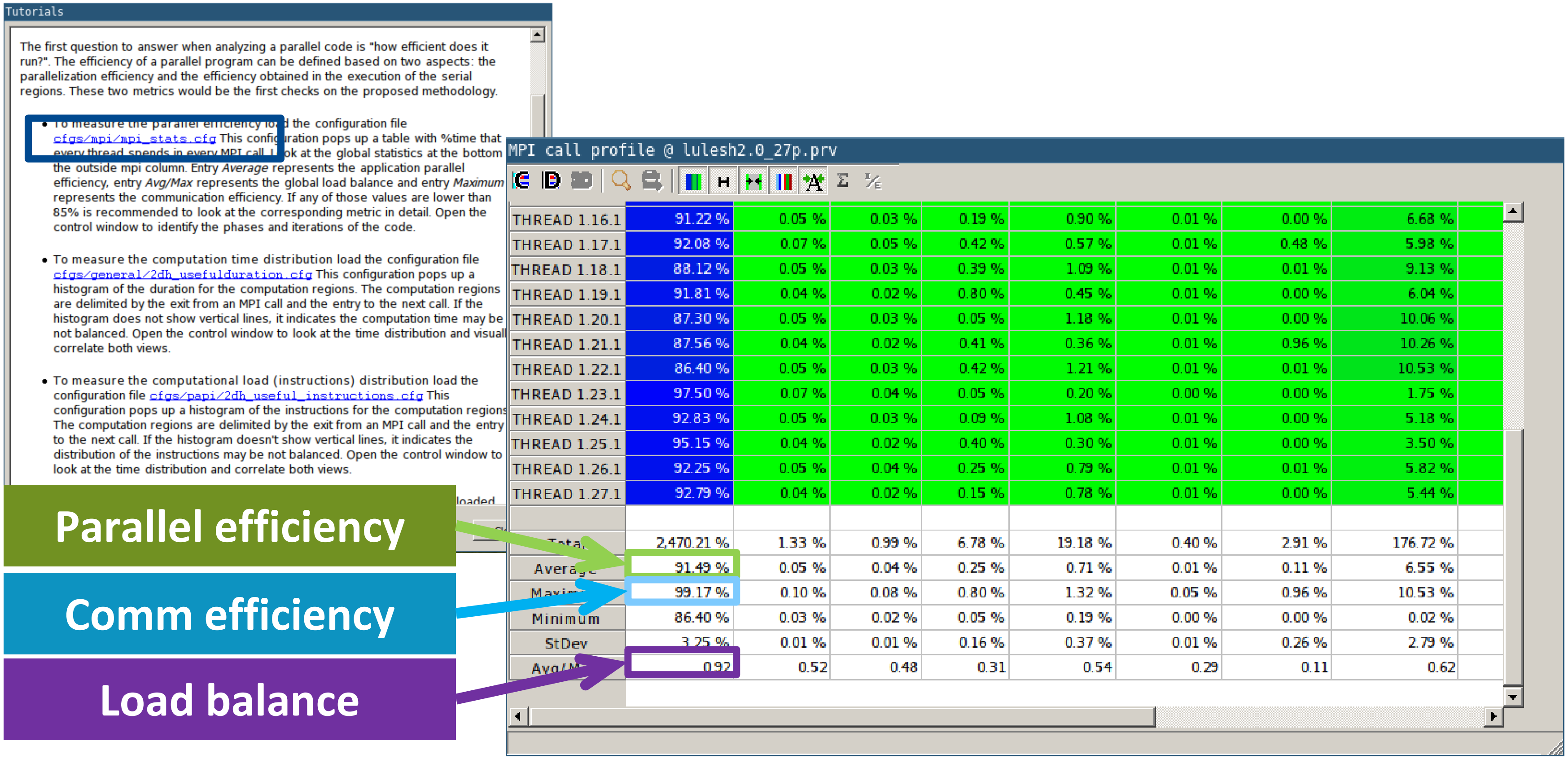
Index

1. Introduction to Analysis with Paraver - MPI
2. Introduction to the Use of Dimemas
3. Introduction to Paraver and Dimemas methodology
4. Analysis with Paraver & Dimemas - Methodology
5. HydroC Tutorial
6. Paraver trace preparation

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Measure the parallel efficiency

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



Tutorials

The first question to answer when analyzing a parallel code is "how efficient does it run?". The efficiency of a parallel program can be defined based on two aspects: the parallelization efficiency and the efficiency obtained in the execution of the serial regions. These two metrics would be the first checks on the proposed methodology.

- To measure the parallel efficiency load the configuration file cfgs/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 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 cfgs/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 visuall correlate both views.

- To measure the computational load (instructions) distribution load the configuration file cfgs/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.

loaded

MPI call profile @ lulesh2.0_27p.prv

THREAD 1.16.1	91.22 %	0.05 %	0.03 %	0.19 %	0.90 %	0.01 %	0.00 %	6.68 %
THREAD 1.17.1	92.08 %	0.07 %	0.05 %	0.42 %	0.57 %	0.01 %	0.48 %	5.98 %
THREAD 1.18.1	88.12 %	0.05 %	0.03 %	0.39 %	1.09 %	0.01 %	0.01 %	9.13 %
THREAD 1.19.1	91.81 %	0.04 %	0.02 %	0.80 %	0.45 %	0.01 %	0.00 %	6.04 %
THREAD 1.20.1	87.30 %	0.05 %	0.03 %	0.05 %	1.18 %	0.01 %	0.00 %	10.06 %
THREAD 1.21.1	87.56 %	0.04 %	0.02 %	0.41 %	0.36 %	0.01 %	0.96 %	10.26 %
THREAD 1.22.1	86.40 %	0.05 %	0.03 %	0.42 %	1.21 %	0.01 %	0.01 %	10.53 %
THREAD 1.23.1	97.50 %	0.07 %	0.04 %	0.05 %	0.20 %	0.00 %	0.00 %	1.75 %
THREAD 1.24.1	92.83 %	0.05 %	0.03 %	0.09 %	1.08 %	0.01 %	0.00 %	5.18 %
THREAD 1.25.1	95.15 %	0.04 %	0.02 %	0.40 %	0.30 %	0.01 %	0.00 %	3.50 %
THREAD 1.26.1	92.25 %	0.05 %	0.04 %	0.25 %	0.79 %	0.01 %	0.01 %	5.82 %
THREAD 1.27.1	92.79 %	0.04 %	0.02 %	0.15 %	0.78 %	0.01 %	0.00 %	5.44 %
Total	2,470.21 %	1.33 %	0.99 %	6.78 %	19.18 %	0.40 %	2.91 %	176.72 %
Average	91.49 %	0.05 %	0.04 %	0.25 %	0.71 %	0.01 %	0.11 %	6.55 %
Maximum	99.17 %	0.10 %	0.08 %	0.80 %	1.32 %	0.05 %	0.96 %	10.53 %
Minimum	86.40 %	0.03 %	0.02 %	0.05 %	0.19 %	0.00 %	0.00 %	0.02 %
StDev	3.25 %	0.01 %	0.01 %	0.16 %	0.37 %	0.01 %	0.26 %	2.79 %
Avg/Max	0.92	0.52	0.48	0.31	0.54	0.29	0.11	0.62

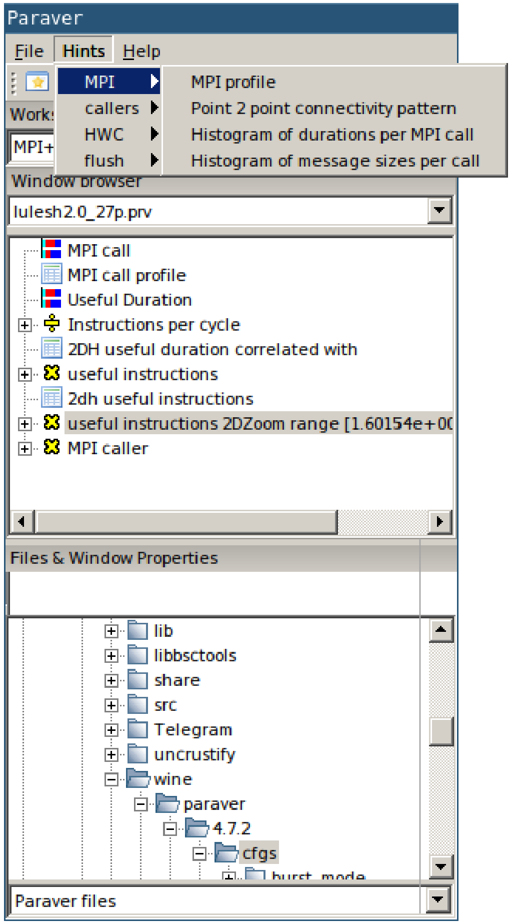
Parallel efficiency

Comm efficiency

Load balance

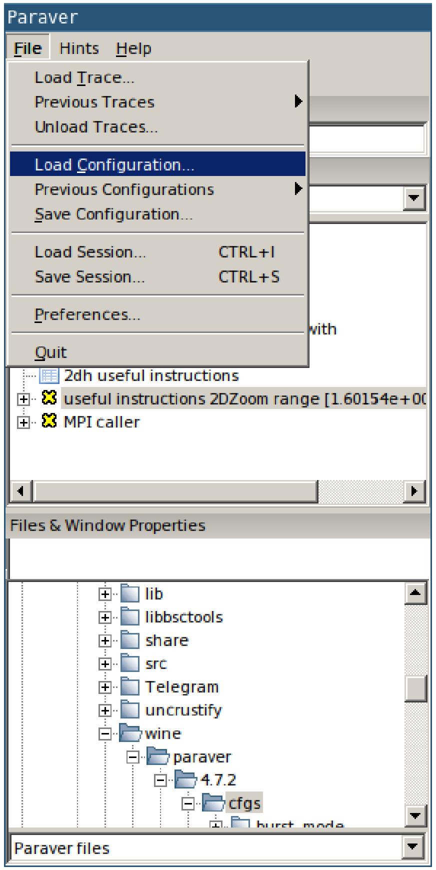
Hints: a good place to start!

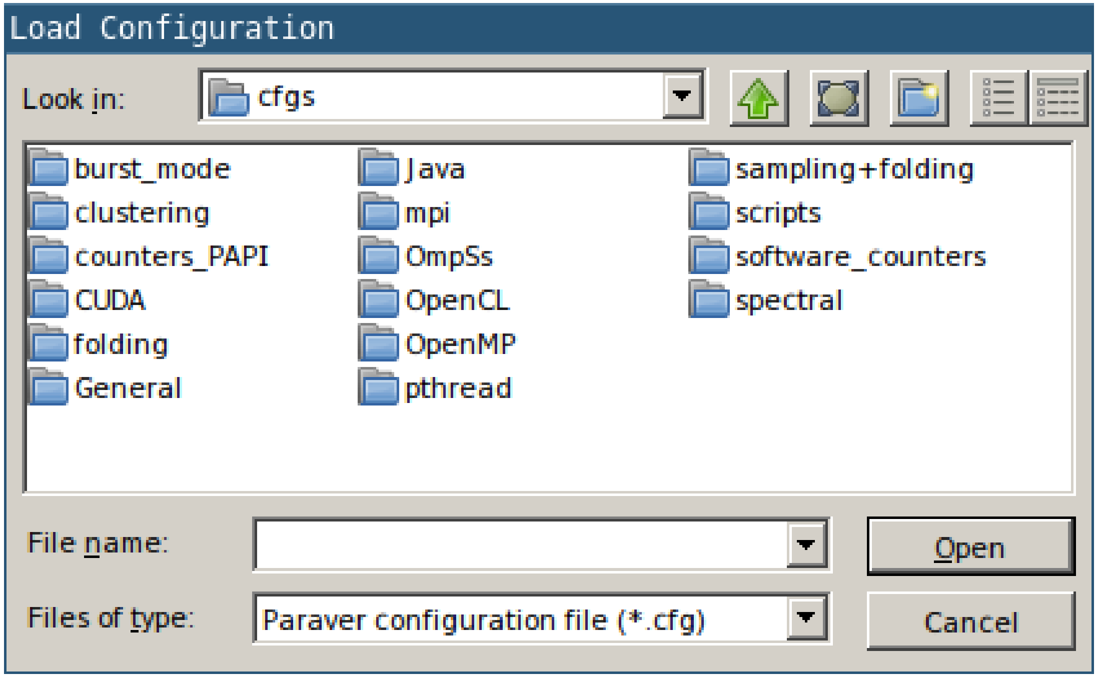
- Paraver suggests CFG's based on the information present in the trace



CFG’s distribution

- Paraver comes with many more included CFG’s





Look in: cfgs

Files of type: Paraver configuration file (*.cfg)