BSCTools Hands-On

Judit Giménez, Germán Llort

Barcelona Supercomputing Center

Getting a trace with Extrae

Extrae features

- Parallel programming models
  - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Intel MIC…
- Performance Counters
  - Using PAPI and PMAPI interfaces
- Link to source code
  - Callstack at MPI routines
  - OpenMP outlined routines and their containers
  - Selected user functions
- Periodic samples
- 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)

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 it!

- 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 Hornet and copy the example files to your home directory

```
> ssh <user>@hornet.hww.de

> cp -r ~xhpgl/paraver-tutorial $HOME

> cd $HOME/paraver-tutorial
```

Example: Adapt the job script to load Extrae with LD_PRELOAD

run-27procs.pbs

```
#!/bin/bash
#PBS -l nodes=2:ppn=24
#PBS -l walltime=00:10:00
#PBS -N LULESH
#PBS -e $PBS_JOBID.err
#PBS -o $PBS_JOBID.out
#PBS -q R_vihps04

cd $PBS_O_WORKDIR
module switch PrgEnv-cray PrgEnv-gnu

time aprun -n 27 ./lulesh
```

Example: Adapt the job script to load Extrae with LD_PRELOAD

run-and-trace-27procs-24+3.pbs

```
#!/bin/bash
#PBS -l nodes=2:ppn=24
#PBS -l walltime=00:10:00
#PBS -N LULESH
#PBS -e $PBS_JOBID.err
#PBS -o $PBS_JOBID.out
#PBS -q R_vihps04

cd $PBS_O_WORKDIR
module switch PrgEnv-cray PrgEnv-gnu

export TRACE_NAME=lulesh.27procs.24+3.prv

time aprun -n 27 ./trace.sh ./lulesh
```

Example: Adapt the job script to load Extrae with LD_PRELOAD

run-and-trace-27procs-24+3.pbs

```
#!/bin/bash
#PBS -l nodes=2:ppn=24
#PBS -l walltime=00:10:00
#PBS -N LULESH
#PBS -e $PBS_JOBID.err
#PBS -o $PBS_JOBID.out
#PBS -q R_vihps04

cd $PBS_O_WORKDIR
module switch PrgEnv-cray PrgEnv-gnu

export TRACE_NAME=lulesh.27procs.24+3.prv

time aprun -n 27 ./trace.sh ./lulesh
```

trace.sh

```
#!/bin/bash

rm -rf TRACE*.sym TRACE*.mpits TRACE*.spawn set-*

# Configure Extrae
export EXTRAE_HOME=~xhpgl/tools/extrae/default
source $EXTRAE_HOME/etc/extrae.sh
export EXTRAE_CONFIG_FILE=./extrae-config.xml

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

# Run the program
$*
```

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

Extrae XML configuration: extrae_config.xml

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

<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 MPI calls + HW counters

Trace call-stack events @ MPI calls

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,PAPI_L2_DCM
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L3_TCM,RESOURCE_STALLS:SB,RESOURCE_STALLS:ROB
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_SR_INS,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_LD_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,RESOURCE_STALLS,RESOURCE_STALLS:RS
    </set>
  </cpu>

  <network enabled="no" />

  <resource-usage enabled="no" />

  <memory-usage enabled="no" />
</counters>
```

Define which HW counters are measured

Extrae XML configuration: extrae_config.xml (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 it!

- Submit your job

```
> cd $HOME/paraver-tutorial
> qsub run-and-trace-27procs-24+3.pbs
```

- Copy the resulting trace to your laptop

```
> gzip lulesh.27procs.24+3.prv
> scp <user>@hornet.hww.de:paraver-tutorial/lulesh.27procs.24+3.\* $HOME
```

- Or click to download from: (3 files, lulesh.27procs.24+3.[prv.gz | row | pcf])
  https://www.dropbox.com/sh/42teru7vunc4yot/AACy1ckEr2sUKgp-3re3IK8Aa?dl=0

Installing Paraver

Install Paraver in your laptop

- Download the Paraver binaries

Linux 64 bits	Click to download Paraver for Linux 64 bits from: https://www.dropbox.com/s/hu6u2pfoa8zx0ha/wxparaver-4.5.6-linux-x86_64.tar.gz?dl=0
Linux 32 bits	Click to download Paraver for Linux 32 bits from: https://www.dropbox.com/s/jtkik0mrazu0ue2/wxparaver-4.5.6-linux-x86_32.tar.gz?dl=0
Mac	Click to download Paraver for Mac from: https://www.dropbox.com/s/7vq6ereyo5incb6/wxparaver_mac.zip?dl=0
Windows	Click to download Paraver for Windows from: https://www.dropbox.com/s/pwcruoezjuyoqfc/wxparaver-4.5.6-win.zip?dl=0

Install Paraver in your laptop (II)

- Uncompress into your home directory

```
> tar xvfz wxparaver-4.5.6-linux-x86_64.tar.gz -C $HOME

> cd $HOME

> ln -s wxparaver-4.5.6-linux-x86_64 paraver
```

- Download Paraver tutorials and uncompress into the Paraver directory

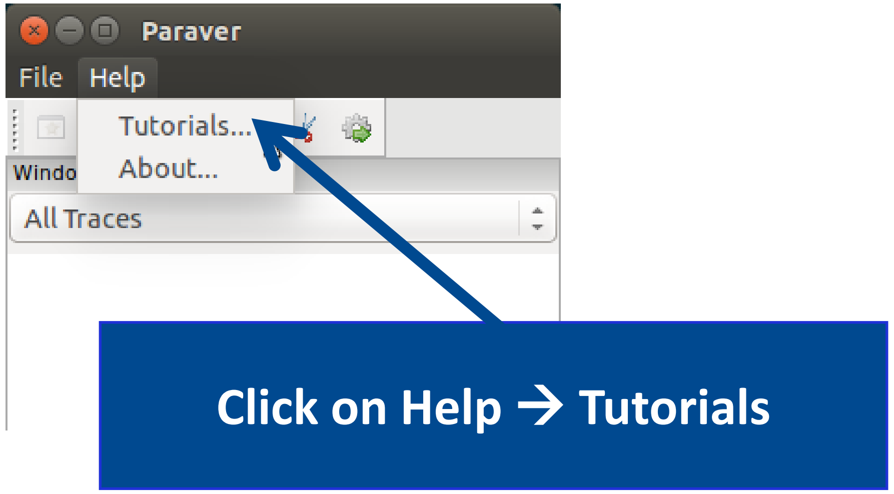
```
Click to download Paraver tutorials from:
https://www.dropbox.com/s/ece15xzqxa4j2qo/paraver-tutorials-20141016.tar.gz?dl=0

> tar xvfz paraver-tutorials-20141016.tar.gz -C $HOME/paraver --strip=1
```

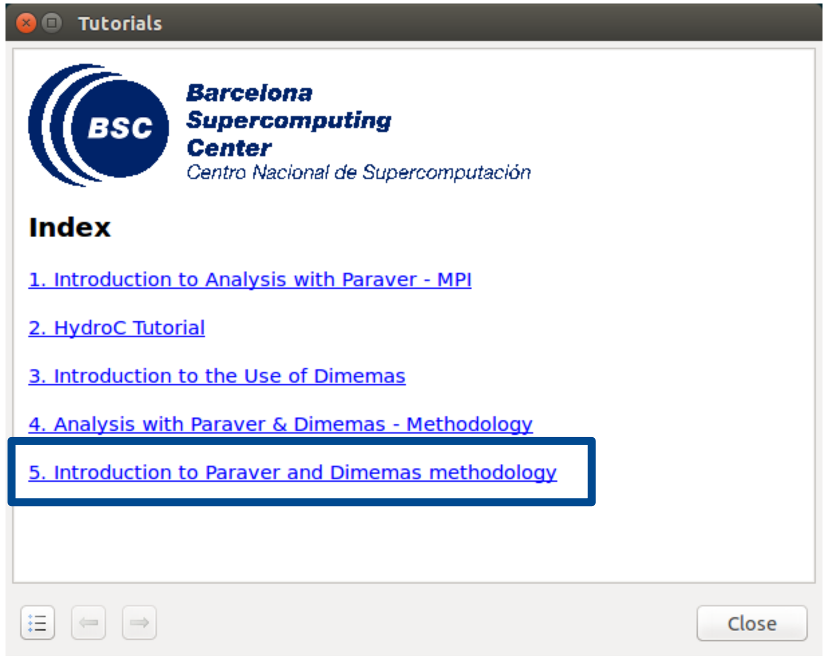
First steps of analysis

- Load the trace with Paraver

```
> $HOME/paraver/bin/wxparaver lulesh.27procs.24+3.prv
```

- Open Tutorial #5
  - Help → Tutorials



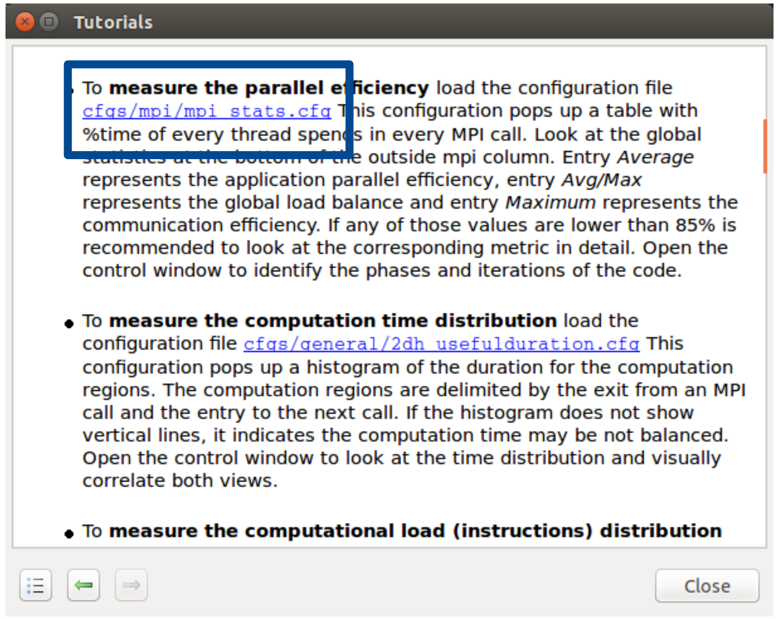

Tutorials
Barcelona Supercomputing Center
Centro Nacional de Supercomputación
Close

Index

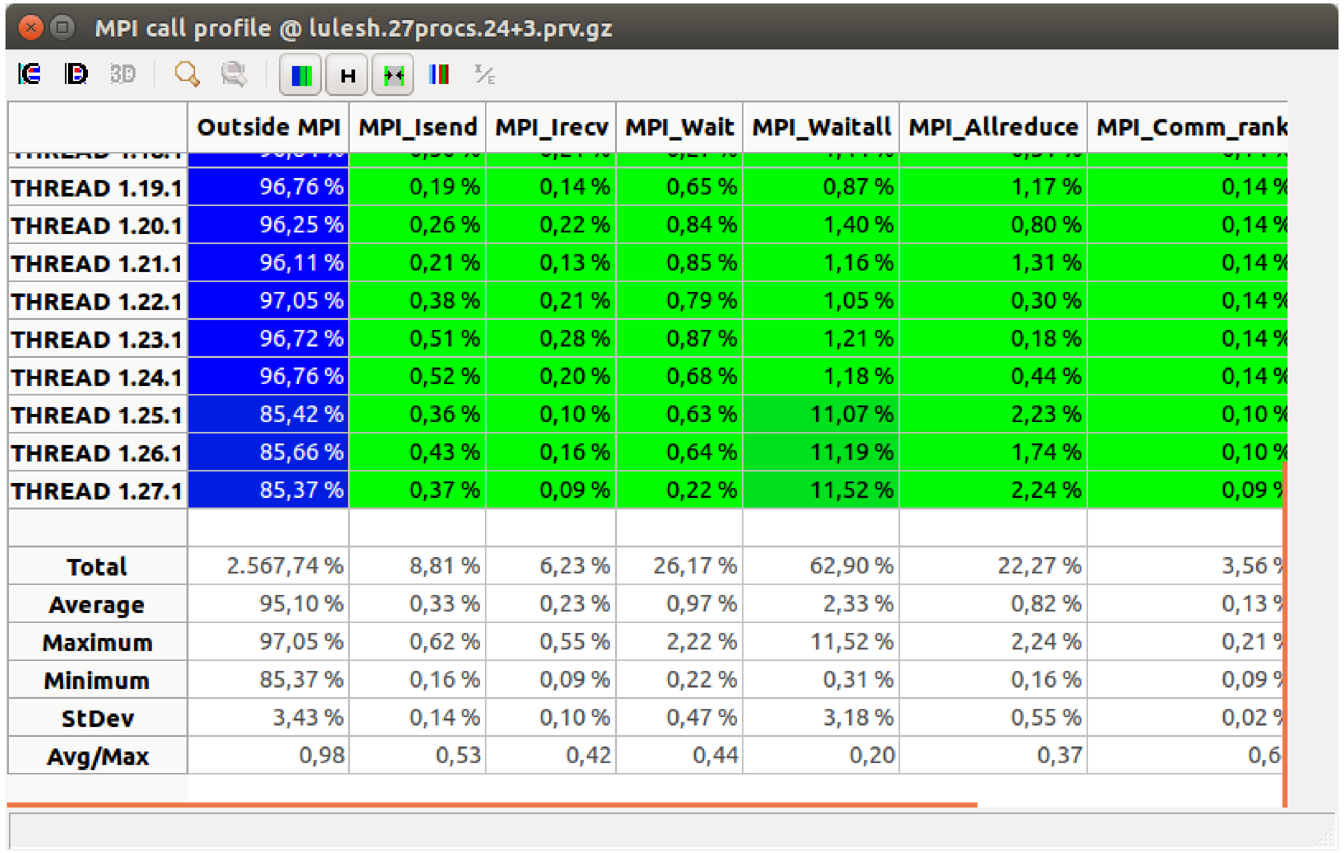
1. Introduction to Analysis with Paraver - MPI
2. HydroC Tutorial
3. Introduction to the Use of Dimemas
4. Analysis with Paraver & Dimemas - Methodology
5. Introduction to Paraver and Dimemas methodology

Measure the parallel efficiency

- Click on the “mpi_stats.cfg”



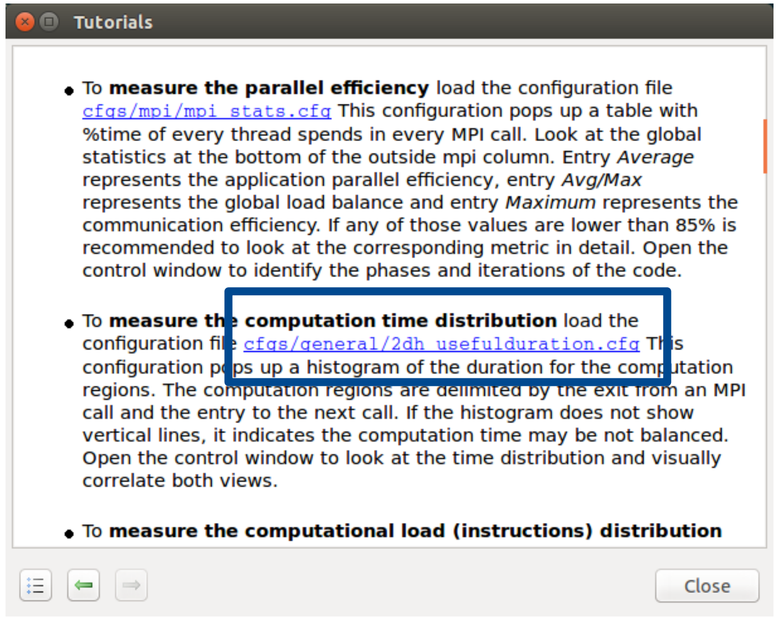
- To **measure the parallel efficiency** load the configuration file cfgs/mpi/mpi_stats.cfg This configuration pops up a table with %time of every thread spends in every MPI call. Look at the global statistics at the bottom of the outside mpi column. Entry *Average* represents the application parallel 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**



	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank
THREAD 1.19.1	96,76 %	0,19 %	0,14 %	0,65 %	0,87 %	1,17 %	0,14 %
THREAD 1.20.1	96,25 %	0,26 %	0,22 %	0,84 %	1,40 %	0,80 %	0,14 %
THREAD 1.21.1	96,11 %	0,21 %	0,13 %	0,85 %	1,16 %	1,31 %	0,14 %
THREAD 1.22.1	97,05 %	0,38 %	0,21 %	0,79 %	1,05 %	0,30 %	0,14 %
THREAD 1.23.1	96,72 %	0,51 %	0,28 %	0,87 %	1,21 %	0,18 %	0,14 %
THREAD 1.24.1	96,76 %	0,52 %	0,20 %	0,68 %	1,18 %	0,44 %	0,14 %
THREAD 1.25.1	85,42 %	0,36 %	0,10 %	0,63 %	11,07 %	2,23 %	0,10 %
THREAD 1.26.1	85,66 %	0,43 %	0,16 %	0,64 %	11,19 %	1,74 %	0,10 %
THREAD 1.27.1	85,37 %	0,37 %	0,09 %	0,22 %	11,52 %	2,24 %	0,09 %
Total	2.567,74 %	8,81 %	6,23 %	26,17 %	62,90 %	22,27 %	3,56 %
Average	95,10 %	0,33 %	0,23 %	0,97 %	2,33 %	0,82 %	0,13 %
Maximum	97,05 %	0,62 %	0,55 %	2,22 %	11,52 %	2,24 %	0,21 %
Minimum	85,37 %	0,16 %	0,09 %	0,22 %	0,31 %	0,16 %	0,09 %
StDev	3,43 %	0,14 %	0,10 %	0,47 %	3,18 %	0,55 %	0,02 %
Avg/Max	0,98	0,53	0,42	0,44	0,20	0,37	0,6

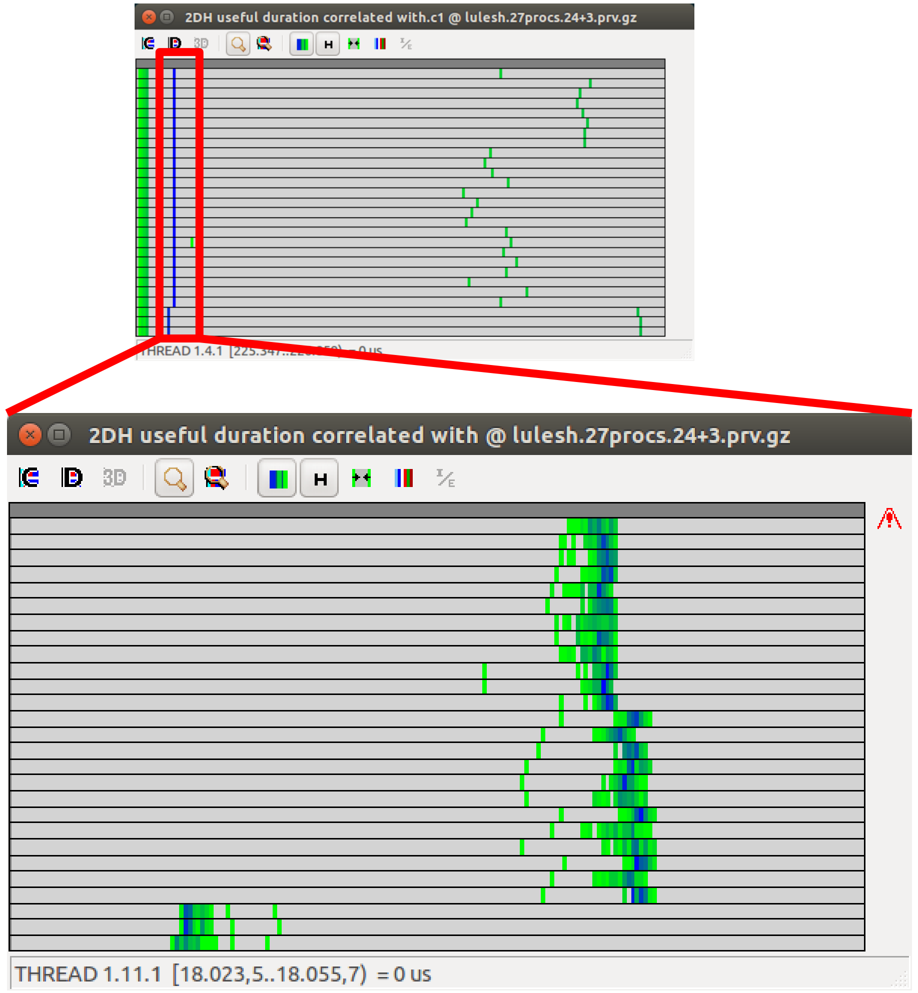
Measure the computation time distribution

- 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


2DH useful duration correlated with.c1 @ lulesh.27procs.24+3.prv.gz
THREAD 1.4.1 [225.347...
2DH useful duration correlated with @ lulesh.27procs.24+3.prv.gz
3D
H
THREAD 1.11.1 [18.023,5..18.055,7) = 0 us

Compare with other configurations

- Nodes more balanced (14 processes in node 1, 13 in node 2)
  - If you want to get the trace…

```
> cd $HOME/paraver-tutorial
> qsub run-and-trace-27procs-14+13.pbs
```

- CPU sockets more balanced (max 7 processes per socket)
  - If you want to get the trace…

```
> cd $HOME/paraver-tutorial
> qsub run-and-trace-27procs-14+13-7PExSOCKET.pbs
```

- Or click to download from:

https://www.dropbox.com/sh/42teru7vunc4yot/AACy1ckEr2sUKgp-3re3IK8Aa?dl=0

Thank you!