Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Understanding applications with Paraver

Judit Gimenez

judit@bsc.es

Oct 7-10, 2019

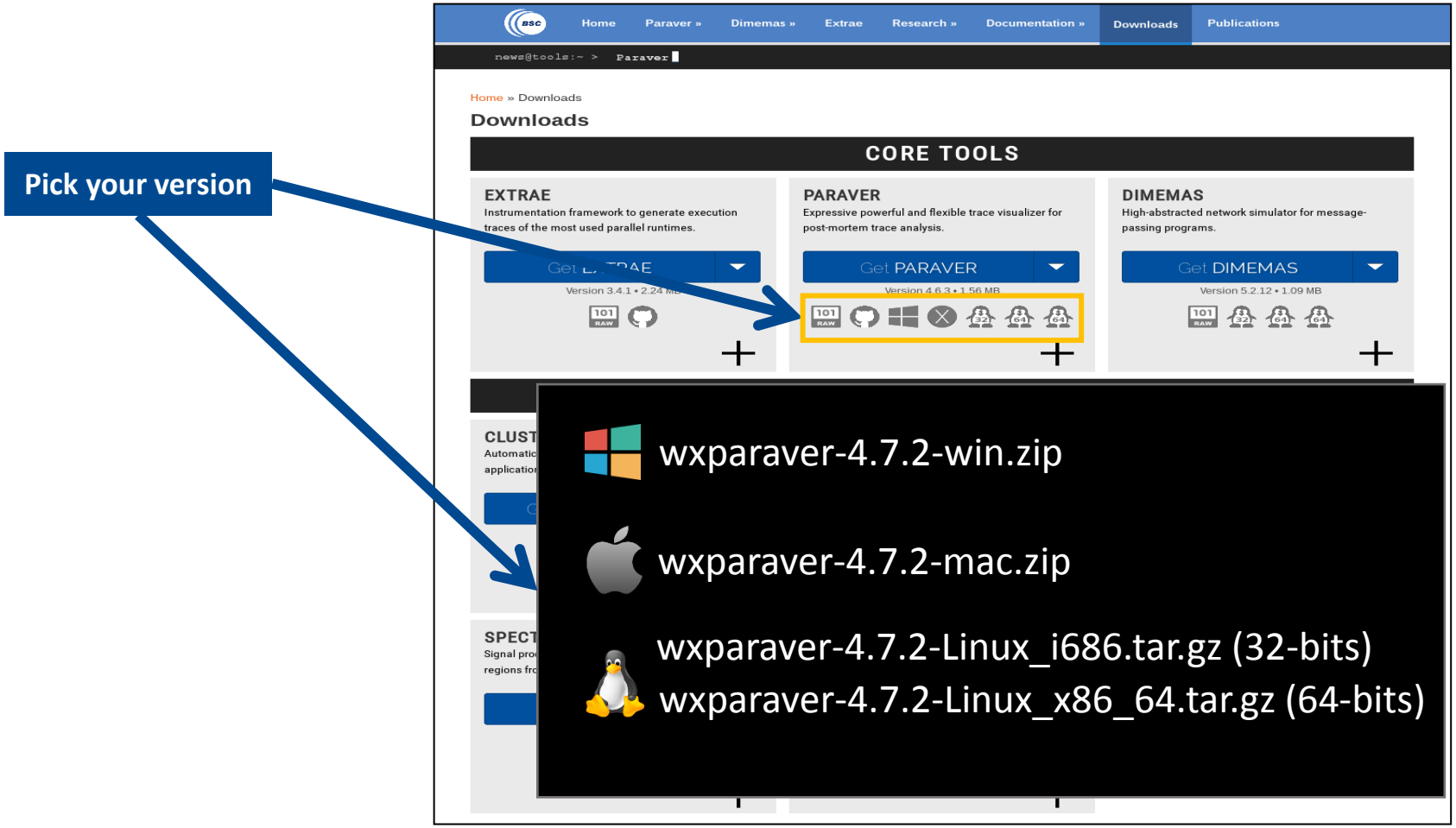
EoCoE-II Performance Evaluation Workshop

Extrae and Paraver Hands-on

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

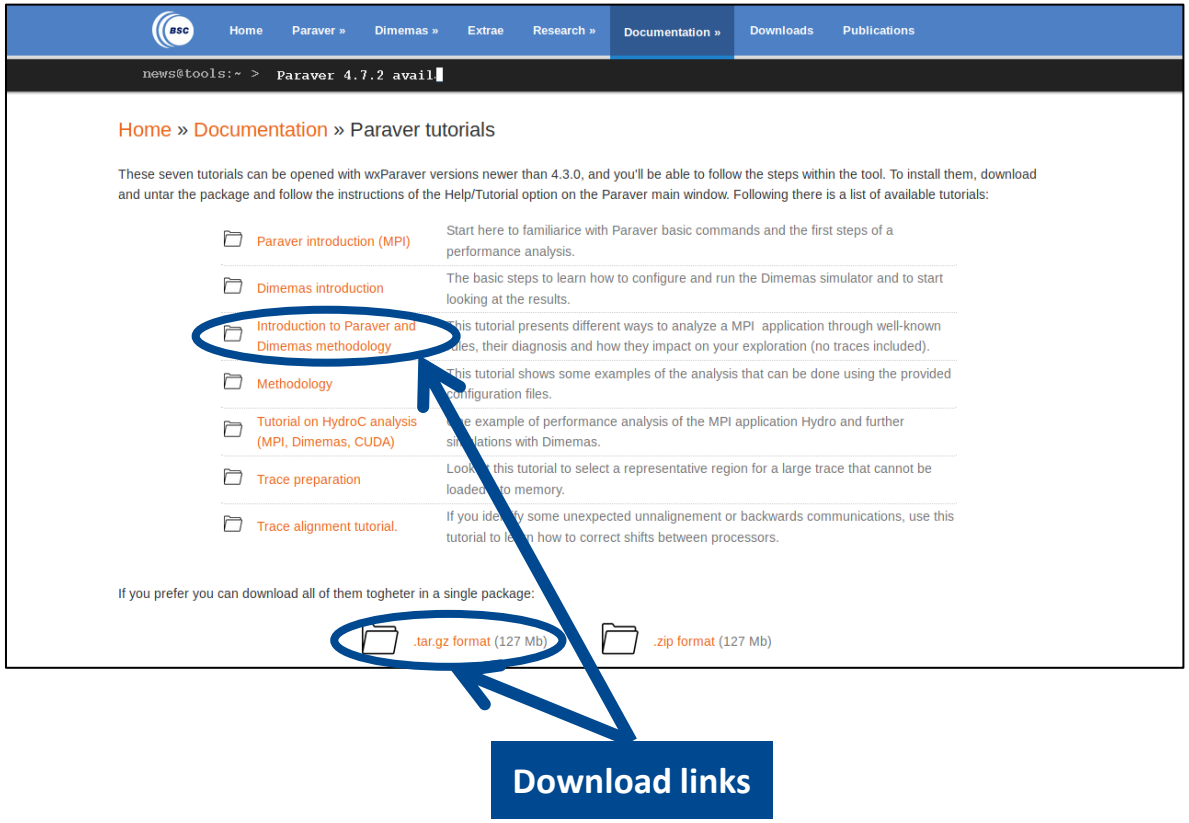
Install Paraver

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



Install Paraver (II)

- Download tutorials:
  - Documentation
    - Paraver tutorials



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
```

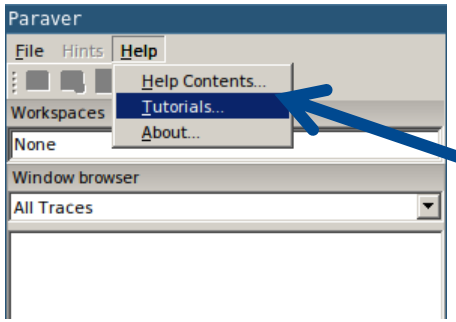
Check that everything works

- Start Paraver

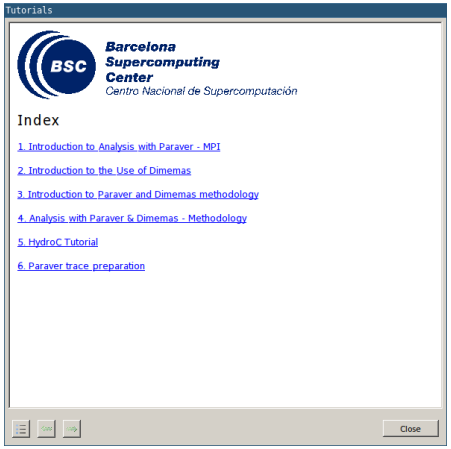
@ your laptop

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

- Check that tutorials are available



Click on Help → Tutorials



- Trouble installing locally? Remote open from emmy

@emmy.rrze.uni-erlangen.de

```
➢ ssh -Y <USER>@emmy.rrze.uni-erlangen.de
➢ source ~j75n0024/tools/setup.sh
> wxparaver
```

Log in to emmy

@ your laptop

```
➢ > ssh -Y <USER>@emmy.rrze.uni-erlangen.de
```

- Copy the examples to your home folder:

@emmy.rrze.uni-erlangen.de

```
> cp -r ~j75n0024/BSC-handson $HOME

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

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	emmy
Event	150 - 200 ns	190 ns
Event + PAPI	750 ns – 1 us	1405 ns
Event + callstack (1 level)	600 ns	7 us
Event + callstack (6 levels)	2 us	15 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

@emmy.rrze.uni-erlangen.de

```
> vi $HOME/BSC-handson/apps/lulesh/job.8
```

job.27

```
#!/bin/bash -l
#
#PBS -l nodes=1:ppn=40       ,walltime=00:05:00
#PBS -N lulesh_33

module load openmpi/3.1.3-gcc8.2.0+q6rr7p

cd  ${PBS_O_WORKDIR}




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

Step 1: Adapt the job script to load Extrae

@emmy.rrze.uni-erlangen.de

```
> vi $HOME/BSC-handson/apps/lulesh/job.8
```

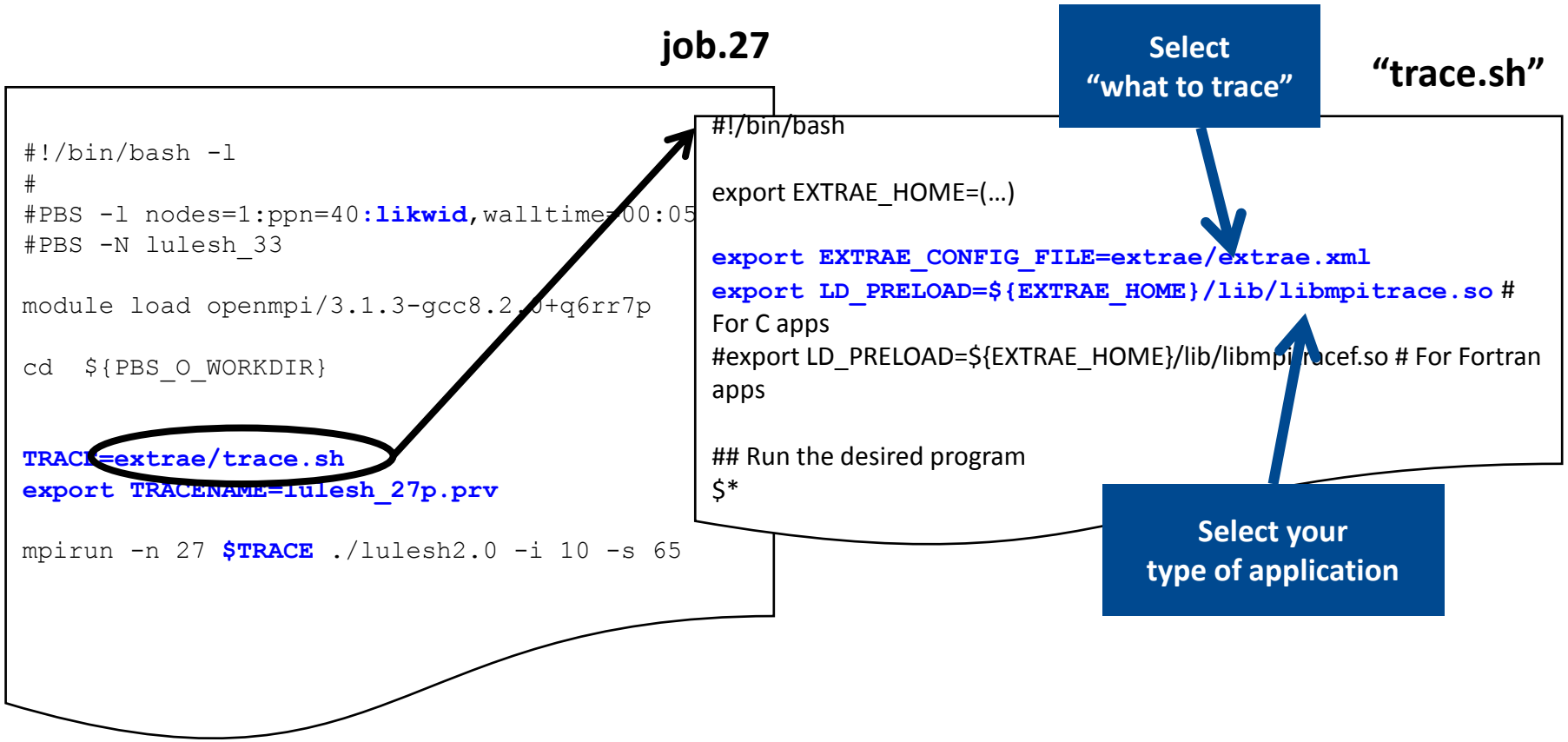
job.27

```
#!/bin/bash -l
#
#PBS -l nodes=1:ppn=40:likwid,walltime=00:05:00
#PBS -N lulesh_33

module load openmpi/3.1.3-gcc8.2.0+q6rr7p

cd  ${PBS_O_WORKDIR}


TRACE=extrae/trace.sh
export TRACENAME=lulesh_27p.prv

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

Step 1: Adapt the job script to load Extrae

@emmy.rrze.uni-erlangen.de

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



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

@emmy.rrze.uni-erlangen.de

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

> qsub job.27
```

- Easy! ☺

Step 2: Extrae XML configuration

@emmy.rrze.uni-erlangen.de

```
> 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)

@emmy.rrze.uni-erlangen.de

```
> 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)

@emmy.rrze.uni-erlangen.de

```
> 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 `qstat`) you will have the trace (3 files):

@emmy.rrze.uni-erlangen.de

```
➢ ls -l $HOME/BSC-handson/apps/lulesh
   ...
   lulesh_27p.pcf
   lulesh_27p.prv
   lulesh_27p.row
```

- Trouble getting in the queues? Already available at ~/BSC-handson/traces/lulesh/lulesh_27p.*

- Now let's look into it !

First steps of analysis

- Copy the trace to your laptop

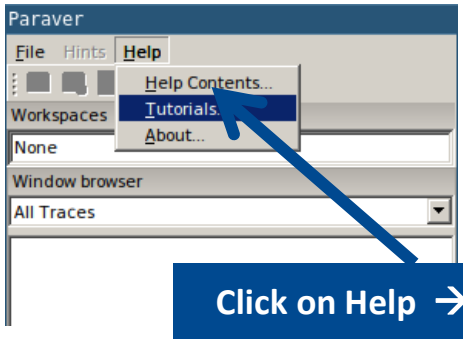
@ your laptop

```
> scp <USER>@emmy.rrze.uni-erlangen.de: \
    BSC_handson/apps/lulesh/lulesh_27p.* $HOME
```

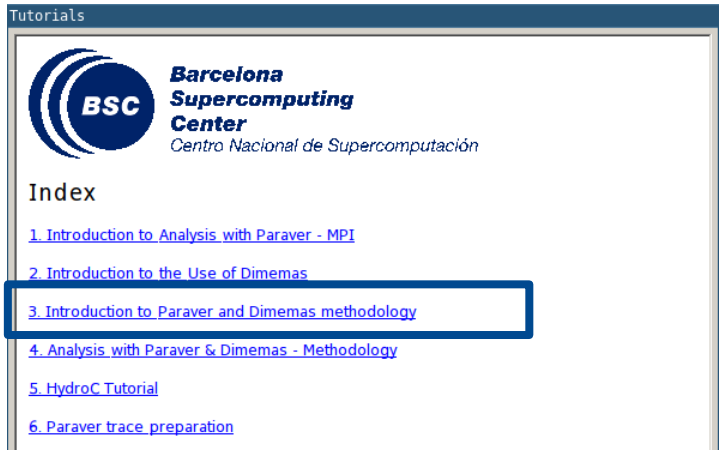
- Load the trace with Paraver

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

- Follow Tutorial #3

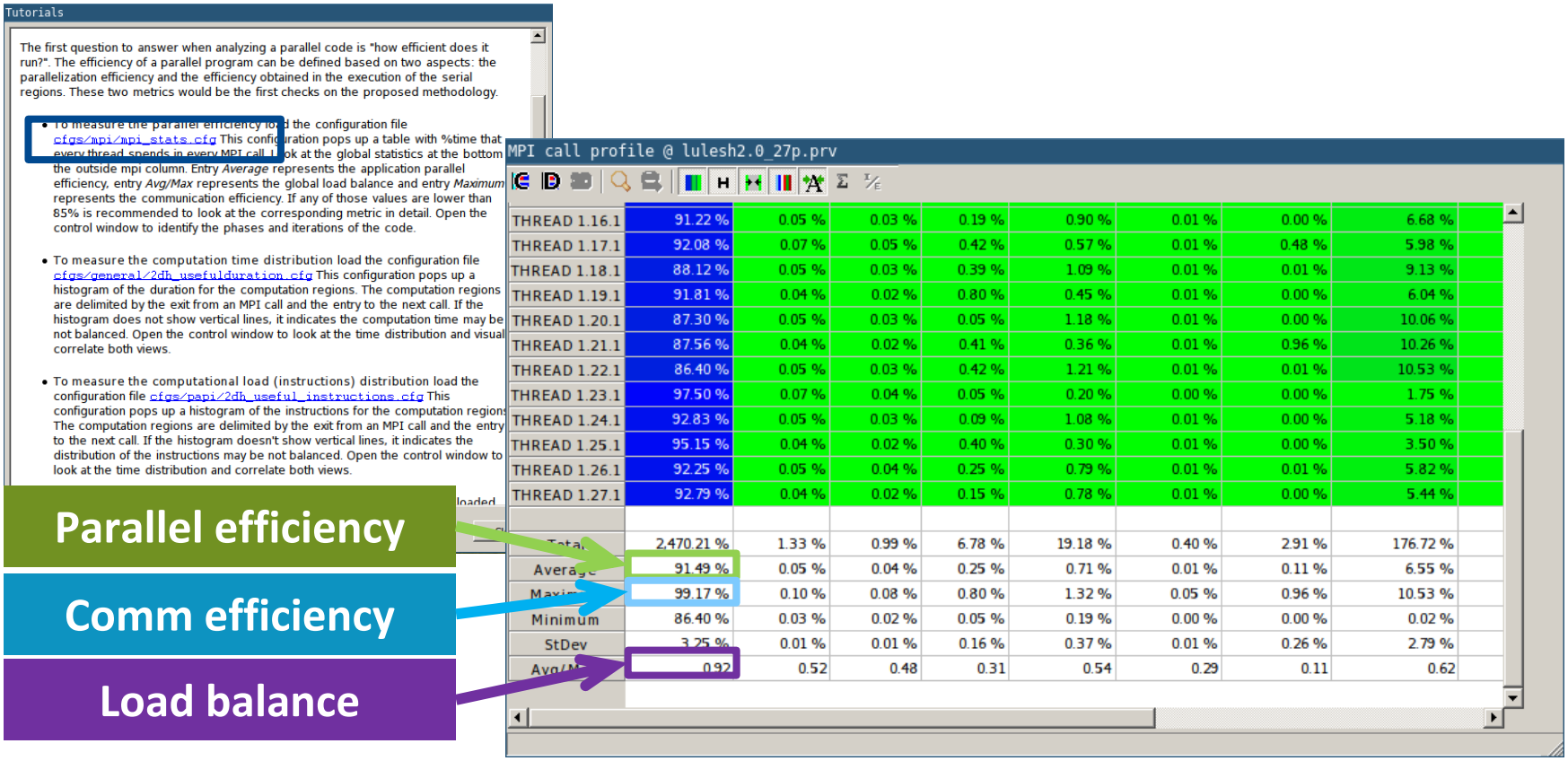


Click on Help → Tutorials



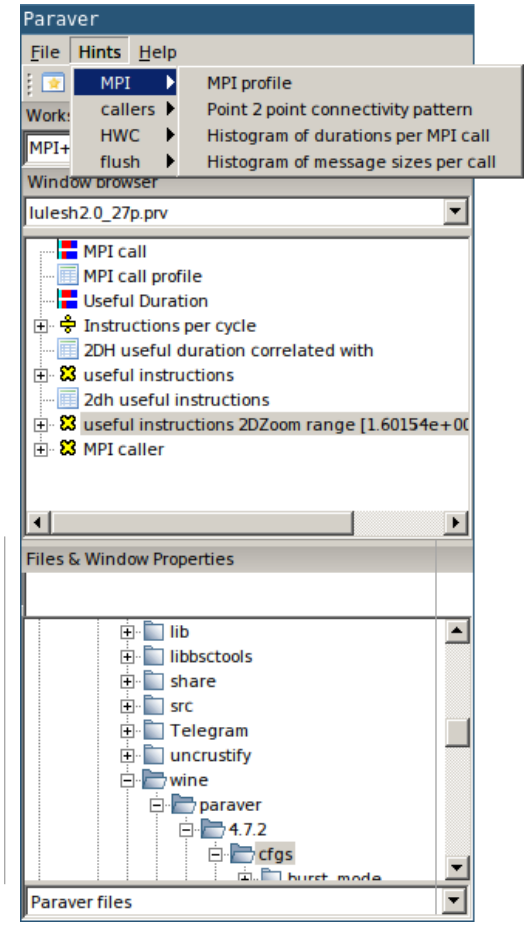
Measure the parallel efficiency

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



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