www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Performance analysis with BSC-tools

Judit Giménez, Juan González, Pedro González, Jesús Labarta, Germán Llort, Eloy Martínez, Xavier Pegenaute

Presenter: Harald Servat
BSC / Computer Sciences

PATC Course
Parallel Programming Workshop
26-30th November @ Barcelona

Performance analysis tools objective

Help generate hypotheses

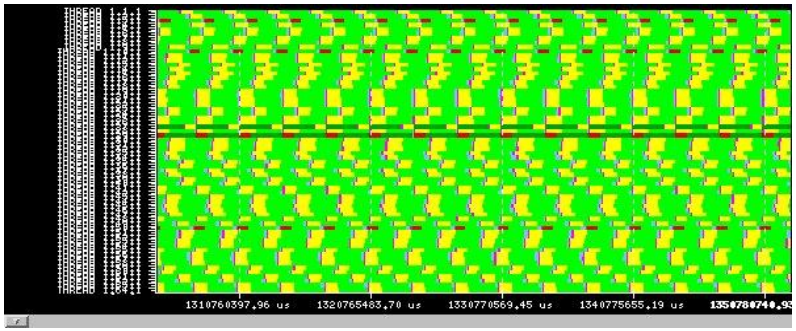
Help validate hypotheses

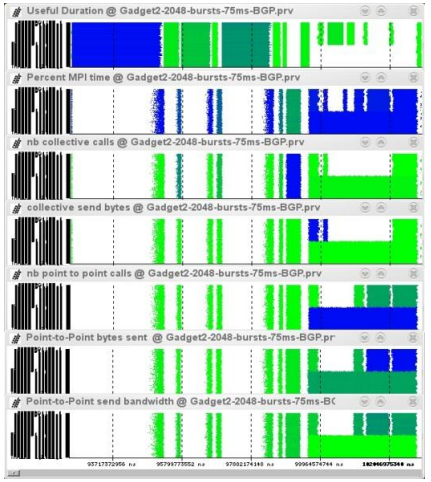
qualitatively

quantitetively

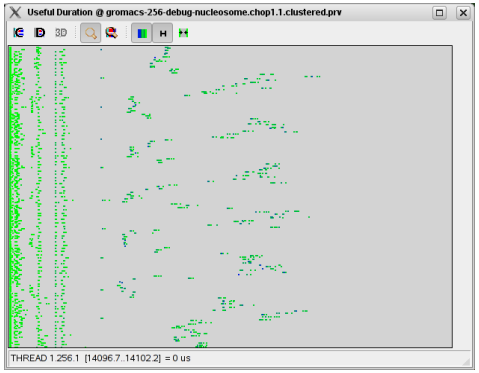
Our tools

- Since 1991
- Based on traces
- Open Source
  - http://www.bsc.es/paraver





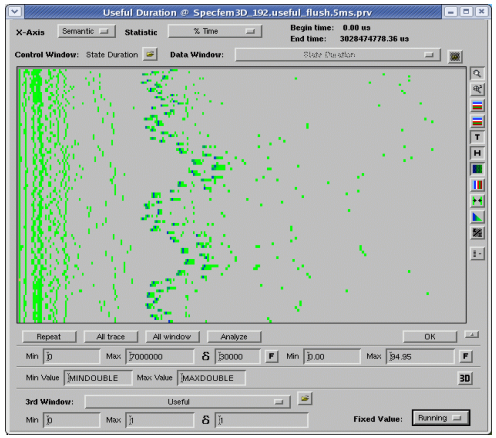
- Core tools
  - Paraver (paramedir) – offline trace analysis
  - Dimemas – message passing simulator
  - Extrae – instrumentation



- Focus
  - Detail, flexibility and intelligence

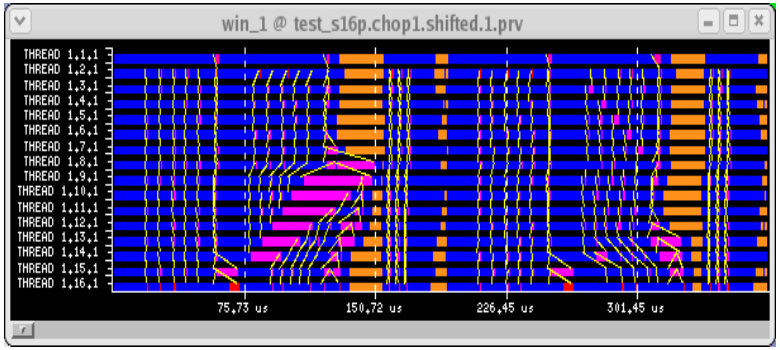
Why traces?

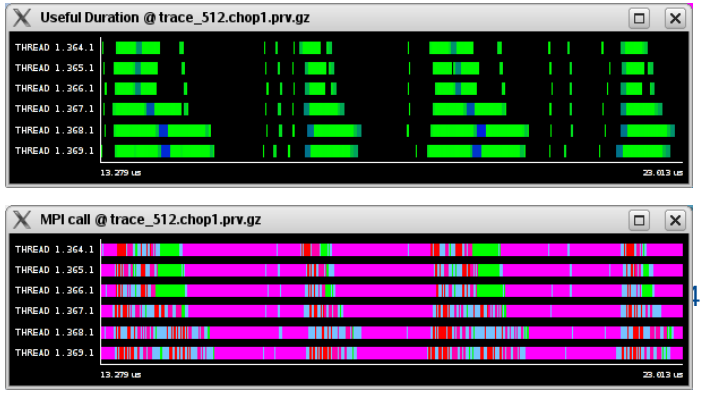
- Detail and variability are important
  - Along time, across processors



- Highly non-linear systems
  - Small effects may have huge impact

- Useful to develop/test analysis techniques





… at the end of the session

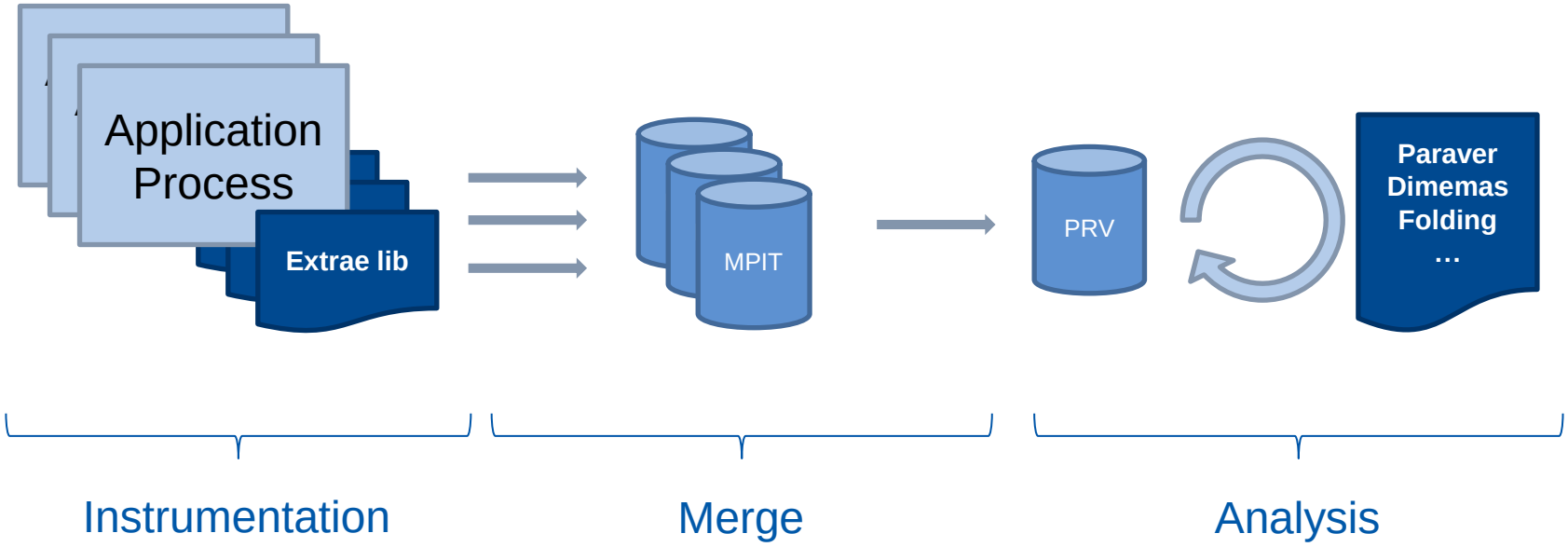
- One image is worth 1000 words!
  - Learning to read it pays off!

- Do not speculate about your code performance, look at it!

Outline

- **Extrae**
- Paraver
  - General description
  - Examples
- Research topics
  - Structure detection
  - On-line analysis
  - Folding
- Hands-on session

Trace Generation Workflow



Extrae

- Parallel programming model runtime
  - MPI, OpenMP, pthreads, *Ss, CUDA, MIC…
- Counters
  - CPU counters
    - Using PAPI and PMAPI interfaces
  - Network counters
  - OS counters
- Link to source code
  - Callstack at MPI
  - OpenMP outlined routines and their containers
  - User functions selected
- Periodic samples
- User events

How does Extrae work?

- Dynamic instrumentation
  - Based on DynInst (developed by U.Wisconsin/U.Maryland)
    - Instrumentation in memory
    - Binary rewriting
- LD_PRELOAD
  - Specific libraries for each combination of runtimes
    - MPI
    - OpenMP
    - OpenMP+MPI
    - …
- Alternatives
  - OmpSs
  - Statically link (i.e., PMPI)

How to use Extrae?

- Adapt job submission script
- Adapt .xml configuration file
  - Bunch of examples distributed in the package
    - Look at $EXTRAE_HOME/share/example
- Run it!

Extrae with DynInst

appl.job

```
#!/bin/bash
…
# @ total_tasks = 4
# @ cpus_per_task = 1
# @ tasks_per_node = 4
…

srun ./trace.sh ./my_MPI_binary
```

trace.sh

```
#!/bin/sh

export EXTRAE_HOME=…
export EXTRAE_CONFIG_FILE=extrae.xml
source ${EXTRAE_HOME}/etc/extrae.sh

# Run the desired program
${EXTRAE_HOME}/bin/extrae -v $*
```

Extrae with LD_PRELOAD

appl.job

```
#!/bin/bash
…
# @ total_tasks = 4
# @ cpus_per_task = 1
# @ tasks_per_node = 4
…

srun ./trace.sh ./my_MPI_binary
```

trace.sh

```
#!/bin/sh

export EXTRAE_HOME=…
export EXTRAE_CONFIG_FILE=extrae.xml
export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitrace.so

# Run the desired 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]		✓			✓

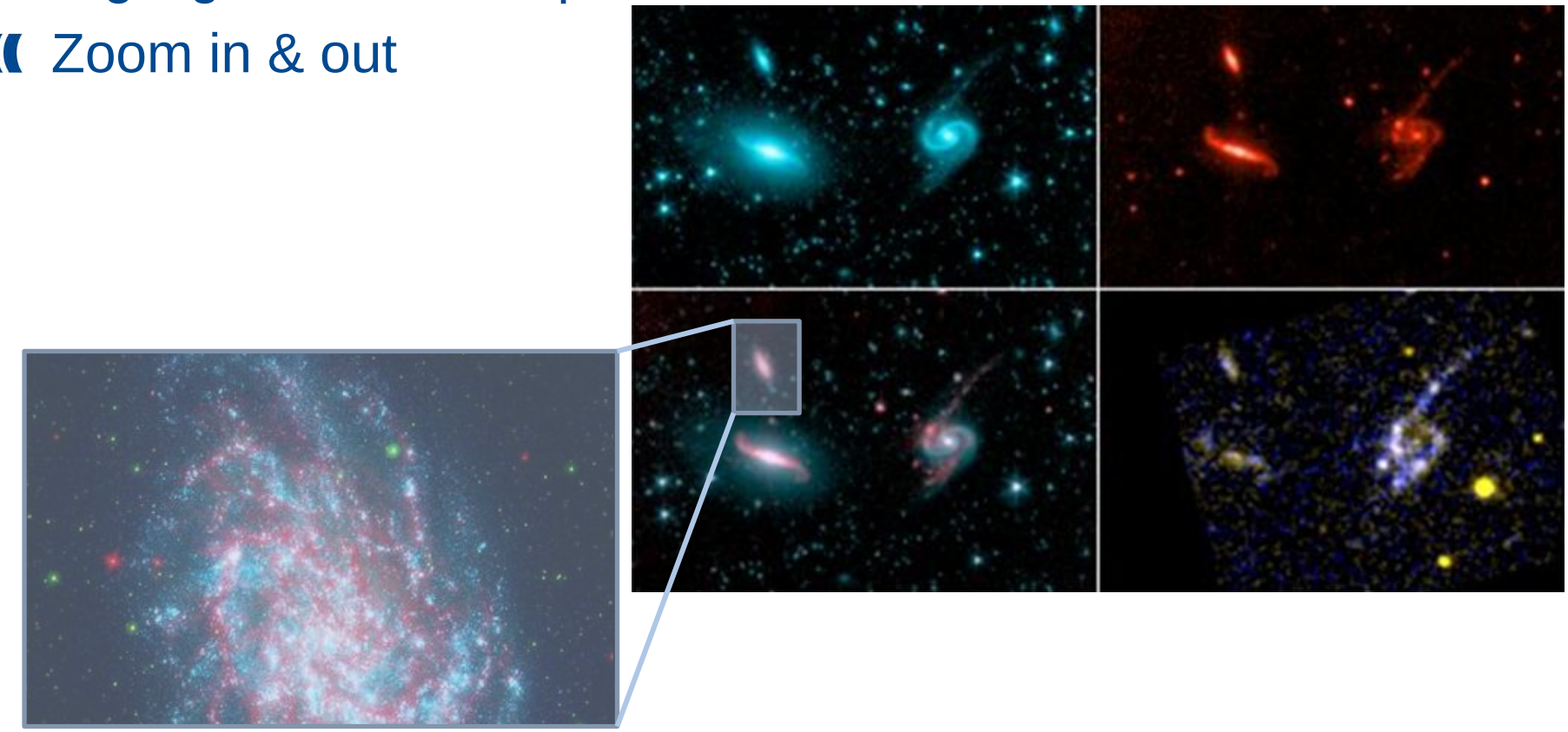
[1] for Fortran codes

Outline

- Extrae
- **Paraver**
  - **General description**
  - Examples
- Research topics
  - Structure detection
  - On-line analysis
  - Folding
- Hands-on session

Multispectral imaging

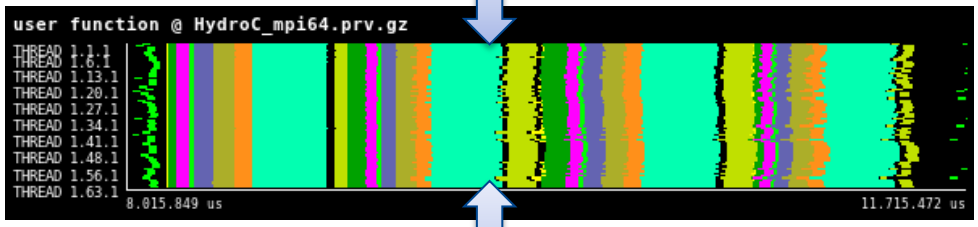
- Different looks at one reality
  - Different light sources using filters
- Highlight different aspects
- Zoom in & out



Paraver

Raw performance data
MPI calls, OpenMP regions, user functions, p2p & collective communications, performance counters, samples, …

Timelines



2D / 3D tables (statistics)

- Our goal is flexibility
  - No semantics
  - Programmable
- Configuration files
  - Distributed with Paraver
  - You can write your own

Timelines: Description

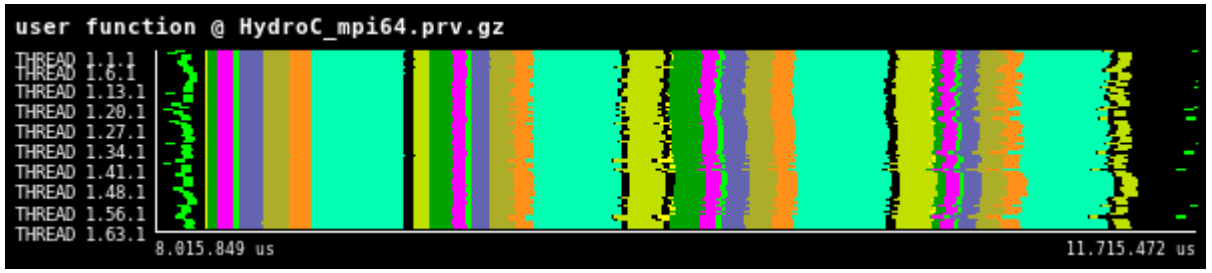
Objects

Process dimension

- **Thread (default)**
- Process
- Application
- Workload

Resource dimension

- CPU
- Node
- System



Time

Timelines: Semantics

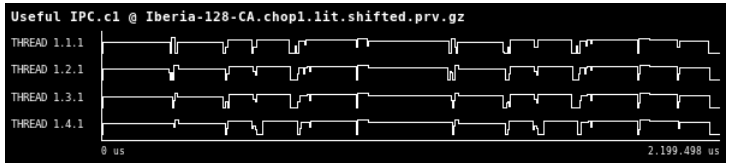
- Each window displays one view
  - Piecewise constant function of time $S(t) = S_i, i \in [t_i, t_{i+1})$
  - One such function of time per object:
    - Thread, process, application, workload, CPU, node

- Types of functions
  - Categorical $S_i \in [0, \mathrm{n}] \subset N$
    - State, user function, which MPI call
  - Logical $S_i \in \{0, 1\}$
    - In specific user function, In MPI call, In long MPI call
  - Numerical $S_i \in R$
    - IPC, L2 miss ratio, Duration of MPI call, duration of computation burst

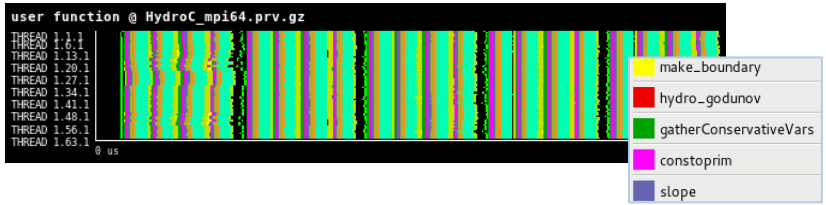
Timelines: Display

Representation

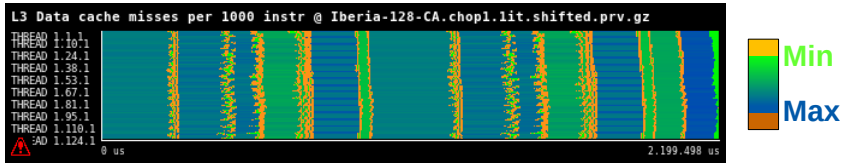
- Function of time



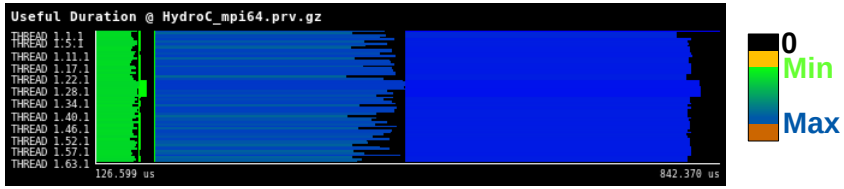
- Color encoding



- Gradient color
  - From **light green to dark blue**
  - Limits in yellow and orange

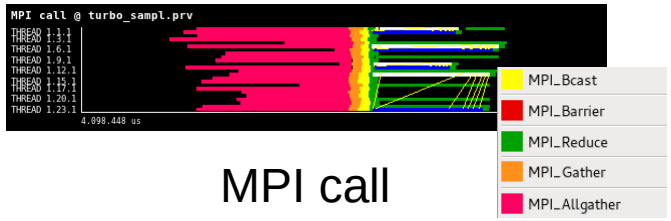


- Not null gradient
  - **Black** (or background) for zero value
  - From **light green to dark blue**
  - Limits in yellow and orange

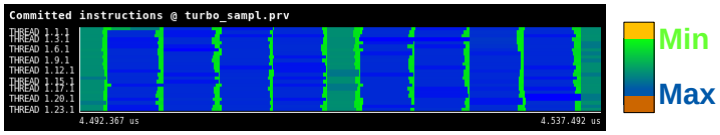


Timelines: Deriving

Basic metrics



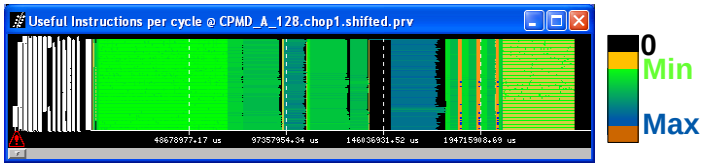
MPI call



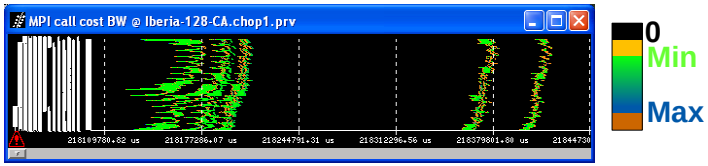
Instructions

Derived metrics

$$useful_IPC = \frac{\#\,instr}{\#\,cycles} * useful$$

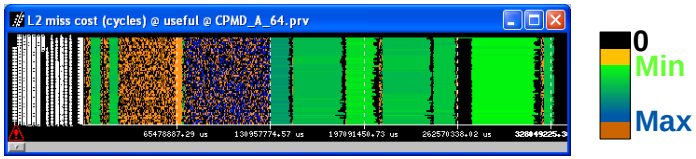


$$MPI_call_Cost = \frac{MPI_call_duration}{\#\,bytes}$$



Models

$$L2_{miss_{latency}} = \frac{cycles - instr / idealIPC}{L2misses}$$

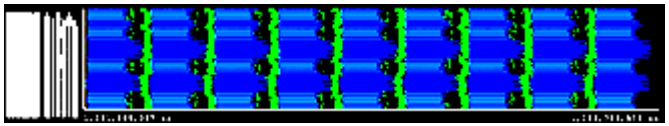


Tables: profiles, histograms and correlations

- Huge number of statistics computed from any timeline

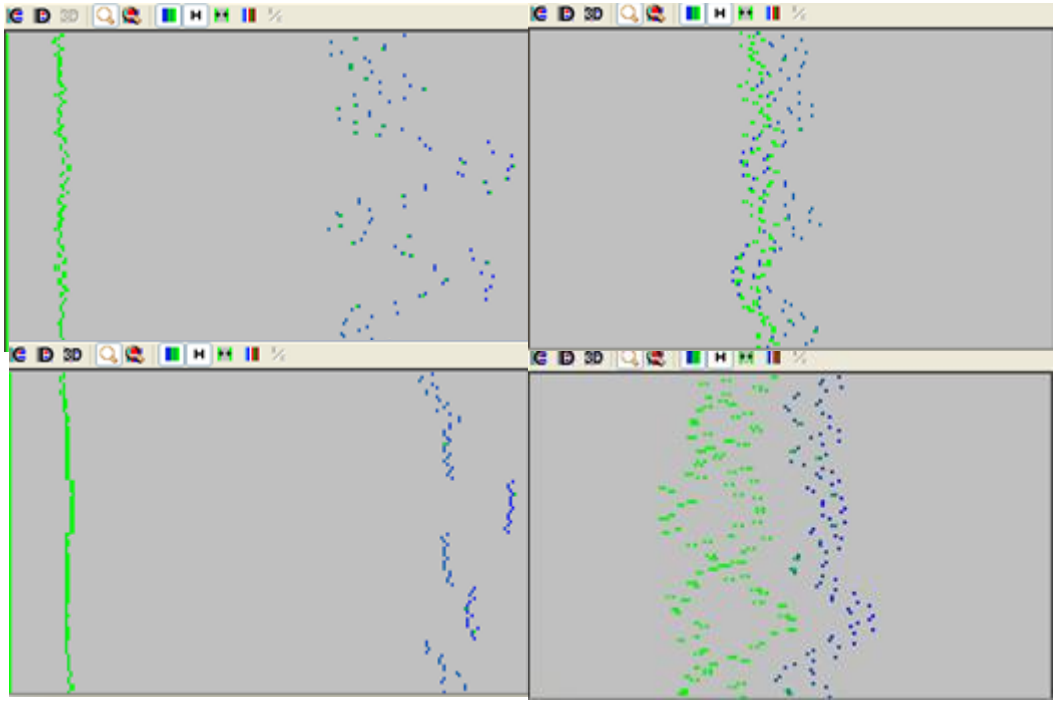
	MPI_Isend	MPI_Irecv	MPI_Alltoall	MPI_Allgather	MPI_Waitany	MPI_Request_free
THREAD 1.503.1	0.77 %	0.36 %	69.84 %	26.15 %	2.59 %	0.30 %
THREAD 1.504.1	1.07 %	0.27 %	70.76 %	25.70 %	1.99 %	0.20 %
THREAD 1.505.1	0.81 %	0.28 %	66.60 %	29.94 %	2.20 %	0.18 %
THREAD 1.506.1	1.12 %	0.45 %	71.00 %	23.53 %	3.57 %	0.32 %
THREAD 1.507.1	0.95 %	0.22 %	68.92 %	28.04 %	1.70 %	0.17 %
THREAD 1.508.1	0.38 %	0.34 %	67.89 %	27.31 %	3.86 %	0.23 %
THREAD 1.509.1	2.32 %	0.36 %	62.98 %	33.21 %	0.83 %	0.29 %
THREAD 1.510.1	0.81 %	0.31 %	68.68 %	25.86 %	4.11 %	0.23 %
THREAD 1.511.1	2.45 %	0.56 %	70.48 %	25.05 %	1.06 %	0.40 %
THREAD 1.512.1	1.20 %	0.28 %	67.03 %	30.21 %	1.05 %	0.23 %
Total	525.20 %	202.46 %	35,644.50 %	13,711.63 %	977.46 %	138.76 %
Average	1.03 %	0.40 %	69.62 %	26.78 %	1.92 %	0.27 %
Maximum	3.25 %	2.46 %	77.63 %	41.15 %	6.25 %	1.18 %
Minimum	0.05 %	0.05 %	56.00 %	12.80 %	0.16 %	0.04 %
StDev	0.56 %	0.24 %	2.92 %	3.45 %	1.26 %	0.13 %
Avg/Max	0.32	0.16	0.90	0.65	0.31	0.23

MPI call profile



Useful duration

IPC



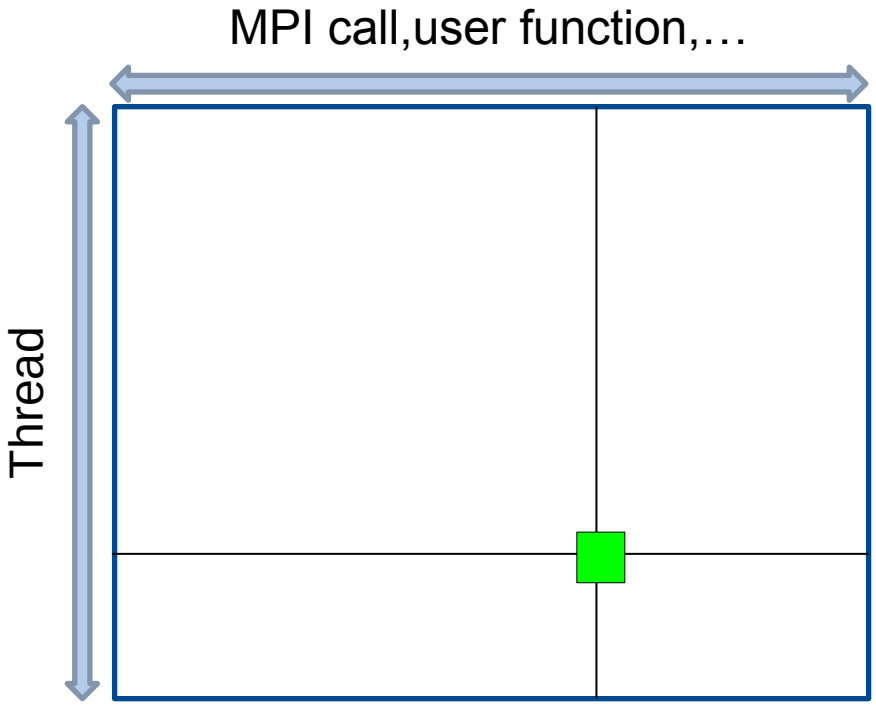
Instructions

L2 miss ratio

Tables: How to read profiles

One column per categorical value of **Control window**

	End	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Allreduce	MPI_Com
THREAD 1.1.1	86,98 %	0,06 %	0,08 %	11,12 %	1,75 %	
THREAD 1.2.1	88,29 %	0,10 %	0,10 %	9,95 %	1,56 %	
THREAD 1.3.1	88,33 %	0,13 %	0,10 %	9,92 %	1,51 %	
THREAD 1.4.1	89,75 %	0,10 %	0,09 %	8,62 %	1,44 %	
THREAD 1.5.1	89,47 %	0,11 %	0,10 %	8,85 %	1,46 %	
THREAD 1.6.1	88,76 %	0,12 %	0,09 %	9,54 %	1,48 %	
THREAD 1.7.1	91,77 %	0,13 %	0,10 %	6,51 %	1,49 %	
THREAD 1.8.1	90,23 %	0,06 %	0,08 %	8,13 %	1,50 %	
THREAD 1.9.1	91,88 %	0,13 %	0,09 %	6,73 %	1,17 %	
THREAD 1.10.1	93,24 %	0,18 %	0,11 %	5,41 %	1,05 %	
THREAD 1.11.1	93,25 %	0,18 %	0,11 %	5,45 %	1,00 %	
THREAD 1.12.1	94,63 %	0,17 %	0,11 %	4,16 %	0,93 %	
THREAD 1.13.1	93,40 %	0,17 %	0,11 %	5,35 %	0,96 %	
THREAD 1.14.1	94,99 %	0,20 %	0,11 %	3,77 %	0,93 %	
THREAD 1.15.1	96,80 %	0,22 %	0,11 %	1,92 %	0,95 %	
THREAD 1.16.1	95,73 %	0,12 %	0,09 %	2,99 %	1,06 %	

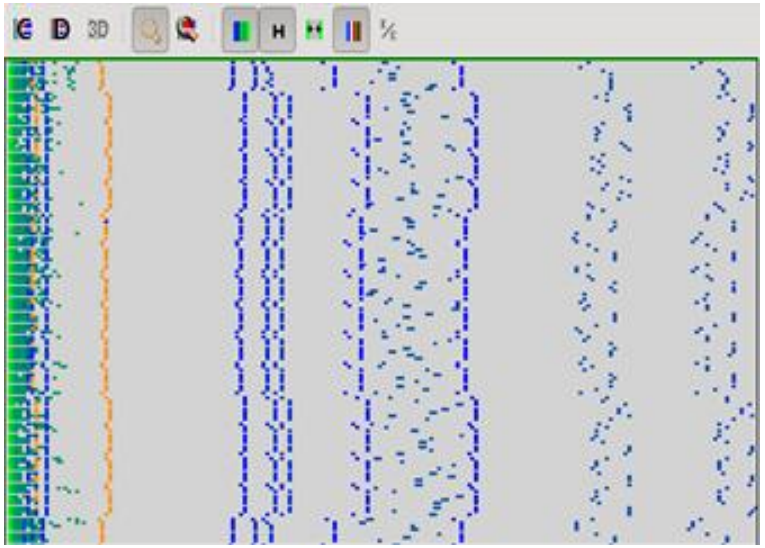


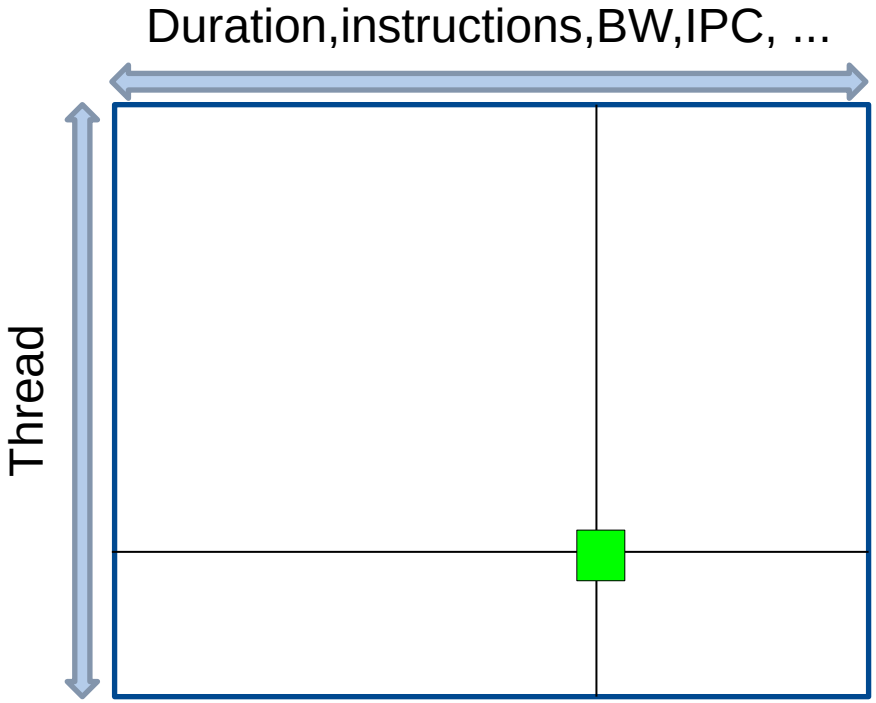
Value/color is a statistic computed for the given thread
when control window had the value corresponding to the column

Relevant statistics:
time, %time, avg. burst time, average of **data window**

Tables: How to read histograms

Columns correspond to bins of values of a numeric **Control window**



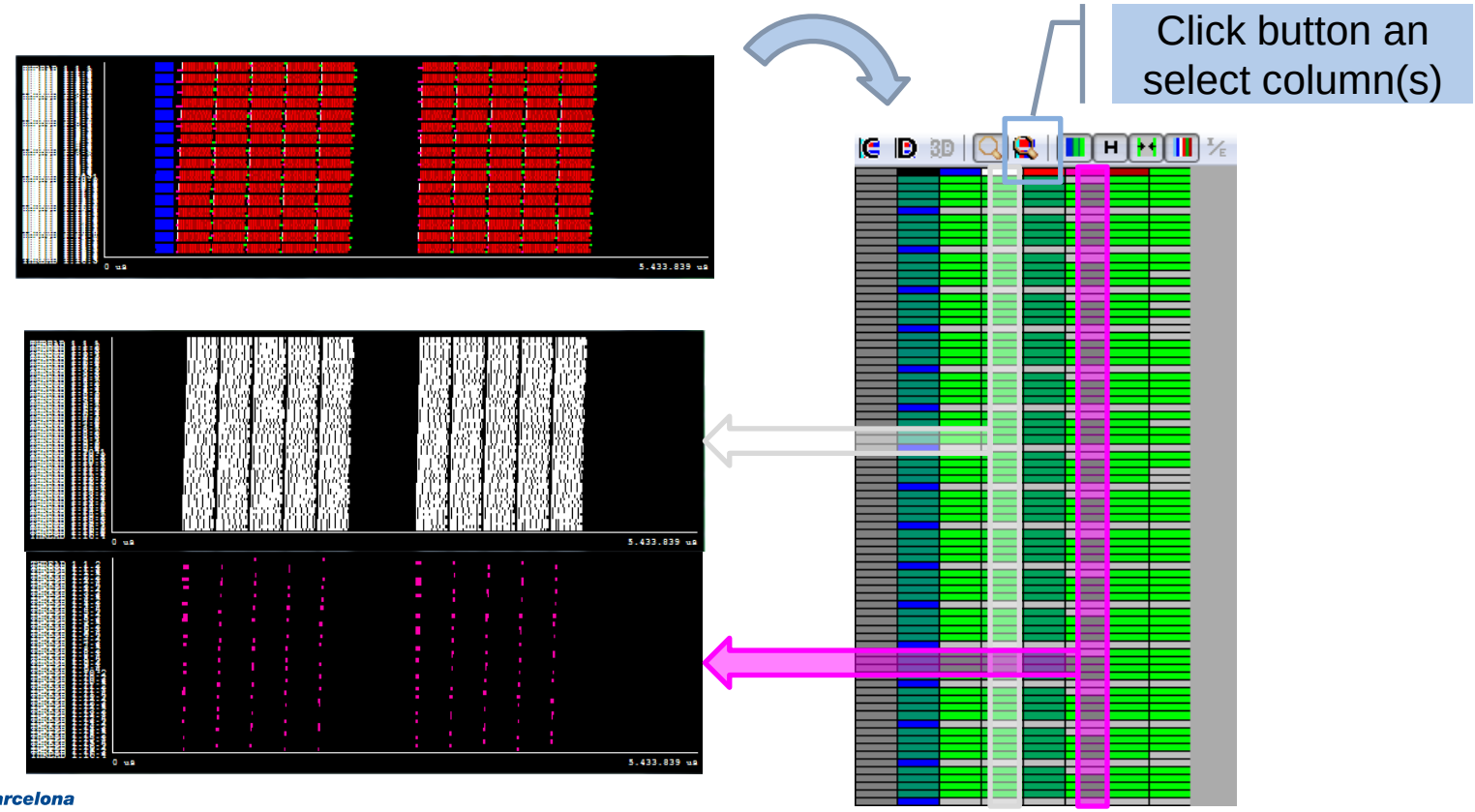


Value/color is a statistic computed for the given thread
when control window had the value corresponding to the column

Relevant statistics:
time, %time, avg. burst time, average of **data window**

Tables: back to timelines

- Where in the timeline do the values in certain table columns appear?
  - i.e. which is the time distribution of a given routine?
  - i.e. when does a routine occur in the timeline?



Comparative analyses

- Possible to load several traces
  - And thus perform execution comparison
- Copy and paste between windows
  - right click menu
  - from one window to another: time, duration, size, objects displayed,…
  - time between windows and tables
    - analysis computed only for the selected time interval
- Synchronize windows

Distribution of cfg directories

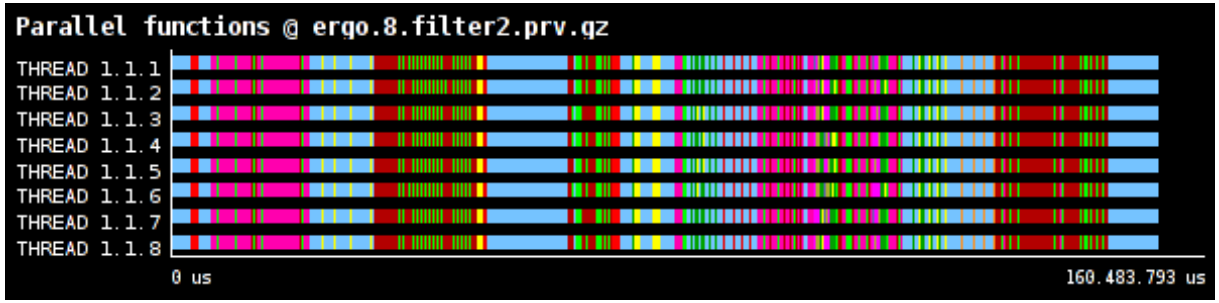
- CFG are programmed Paraver windows
  - General
    - including basic views (timelines) and analysis (2/3D profiles), including views of the user functions and call-stack
  - Counters_PAPI
    - Hardware counter derived metrics. Grouped in directories for
      - Program: related to algorithmic/compilation (i.e. instructions,FP ops,…)
      - Architecture: related to execution on specific architectures (i.e. cache misses,…)
      - Performance: metrics reporting rates per time (i.e. MFLops, MIPS, IPC,…)
  - MPI
    - Grouped in directories displaying views and analysis. Further separated into point to point and collectives.
  - OpenMP
    - Grouped in directories displaying views and analysis

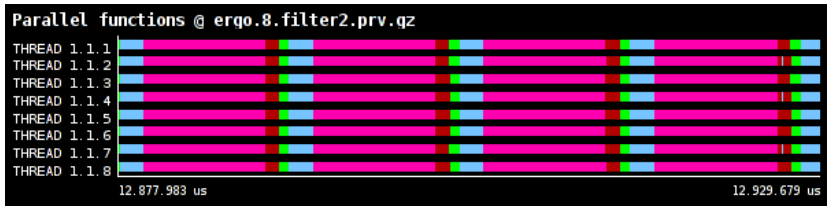
Outline

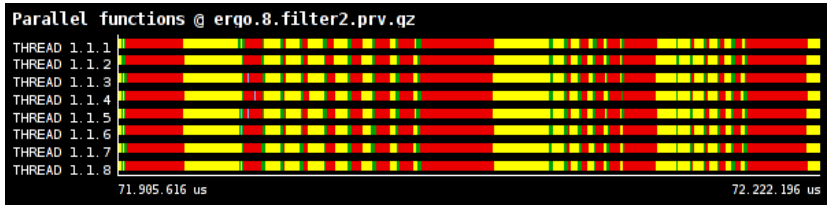
- Extrae
- **Paraver**
  - General description
  - **Examples**
- Research topics
  - Structure detection
  - On-line analysis
  - Folding
- Hands-on session

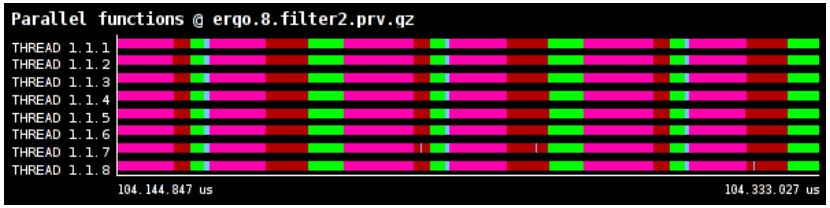
Example 1: Analysing ERGO (OpenMP)

- ERGO – 8 OpenMP threads
- Parallel functions view
  - Structure? Very different phases
  - Some phases delimited by sequential code (light blue)



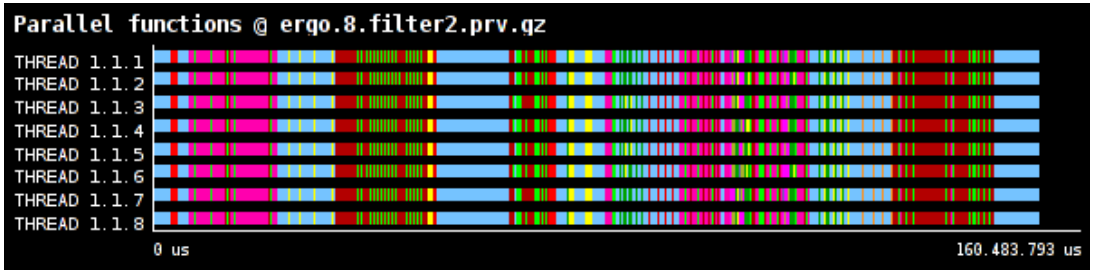




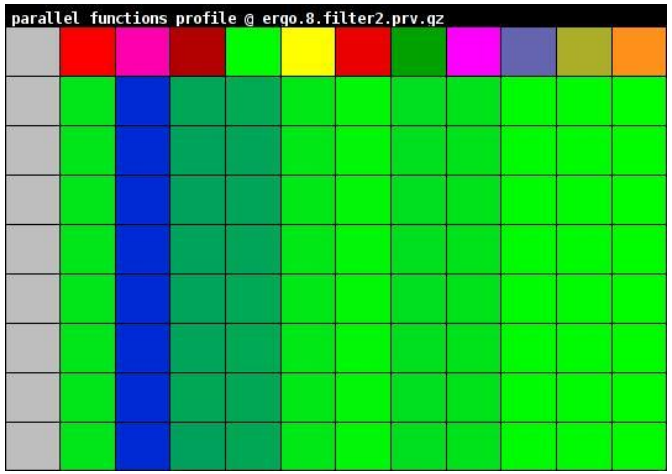


Example 1: Analysing ERGO (OpenMP)

- Loops balanced between threads
- Less than 30% time is paralellised (limited scalability)



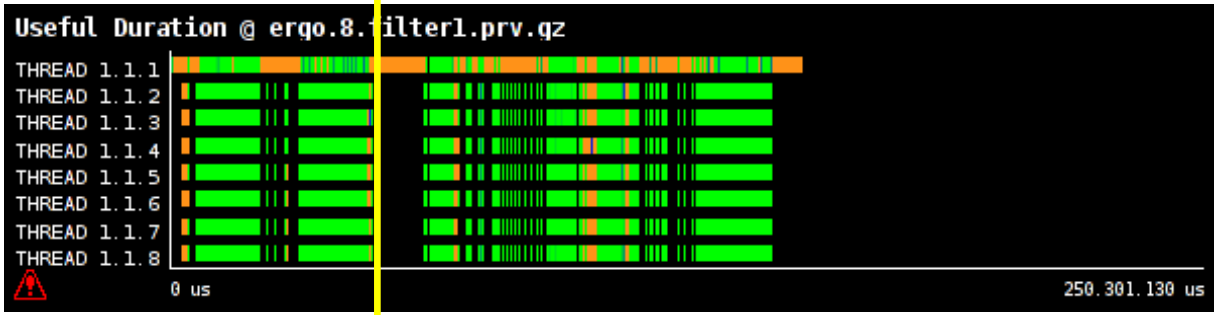
	THREAD 1.1.1	THREAD 1.1.2	THREAD 1.1.3
compute_operator_matrix_sparseBasisInfoStruct_par_region0	1,48 %	1,48 %	1,48 %
mat6VectorMatrix__par_region0	12,51 %	12,51 %	12,51 %
mat6VectorgemvMatrix_par_loop0_2	5,16 %	5,42 %	5,45 %
mat6VectorgemvMatrix_par_loop1_2	4,92 %	5,27 %	5,28 %
mat6Matrixgemm_par_loop0	1,36 %	1,38 %	1,38 %
mat6Matrixgemm_par_loop1	0,45 %	0,45 %	0,45 %
mat6Matrixsymm_par_loop0	1,84 %	1,84 %	1,82 %
mat6Matrixsysq_par_region0	1,62 %	1,62 %	1,62 %
mat6Matrixsyrk_par_region0	0,18 %	0,18 %	0,18 %
mat6Matrixsyrk_par_region3	0,17 %	0,17 %	0,17 %
mat6Matrixgemm_par_loop2	0,03 %	0,04 %	0,04 %
Total	29,73 %	30,36 %	30,38 %



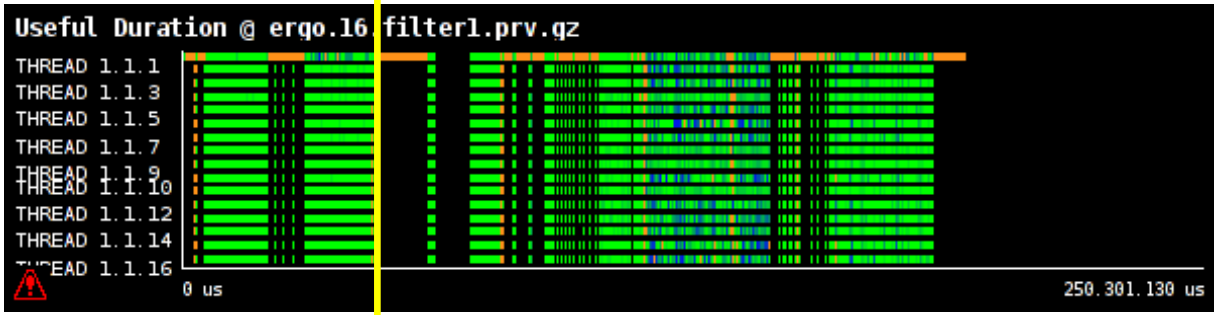
Example 1: Analysing ERGO (OpenMP)

- ERGO scalability

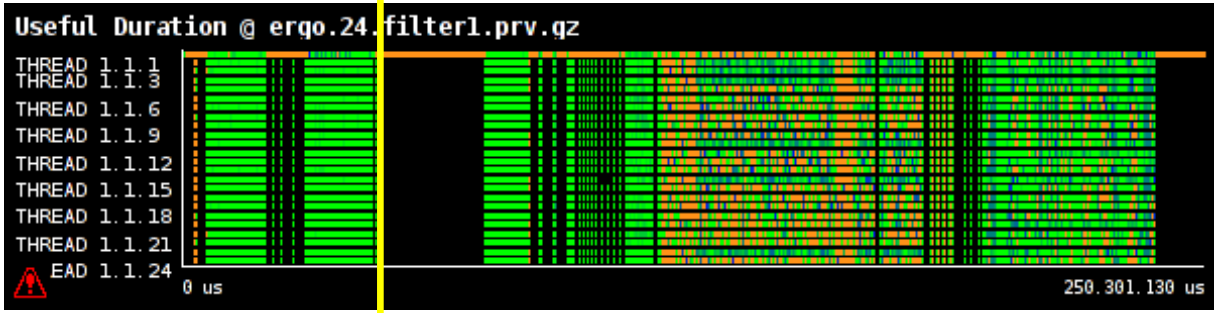
8 threads



16 threads



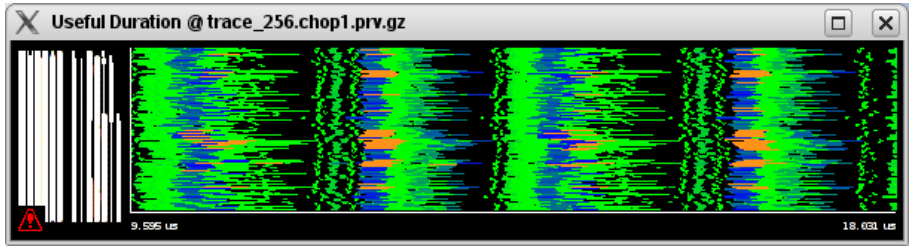
24 threads

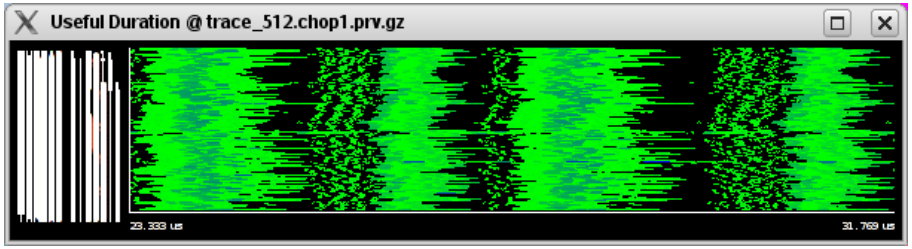


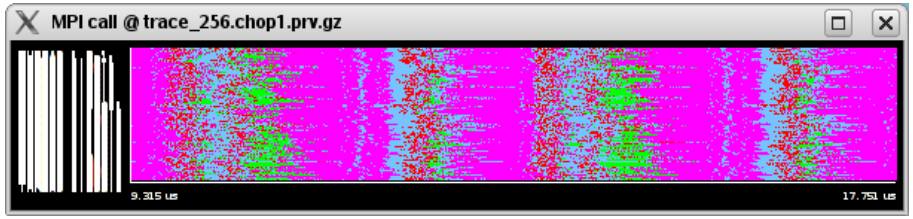
Example 2: FrontFlowRed

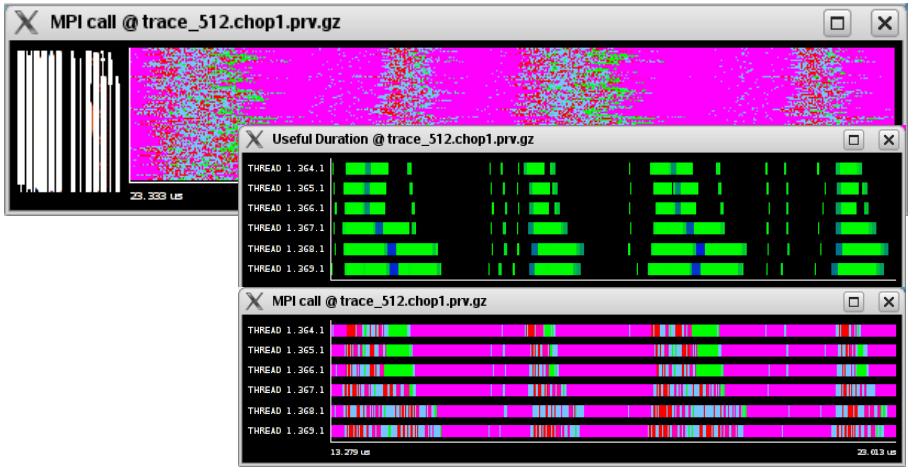
- 256 vs 512 tasks
  - Imbalance and MPI time increases
  - Sequence of MPI_Allreduce
  - Poor IPC, high cache misses

Metric	256	512
Parallel efficiency	27%	19%
Load balance	56%	48%
% MPI_Allreduce	45%	62%









Example 2: FrontFlowRed

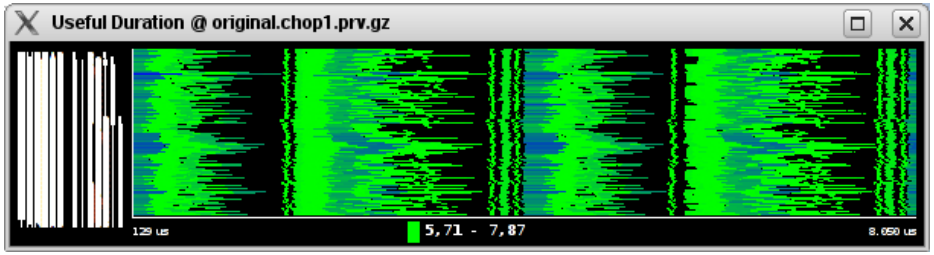
- Modifications in the source code
  - Improved load balance – weight nodes based on their connectivity
  - Reduction of alltoall communications
  - Reduce cache misses – reordering node number

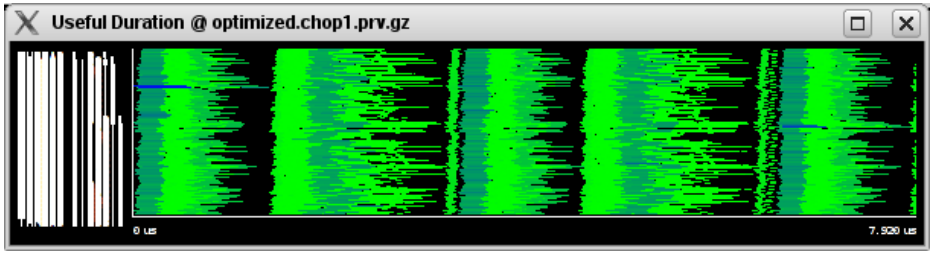
- ½ hour meeting for the analysis + few days modifying code

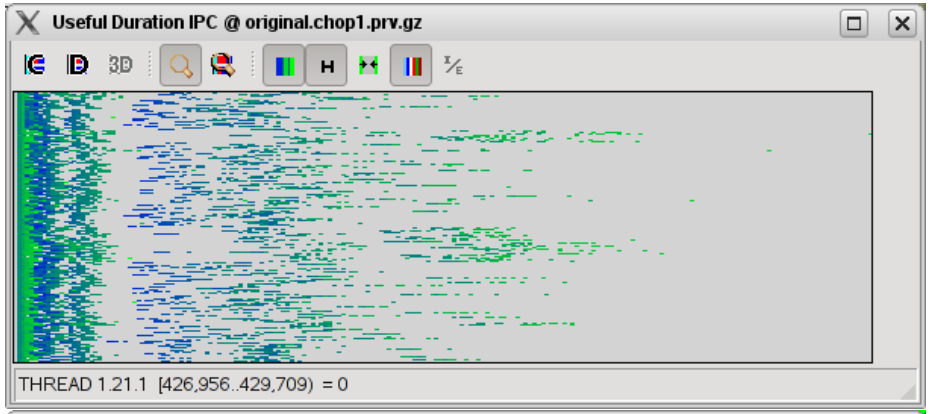
  25% improvement

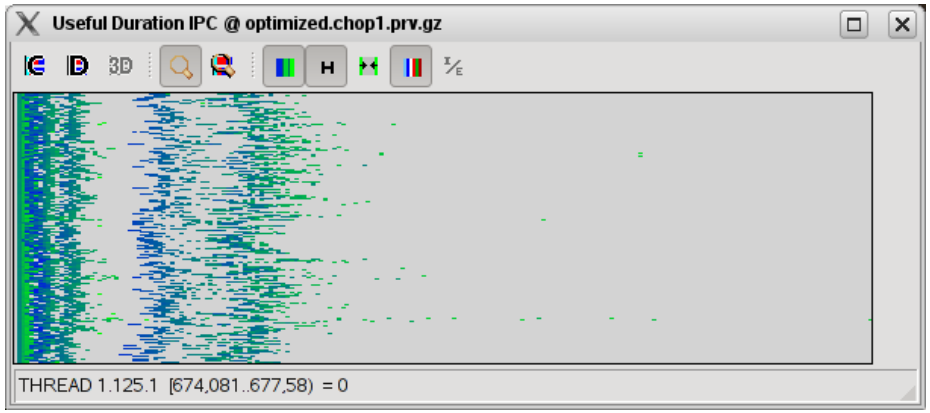
Example 2: FrontFlowRed

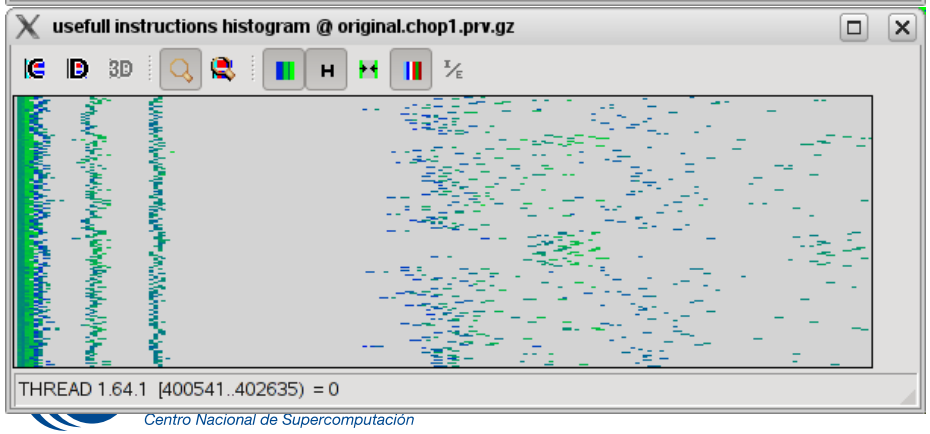
- 256 tasks – Load balance

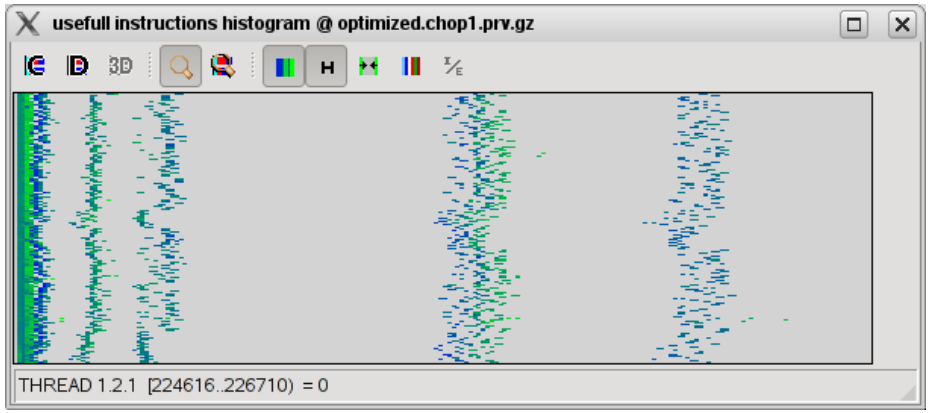






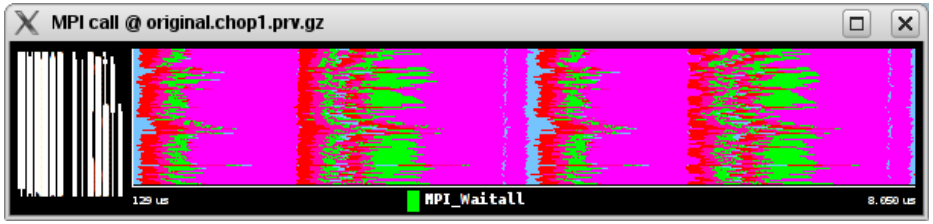


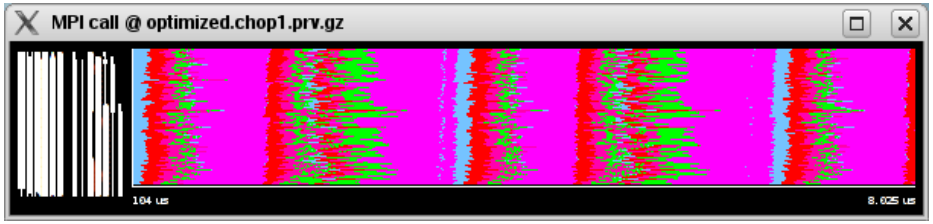




Example 2: FrontFlowRed

- 256 tasks – MPI time



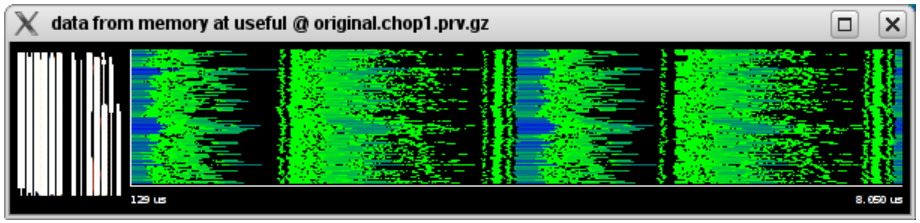


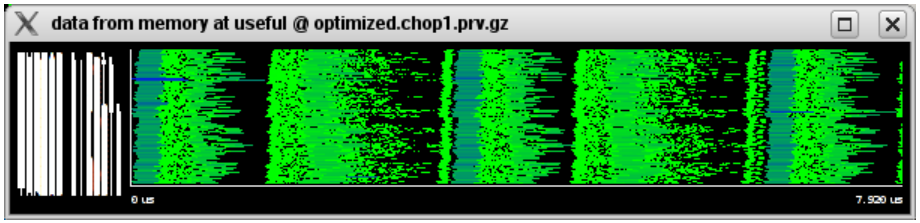
	End	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Allreduce
Total	7.617,81 %	2.432,05 %	1.772,21 %	2.245,19 %	11.532,74 %
Average	29,76 %	9,50 %	6,92 %	8,77 %	45,05 %
Maximum	51,85 %	18,11 %	13,00 %	28,59 %	67,69 %
Minimum	17,59 %	1,65 %	1,37 %	1,46 %	14,88 %
StDev	8,04 %	3,33 %	2,39 %	6,05 %	11,03 %
Avg/Max	0,57	0,52	0,53	0,31	0,67

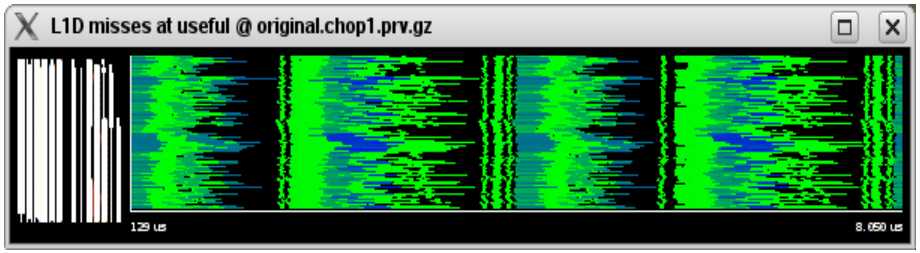
	End	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Allreduce
Total	9.089,05 %	3.100,13 %	2.107,96 %	1.511,02 %	9.791,83 %
Average	35,50 %	12,11 %	8,23 %	5,90 %	38,25 %
Maximum	51,05 %	20,77 %	14,59 %	21,31 %	64,34 %
Minimum	22,99 %	3,47 %	3,21 %	1,72 %	10,90 %
StDev	5,92 %	3,62 %	2,61 %	3,89 %	10,13 %
Avg/Max	0,70	0,58	0,56	0,28	0,59

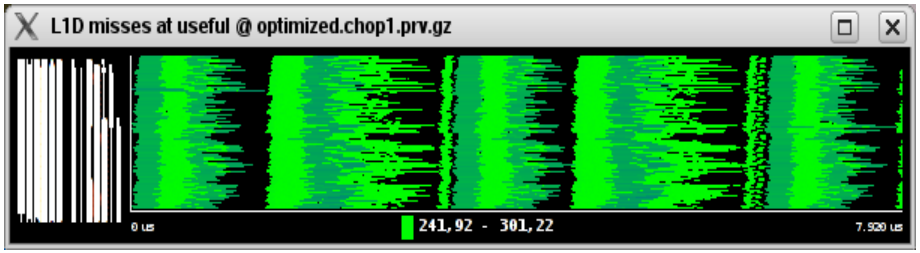
Example 2: FrontFlowRed

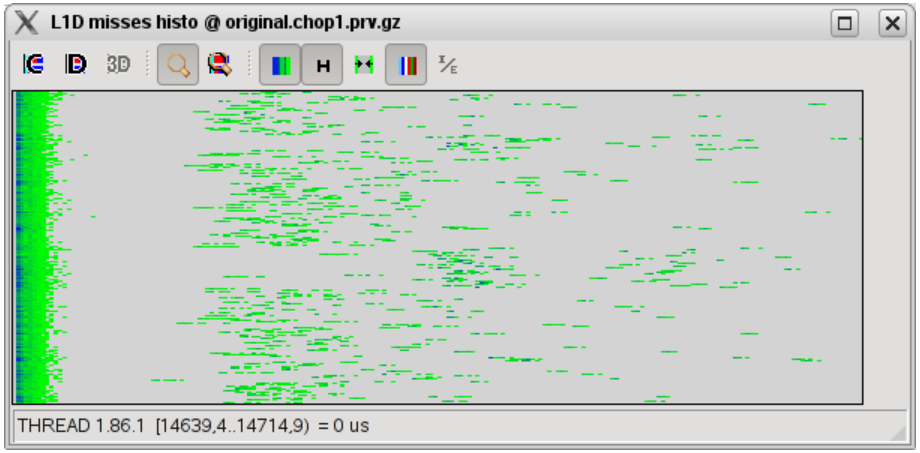
- 256 tasks – memory access

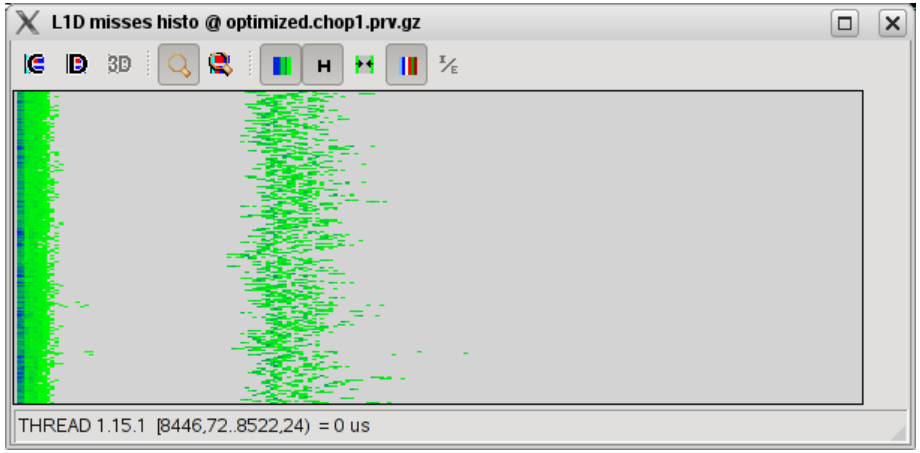












Outline

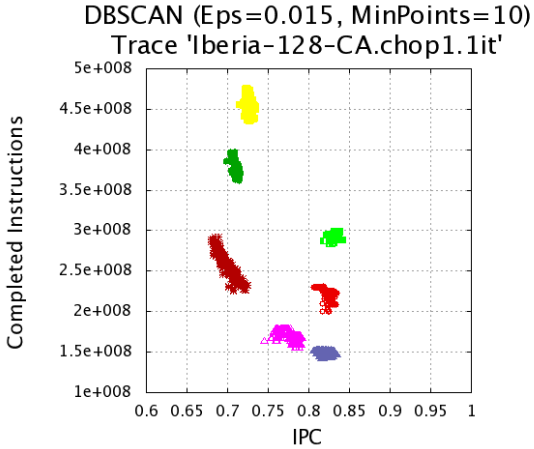
- Extrae
- Paraver
  - General description
  - Examples
- **Research topics**
  - **Structure detection**
  - On-line analysis
  - Folding
- Hands-on session

Performance @ serial computation bursts

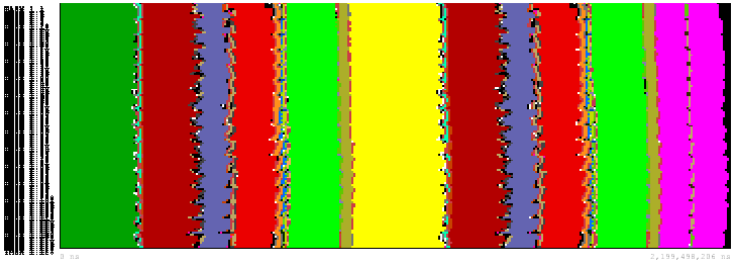
- Burst = continuous computation region
  - i.e., between exit of an MPI call and entry to the next
- Scatter-plot representation of bursts
  - Collapse time dimension
  - N dimensional space of HWC metric
    - Instructions: computational complexity, computational load imbalance,…
    - IPC: absolute performance and performance imbalance
- Structure
  - Clouds, clusters: Burst of *similar* characteristics
  - How these similarities spread along the timeline?

Clustering module – detecting structure

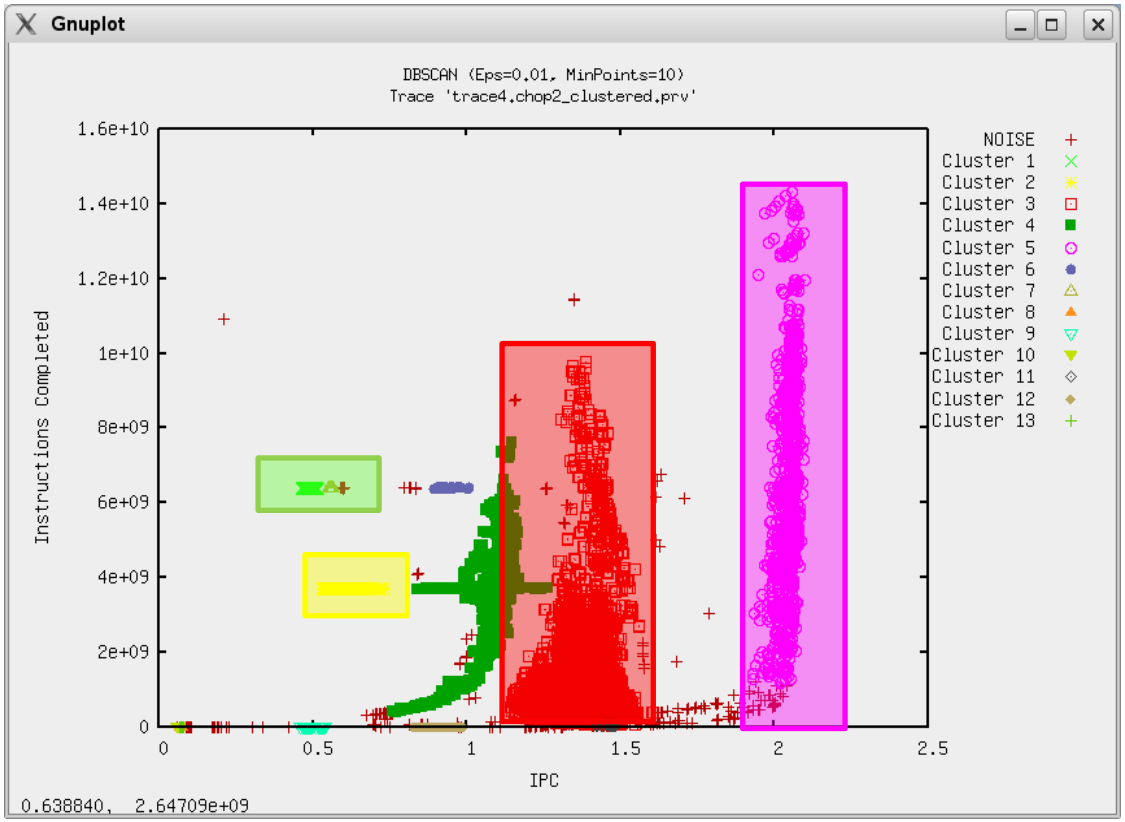
Scatterplot of clustered metrics

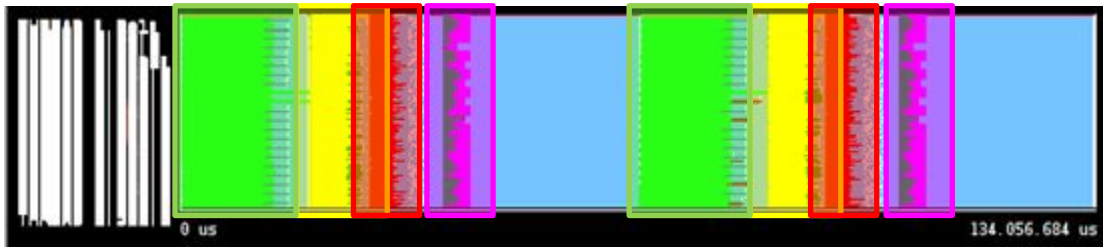


Clusters distribution over time



Clustering module – detecting structure





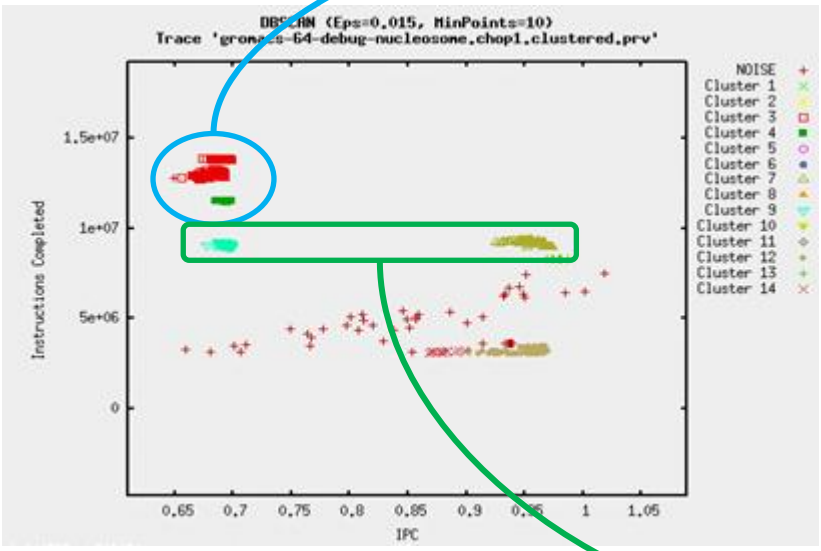
Most relevant computing regions (colored in green and yellow) obtain poor IPC

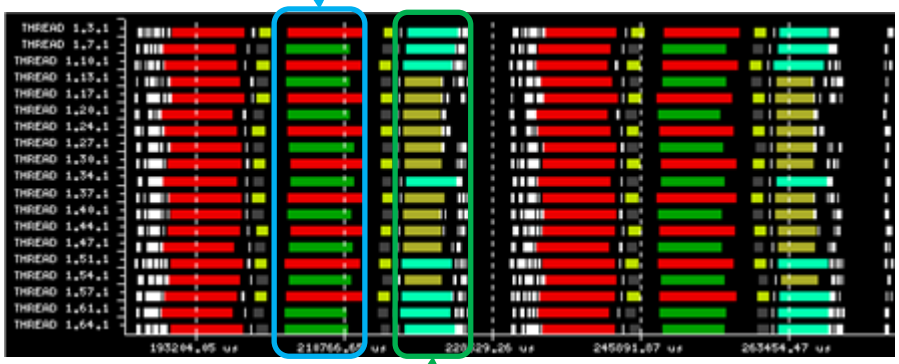
Huge instructions variability in red and pink regions

Gromacs example

Gromacs FFTs balance

Instructions imbalance





IPC imbalance

Outline

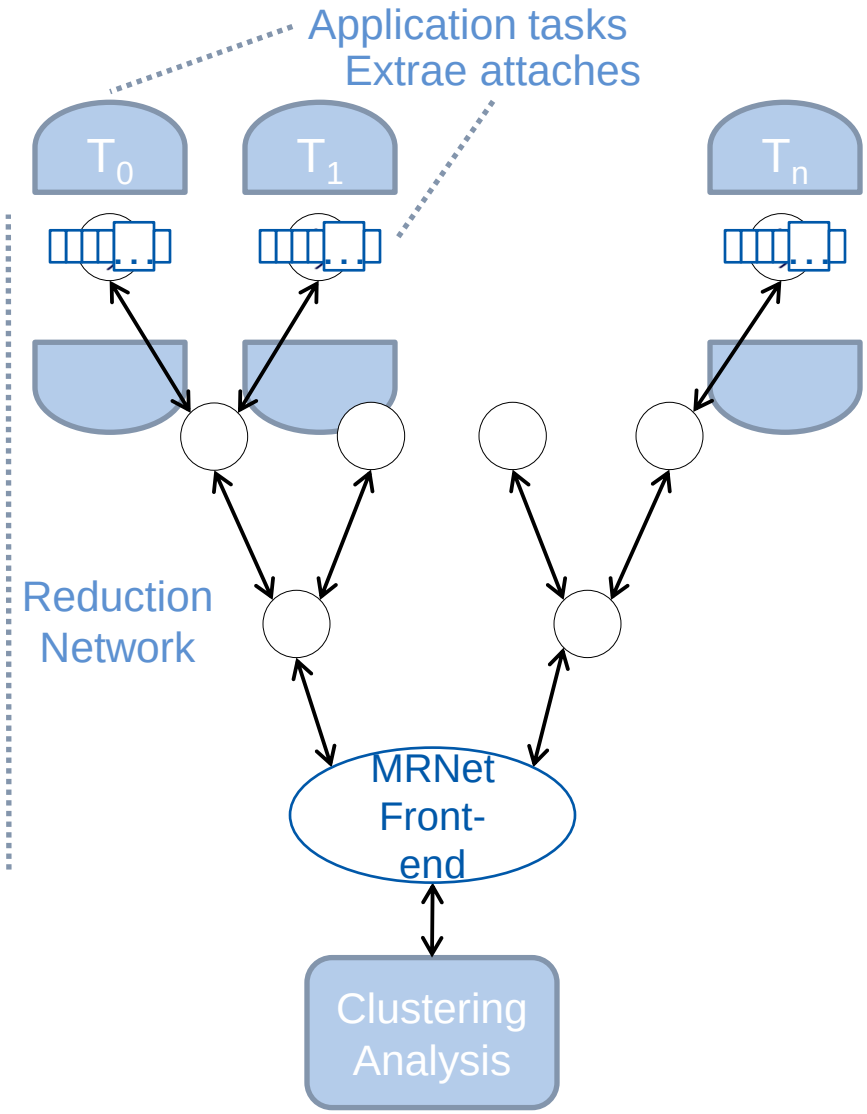
- Extrae
- Paraver
  - General description
  - Examples
- **Research topics**
  - Structure detection
  - **On-line analysis**
  - Folding
- Hands-on session

On-line analysis

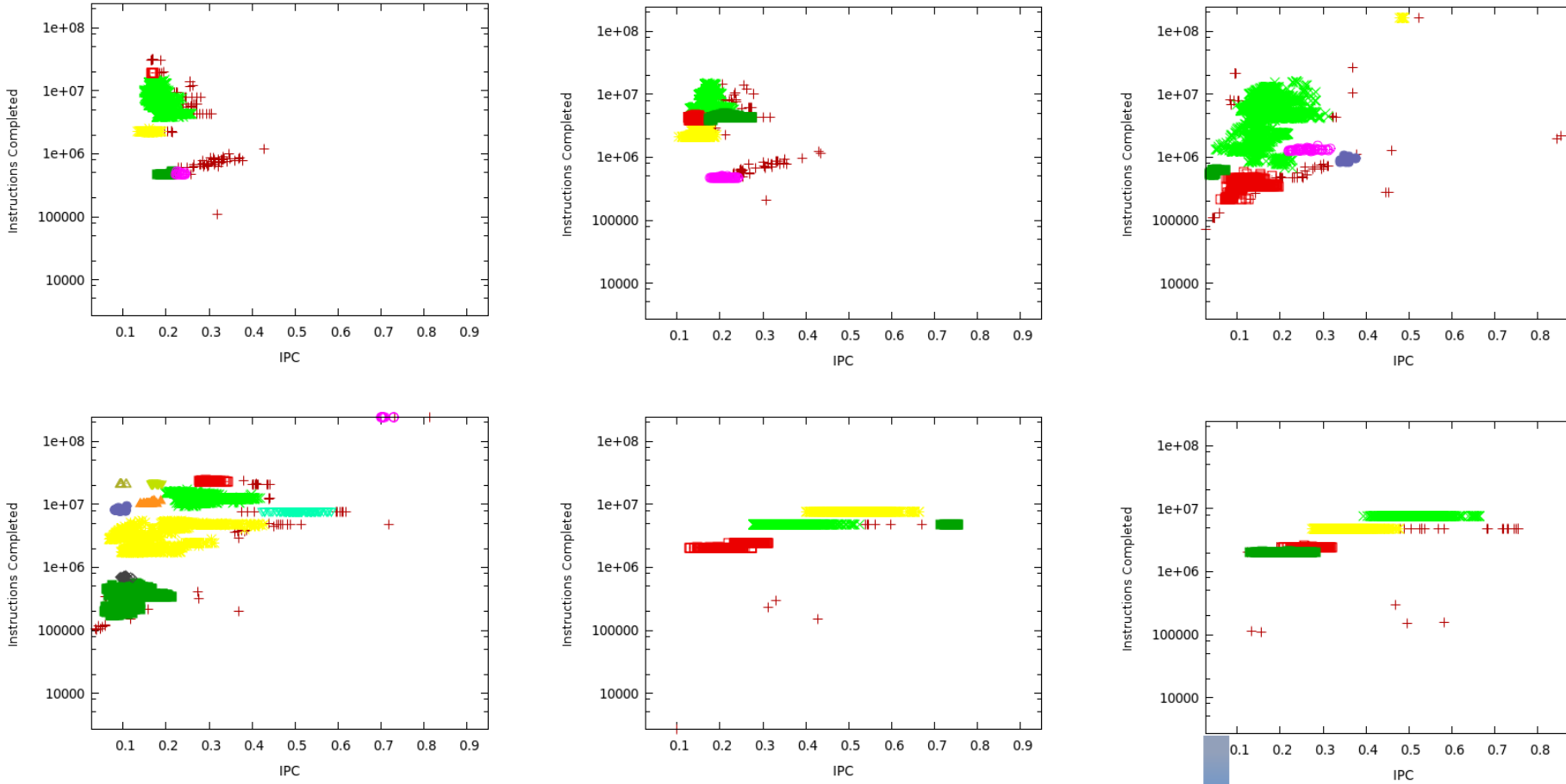
- Traces of “100 Mbytes”
  - Trigger clustering analysis periodically
    - Sequence of structure snapshots
  - Compare subsequent clusterings
    - See changes in application behavior
  - Find a representative region
    - Most application are highly iterative

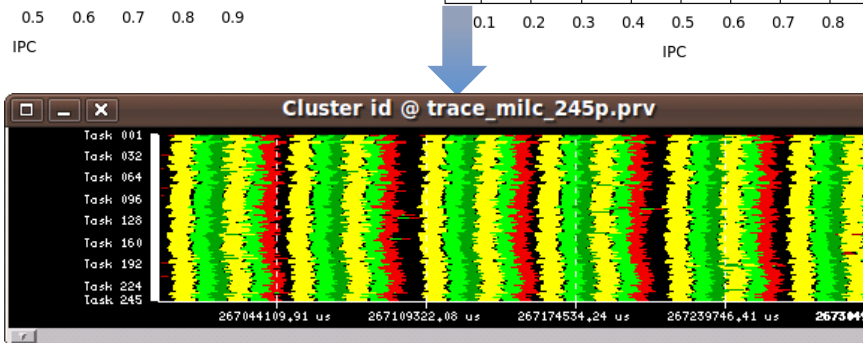
Modules integration

- Data acquisition
  - Extrae (BSC)
    - PMPI wrappers
    - Extrae communication thread
- Data transmission
  - MRNet (U. Wisconsin)
    - Scalable master/worker
    - Tree topology
- Data analysis
  - Clustering analysis (BSC)
    - Find structure



On-line analysis: MILC





- 60 Mb, 6 iterations

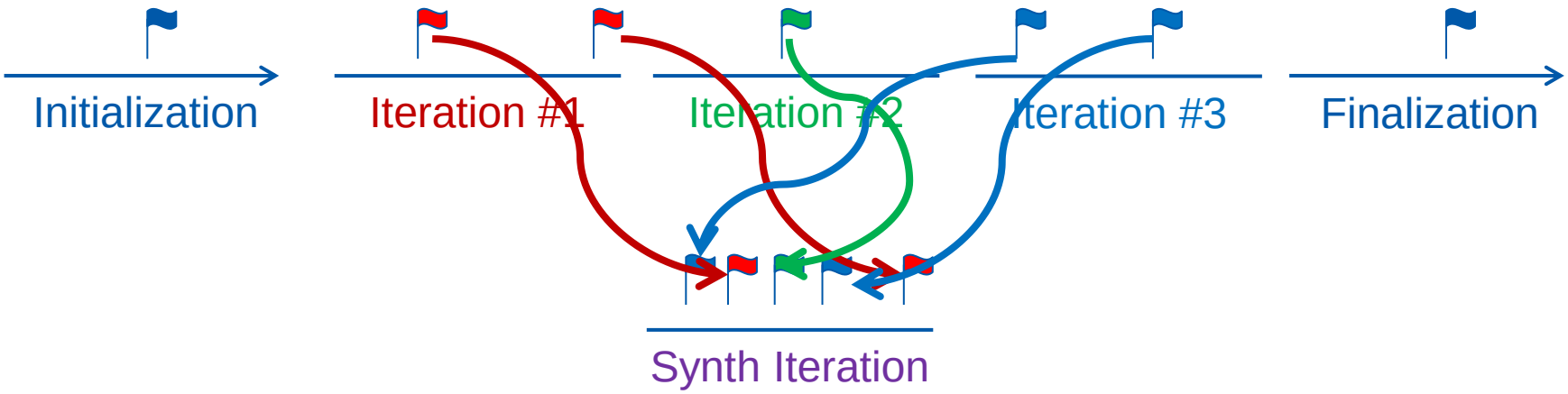
Outline

- Extrae
- Paraver
  - General description
  - Examples
- **Research topics**
  - Structure detection
  - On-line analysis
  - **Folding**
- Hands-on session

Folding

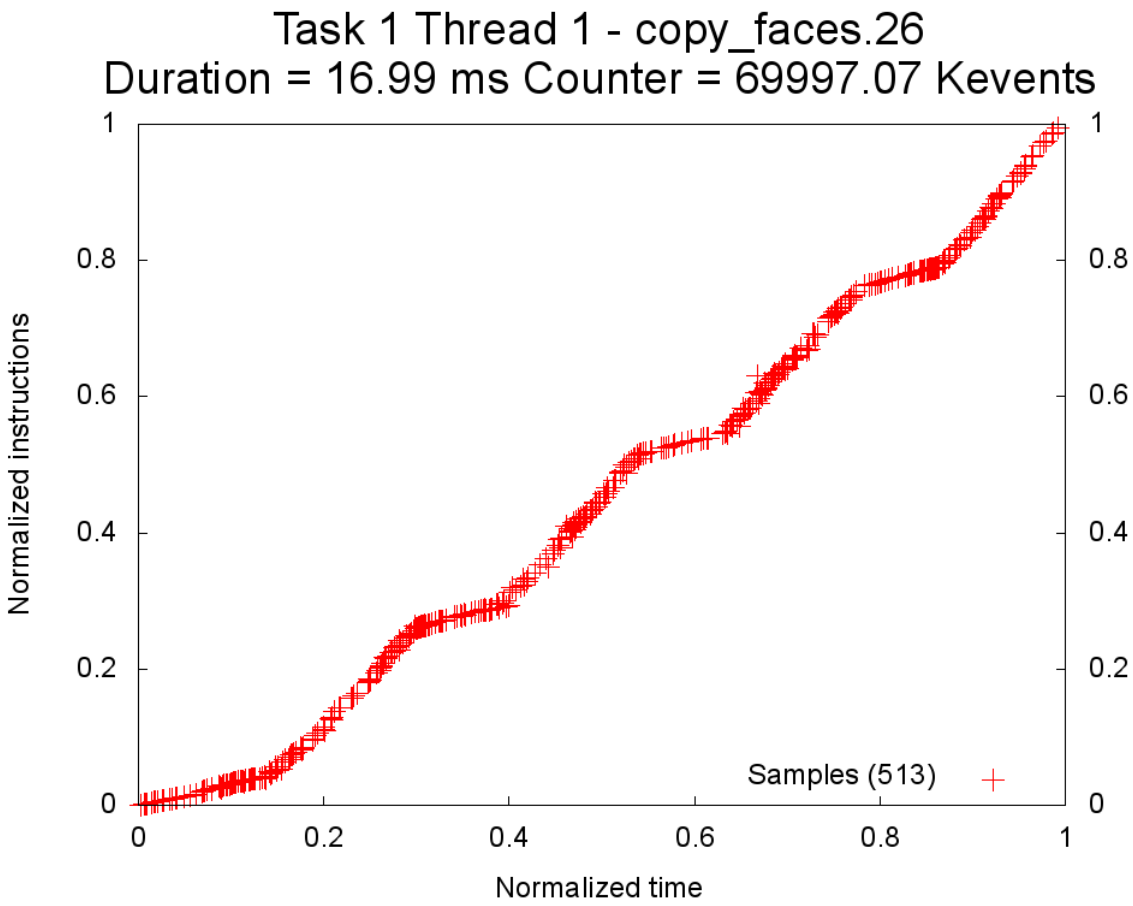
- Get detailed metrics with few samples
  - Detailed sampling is costly
  - Combines instrumentation and sampling

- Takes advantage of long iterative code regions



Folding: hardware counters

- Instructions evolution for copy_faces of NAS MPI BT.B

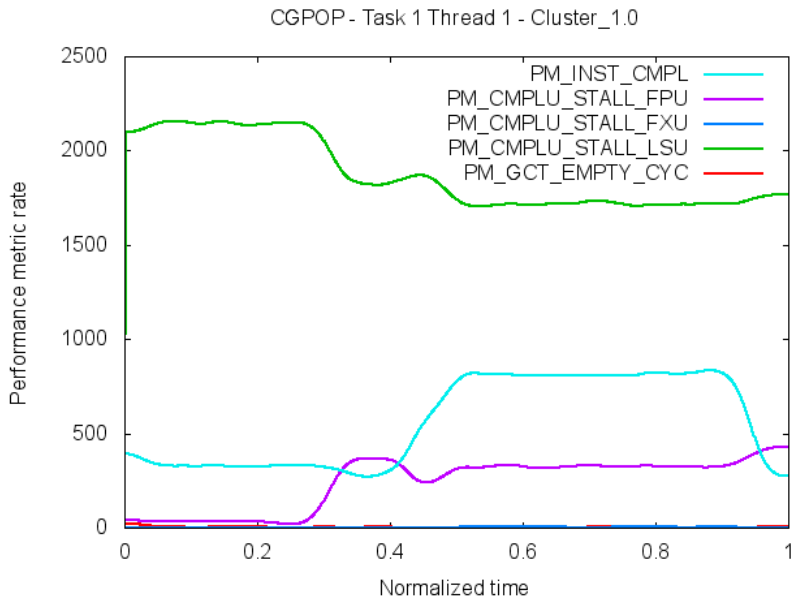


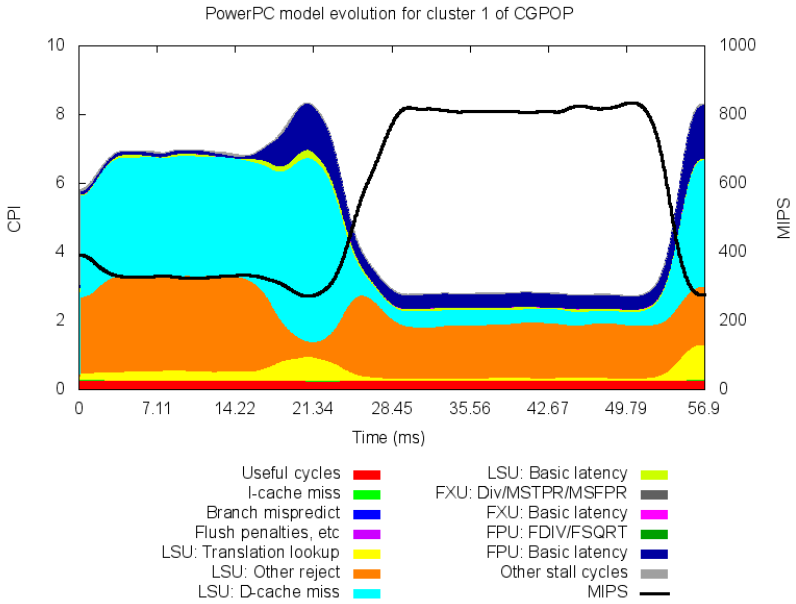
Red crosses represent the folded samples and show the completed instructions from the start of the routine

Blue line is the derivative of the curve fitting over time (*counter rate*)

Folding + models

- Combine multiple performance counters in a plot





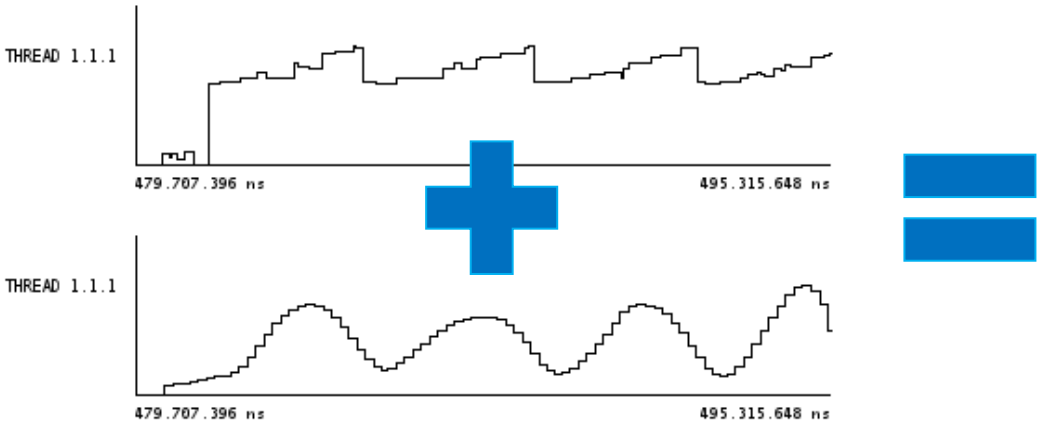
POWER5 CPI BREAK-DOWN MODEL

		CPI break-down component		
Total cycles	*Completion cycles*	PowerPC Base completion cycles		
		Overhead of μ-coding and grouping restriction		
	Completion Table empty cycles	I-cache miss penalty		
		Branch redirection penalty		
		Others (Flush penalty, *etc.*)		
	Completion stall cycles	by LSU inst	by reject	by ERAT miss
				Other reject
			by D-cache miss	
			by LSU basic latency	
		by FXU inst	by DIV/MTSPR/MFSPR inst	
			by FXU basic latency	
		by FPU inst	by FDIV/FSQRT inst	
			by FPU basic latency	
		Others: Stall by BRU/CRU, flush penalty, *etc.*		

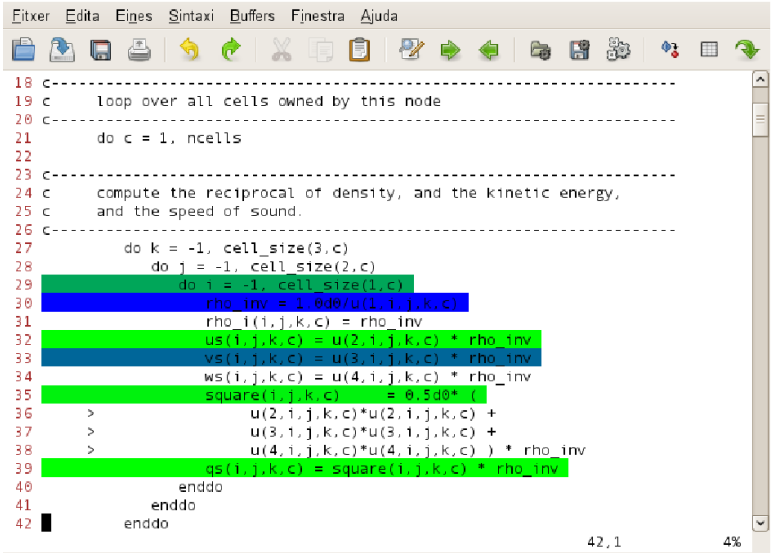
Folding: reference to source code

- Call-site sampling information is folded
  - Correlation between hwc and call-sites
  - GVIM add-on to show performance within source code
    - Timeless but useful to point performance issues

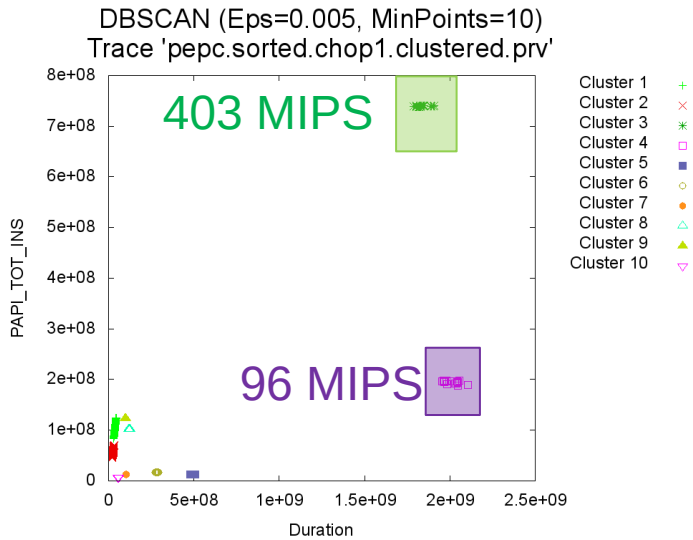
Folded source code line

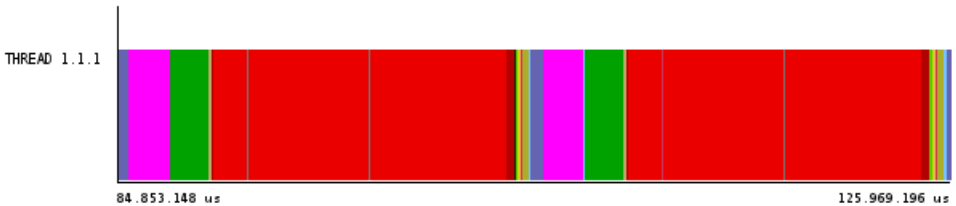


Folded instructions

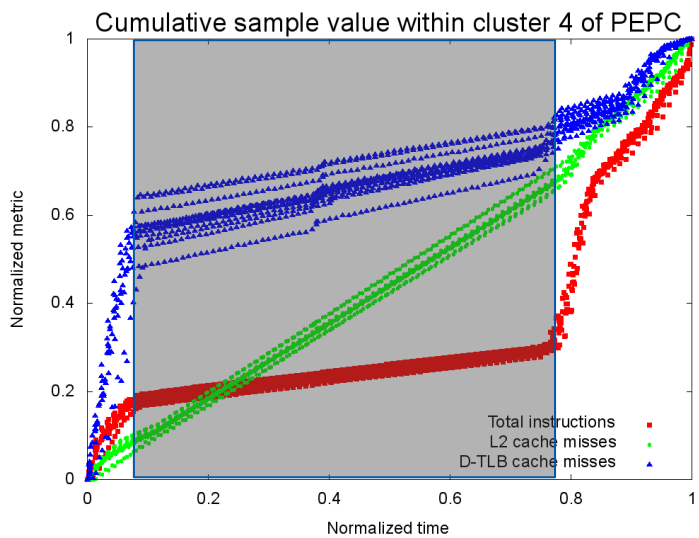


Example: PEPC





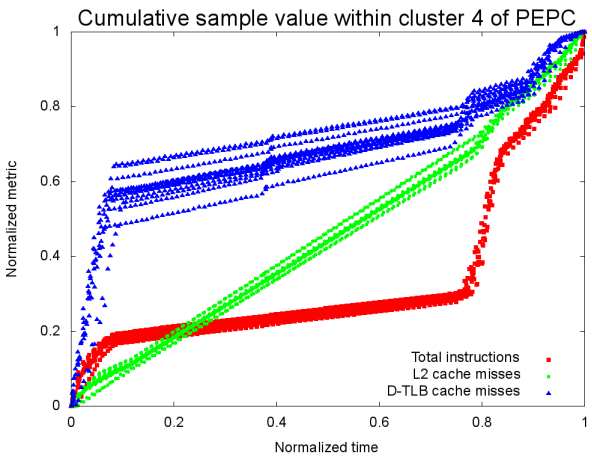
Performance metrics
16 MIPS
2.3 L2 misses/s
0.1 TLB misses/s

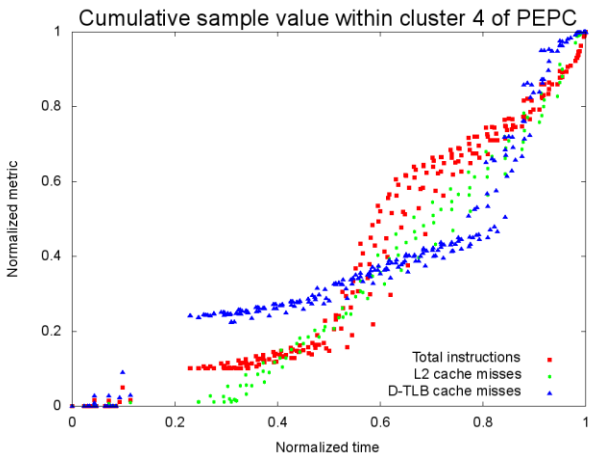



```
htable%node = 0
htable%key = 0
htable%link = -1
htable%leaves = 0
htable%childcode = 0
```

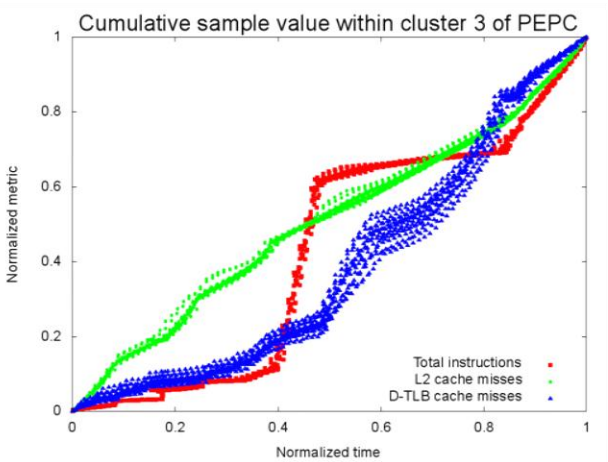
```
do i = 1, n
 htable(i)%node = 0
 htable(i)%key = 0
 htable(i)%link = -1
 htable(i)%leaves = 0
 htable(i)%childcode = 0
End do
```

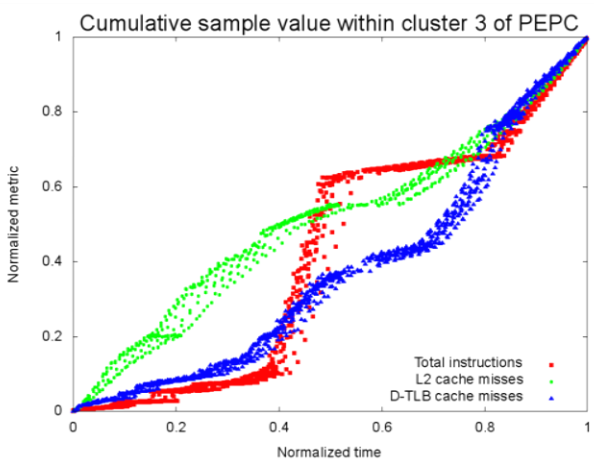
Example: PEPC (optimized)





Changes
-70% time
-18% instructions
-63% L2 misses
-78% TLB misses
254 MIPS (+163%)





Changes
-30% time
-1% instructions
-10% L2 misses
-32% TLB misses
544 MIPS (+34%)

Outline

- Extrae
- Paraver
  - General description
  - Examples
- Research topics
  - Structure detection
  - Folding
  - On-line analysis
- **Hands-on session**

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you!

For further information please contact us at
tools@bsc.es