

www.bsc.es



**Barcelona
Supercomputing
Center**

Centro Nacional de Supercomputación

Tools Hands-on

Judit Gimenez
judit@bsc.es

NERSC - Berkeley, August 2016

www.bsc.es



**Barcelona
Supercomputing
Center**

Centro Nacional de Supercomputación

Getting a trace with Extrae

Extrae features

Parallel programming models

- MPI, OpenMP(*), pthreads, OmpSs, CUDA, OpenCL, Java, Python...

Platforms

- Intel, Cray, BlueGene,, Fujitsu Sparc, Xeon Phi,, ARM, Android...

Performance Counters

- Using PAPI

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!**

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

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

Extrae overheads

	Average values	Cori (intel – gnu)
Event	120-200 ns	120 - 140 ns
Event + PAPI	725 ns – 1 us	730 - 725 us
Event + callstack (1 level)	600 ns	570 - 518 ns
Event + callstack (6 levels)	2 us	1.8 us

How to use Extrae

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 to cori and copy the example to your home

@ your laptop

```
> ssh -X <USER>@cori.nersc.gov
```

⌘ Copy the examples to your home folder:

@ cori.nersc.gov

```
> cp -r ~judit/handson_extrae $HOME  
  
> ls -l $HOME/handson_extrae  
... lulesh  
... mpi_ping  
... extrae
```

Adapt the job script to load Extrae (LD_PRELOAD)

@ cori.nersc.gov

```
> vi $HOME/handson_extrae/lulesh/job1n.sh
```

job1n.sh

```
#!/bin/bash -l
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --time=00:10:00
#SBATCH --license=SCRATCH #note: specify
license need for the file systems your job
needs, such as SCRATCH,project

module swap PrgEnv-intel PrgEnv-gnu

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


Adapt the job script to load Extrae (LD_PRELOAD)

@ cori.nersc.gov

```
> vi $HOME/handson_extrae/lulesh/job1n.sh
```

job1n.sh

```
#!/bin/bash -l
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --time=00:10:00
#SBATCH --license=SCRATCH #note: specify
license need for the file systems your job
needs, such as SCRATCH,project

module swap PrgEnv-intel PrgEnv-gnu
module load extrae

TRACE=./extrae/trace.sh
export TRACE_NAME=lulesh27_1node.prv

srun -n 27 $TRACE ./lulesh2.0 -i 10 -p -s 65
```

Adapt the job script to load Extrae (LD_PRELOAD)

@ cori.nersc.gov

```
> vi $HOME/handson_extrae/lulesh/job1n.sh
```

job1n.sh

```
#!/bin/bash -l
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --time=00:10:00
#SBATCH --license=SCRATCH #note: specify
license need for the file systems your job
needs, such as SCRATCH,project

module swap PrgEnv-intel PrgEnv-gnu
module load extrae

TRACE=./extrae/trace.sh
export TRACE_NAME=lulesh27_1node.prv

srun -n 27 $TRACE /lulesh2.0 -i 10 -p -s 65
```

trace.sh

```
#!/bin/bash

export EXTRAE_HOME=$EXTRAE_DIR
export EXTRAE_CONFIG_FILE=$HOME/
handson_extrae/extrae/extrae.xml

export LD_PRELOAD=${EXTRAE_HOME}/lib/
libmpitrace.so
# For C apps
#exportLD_PRELOAD=${EXTRAE_HOME}/lib/
libmpitracef.so
# For Fortran apps

## Run the desired program
$*
```

Pick tracing
library

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

Extrae XML configuration

```
<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

Extrae XML configuration (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_CN,PAPI_BR_UCN
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_BR_MSP,PAPI_SR_INS,PAPI_LD_INS,RESOURCE_STALLS:SB
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L2_DCM,RESOURCE_STALLS:ROB,RESOURCE_STALLS:RS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,RESOURCE_STALLS
    </set>
  </cpu>
  <network enabled="no" />
  <resource-usage enabled="no" />
  <memory-usage enabled="no" />
</counters>
```

Select which HW counters
are measured

Extrae XML configuration (III)

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

Enable sampling

Merge intermediate files into Paraver trace

Run with instrumentation

« Submit your job

@ cori.nersc.gov

```
> cd $HOME/handson_extrae  
> sbatch job1n.sh
```

- Once finished (check with “squeue”) you will have the trace:
 - Lulesh27_1node.{prv,pcf,row} (3 files)

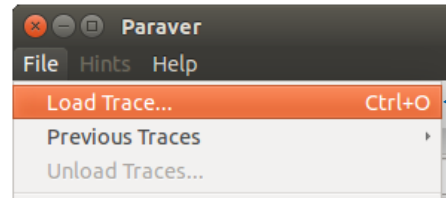
First steps of analysis

Start Paraver

@ cori.nersc.gov

```
> source $HOME/handson_extrae/setup_bsctools.sh  
> wxparaver &
```

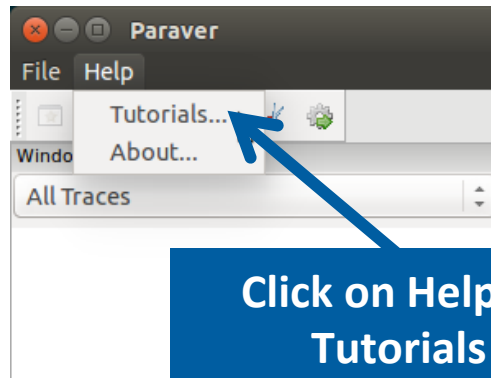
Load the trace



Click on File → Load Trace → Browse to lulesh27_1node.prv

Follow Tutorial #3

- Introduction to Paraver and Dimemas methodology



Click on Help → Tutorials



Is the application efficient?

« Click on “mpi_stats.cfg”

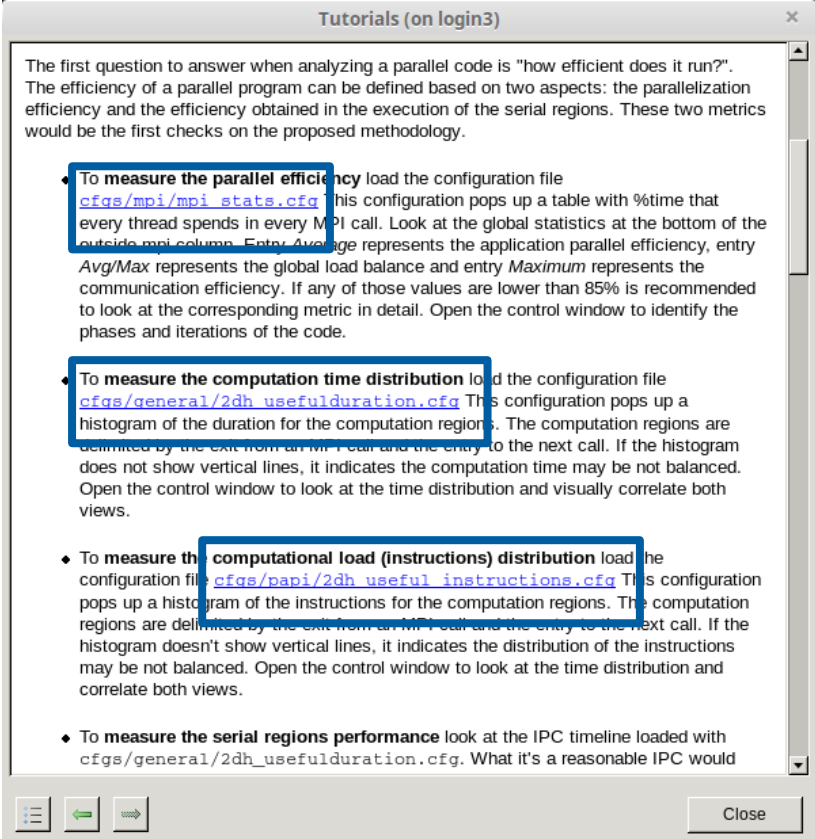
- Check the Average for the column labeled “Outside MPI”
 - Parallel efficiency is high?

« Click on “2dh_usefulduration.cfg”

- Histogram of duration of the computing regions
 - Vertical lines or dispersion? Fast/slow processes?

« Click on “2dh_useful_instructions.cfg”

- Histogram of instructions of the computing regions
 - Work is well distributed?



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 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 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.
- To measure the serial regions performance look at the IPC timeline loaded with `cfgs/general/2dh_usefulduration.cfg`. What it's a reasonable IPC would

The trace is HUGE?

« Chop a region of interest from the trace-file

1. Filter the trace discarding very small regions

```
> $PARAVER_HOME/bin/paramedir -f filter.xml <trace.prv>
```

Example file in cori:

`~judit/handson-extrae/tools_cfgs/filter.xml`

- This gets a trace showing only the most relevant regions for the whole run
2. On the filtered trace: File → Load configuration → `cfgs/General/view/useful-duration.cfg`
 - Identify a periodic pattern / small region of interest
 3. Apply the “scissors” tool → Cut the region of interest from the big trace
 - This gets a small trace with all the details for a small time interval

www.bsc.es



**Barcelona
Supercomputing
Center**

Centro Nacional de Supercomputación

Installing Paraver in your laptop

Installing Paraver in your laptop (I)

⌘ Download the Paraver binaries to your laptop

@ your laptop

```
> scp <USER>@cori.nersc.gov:~judit/packages/<VERSION> $HOME
```

Pick your version

Linux 64 bits

```
wxparaver-4.6.2-linux-x86_64.tar.gz
```

Linux 32 bits

```
wxparaver-4.6.2-linux-x86_32.tar.gz
```

Mac

```
wxparaver-4.6.2-mac.zip
```

Windows

```
wxparaver-4.6.2-win.zip
```

Installing Paraver in your laptop (II)

- ⌘ Uncompress the package into your home directory

@ your laptop

```
> tar xvfz wxparaver-4.6.2-linux-x86_64.tar.gz -C $HOME  
> ln -s $HOME/wxparaver-4.6.2-linux-x86_64 $HOME/paraver
```

- ⌘ Download Paraver tutorials and uncompress into the Paraver directory

@ your laptop

```
> scp <USER>@cori.nersc.gov:~judit/packages/paraver-  
tutorials.tar.gz $HOME  
> tar xvfz $HOME/paraver-tutorials.tar.gz -C $HOME/paraver
```

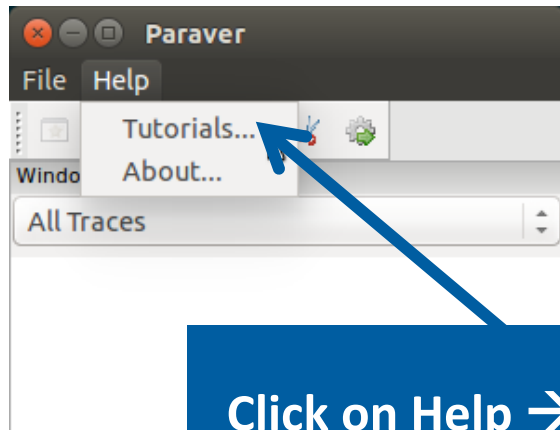
Check that everything works

« Start Paraver

@ your laptop

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

« Check that tutorials are available



Click on Help → Tutorials

