

BSC Tools Hands-On

Judit Giménez, Germán Llort (gllort@bsc.es)

Barcelona Supercomputing Center

Getting a trace with 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	UV2
Event	150-200 ns	530 ns
Event + PAPI	750 ns - 1 us	1.5 us
Event + callstack (1 level)	600 ns	930 ns
Event + callstack (6 levels)	1.9 us	2.9 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
- <http://www.bsc.es/computer-sciences/performance-tools/documentation>

Log in and copy the examples to your home directory

```
> ssh -Y <USER>@lxlogin1.lrz.de  
  
> ssh -Y ice1-login  
  
> cp -r /lrz/sys/courses/VIHPS21TW/bsc/tools-material $HOME  
  
> ls $HOME/tools-material  
...apps  
...slides  
...extrae  
...traces
```



Here you have a copy of this slides.

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

```
> vi $HOME/tools-material/extrae/job_27p.sh
```

```
#!/bin/bash
#SBATCH -o lulesh.out
#SBATCH -J lulesh
#SBATCH --get-user-env
#SBATCH --clusters=uv2
#SBATCH --ntasks=27
#SBATCH --time=00:05:00
#SBATCH -reservation=VI-HPS_Workshop

source /etc/profile.d/modules.sh
module unload mpi.mpt
module load mpi.intel

export I_MPI_DEVICE=shm
export OMP_NUM_THREADS=1

srun_ps -n 27 ../apps/lulesh2.0 -i 10 -p -s 65
```


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

```
> vi $HOME/tools-material/extrae/job_27p.sh
```

```
#!/bin/bash
#SBATCH -o lulesh.out
#SBATCH -J lulesh
#SBATCH --get-user-env
#SBATCH --clusters=uv2
#SBATCH --ntasks=27
#SBATCH --time=00:05:00
#SBATCH -reservation=VI-HPS_Workshop

source /etc/profile.d/modules.sh
module unload mpi.mpt
module load mpi.intel

export I_MPI_DEVICE=shm
export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh_27p.prv

srun_ps -n 27 ./trace.sh ../apps/lulesh2.0
-i 10 -p -s 65
```

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

```
> vi $HOME/tools-material/extrae/job_27p.sh
```

```
#!/bin/bash
#SBATCH -o lulesh.out
#SBATCH -J lulesh
#SBATCH --get-user-env
#SBATCH --clusters=uv2
#SBATCH --ntasks=27
#SBATCH --time=00:05:00
#SBATCH -reservation=VI-HPS_Workshop

source /etc/profile.d/modules.sh
module unload mpi.mpt
module load mpi.intel

export I_MPI_DEVICE=shm
export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh_27p.prv

srun_ps -n 27 ./trace.sh ...apps/lulesh2.0
-i 10 -p -s 65
```

trace.sh

```
#!/bin/bash

source /lrz/sys/courses/VIHPS21TW/bsc/setup.sh

# Configure Extrae
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/libmpitrace.so
```

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		✓			
libomptrace			✓		
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

@ ice1-login

```
> cd $HOME/tools-material/extrae  
> sbatch job_27p.sh
```

Step 2: Extrae XML configuration: extrae_config.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_config.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
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_LD_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_SR_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_FP_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_BR_MSP
    </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_config.xml (III)

```
<buffer enabled="yes">
```

```
  <size enabled="yes">500000</size>
```

Trace buffer size

```
  <circular enabled="no" />
```

```
</buffer>
```

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

Enable sampling

```
<merge enabled="yes"
```

```
  synchronization="default"
```

```
  tree-fan-out="16"
```

```
  max-memory="512"
```

```
  joint-states="yes"
```

```
  keep-mpits="yes"
```

```
  sort-addresses="yes"
```

```
  overwrite="yes"
```

Merge intermediate
files into Paraver
trace

```
>
```

```
  $TRACE$
```

```
</merge>
```

Check the resulting trace

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

@ ice1-login

```
> ls -ltr $HOME/tools-material/extrae
...
lulesh_27p.prv
lulesh_27p.pcf
lulesh_27p.row
```

- Any trouble? Traces already generated here:

@ ice1-login

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


First steps of analysis

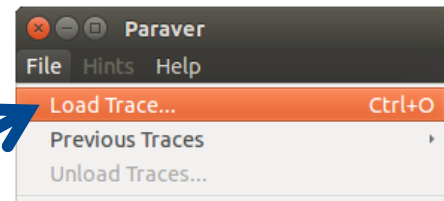
- Run Paraver from lxlogin1

@ lxlogin1

```
> ssh -Y <USER>@lxlogin1.lrz.de
> source /lrz/sys/courses/VIHPS21TW/bsc/setup.sh
> wxparaver
```

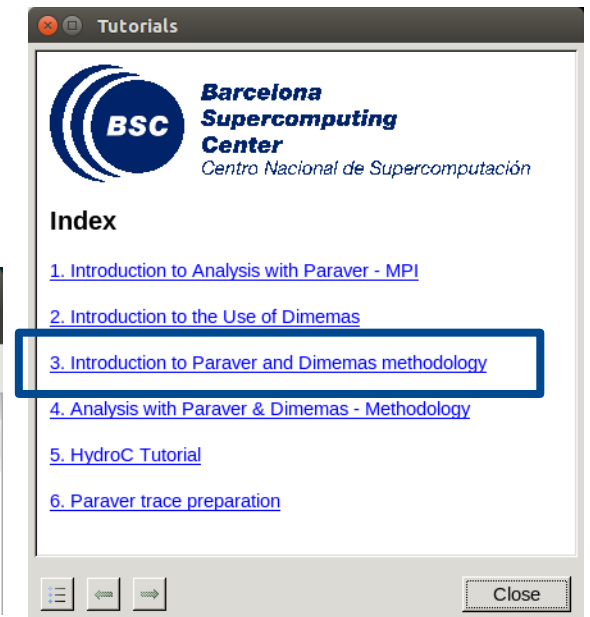
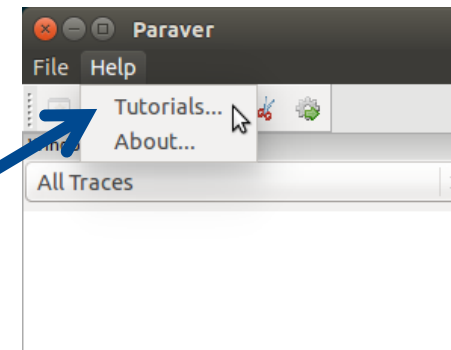
- Load the trace

Click on File → Load Trace → Browse to "lulesh_27p.prv"



- Follow Tutorial #3
 - Introduction to Paraver and Dimemas methodology

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 `cfqs/mpi/mpi_stats.cfg`. This configuration pops up a table with %time of 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 `cfqs/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**

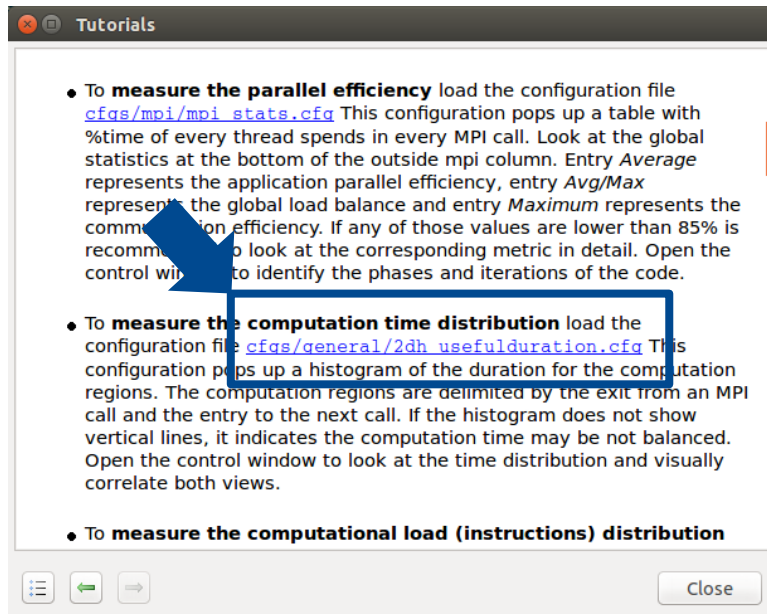
Close

MPI call profile @ lulesh_27p.prv

	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Barrier	MPI_Reduce	MF
THREAD 1.20.1	83.83 %	0.06 %	0.02 %	0.38 %	2.62 %	0.01 %	0.00 %	
THREAD 1.21.1	82.49 %	0.03 %	0.02 %	0.41 %	1.63 %	0.01 %	1.02 %	
THREAD 1.22.1	75.54 %	0.04 %	0.02 %	1.16 %	7.11 %	0.05 %	0.00 %	
THREAD 1.23.1	92.53 %	0.05 %	0.02 %	0.28 %	2.51 %	0.00 %	0.00 %	
THREAD 1.24.1	89.06 %	0.05 %	0.02 %	0.08 %	2.89 %	0.00 %	0.00 %	
THREAD 1.25.1	80.52 %	0.03 %	0.02 %	1.02 %	7.23 %	0.08 %	0.47 %	
THREAD 1.26.1	90.18 %	0.05 %	0.02 %	0.49 %	1.62 %	0.00 %	0.00 %	
THREAD 1.27.1	88.45 %	0.04 %	0.01 %	0.04 %	3.72 %	0.01 %	0.00 %	
Total	2,306.34 %	1.21 %	0.59 %	15.78 %	98.15 %	0.32 %	4.57 %	
Average	85.42 %	0.04 %	0.02 %	0.58 %	3.64 %	0.01 %	0.17 %	
Maximum	94.38 %	0.06 %	0.03 %	1.61 %	7.61 %	0.08 %	1.02 %	
Minimum	75.54 %	0.02 %	0.01 %	0.04 %	1.62 %	0.00 %	0.00 %	
StDev	5.19 %	0.01 %	0.01 %	0.38 %	2.07 %	0.02 %	0.28 %	
Avg/Max	0.91	0.69	0.64	0.36	0.48	0.15	0.17	

Measure the computation time distribution

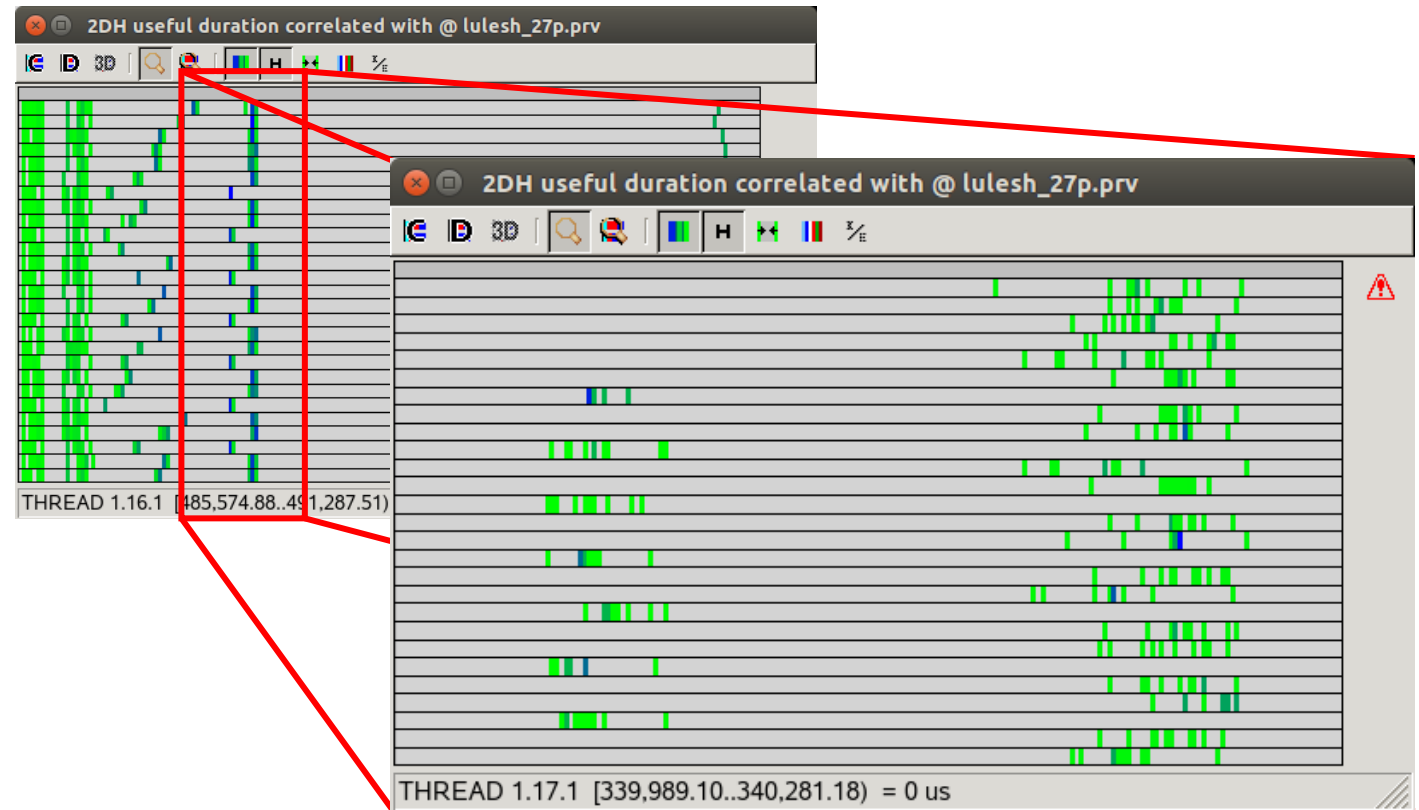
- Click on the “2dh_usefulduration.cfg”



Tutorials

- To **measure the parallel efficiency** load the configuration file [cfgs/mpi/mpi_stats.cfg](#). This configuration pops up a table with %time of 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 [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 visually correlate both views.
- To **measure the computational load (instructions) distribution**

Close



Measure the computation time distribution

- Click on the “2dh_usefulduration.cfg”

The image displays a tutorial window on the left and two performance analysis windows on the right. The tutorial window, titled "Tutorials", contains three bullet points:

- To **measure the parallel efficiency** load the configuration file `cfgs/mpi/mpi_stats.cfg`. This configuration pops up a table with %time of 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 `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 visually correlate both views.
- To **measure the computational load (instructions) distribution**

The two performance analysis windows, both titled "2DH useful duration correlated with @ lulesh_27p.prv", show a Gantt chart of computation time distribution. The top window shows a red box highlighting a region of the chart. The bottom window shows a blue box highlighting a region of the chart. A blue callout box labeled "20 SLOW tasks" points to the right window, and a blue callout box labeled "7 FAST tasks" points to the bottom window. The bottom window also shows a status bar with the text "THREAD 1.16.1 [485,574.88..491,287.51]" and "READ 1.17.1 [339,989.10..340,281.18] = 0 us".

Installing Paraver locally

Installing Paraver locally

- Download the Paraver binaries to your laptop

@ your laptop

```
> scp <USER>@lxlogin1.lrz.de:/lrz/sys/courses/VIHPS21TW/bsc/  
tools-packages/<VERSION> $HOME
```

Pick your version

Linux 64 bits

```
wxparaver-4.6.1-linux-x86_64.tar.gz
```

Linux 32 bits

```
wxparaver-4.6.1-linux-x86_32.tar.gz
```

Mac

```
wxparaver-4.6.1-mac.zip
```

Windows

```
wxparaver-4.6.1-win.zip
```

Installing Paraver (II)

- Uncompress the package into your home folder (Linux example)

@ your laptop

```
> tar xvfz wxparaver-4.6.1-linux-x86_64.tar.gz -C $HOME  
> ln -s $HOME/wxparaver-4.6.1-linux-x86_64 $HOME/paraver
```

- Download Paraver tutorials and uncompress into the Paraver folder (Linux example)

@ your laptop

```
> scp <USER>@lxlogin1.lrz.de:/lrz/sys/courses/VIHPS21TW/bsc/  
tools-packages/paraver-tutorials-20150526.tar.gz $HOME  
> tar xvfz $HOME/paraver-tutorials-20150526.tar.gz -C $HOME/paraver
```

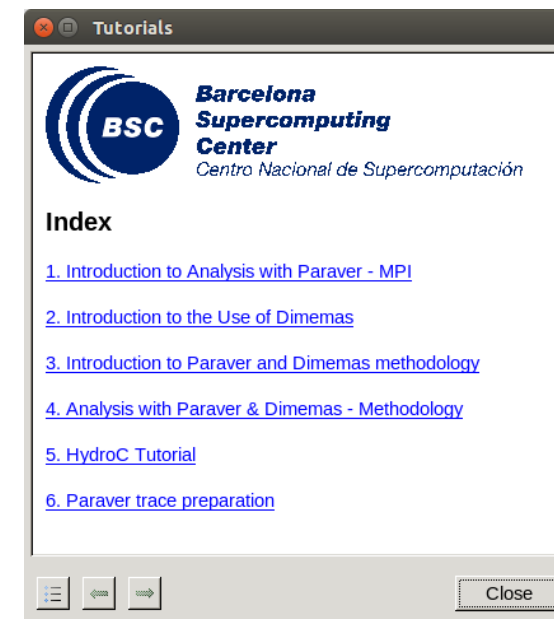
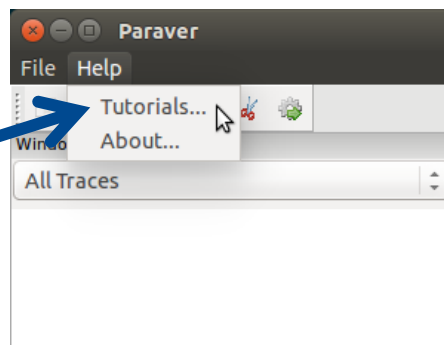
Check that everything works

- Launch Paraver from your laptop

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

- Check that tutorials are available

Click on Help → Tutorials



- Copy the trace to your laptop and open it locally

```
> scp <USER>@lxlogin1.lrz.de:tools-material/extrae/lulesh_27p.\* $HOME
```