



**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*



# Programming Distributed Computing Platforms with COMPSs

Workflows & Distributed Computing Group

24-25/01/2023

Barcelona

# Supercomputers Hands-on



**Barcelona  
Supercomputing  
Center**

Centro Nacional de Supercomputación

# Supercomputers Hands-on

- Execution in MareNostrum 4
- Tracing Analysis Overview

# Execution in MareNostrum 4

- How to connect to MareNostrum?
  - **ssh -X [nct01XXX@mn1.bsc.es](mailto:nct01XXX@mn1.bsc.es)**

(Where XXX is 191–195 or 201-285)

Password: **pvcte9.XXX**

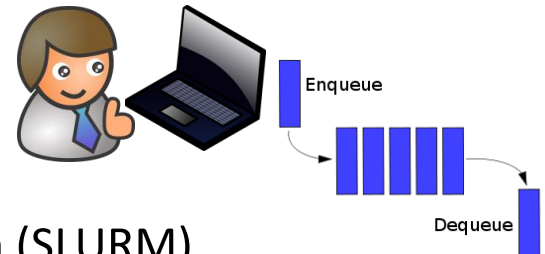
- Load COMPSs:
  - **export COMPSS\_PYTHON\_VERSION=3**
  - **module load COMPSs/3.1**
- Where is the source code?
  - **cd**
  - **cp -r /gpfs/home/nct00/nct00006/source .**
- Available editors
  - **vi**
  - **emacs**



# WordCount@ Sequential

- Remember the dataset path
- How to launch with python sequentially?
  - > cd source/0\_wordcount/src
  - > python wordcount.py /gpfs/home/nct00/nct00006/dataset/dataset\_4f\_16mb
  - Results:

```
user@login:~> python wordcount.py /path/to/dataset/  
Elapsed Time (s): 3.959941864014  
Words: 10206202
```



- Submit jobs to MareNostrum 4:
  - All jobs should be submitted to the queuing system (SLURM)
  - We will use the **pycompss-cli**
  - Useful commands:
    - squeue – This command shows the status of the job.
    - scancel jobId – This command kills a job with id 'jobId'.



# Execution in MareNostrum 4 - HandsOn

- `launch_with_pycompss.sh`

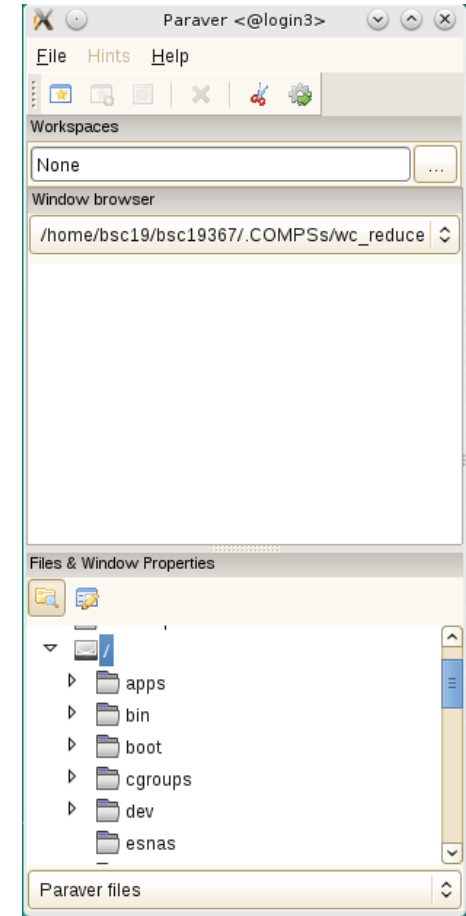
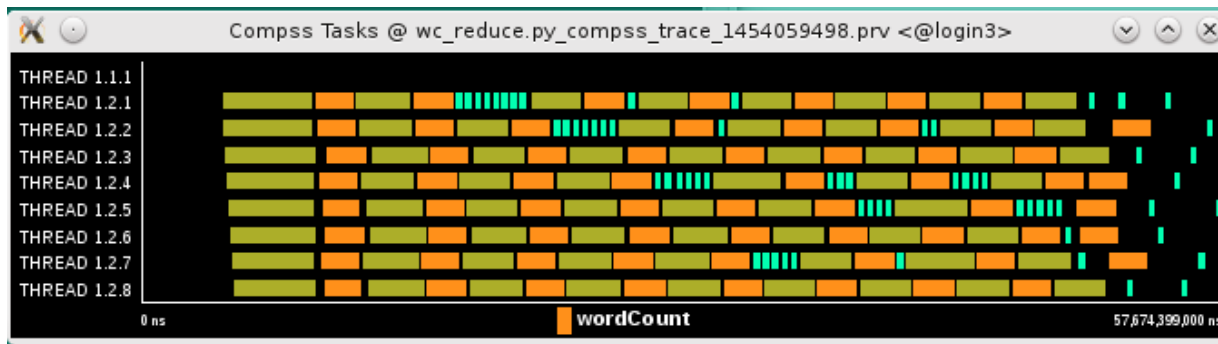
```
#!/bin/bash
```

```
pycompss job submit \  
  --qos=training \  
  --num_nodes=2 \  
  --exec_time=10 \  
  --reservation=COMPSS2023 \  
  --lang=python \  
  --tracing=true \  
  --graph=true \  
  /home/nct01/nct01XXX/source/0_wordcount/src/wordcount.py /gpfs/home/nct00/nct00006/dataset/dataset_64f_16mb
```

- Parameters:
  - `num_nodes`: amount of nodes where to execute (1 master + 1 worker).
  - Dataset path: `/gpfs/home/nct00/nct00006/dataset/dataset_64f_16mb`
- How to execute with PyCOMPSs?
  - `chmod 755 launch_with_pycompss.sh`
  - `./launch_with_pycompss.sh`

# Wordcount @ Performance Analysis

- Paraver is the BSC tool for trace visualization
  - Trace events are encoding in Paraver (.prv) format by Extrae
  - Paraver is a powerful tool for trace visualization.
  - An experimented user could obtain many different views of the trace events.
- For more information about Paraver visit:
  - <https://tools.bsc.es/paraver>



# Wordcount @ Performance Analysis

- COMPSs can generate post-execution traces of the distributed execution of the application
  - Useful for performance analysis and diagnosis
- How it works?
  - Task execution and file transfers are application events
  - An XML file is created at workers to keep track of these events
  - At the end of the execution all the XML files are merged to get the final trace file
  - COMPSs uses Extrae tool to dynamically instrument the application
    - In a worker:
      - Extrae keeps track of the events in an intermediate file
    - In the master:
      - Extrae merges the intermediate files to get the final trace file



# Wordcount @ Performance Analysis

----- Executing wc\_reduce.py -----

**Welcome to Extrae 3.5.3**

**Extrae: Generating intermediate files for Paraver traces.**

**Extrae: Intermediate files will be stored in /gpfs/home/nct01/nct01XXX/sources/examples**

**Extrae: Tracing buffer can hold 500000 events**

**Extrae: Tracing mode is set to: Detail.**

**Extrae: Successfully initiated with 1 tasks**

[ API] - Deploying COMPSs Runtime v3.1

[ API] - Starting COMPSs Runtime v3.1

...

[ API] - No more tasks for app 0

[ API] - Getting Result Files 0

[ API] - Execution Finished

...

**Extrae: Application has ended. Tracing has been terminated.**

**merger: Output trace format is: Paraver**

**merger: Extrae 3.5.3**

**mpi2prv: Selected output trace format is Paraver**

**mpi2prv: Parsing intermediate files**

**mpi2prv: Generating tracefile (intermediate buffers of 745642 events)**

**mpi2prv: Congratulations! ./trace/wc\_reduce.py\_compss\_trace\_1453885329.prv has been generated.**

Extrae starts before  
the user application execution

COMPSs runtime starts

COMPSs runtime ends

The application finishes and  
the tracing process ends

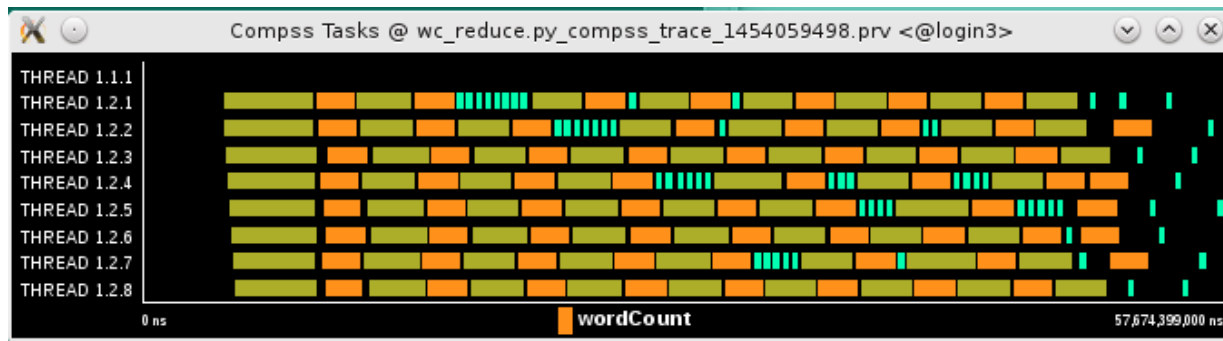
The merge process starts

Intermediate trace files  
are processed

The final trace file is  
generated

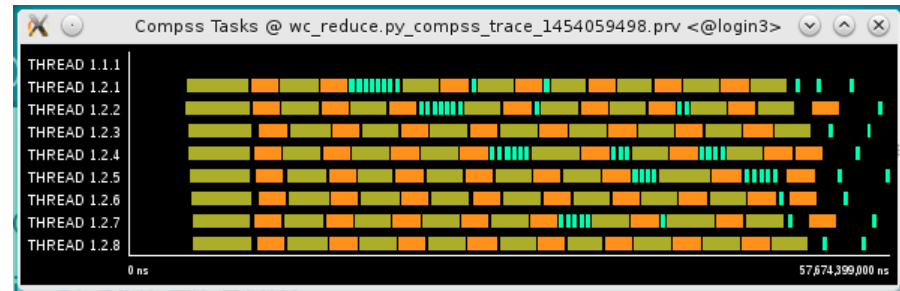
# WordCount @ Performance Analysis

- Open Paraver
    - `module load paraver`
    - `cd $HOME/.COMPSSs/wordcount.py_01`
    - `wxparaver trace/*.prv`
  - COMPSS provides some configuration files to automatically obtain the view of the trace
    - File/Load Configuration...
- (/gpfs/apps/MN4/COMPSSs/3.1/Dependencies/paraver/cfgs/compss\_tasks.cfg

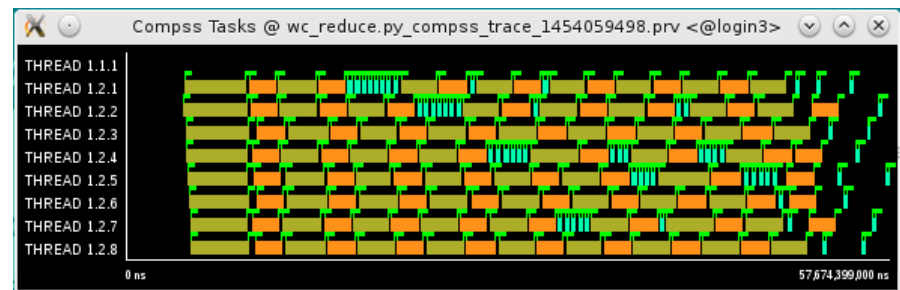


# Wordcount @ Performance Analysis

- Fit window
  - Right click on the trace window
  - Fit Semantic Scale/ Fit Both

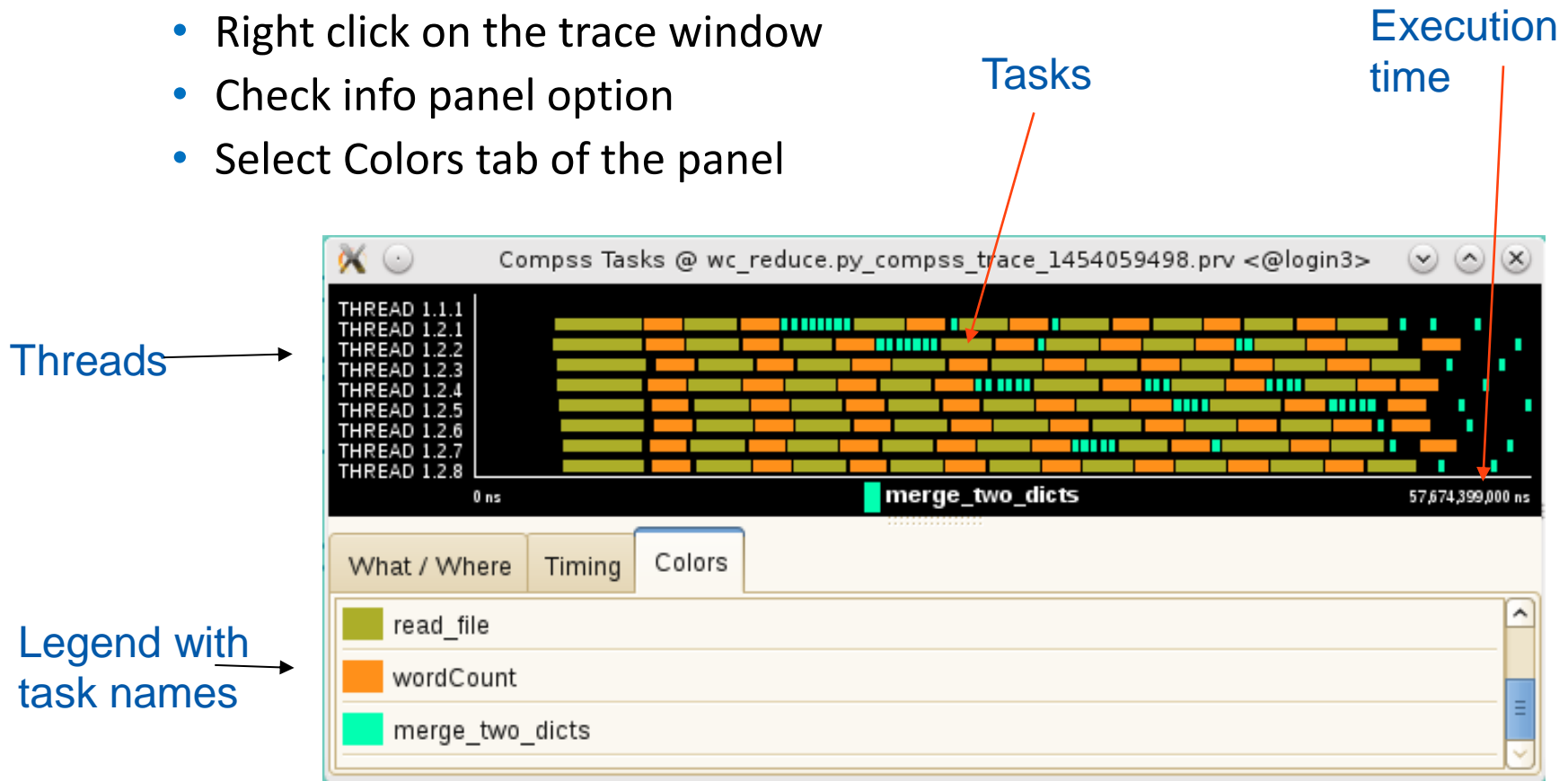


- View Event flags
  - Right click on the trace window
  - View / Event Flags



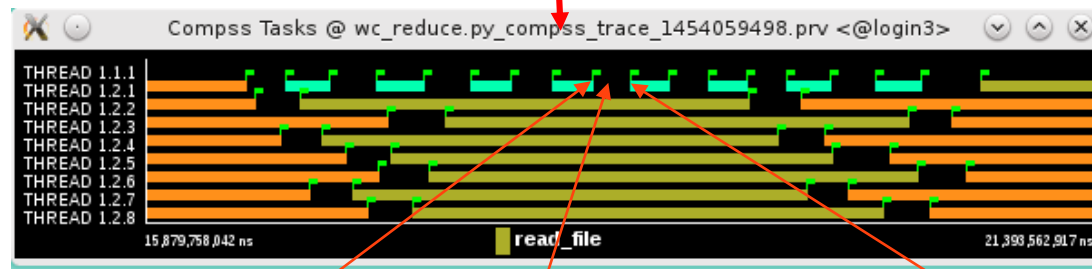
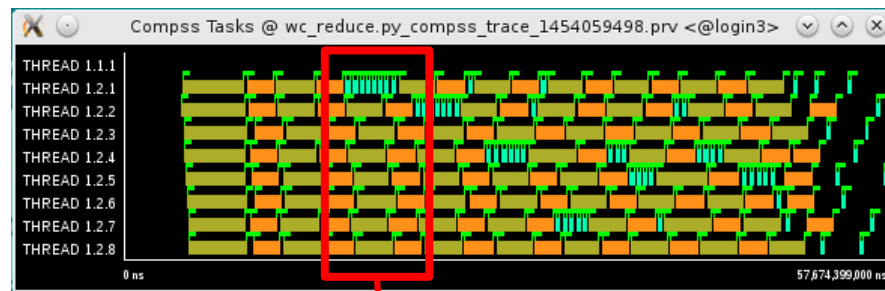
# Wordcount @ Performance Analysis

- Show info Panel
  - Right click on the trace window
  - Check info panel option
  - Select Colors tab of the panel



# Wordcount @ Performance Analysis

- Zoom to see details
  - Select a region in the trace window to see in detail
  - And repeat the process until the needed zoom level
  - The undo zoom option is in the right click panel



Previous task  
ends

Processor is  
idle

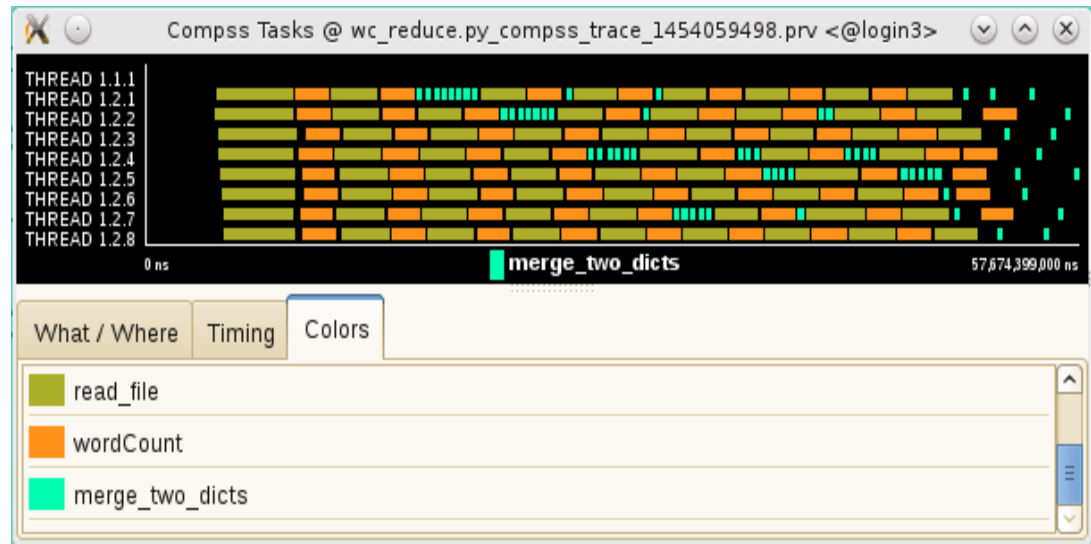
New task starts

# Wordcount @ Performance Analysis

- Summarizing:
  - Lines in the trace:
    - THREAD 1.1.X are the master threads
    - THREAD 1.X.Y are the worker threads

- Meaning of the colours:
  - Black: idle
  - Other colors: task running
    - see the color legend

- Flags (events):
  - Start / end of task





**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación



EXCELENCIA  
SEVERO  
OCHOA

# THANK YOU!

[support-compss@bsc.es](mailto:support-compss@bsc.es)

[www.bsc.es](http://www.bsc.es)