www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

BSC Tools

Jesús Labarta, Judit Gimenez
BSC

Moscow, Nov 27th 2012

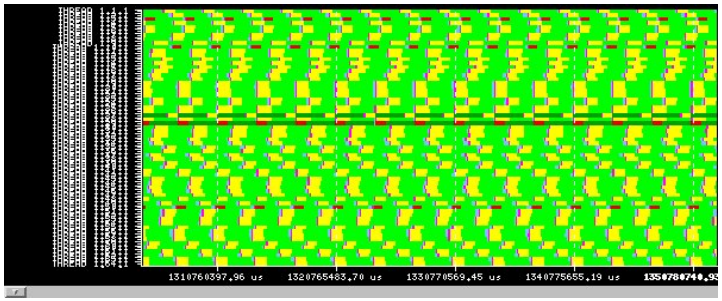
Index

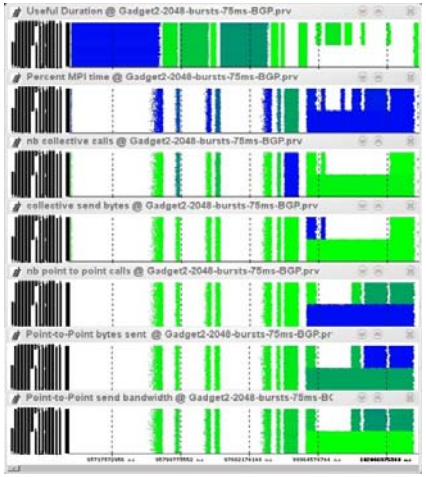
- Performance tools
- Extrae
- Paraver
  - Philosophy
  - Timelines and tables
  - Finding needles in haystacks
  - Scalability
  - Scaling model
  - More examples
  - Analysis recommendations
- Dimemas
  - Concept and uses
  - Using Dimemas
  - Scaling model
- Performance analytics
  - Spectral analysis
  - Clustering
  - Folding
- Tareador

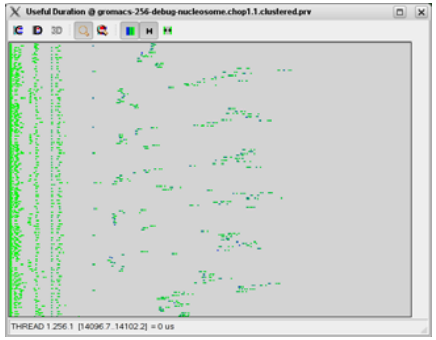
- Additional slides
  - Paraver internals

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, intelligence



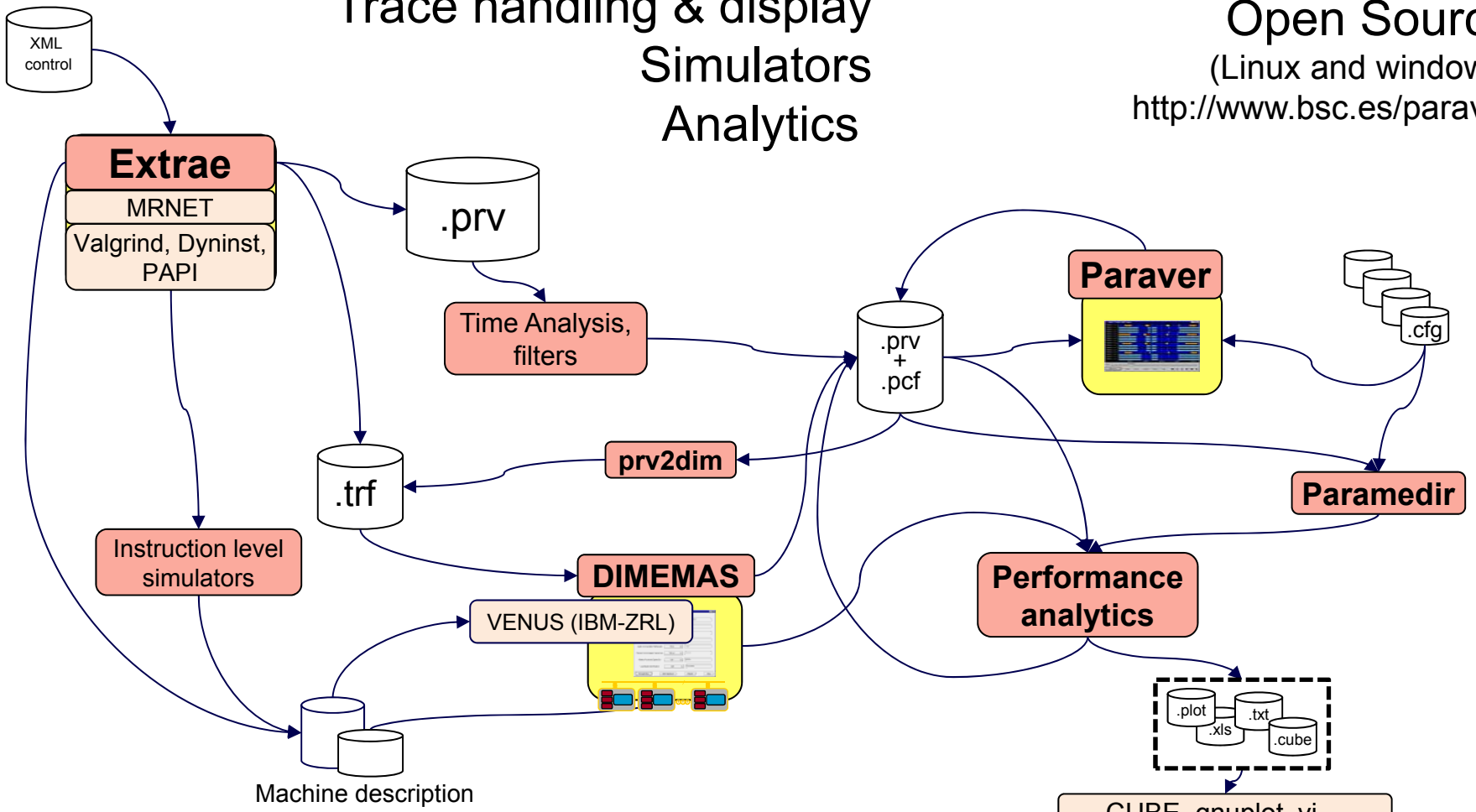




BSC – tools framework

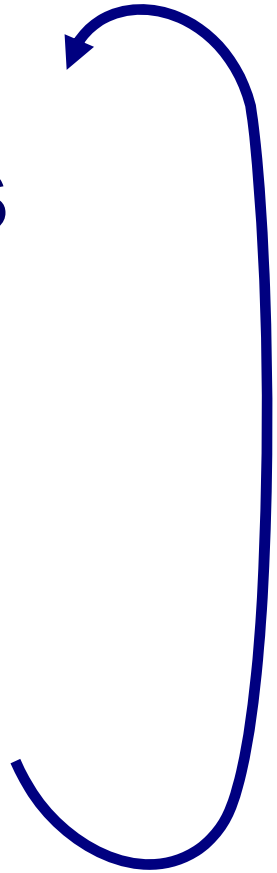
Trace handling & display
Simulators
Analytics

Open Source
(Linux and windows)
http://www.bsc.es/paraver



The importance of detail and intelligence

Performance analysis tools objective

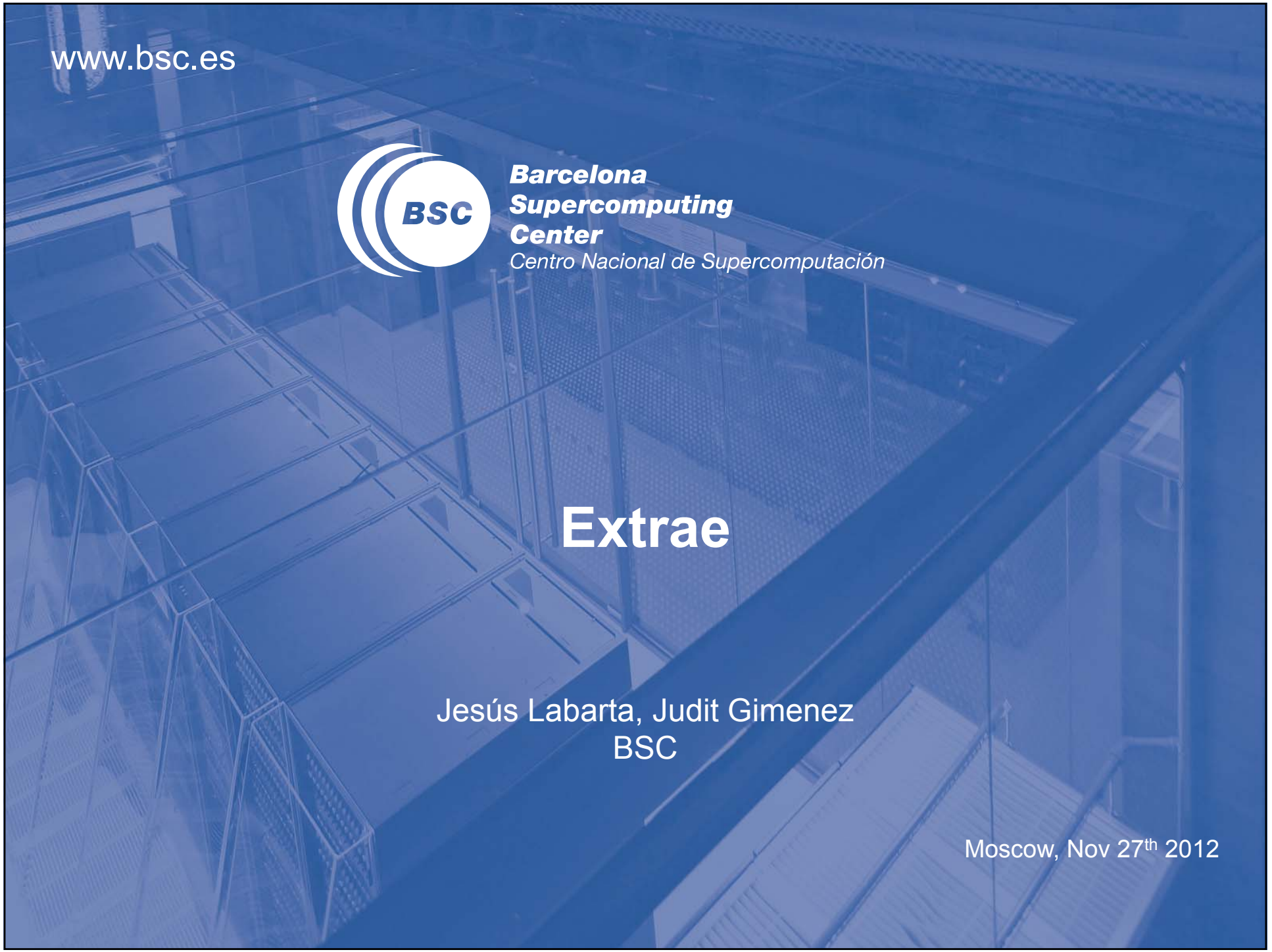


Help generate hypotheses

Help validate hypotheses

Qualitatively

Quantitatively


www.bsc.es
BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
Extrae
Jesús Labarta, Judit Gimenez
BSC
Moscow, Nov 27th 2012

Extrae

- Parallel programming model runtime
  - MPI, OpenMP, pthreads, OmpSs, 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 intercepts your app?

- LD_PRELOAD
  - Specific libraries for each combination of runtimes
    - MPI
    - OpenMP
    - OpenMP+MPI
    - …
- Dynamic instrumentation (not available in Graphit)
  - Based on DynInst (developed by U.Wisconsin/U.Maryland)
    - Instrumentation in memory
    - Binary rewriting
- Other possibilities
  - Link instrumentation library statically (i.e., PMPI @ BG/Q, …)
  - OmpSs (instrumentation calls injected by compiler + linked to library)

How to use Extrae?

- Build normal production binary of your app.
- Adapt job submission script
- If special features required select/adapt .xml configuration file
  - Bunch of examples distributed in the package
    - Look at $EXTRAE_HOME/share/example
- Run it and get the trace!!

Adapt job submission script

appl.job

```
#!/bin/bash

export NP=8
export INPUT=$1

cleo-submit -np $NP                ./HydroC -i $INPUT
```

Adapt job submission script

appl.job

```
#!/bin/bash

export NP=8
export INPUT=$1

cleo-submit -np $NP ./trace.sh ./HydroC -i $INPUT
```

trace.sh

```
#!/bin/bash

export EXTRAE_HOME=/export/hopsa/BSCtools/tools/extrae-2.3
export EXTRAE_CONFIG_FILE=extrae/extrae.xml

export LD_PRELOAD=$EXTRAE_HOME/lib/libmpitrace.so

export EXE=$1
export TRACENAME=${EXE}_$3.prv

$@
```

Trace control .xml

extrae.xml

```
<?xml version='1.0'?>

  <trace enabled="yes“
    home="/home/judit/tools/extrae-2.3"
    initial-mode="detail"
    type="paraver"
    xml-parser-id="Id: xml-parse.c 799 2011-10-20 16:02:03Z harald $"
  >

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

  <openmp enabled="no">
    <locks enabled="no" />
    <counters enabled="yes" />
  </openmp>

  <callers enabled="yes">
    <mpi enabled="yes">1-3</mpi>
    <sampling enabled="no">1-5</sampling>
  </callers>

...
```

Activate MPI tracing and emit hardware counters at MPI calls

Do not activate OpenMP tracing

Emit call stack information (number of levels) at acquisition points

Details in $EXTRAE_HOME/share/example/MPI/extrae_explained.xml

Trace control .xml (cont)

extrae.xml (cont)

```
<user-functions enabled="no" list="/home/bsc41/bsc41273/user-functions.dat">
  <max-depth enabled="no">3</max-depth>
  <counters enabled="yes" />
</user-functions>
```

Add instrumentation at specified user functions
Requires Dyninst based mpitrace

...

Trace control .xml (cont)

extrae.xml (cont)

```
…
  <counters enabled="yes">

    <cpu enabled="yes" starting-set-distribution="1">
      <set enabled="yes" domain="all" changeat-globalops="5">
        PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L2_DCM
        <sampling enabled="no" frequency="100000000">PAPI_TOT_CYC
      </set>
      <set enabled="yes" domain="user" changeat-globalops="5">
        PAPI_TOT_INS,PAPI_FP_INS,PAPI_TOT_CYC
      </set>
    </cpu>

    <network enabled="no" />

    <resource-usage enabled="no" />

    <memory-usage enabled="no" />

  </counters>

…
```

Emit counters or not

When to rotate between groups

Groups

Interconnection network counters
Just at end of trace because of large acquisition overhead

OS info (context switches,….)

Trace control .xml (cont)

mpitrace.xml (cont)

```
...

  <storage enabled="no">
    <trace-prefix enabled="yes">TRACE</trace-prefix>
    <size enabled="no">5</size>
    <temporal-directory enabled="yes" make-dir="no">/scratch</temporal-directory>
    <final-directory enabled="yes" make-dir="no">/gpfs/scratch/</final-directory>
    <gather-mpits enabled="no" />
  </storage>

  <buffer enabled="yes">
    <size enabled="yes">500000</size>
    <circular enabled="no" />
  </buffer>

...
```

Control of emitted trace …

… name, tmp and final dir …

… max (MB) per process size (stop tracing when reached)

Size of in core buffer (#events)

Trace control .xml (cont)

mpitrace.xml (cont)

```
…

  <trace-control enabled="yes">
    <file enabled="no" frequency="5m">/gpfs/scratch/bsc41/bsc41273/control</file>
    <global-ops enabled="no"></global-ops>
     <remote-control enabled="no">
      <signal enabled="no" which="USR1"/>
    </remote-control>
  </trace-control>



  <others enabled=“no">
    <minimum-time enabled="no">10M</minimum-time>
    <terminate-on-signal enabled="no">USR2</terminate-on-signal>
  </others>

…
```

External activation of tracing (creation of file will start tracing)

Stop tracing after elapsed time …

… or when signal received

Trace control .xml (cont)

mpitrace.xml (cont)

...

```
<bursts enabled="no">
   <threshold enabled="yes">500u</threshold>
   <counters enabled="yes" />
   <mpi-statistics enabled="yes" />
 </bursts>
```

… emit only computation bursts of a minimal duration …

… plus summarized MPI events

```
<sampling enabled="no" type="default" period="5m" />
```

Activate/not time based sampling and how often

...

Trace control .xml (cont)

mpitrace.xml (cont)

```
…

  <merge enabled="yes"
    synchronization="default"
    binary="$EXE$"
    tree-fan-out="16"
    max-memory="512"
    joint-states="yes"
    keep-mpits="yes"
    sort-addresses="yes"
  >
    $TRACENAME$
  </merge>


</trace>
```

Merge individual traces into global application trace at end of run …

… into this trace name

LD_PRELOAD library selection

- Library depends on programming model

Programming model	Library
Serial	libseqtrace
Pure MPI	libmpitrace[f][1]
Pure OpenMP	libomptrace
Pure Pthreads	libpttrace
CUDA	libcudatrace
MPI + OpenMP	libompitrace[f] [1]
MPI + Pthreads	libptmpitrace[f] [1]
Mpi + CUDA	libcudampitrace[f] [1]

[1] for Fortran codes

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Paraver

Jesús Labarta, Judit Gimenez
BSC

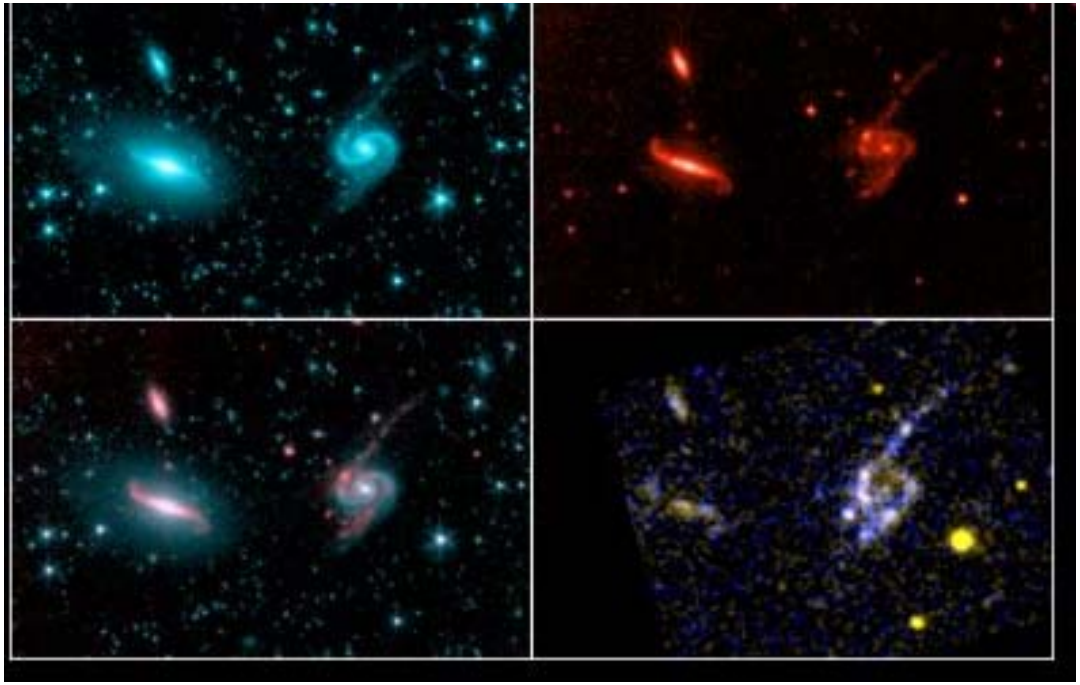
Moscow, Nov 27th 2012

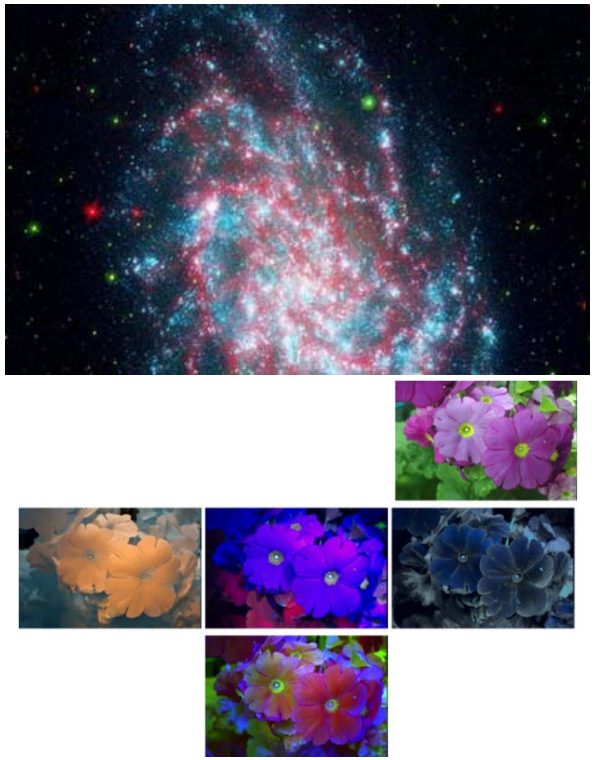
“That what is simple is rarely understood”

my iPads Shangai cookies

Multispectral imaging

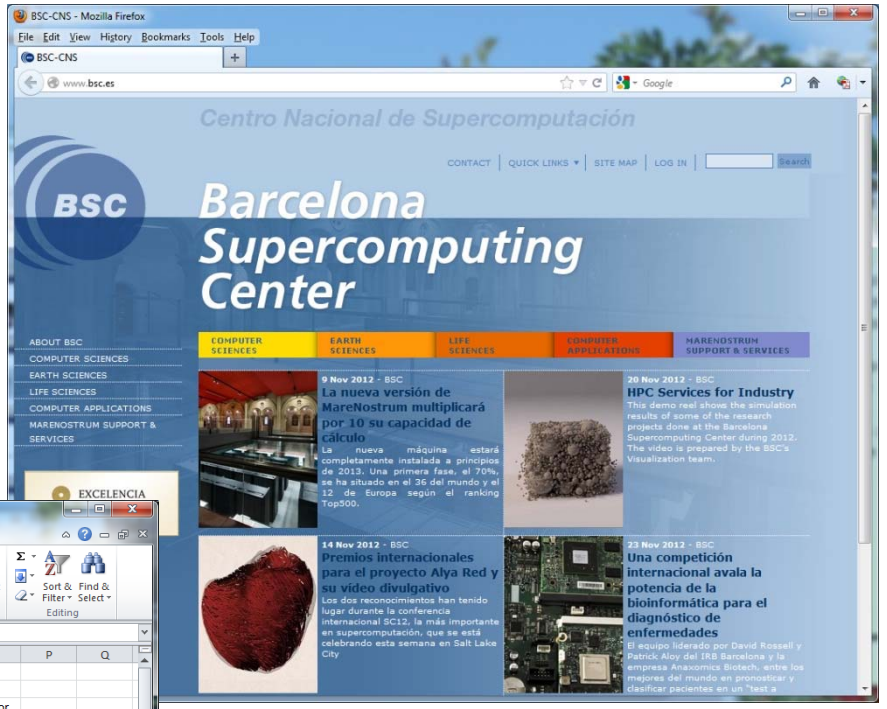
- Different looks at one reality
  - Different spectral bands (light sources and filters)
- Highlight different aspects
  - Can combine into false colored but highly informative images

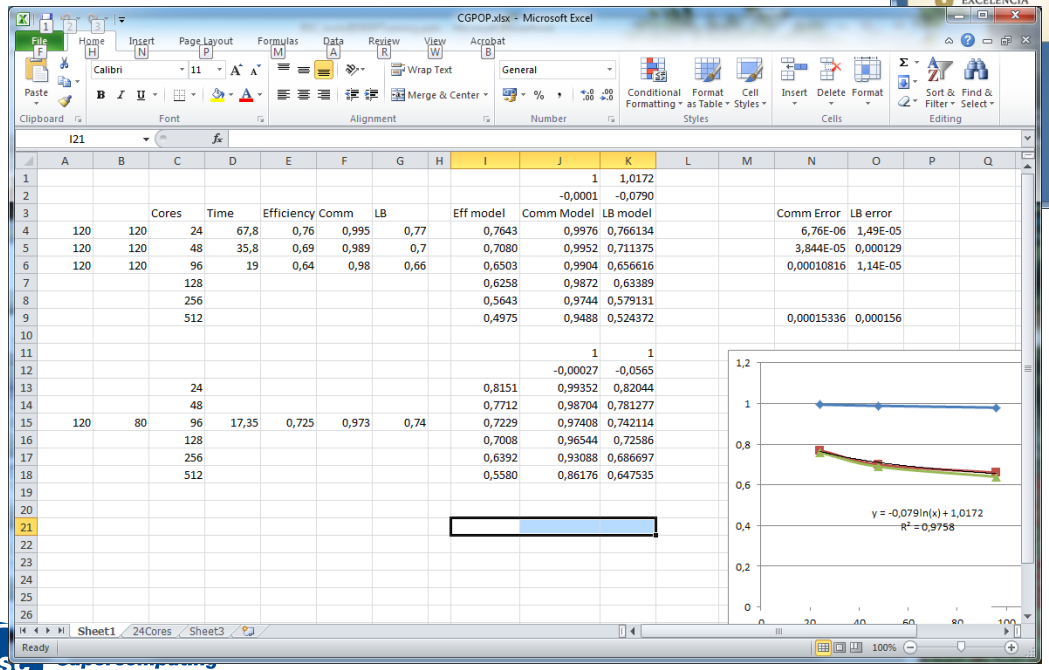




Spreadsheets and browsers

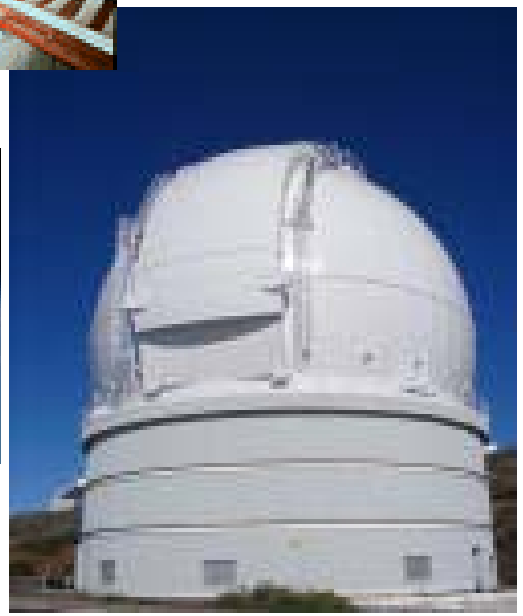
- Display, manipulate data
- User defined operations





Instruments

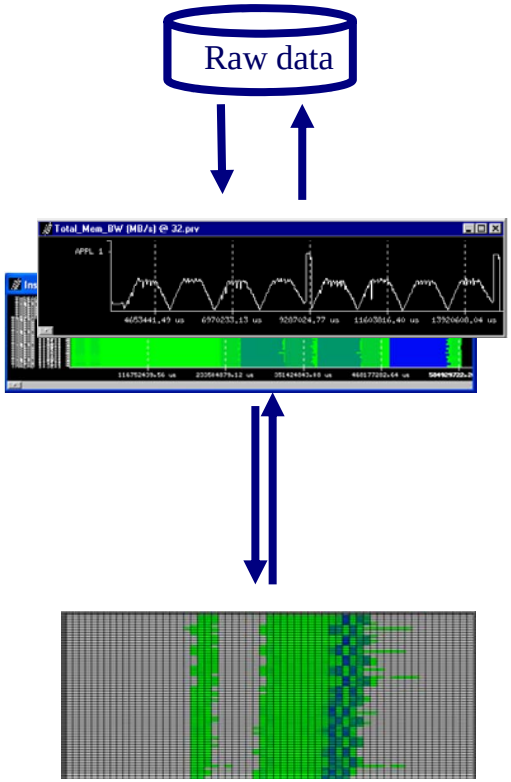
- One experiment
  - “Expensive” resources
- Lots of analysis
- To obtain sufficient information/insight
  - Avoid flying blind
  - Identification of productive next steps



What is Paraver

- A browser …
- …to manipulate (visualize, filter, cut, combine, …) ….
- … sequences of time-stamped events …
- … with a multispectral philosophy …
- … and a mathematical foundation …
- … that happens to be mainly used for **performance analysis**

Paraver – Performance data browser



Timelines

2/3D tables (Statistics)

Trace visualization/analysis

+ trace manipulation

Goal = Flexibility

No semantics

Programmable

Configuration files

Distribution

Your own

Comparative analyses

Multiple traces

Synchronize scales

Paraver mathematical foundation

- Every behavioral aspect/metric described as a function of time
  - Possibly aggregated along
    - the process model dimension (**thread**, process, application, workload)
    - The resource model dimension (core, node, system)
  - Need a language to describe how to compute such functions of time.
    - Basic operators (from) trace records
    - Ways of combining them
- Those functions of time can be rendered into a 2D image
  - Timeline
- Statistics can be computed for each possible value or range of values of that function of time
  - Tables: profiles and histograms

Timelines

- Each window displays one view
  - **Piecewise constant** function of time

$$S(t) = S_i, i \in [t_i, t_{i+1})$$

- Types of functions
  - Categorical

$$S_i \in [0, \mathrm{n}] \subset N, \quad n <$$

    - State, user function, outlined routine
  - Logical

$$S_i \in \{0, 1\}$$

    - In specific user function, In MPI call, In long MPI call
  - Numerical

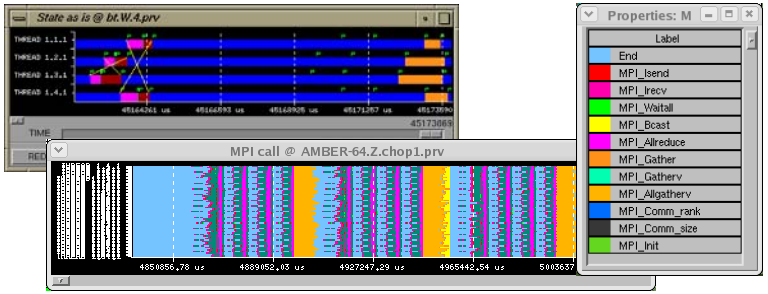
$$S_i \in R$$

    - IPC, L2 miss ratio, Duration of MPI call, duration of computation burst

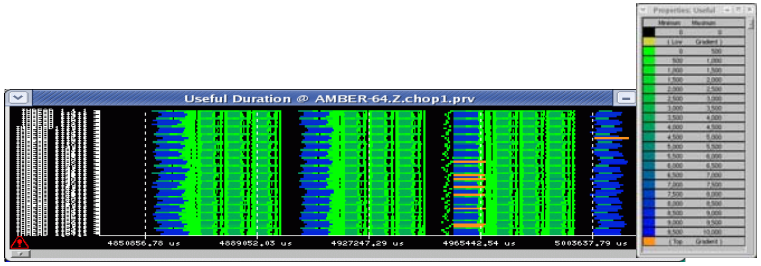
Timelines

- Representation
  - Function of time

  - Colour encoding



  - Not null gradient
    - Black for zero value
    - Light green → Dark blue



- Non linear rendering to address scalability

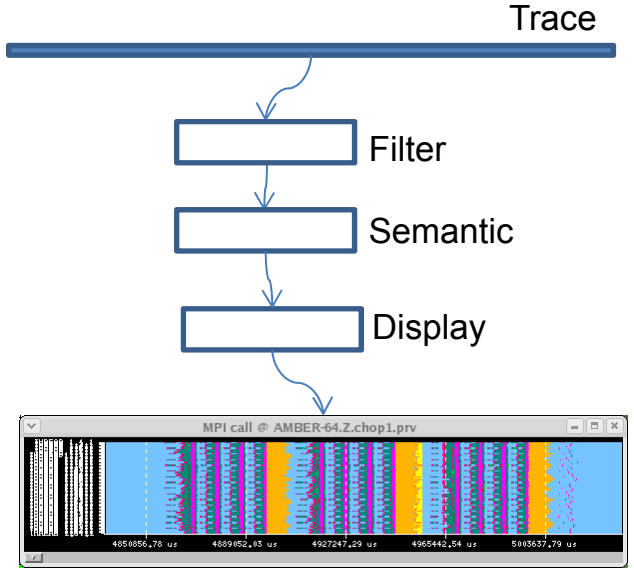
Basic functions of time

- The **filter module** presents a subset of the trace to the semantic module. Each thread th is described by
  - A sequence of events $Ev_i, i \in N$, states $St_i, i \in N$ and communications $C_i, i \in N$
  - For each event let $T(Ev_i)$ be its time and $V(Ev_i)$ its value
  - For each state let $T_s(St_i)$ be its start time $T_e(St_i)$ its stop time and $V(St_i)$ its value
  - For each Communication let $T_S(C_i)$ be its send time, $T_R(C_i)$ its receive time, $Sz(C_i)$ its size.
  - $Partner(C_i)$ and $Dir(C_i) \in \{send, recv\}$ identify the partner process and direction of the transfer

- **Semantic module** builds

$$s(t) = S(i), t \in [t_i, t_{i+1}), i \in \mathrm{N}$$

Function of time Series of values



Basic functions of time

Semantic module

- From Events to functions of time
  - Last event value $s(i) = V(Ev_i)$
  - Next event value $S(i) = V(Ev_{i+1})$
  - Average Next Event Value $s(i) = \frac{V(Ev_{i+1})}{T(Ev_{i+1}) - T(Ev_i)}$
  - Interval btw. Events $S(i) = T(Ev_{i+1}) - T(Ev_i)$

Semantic module

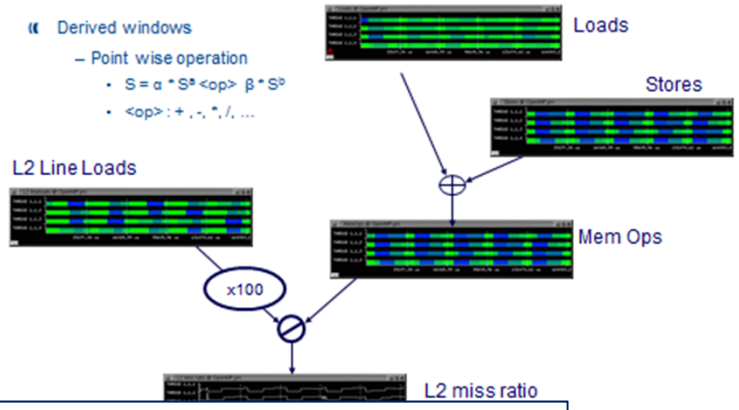
- From communication records to functions of time
  - Send Bytes $s(t) = \sum_j Sz(C_j), j \mid (T_s(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$
  - Send Bandwidth $s(t) = \sum_j \frac{Sz(C_j)}{T_R(C_j) - T_S(C_j)}, j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$
  - Msgs in transit $s(t) = \sum_j sign(j), j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$
  - Recv. Bandwidth $s(t) = \sum_j \frac{Sz(C_j)}{T_R(C_j) - T_S(C_j)}, j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(C_j) == recv)$
  - Rec. Negative Msgs $s(t) = \sum_j sign(j), j \mid (T_R(C_j) < t) \wedge (T_S(Cj) > t) \wedge (Dir(Cj) == recv)$
  - Comm. Partner $s(t) = Partner(C_j), j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t)$
  - Bytes btw. Events $S(i) = \sum_j Sz(C_j), j \mid T_S(C_j) \in [T(Ev_i), T(Ev_{i+1})) \vee T_R(C_j) \in [T(Ev_i), T(Ev_{i+1}))$

Composition

- S'(t) = f(S(t)) S' = f ° S
  - Sign $S'(t) = sign(S(t))$
  - 1-sign $S'(t) = 1 - sign(S(t))$
  - Select range $S'(t) = S(t) \in [a, b] ? S(t) : 0$
  - Sign ° Is equal $S'(t) = sign(S(t) = a ? S(t) : 0)$
  - Delta $S'(t) = S_{i+1} - S_i$
  - Stacked value

Semantic module

- Derived windows
  - Point wise operation
    - S = α * S^β <op> β * S^δ
    - <op> : + , -, *, /, ...

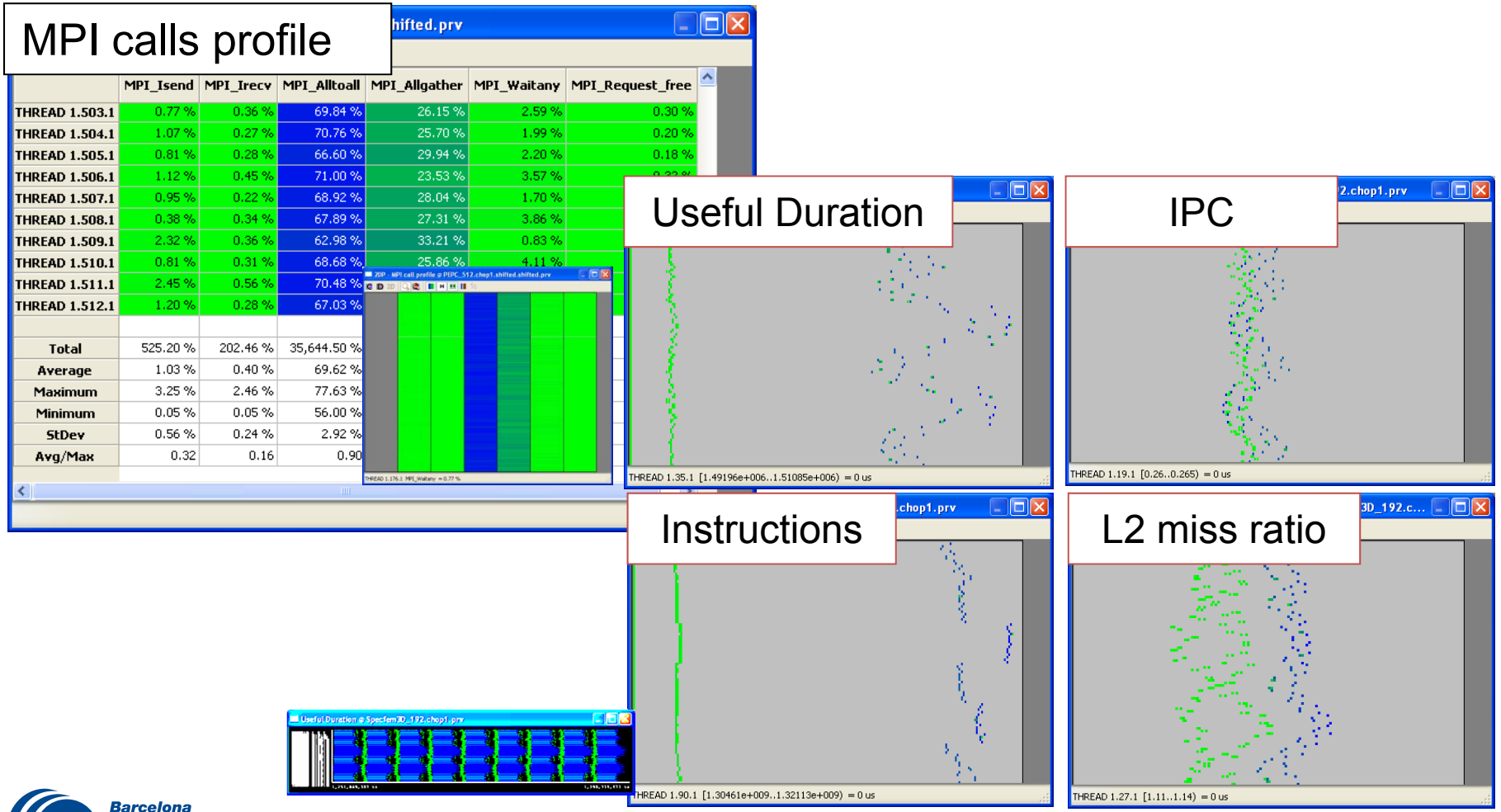


See slides at end of presentation for details

Tables: Profiles, histograms, correlations

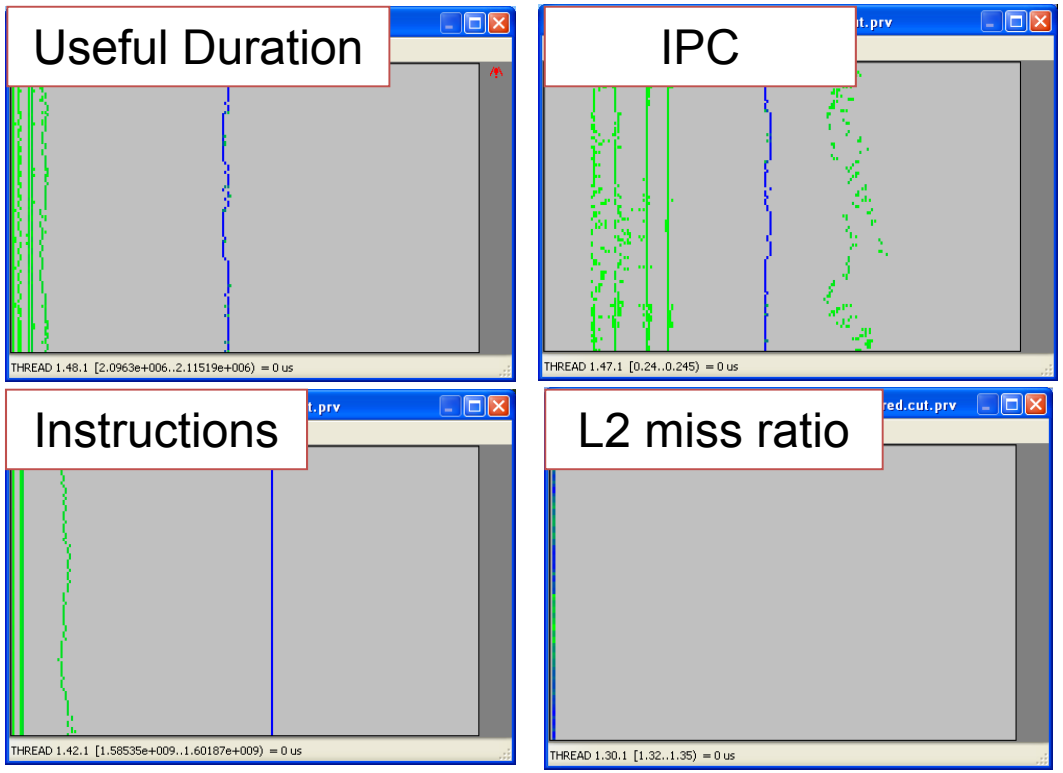
- Huge number of statistics computed from timelines

MPI calls profile



Tables: Profiles, histograms, correlations

- By the way: six months later ….



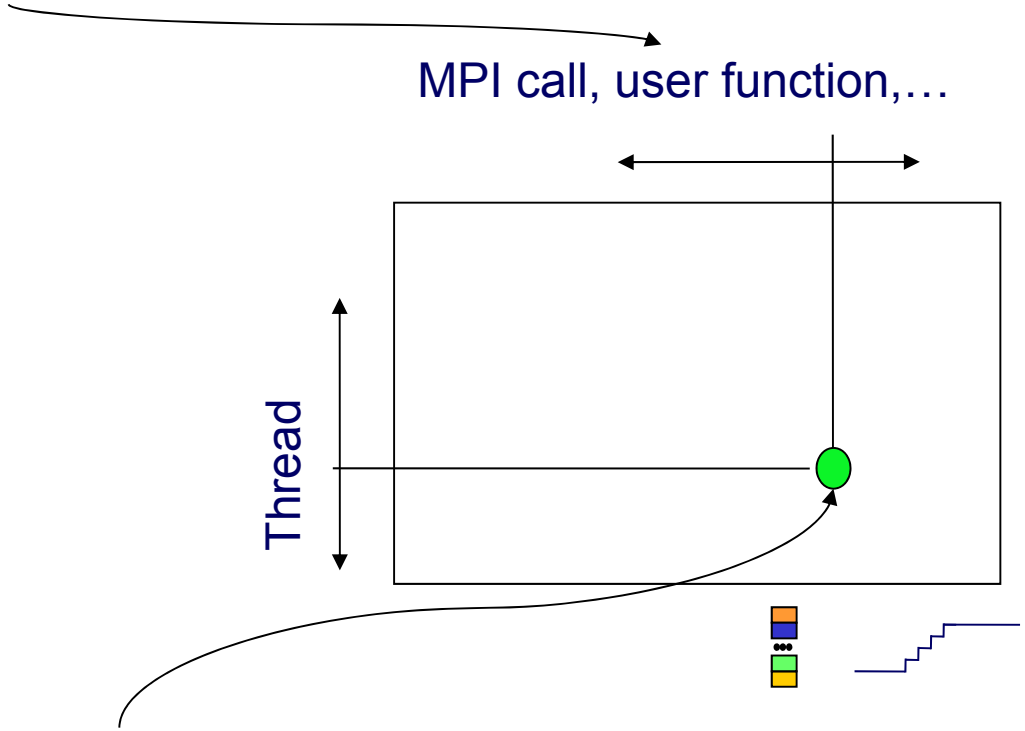
How to read profiles

One columns per specific value of categorical **Control window**

MPI profile @ Iberia-128-CA.chop1.1it.shifted.prv

	End	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Allreduce	MPI_Comm
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 %	

MPI_Wait

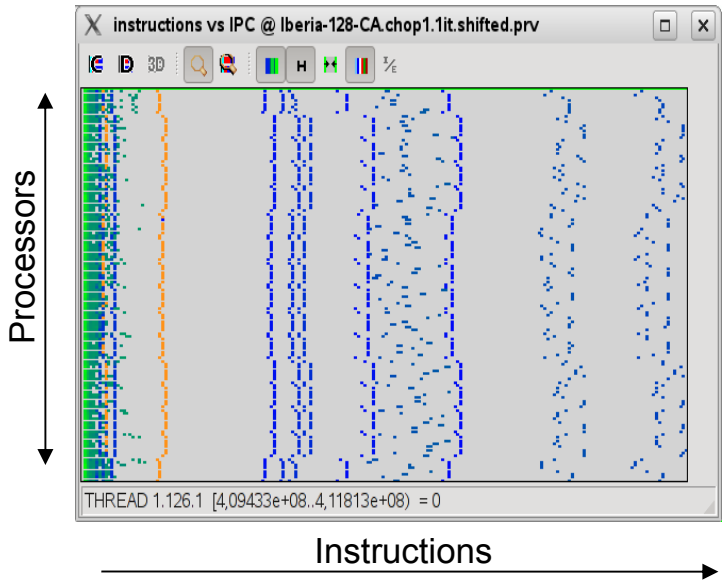


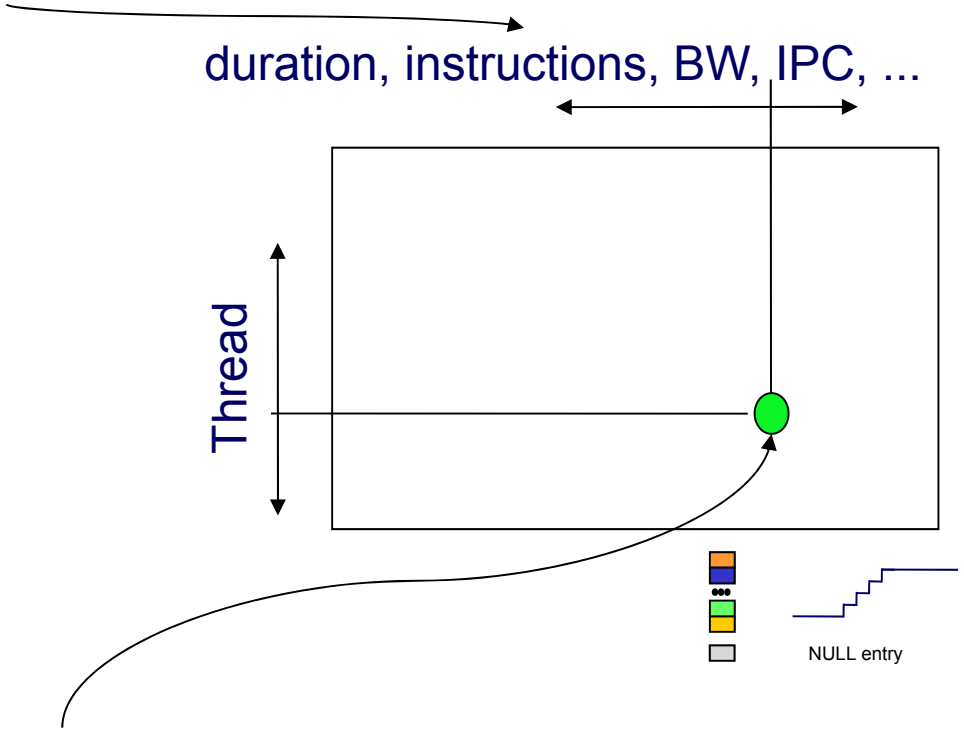
Value/color is a statistic computed for the specific thread
when control window had the value corresponding to the column

Relevant statistics:
Time, %time, #bursts, Avg. burst time
Average of **Data window**

How to read histograms

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



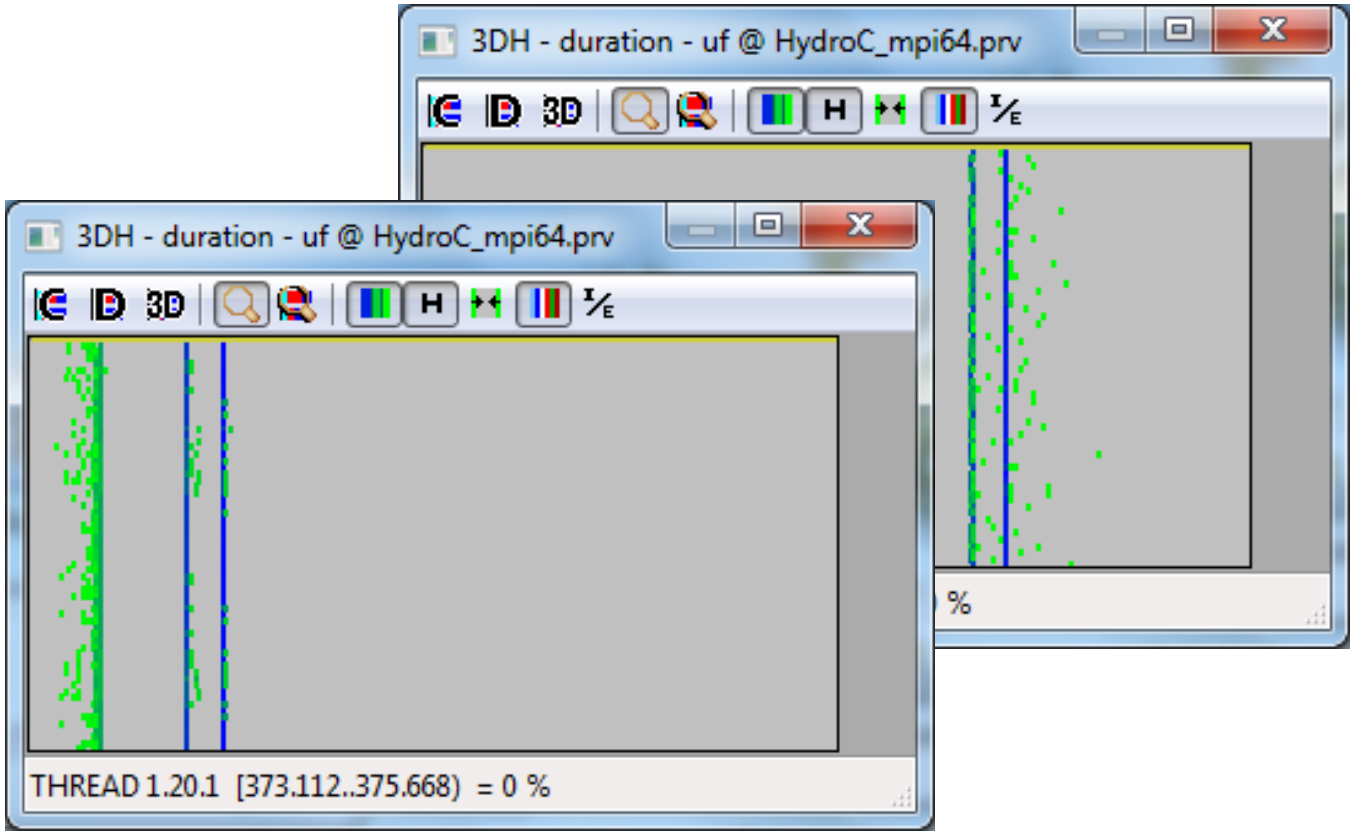


Value/color is a statistic computed for the specific thread
when control window had the value corresponding to the column

Relevant statistics:
Time, %time, #bursts, Avg. burst time
Average of **Data window**

3D

- An additional control dimension
  - One table (plane) per value (or range) of 3D window
  - i.e. histogram of duration of each function



Tables

Tables

- Single flexible quantitative analysis mechanism
- Let
  - cw_1 and cw_2 two views we will call control views
  - dw a view we will call data window

For each window w

$$S^{w}_{th}(t) = S^{w}_{th}(i), t \in [t^{w}_{i}, t^{w}_{i+1})$$

- For each control window we define a set of bins

$$bin^{cw}_{j} = [range^{cw}_{j}, range^{cw}_{j+1}) \qquad range^{cw}_{j+1} = range^{cw}_{j} + delta^{cw}$$

- And the discriminator functions

$$\delta^{cw}_{j}(t) = ((S^{cw}(t) \in bin^{cw}_{j})?1:0)$$
$$\delta_{j,k}(t) = \delta^{cw_1}_{j}(t) * \delta^{cw_2}_{k}(t)$$

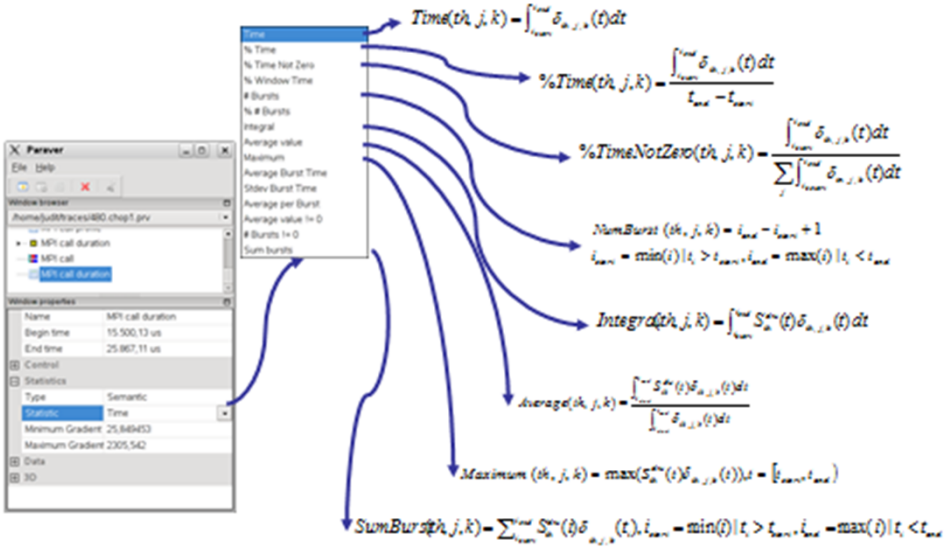
Identify regions with cw's within the (j,k) bin

- The 3D analysis module computes a cube (or plane in the case of 2D) of statistics

$$M(thread, j, k) = statistic(S^{dw}_{th}(t) * \delta_{th,j,k}(t))$$

- Where the statistic can represent the average value, the number of intervals,....

2D analysis module



See slides at end of presentation for details

Paraver:
finding needles in haystacks

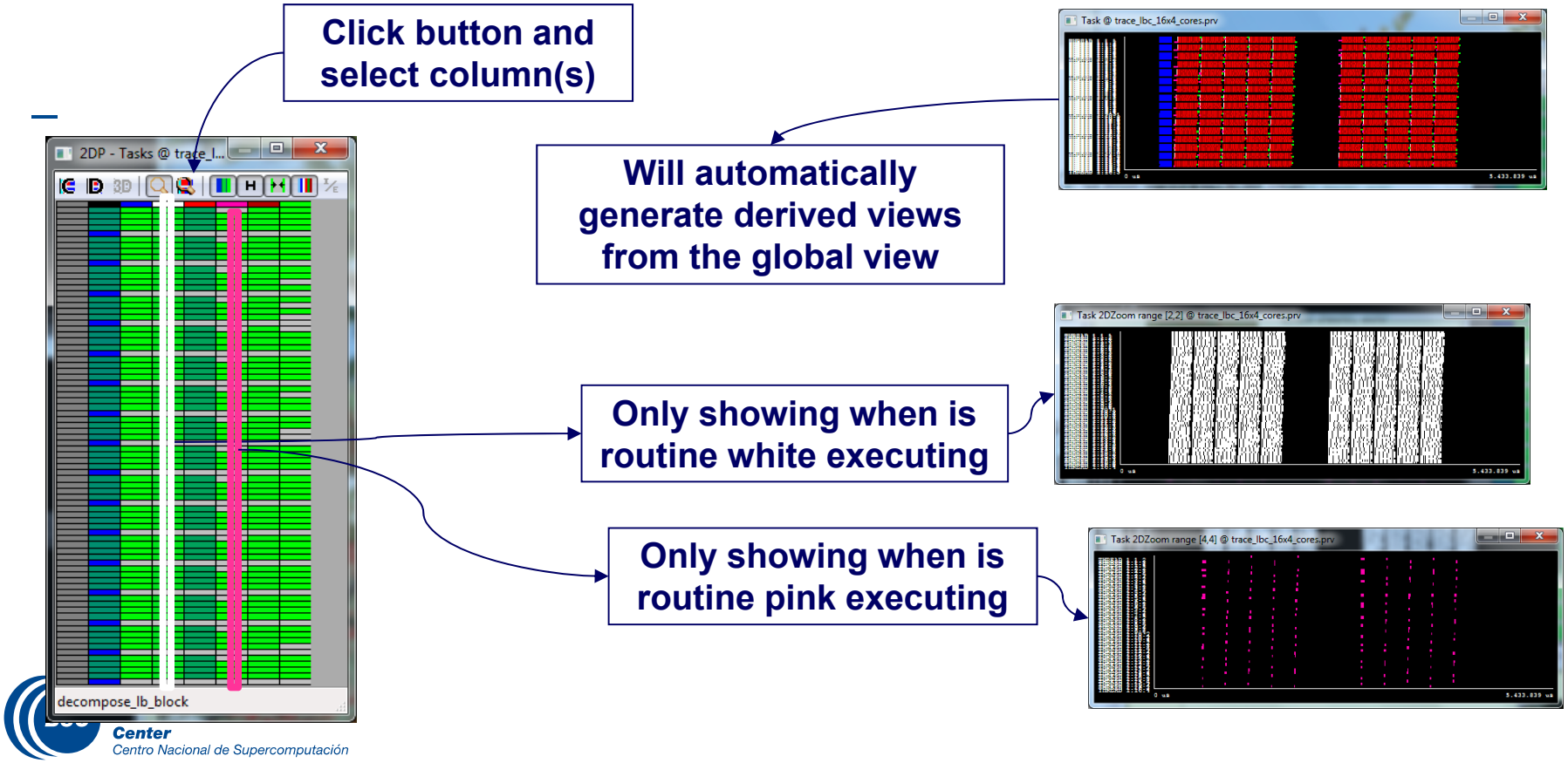
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Comparative analyses

- Possible to load several traces
- Copy and paste
  - right click menu
  - From one window to another: time, duration, size, objects displayed,…
  - Time between windows and tables: analysis will be computed only for the selected time interval

From tables to timelines

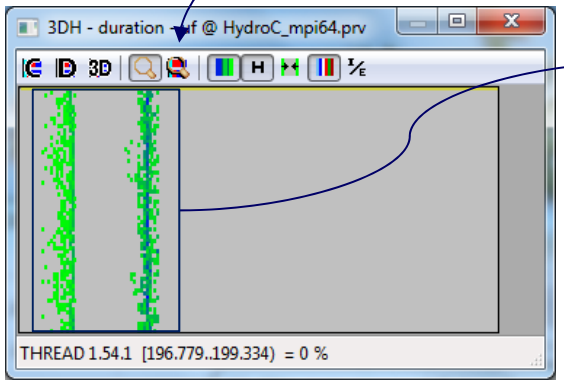
- Where in the timeline do the values in certain table columns appear?
  - ie. want to see the time distribution of a given routine?



From tables to timelines

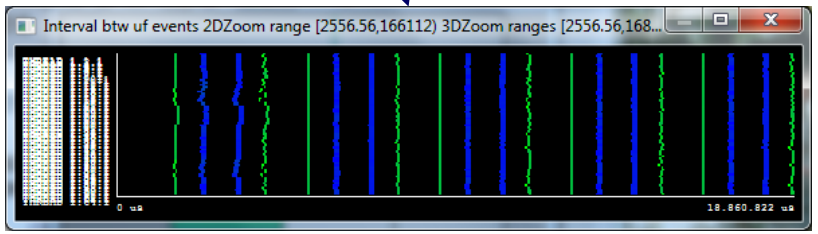
- Where in the timeline do the values in certain table columns appear?
  - ie. want to see where the timeline happen computation bursts of a given length?

Click button and select column(s)



3D histogram of duration of routine foo

Will automatically generate

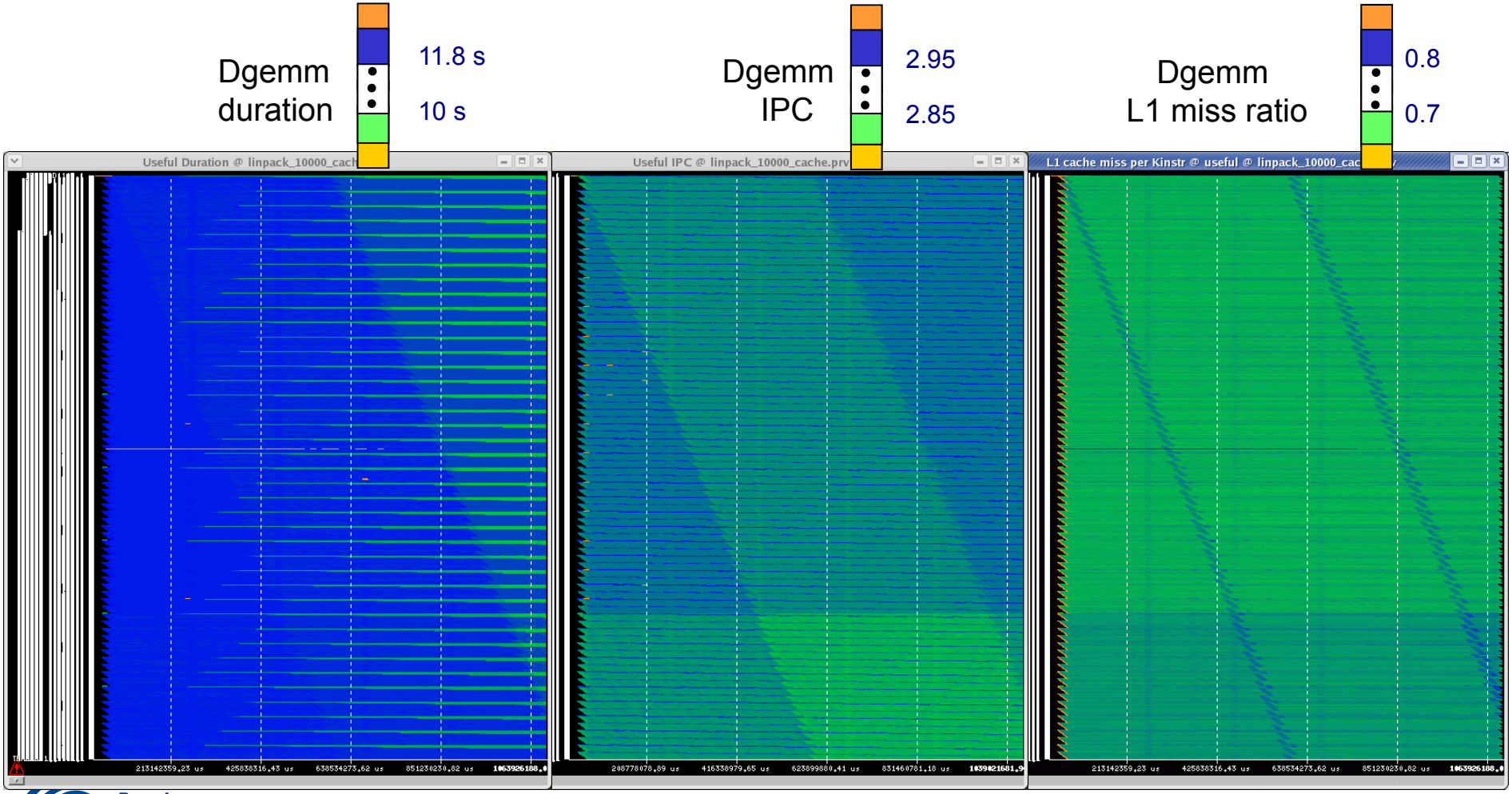


Only showing duration of routine foo

Scalability

Scalability of Presentation

- Linpack @ Marenostrum: 10k cores x 1700 s



Scalability of analysis

Jugene

Jaguar

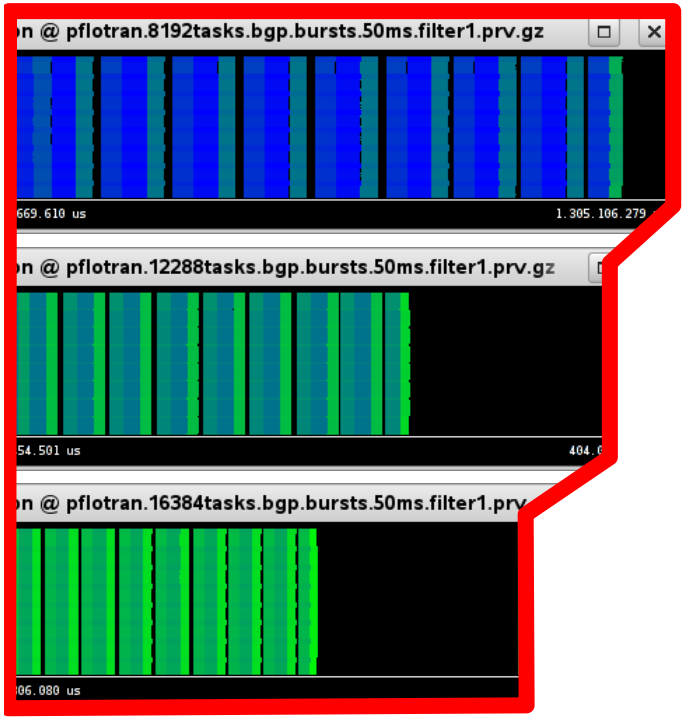
~ 105 seconds

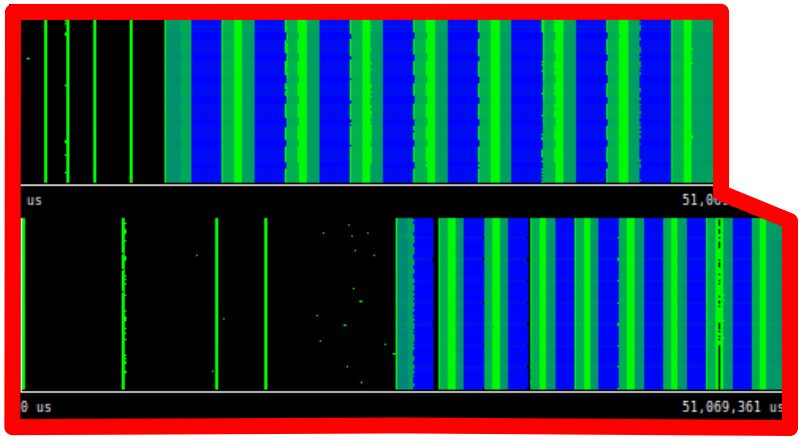
~ 47 seconds

8K cores

12K cores

16K cores





Tran

Flow

Flow

Tran

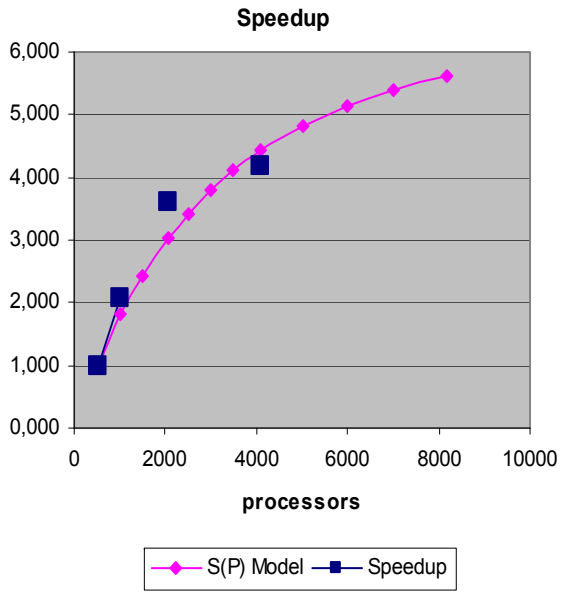
PFLOTRAN

Data reduction techniques

- Software counters
  - Summarize information of some event types (ie. MPI calls) by emitting aggregate counts
  - Emit counts at structurally relevant points (i.e. begin and end of long computation phases)
- Representative cuts
  - Emit full detail only on selected intervals, representative of full program execution
- On and off line combinations
  - By instrumentation
  - By paraver filtering

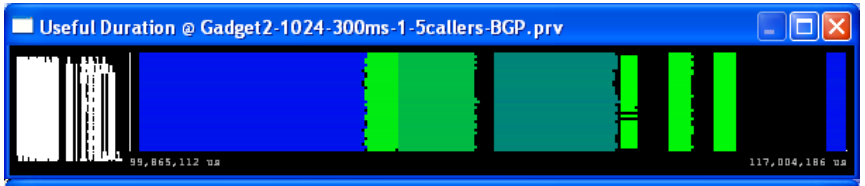
J. Labarta, et al.: "Scalability of tracing and visualization tools", PARCO 2005

Software counters

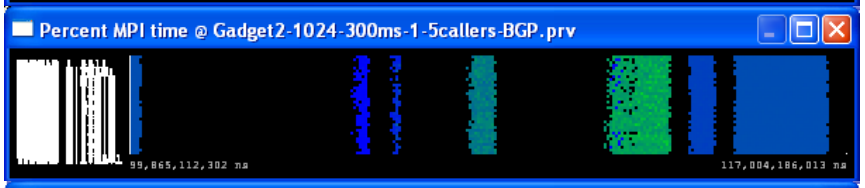


GADGET, PRACE Case A, 1024 procs

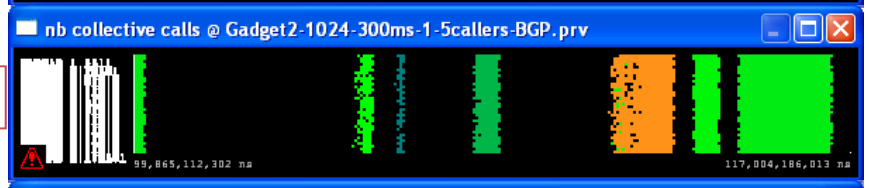
Useful duration



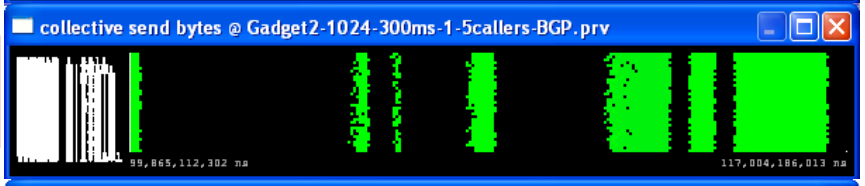
% MPI time



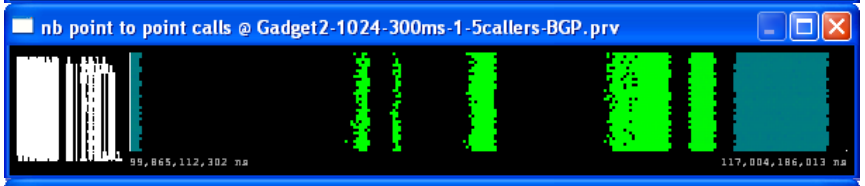
collectives



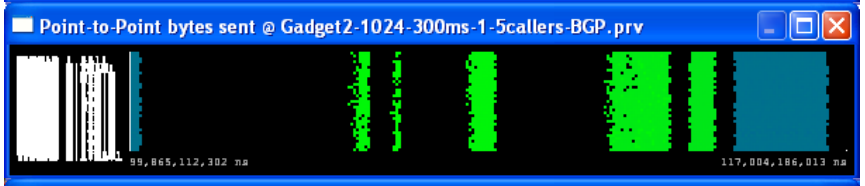
Collective bytes



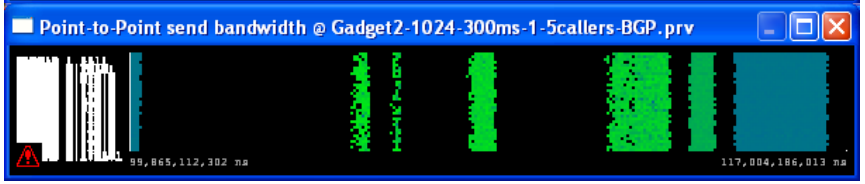
p2p



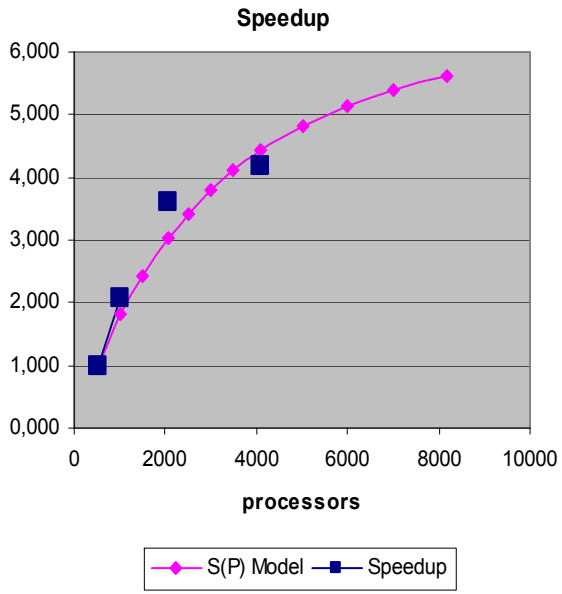
p2p bytes



p2p BW

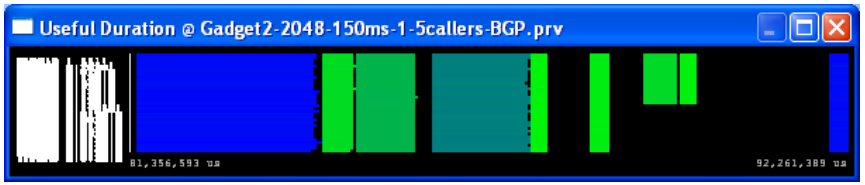


Software counters

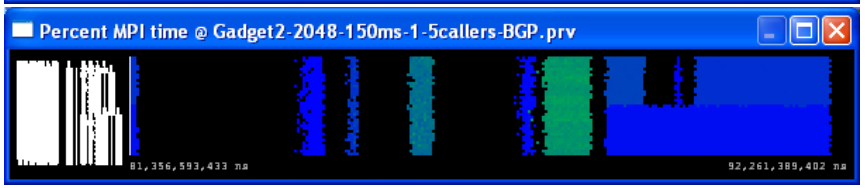


GADGET, PRACE Case A, 2048 procs

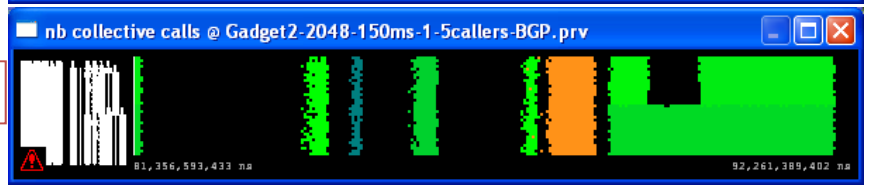
Useful duration



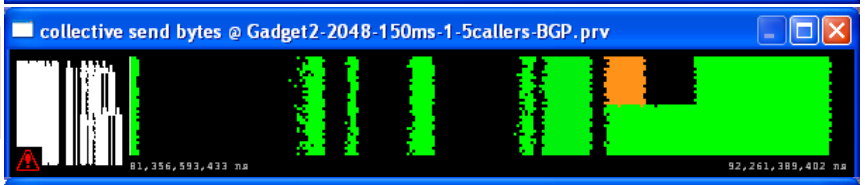
% MPI time



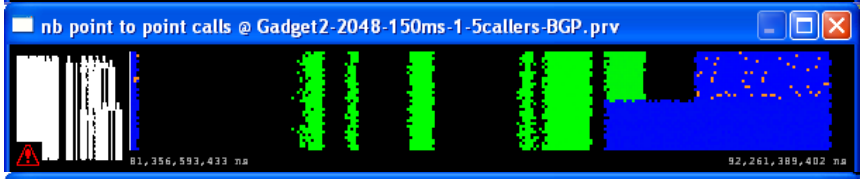
collectives



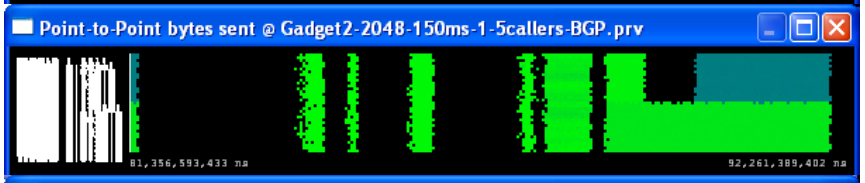
Collective bytes



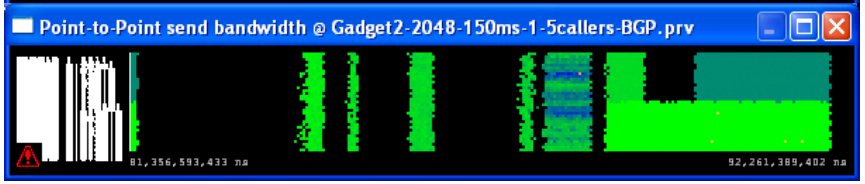
p2p



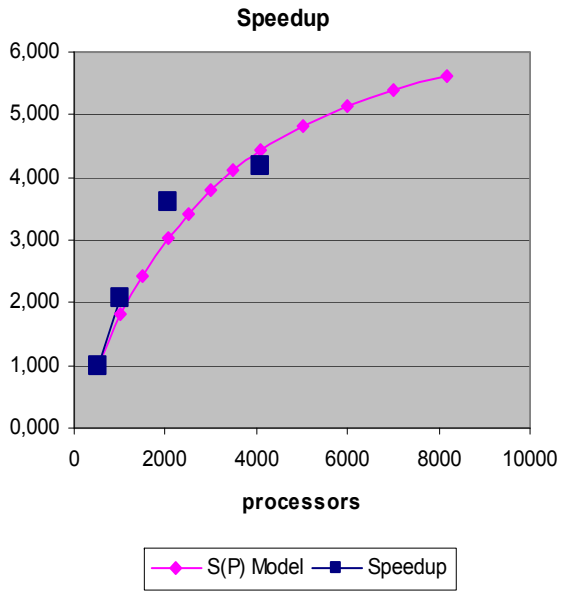
p2p bytes



p2p BW

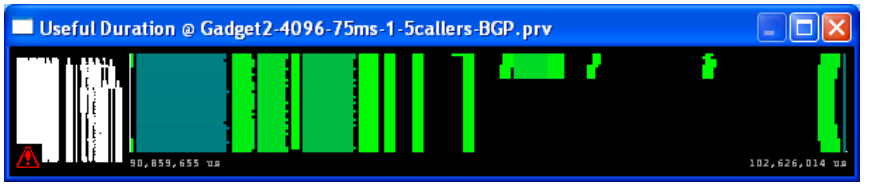


Software counters

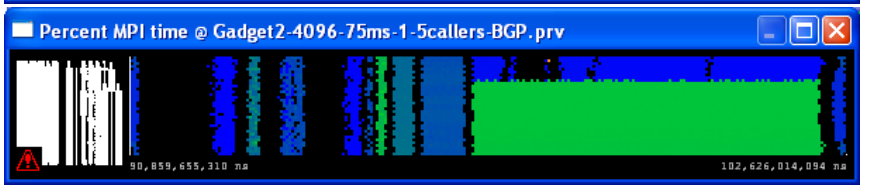


GADGET, PRACE Case A, 4096 procs

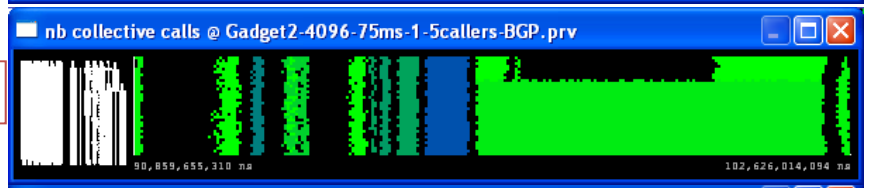
Useful duration



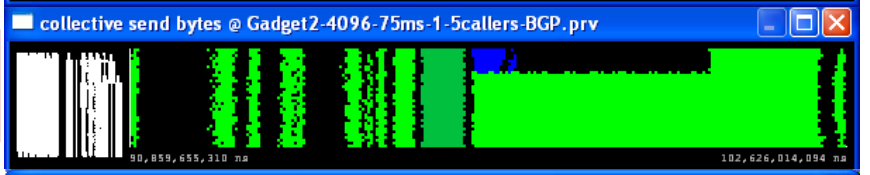
% MPI time



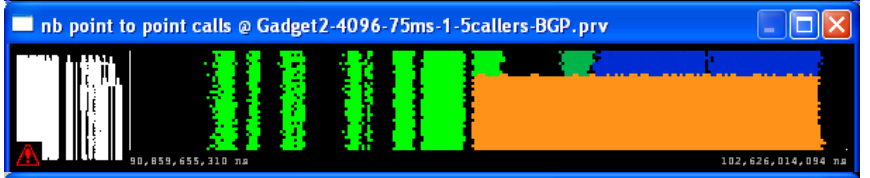
collectives



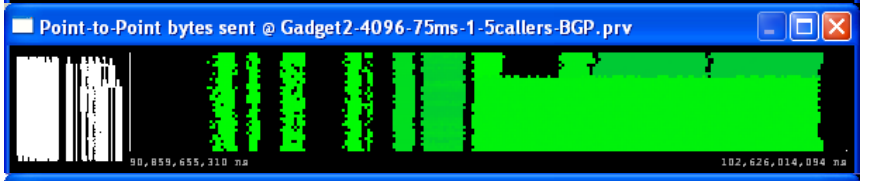
Collective bytes



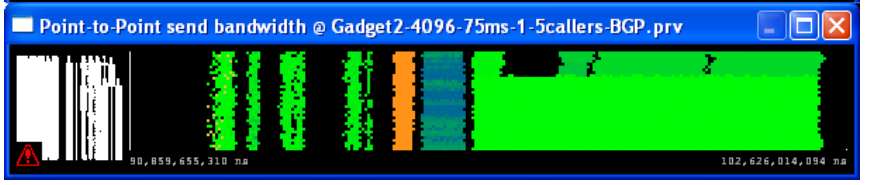
p2p



p2p bytes



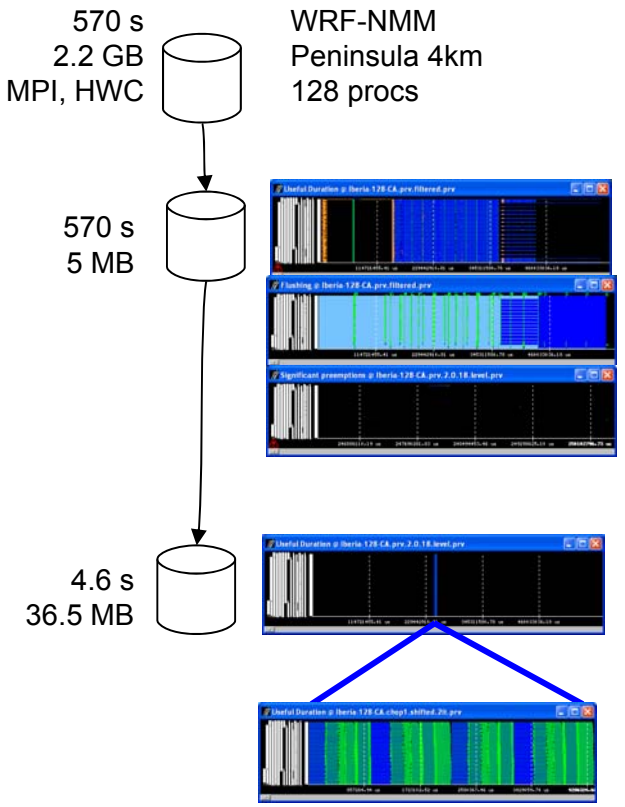
p2p BW



Handling very large traces

- Paraver data handling utilities
  - If trying to load a very large trace, Paraver will ask if you want to filter it

- Three steps:
  - Filter original trace discarding most of the records only keeping most relevant information (typically computation bursts longer than a given lower bound)
  - Analyze coarse grain structure of trace. Typically useful_duration.cfg
  - Cut original trace to obtain a fully detailed trace for the time interval considered representative or of interest



Filtering very large traces

Trace to which it will be applied

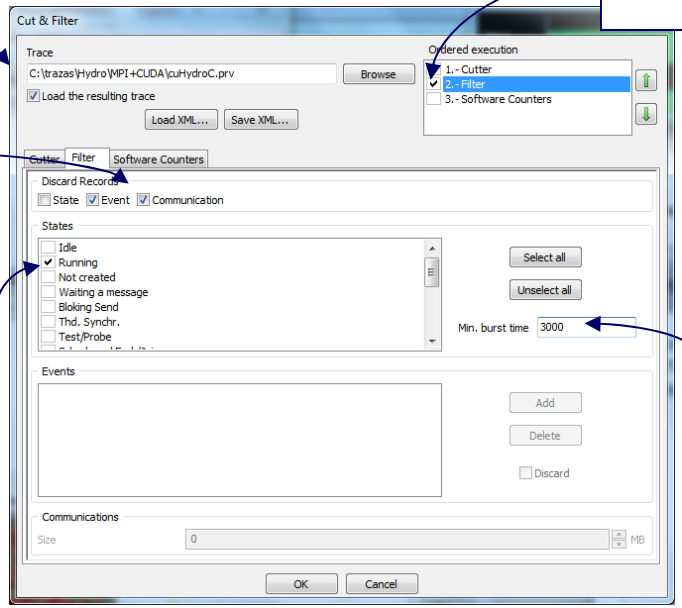
A trace with basename.filter1.prv will be generated

Select filtering option

Discard events and communications

Keep only Running bursts ….

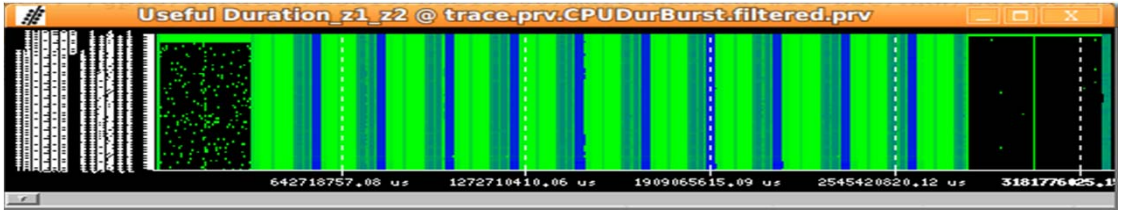
--- longer than 3000 ns



Analyze coarse grain structure

- Filtered trace is a Paraver trace

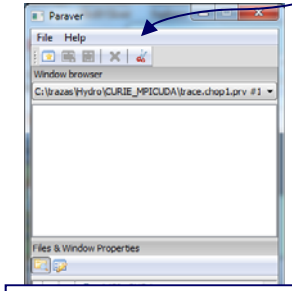
- Can be analyzed with standard cfgs as long as the information they require is still in the trace
  - A typical view that shows a lot of the structure of a trace is useful_duration.cfg
  - Repetitive structure is often apparent
  - Perturbations can also be typically identifed
  - A clean/representative interval can be identified



Cutting very large traces

- Load a filtered trace and use the scissors tool

Scissors tool



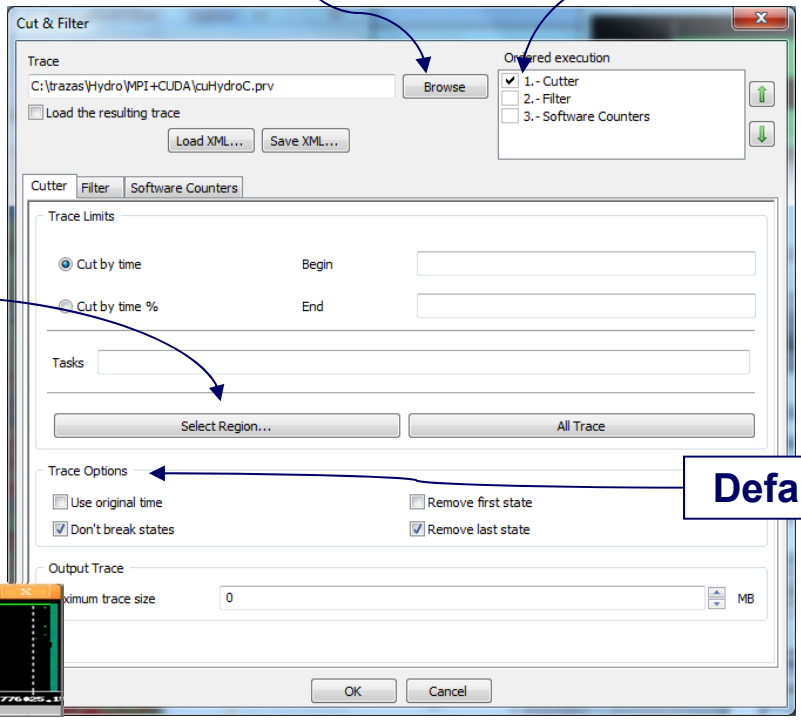
Click to select region

Select time interval by clicking left and right limits in a window of the filtered trace previously loaded

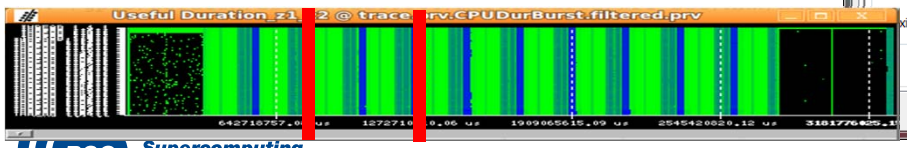
Recommended cuts within long computation bursts

Browse to select file from which the cut will be obtained

Select cutter



Default setups



Scaling model

Presenting application performance

- Factors modeling parallel efficiency
  - **Load balance** (LB)
  - **Micro load balance** (μLB) or **serialization**
  - **Transfer**

$$\eta = LB * \mu LB * Transfer$$

$CommEff$

- Factors describing serial behavior
  - Computational complexity: **#instr**
  - Performance: **IPC**

$$T_{Comp} = \frac{\#instr}{IPC}$$

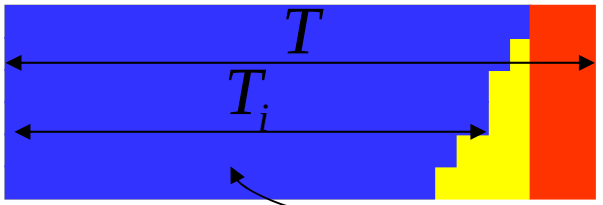
- Scaling model

$$Sup = \frac{P}{P_0} * \frac{\eta}{\eta_0} * \frac{IPC}{IPC_0} * \frac{\#instr_0}{\#instr}$$

M. Casas et al, "Automatic analysis of speedup of MPI applications". ICS 2008.

Scaling model

$$\eta = LB * CommEff$$



$$eff_i = \frac{T_i}{T}$$

$$CommEff = \max(eff_i)$$

$$LB = \frac{\sum_{i=1}^{P} eff_i}{P * \max(eff_i)}$$

IPC

$\#instr$

Directly from real execution metrics

2DP - MPI call profile @ trace_24h_atmos_symbols.cho...

	Outside MPI	MPI_Recv	MPI_Isend	MPI_Irecv
THREAD 1.131.1	88,16 %	9,09 %	0,00 %	0,02 %
THREAD 1.132.1	88,18 %	9,09 %	0,00 %	0,02 %
THREAD 1.133.1	88,18 %	9,09 %	0,00 %	0,02 %
Total	9.309,74 %	306,53 %	1.411,18 %	3,83 %
Average	69,00 %	2,30 %	10,69 %	0,03 %
Maximum	88,18 %	67,62 %	54,97 %	0,04 %
Minimum	30,67 %	0,00 %	0,00 %	0,02 %
StDev	15,27 %	6,06 %	21,40 %	0,00 %
Avg/Max	0,79	0,03	0,19	0,81

η

CommEff

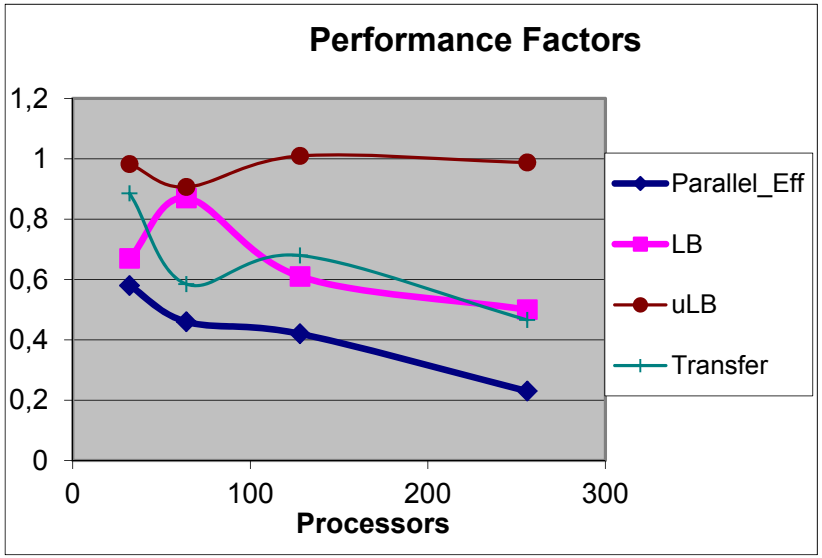
LB

direct computation and presentation of model values to appear in future distributions of Paraver

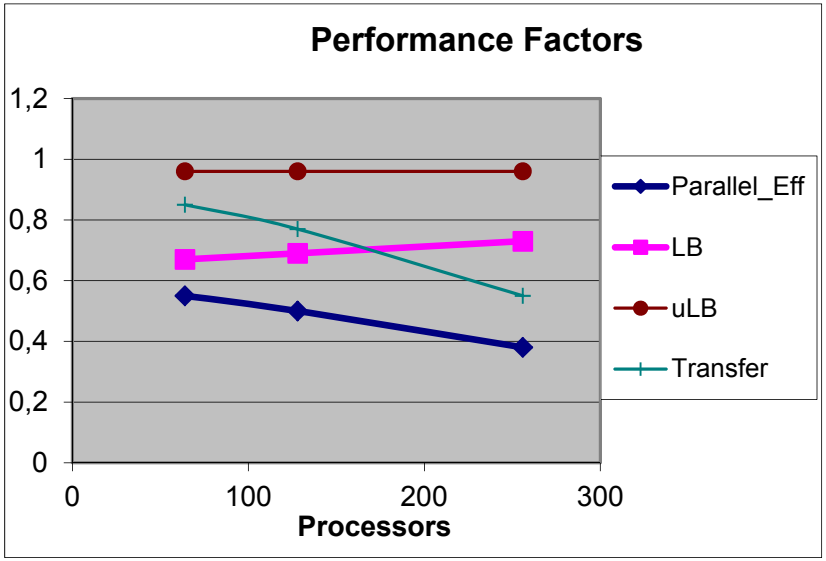
Parallel Performance Models

$$\eta = LB * \mu LB * Transfer$$

Original Version



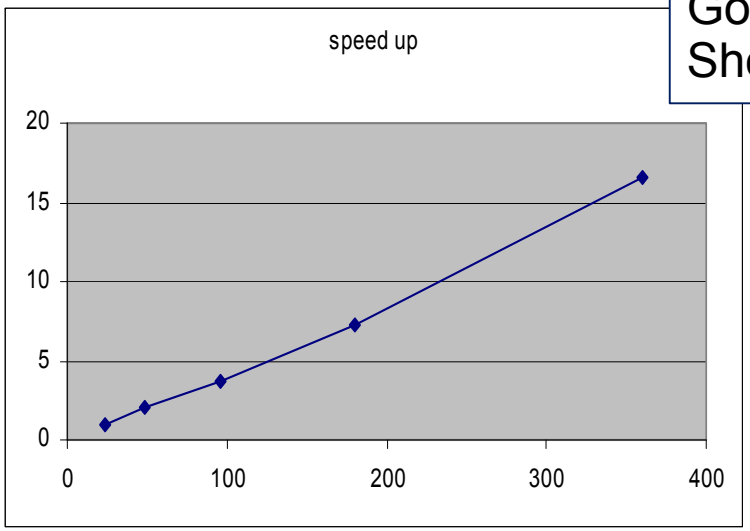
v4.5



GROMACS

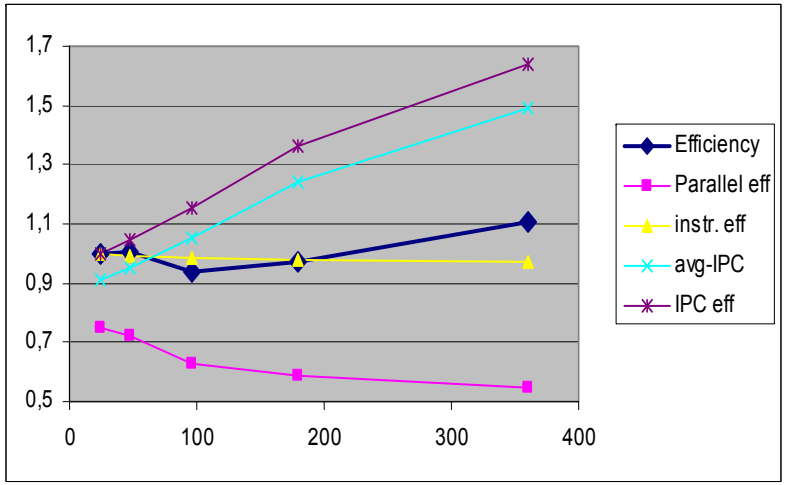
« Old → new version: -43% instr, -52% time

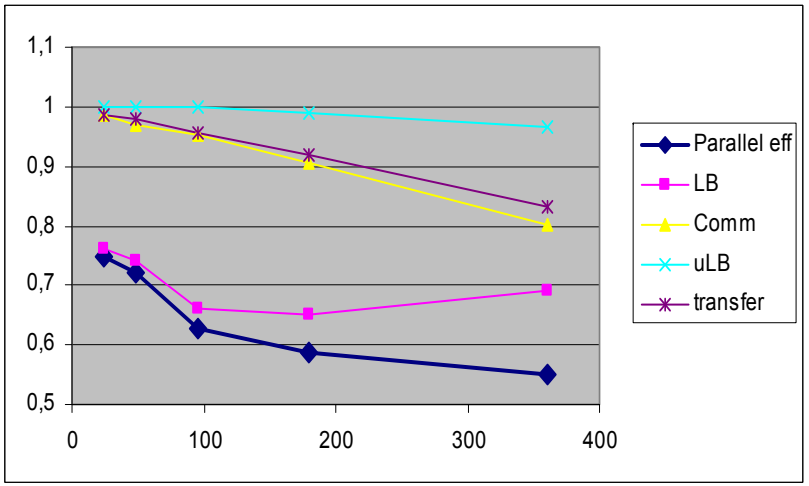
Modelling efficiency: CG-POP mpi2s1D - 180x120



Good scalability !!
Should we be happy?

- Performance Factors/components:
  - Abstracting essential app. characteristics
  - combine/couple → Actual performance
    - May have opposed effects





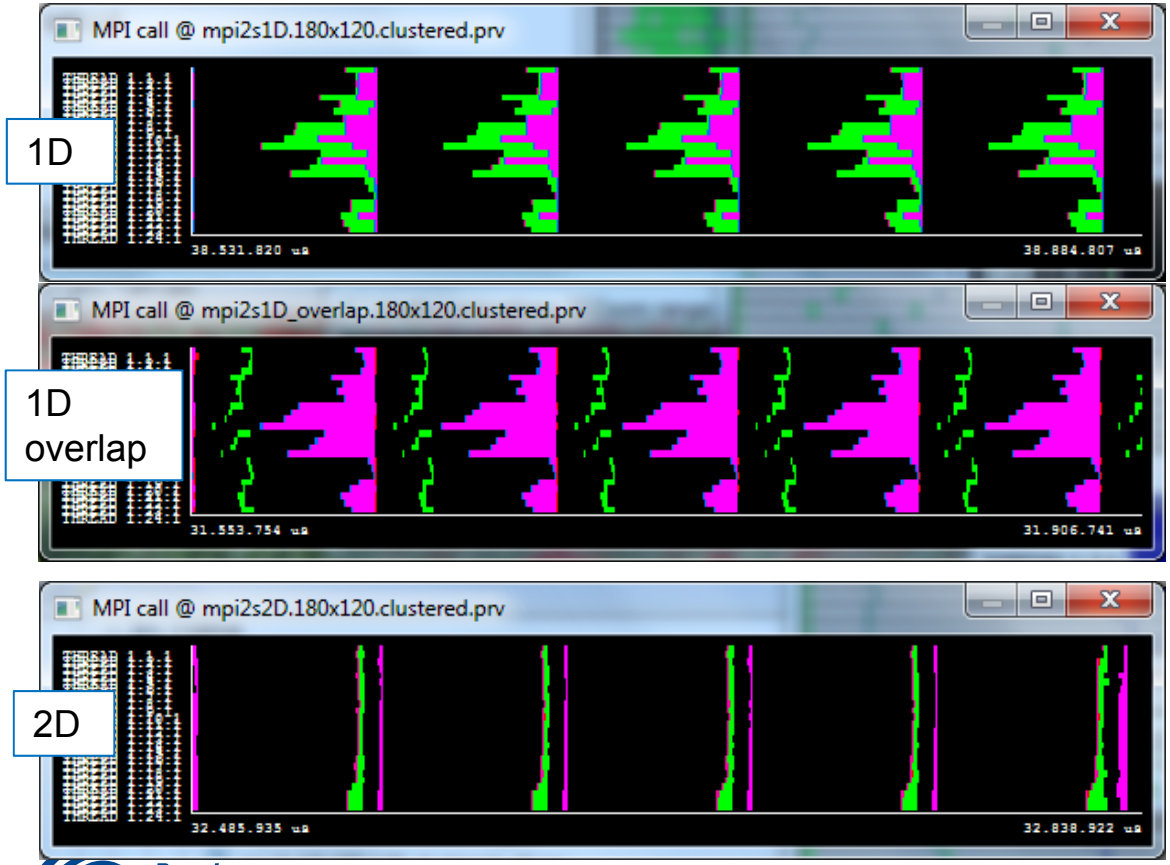
Examples

CGPOP

Comparison between versions

- 24 processes, 180x120

Version	η	Time LB	Comm
1D	0.76	0.77	0.99
1D-ovlp	0.75	0.76	0.99
2D	0.93	0.98	0.95

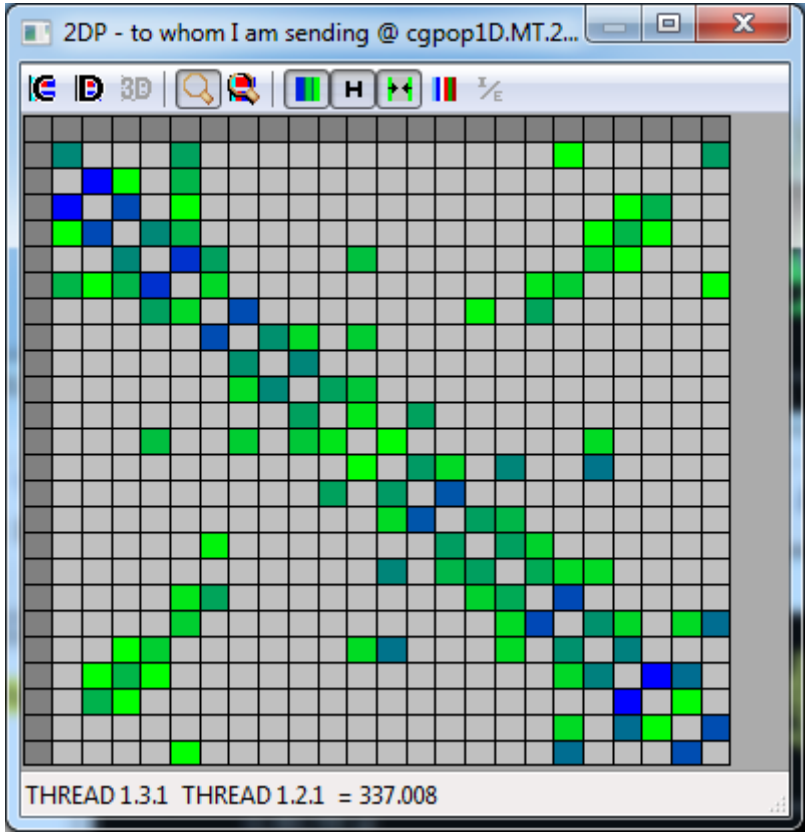


Version	IPC	Comp LB
1D	0.32	0.75
1D-ovlp	0.34	0.75
2D	0.17	0.99

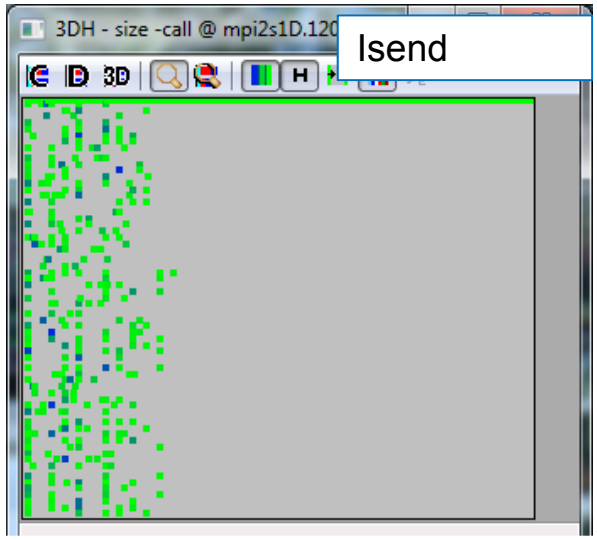
- Load Imbalance
  - Significant improvement in 2D …
- Poor IPC
  - …compensating Load balance improvements in 2D ☹

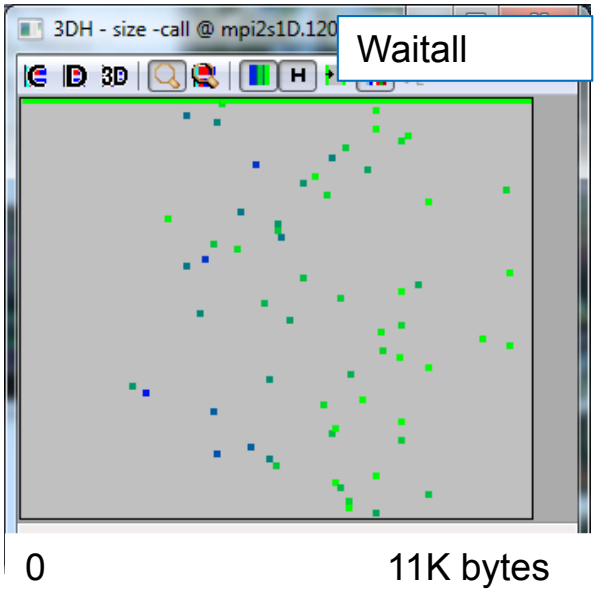
CGPOP: MPI connectivity

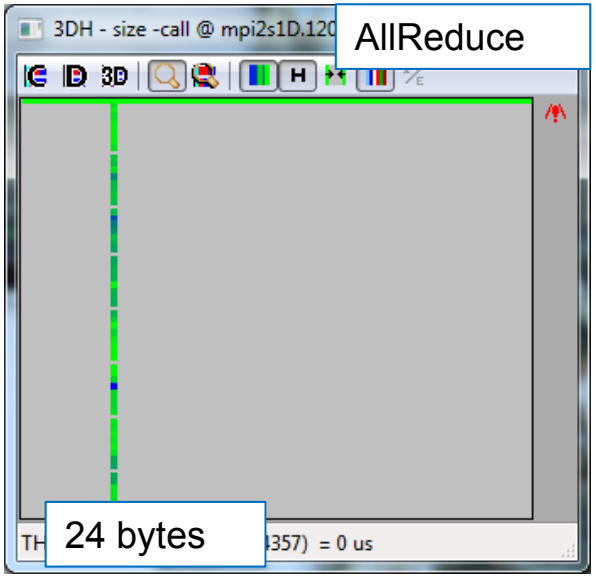
- Who sends to whom
- Leverage full capabilities of Paraver table mechanism
  - Control window: To whom I am sending



CGPOP: Communication analysis





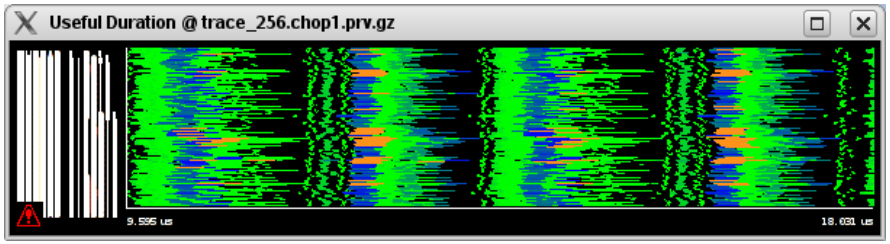


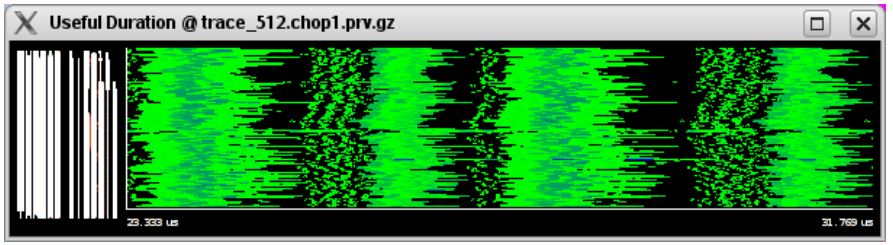
- Significant imbalance in size of p2p communications
  - Will be important at scale !!!!
- Collectives: very small message
  - Introduce asynchrony !!!

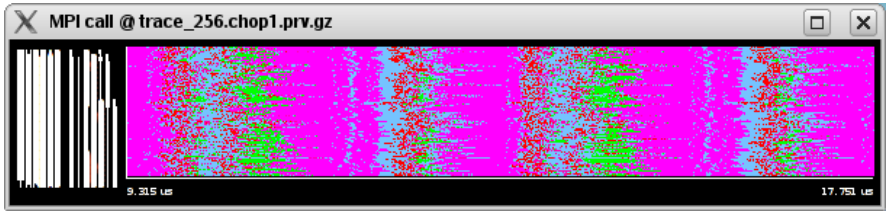
Example 2: Code optimization

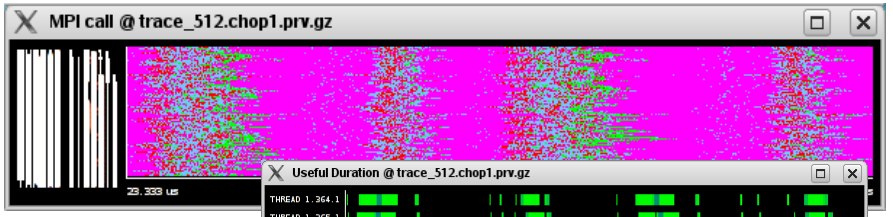
Short diagnosis: FrontFlowRed – 256 vs 512 tasks

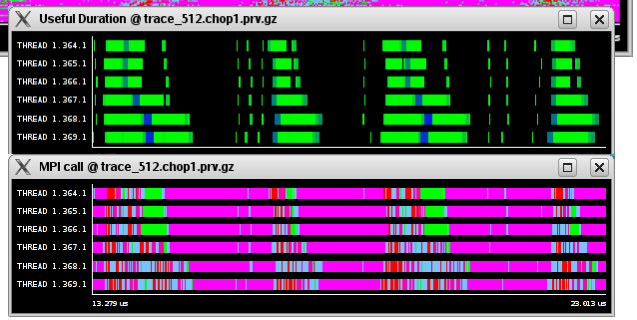
- Unbalance and MPI time increases
- Sequence of MPI_Allreduce calls
- Poor IPC, high cache misses











	256	512
Parallel efficiency	27%	19%
Load balance	56%	48%
% MPI_Allreduce	50%	62%

Example 2: Code optimization

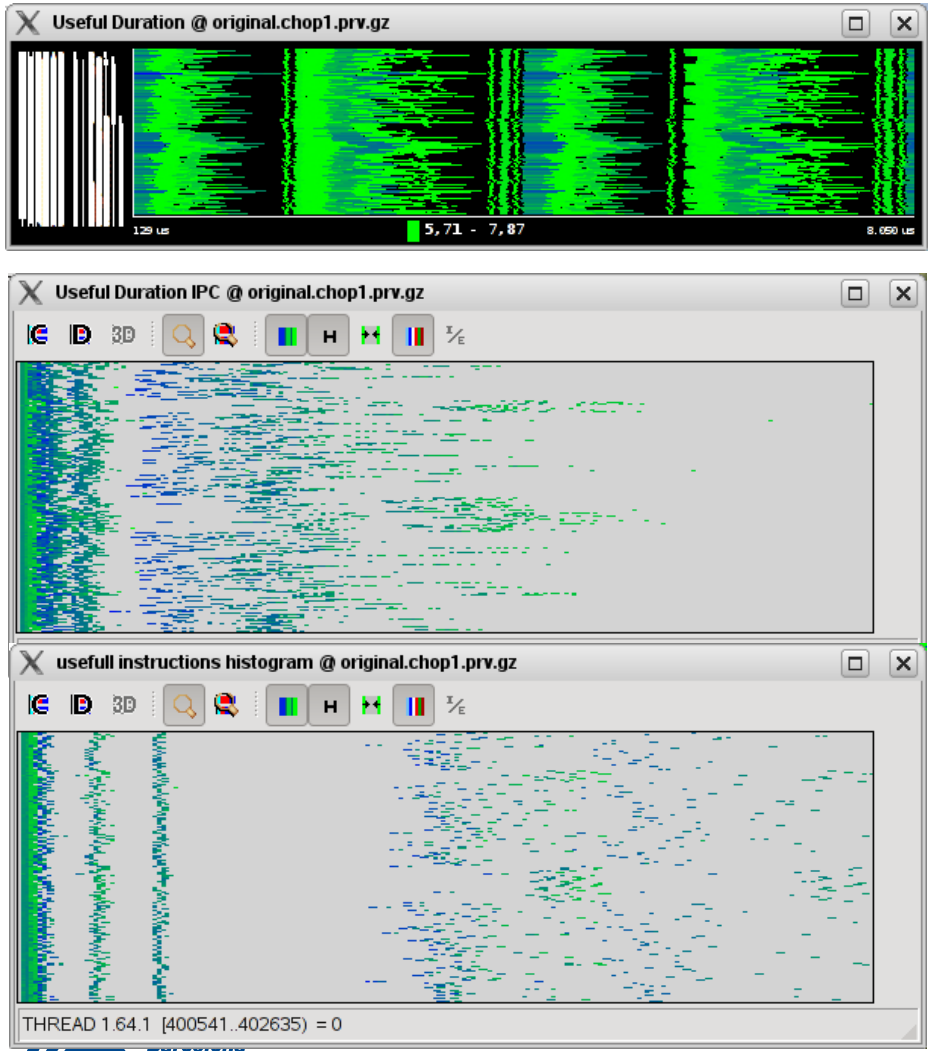
Modifications in the source code

- Improved load balance – weight nodes based on their connectivity, balance total weight
- Reduction of all to all communications – half of MPI_Allreduce calls were eliminated
- Reduce cache misses – reordering node number

Half an hour meeting for the analysis, few days of work to modify code → 25% of improvement

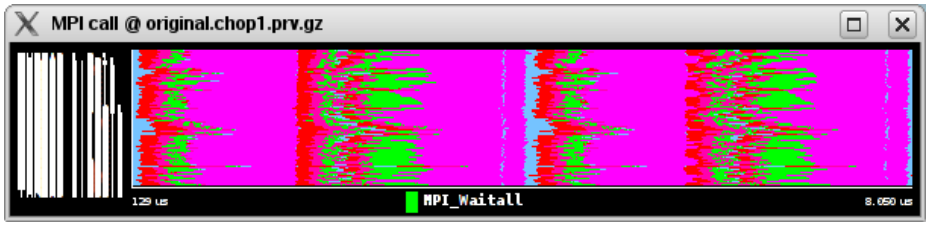
Example 2: Code optimization

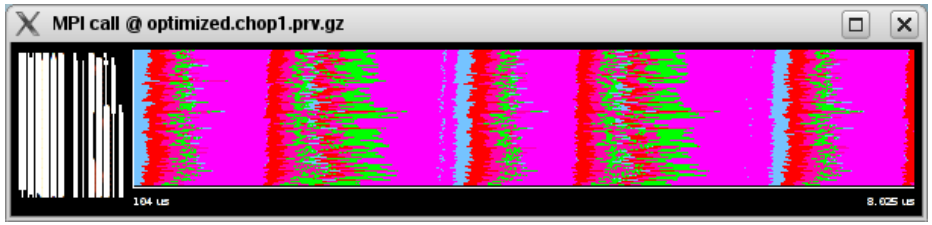
256 tasks – Load balance



Example 2: Code optimization

256 tasks – MPI time



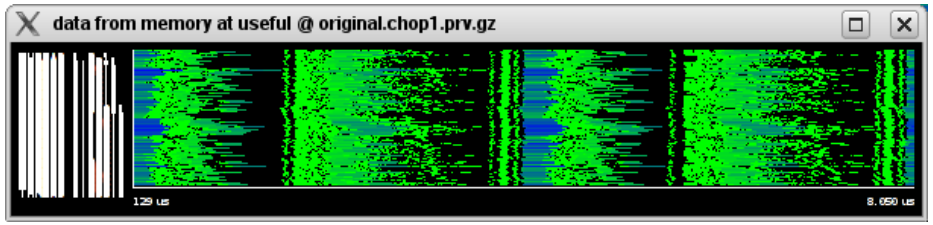


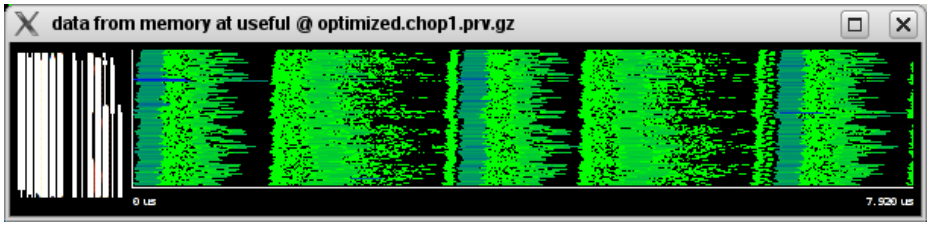
	End	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Allreduce
Total	7.617,81 %	2.432,05 %	1.772,21 %	2.245,19 %	11.532,74 %
Average	29,76 %	9,50 %	6,92 %	8,77 %	45,05 %
Maximum	51,85 %	18,11 %	13,00 %	28,59 %	67,69 %
Minimum	17,59 %	1,65 %	1,37 %	1,46 %	14,88 %
StDev	8,04 %	3,33 %	2,39 %	6,05 %	11,03 %
Avg/Max	0,57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

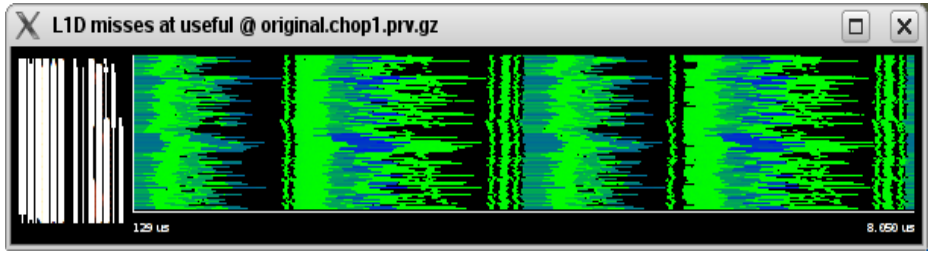
	End	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Allreduce
Total	9.089,05 %	3.100,13 %	2.107,96 %	1.511,02 %	9.791,83 %
Average	35,50 %	12,11 %	8,23 %	5,90 %	38,25 %
Maximum	51,05 %	20,77 %	14,59 %	21,31 %	64,34 %
Minimum	22,99 %	3,47 %	3,21 %	1,72 %	10,90 %
StDev	5,92 %	3,62 %	2,61 %	3,89 %	10,13 %
Avg/Max	0,70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

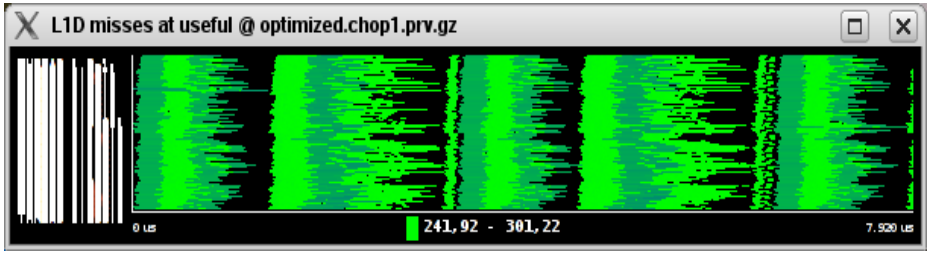
Example 2: Code optimization

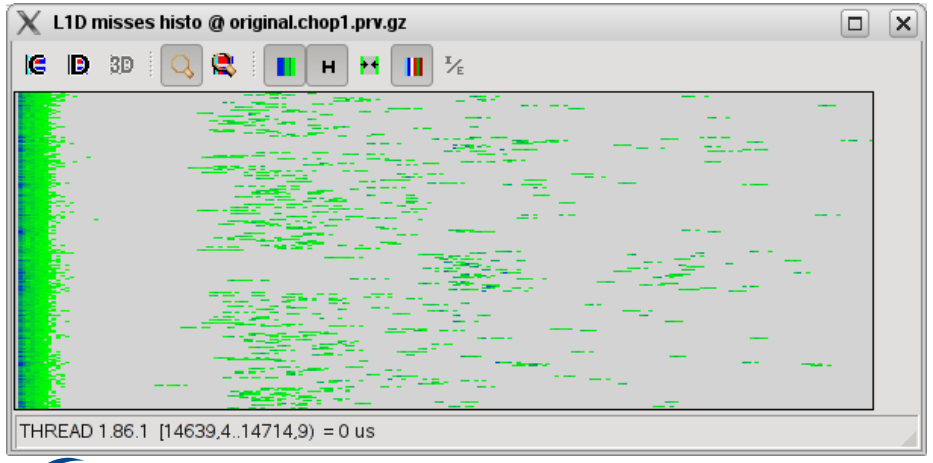
256 tasks – Memory access

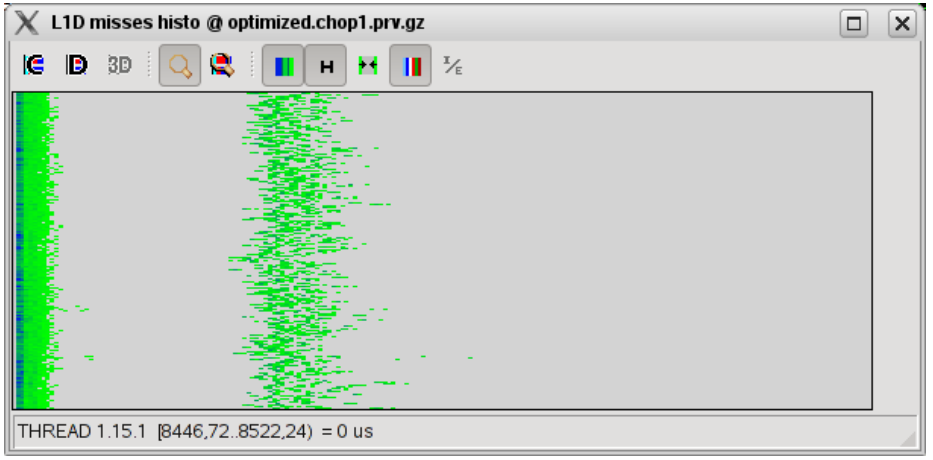












Analysis recommendations

Navigation

- Both for timelines and tables
- First learn to navigate with the tool
  - How to load configurations, zoom, fit coloring scales
  - How to read
  - How to generate timelines form tables
- Second
  - Develop a basic understanding of the process of generation of the timelines and histograms.
- Third: attitude
  - Top down: Do not get obsessed about details!!!! Look at the forest, not the trees !!!

Analysis

- Loop:
  - Question/hypothesis
  - What metric I need to quantify/validate/reject ?
  - What view (timeline or table) would provide the information to answer the question?
  - Do I have a configuration file to show such view?
    - If not, how do I generate it?

- Considerations:
  - May be useful to identify a proper region where to apply the analysis
    - Avoid initialization, perturbed regions, focus on a couple of iterations,….
  - The more iterations you can make out of a single trace the more understanding you will get !!!

Distribution of cfg directories

- CFG

$PARAVER_HOME/cfgs

  - General
    - including basic views (timelines) and analyses (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

Analysis of MPI programs

- Typical initial questions:
  - What is the global efficiency, Load balance, and communication efficiency?
  - If efficiency is good, still need to look at sequential performance.
    - Is IPC good? Everywhere?
    - If bad, is if due to cache misses? Other?
  - If imbalanced
    - is it due to computation or IPC imbalance?
  - If communication efficiency low
    - are message sizes large?
    - Too many messages?
    - Is the effective bandwidth achieved reasonable? For all transfers?
    - Is there serialization of the computations?

Paraver Summary

Summary

- Performance tools are needed more and more !!!!!
  - To tune our applications, to design our system software.
  - To understand what really happens, how our systems really behave,…
- Paraver, …
  - Extremely flexible an powerful.
    - “professional” equipment
  - Attitude: need curiosity to understand, confidence on measurement capability of the tool.
  - A Personal Challenge: squeeze information from the data

I have seen things you people wouldn't believe...

Roy Batty – Blade Runner

Seeing is believing ... measuring is better

Free adaptation of a Spanish saying

Tools web site

- www.bsc.es/paraver
  - downloads
    - Sources / Binaries
  - documentation
    - Training guides
    - Tutorial slides

- Assignment ☺
  - get from Graphit: /export/hopsa/BSCtools/wxparaver_pkgs
    - Paraver (binaries) for your windows/linux/MAC laptop
  - get tutorials from Graphit: /export/hopsa/BSCtools/tutorials
  - Start wxparaver
  - Help → tutorials and follow instructions
  - Follow tutorials 1 (Introduction to Paraver and MPI) and 2 (Tutorial on HydroC)

www.bsc.es

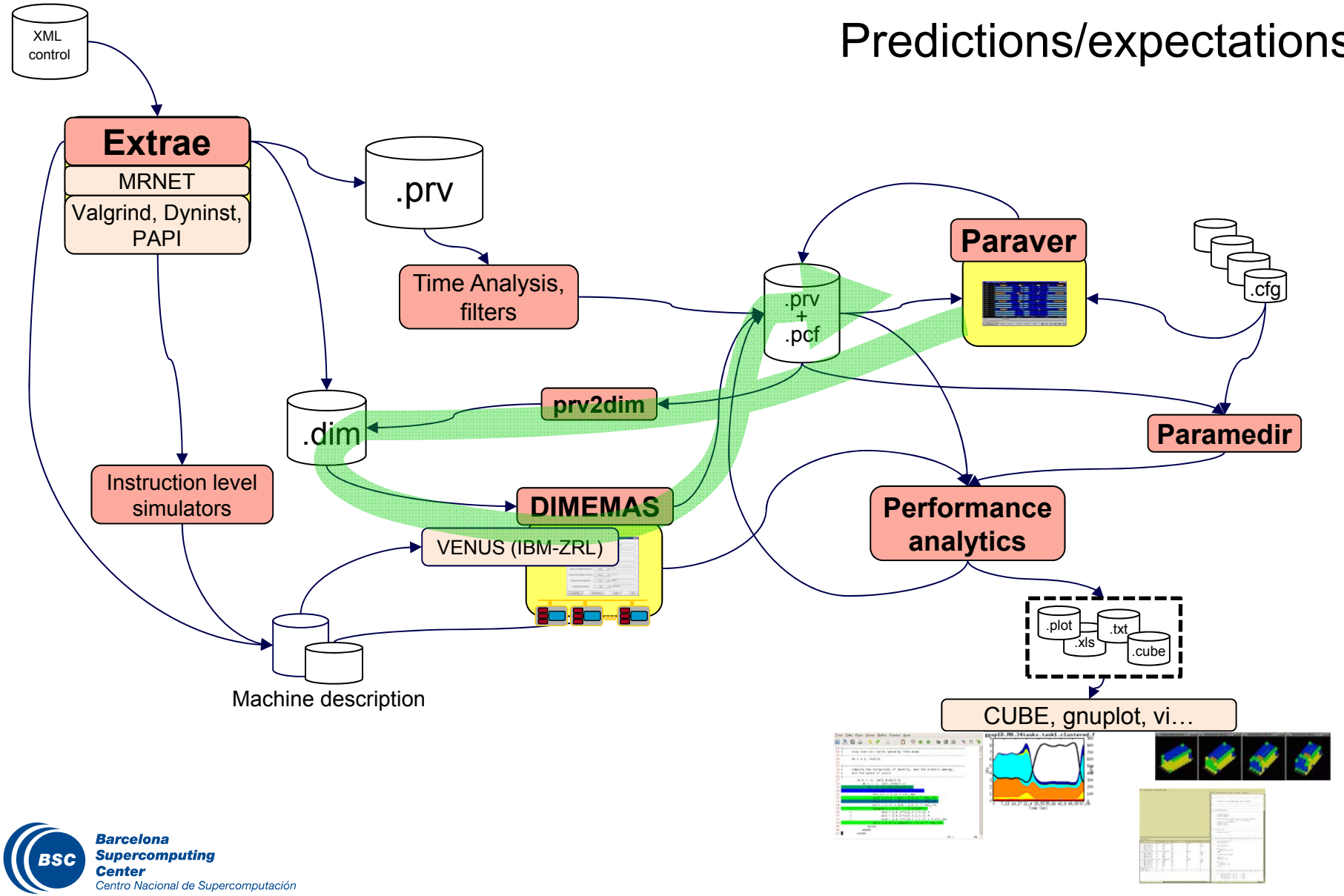
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Dimemas

Jesús Labarta, Judit Gimenez
BSC

Moscow, Nov 27th 2012

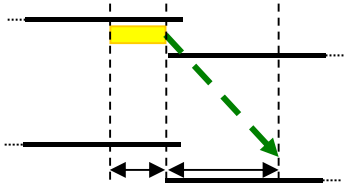
BSC – tools framework

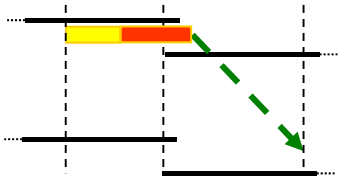


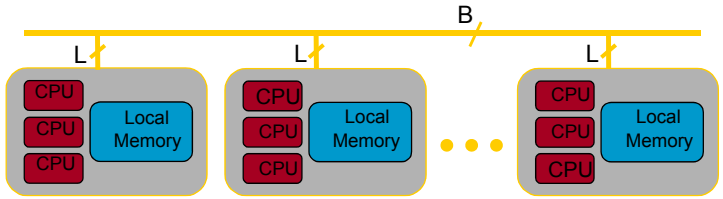
Dimemas: Coarse grain, Trace driven simulation

- Simulation: Highly non linear model
  - Linear components
    - Point to point communication
    - Sequential processor performance
      - Global CPU speed
      - Per block/subroutine
  - Non linear components
    - Synchronization semantics
      - Blocking receives
      - Rendezvous
    - Resource contention
      - CPU
      - Communication subsystem
        - » links (half/full duplex), busses

$$T = \frac{MessageSize}{BW} + L$$

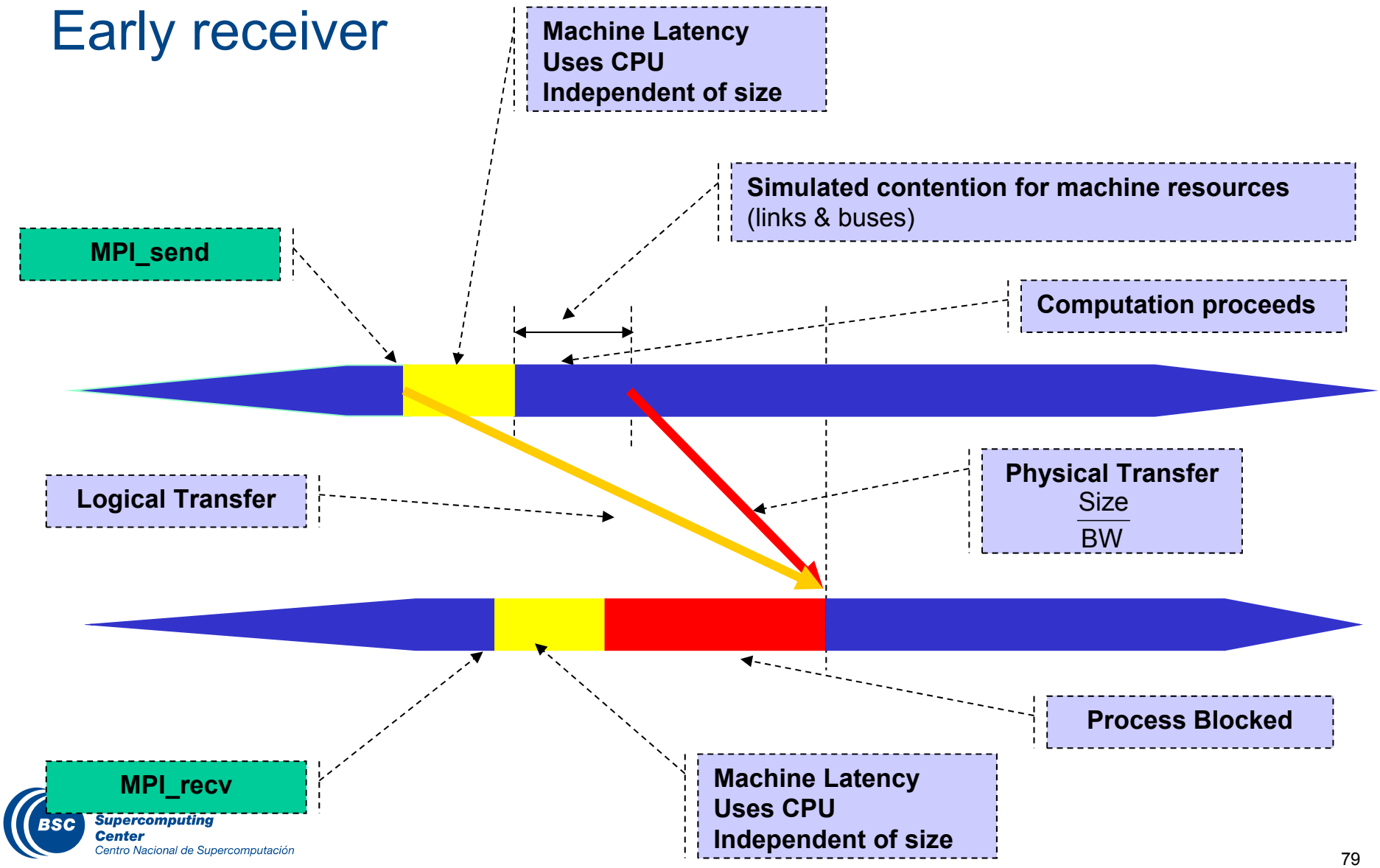






P2P communication model

Early receiver

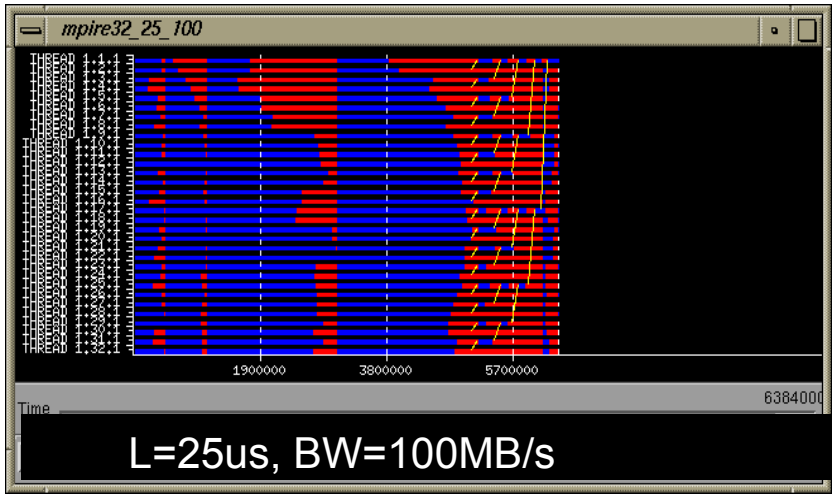


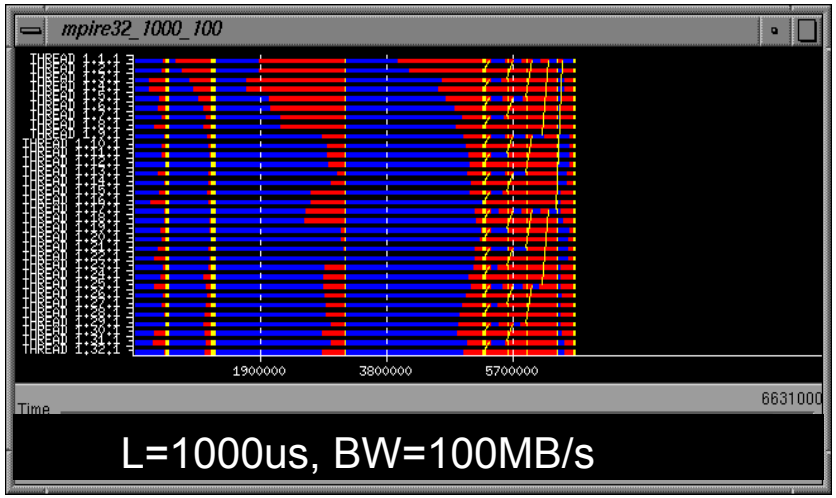
Dimemas

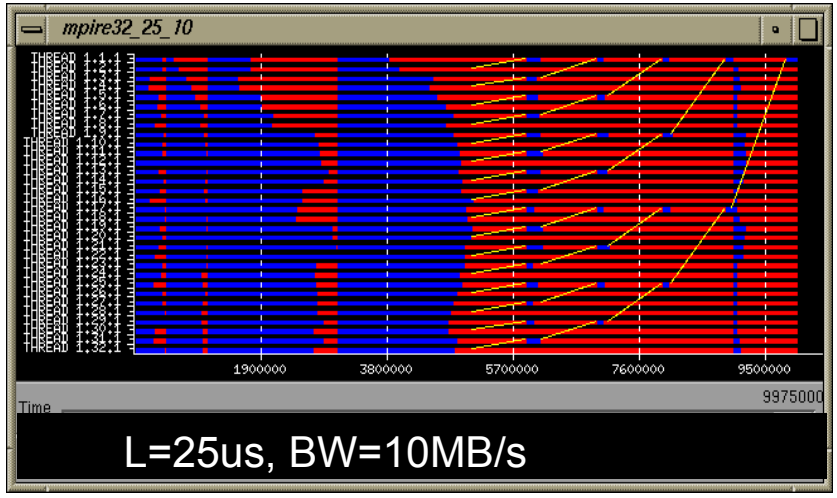
- A model of a hypothetical machine
  - Reference
  - Identification of importance of different factors
- Use examples:
  - Are all parts of an app. equally sensitive to network?
  - Ideal machine
  - Architecture parametric sweeps
  - Estimating impact of ports to MPI+OpenMP/CUDA/…
  - Endpoint contention
  - Core architecture vs. system architecture trade offs

Are all parts of an app. equally sensitive to network?

MPIRE 32 tasks, no network contention



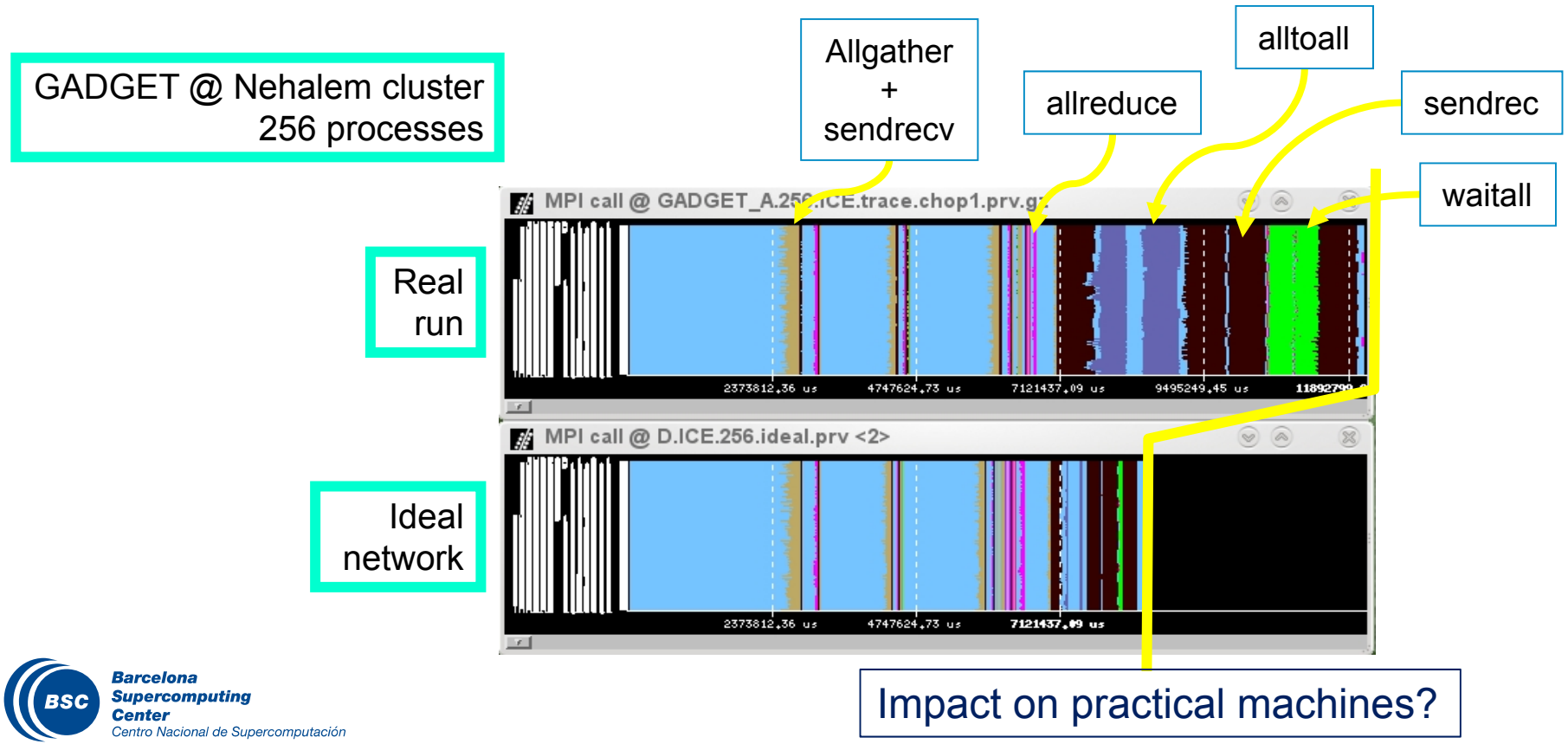




All windows same scale

Ideal machine

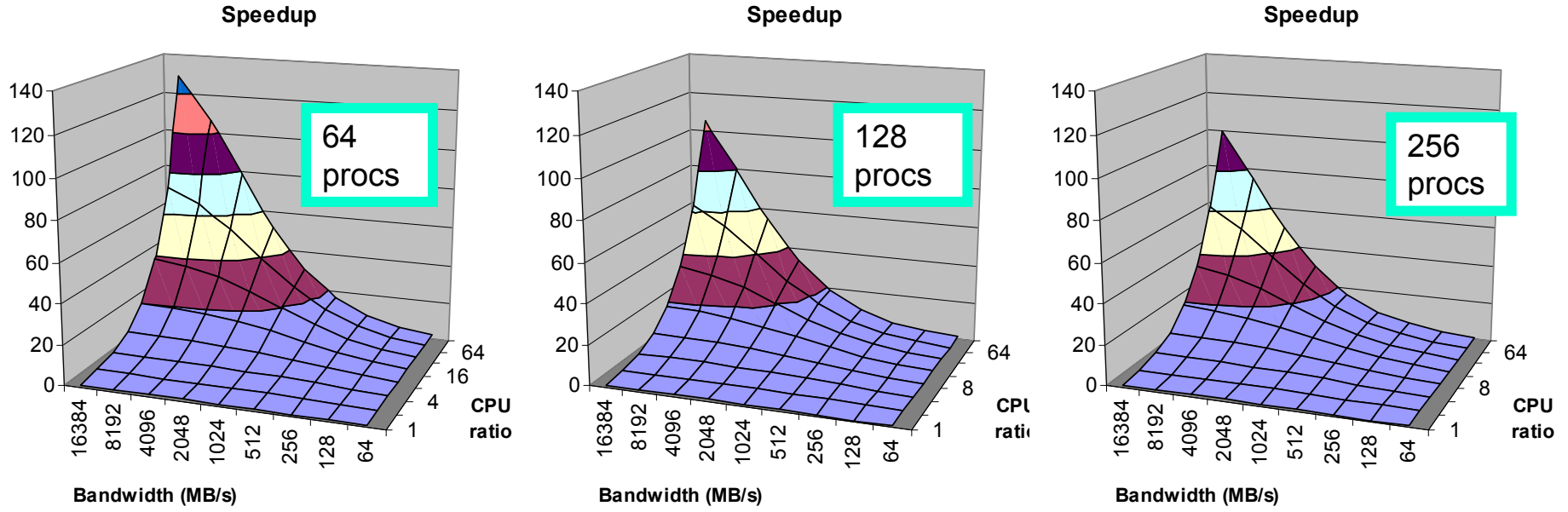
- The impossible machine: BW = ∞, L = 0
- Actually describes/characterizes Intrinsic application behavior
  - Load balance problems?
  - Dependence problems?



Impact of architectural parameters

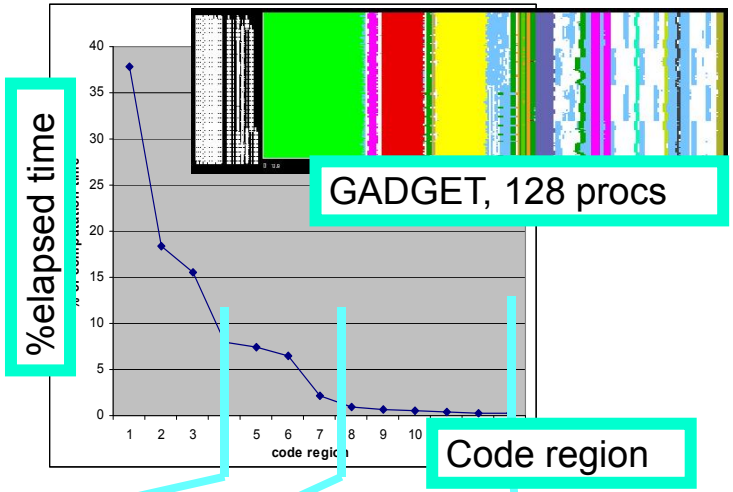
- **Ideal speeding up ALL** the computation bursts by the CPUratio factor
  - The more processes the less speedup (higher impact of bandwidth limitations) !!

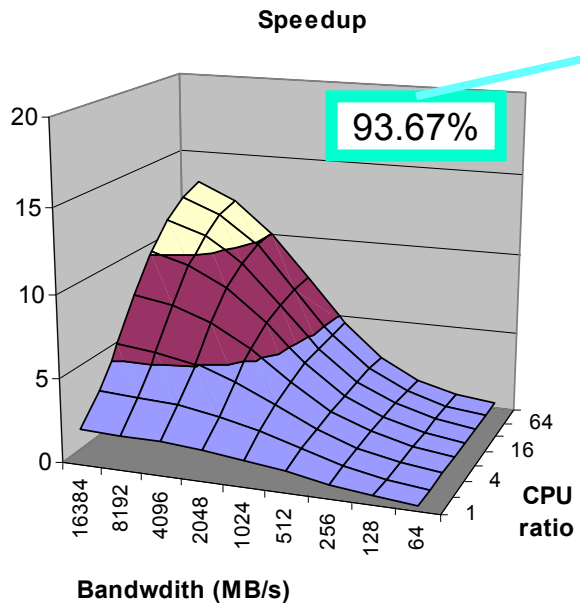
GADGET

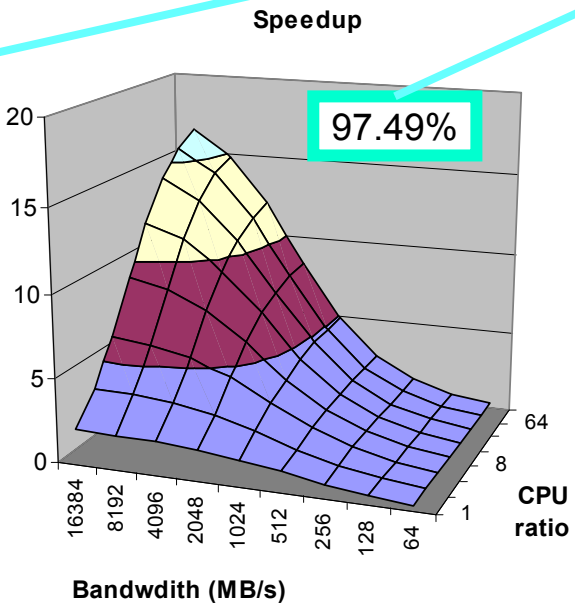


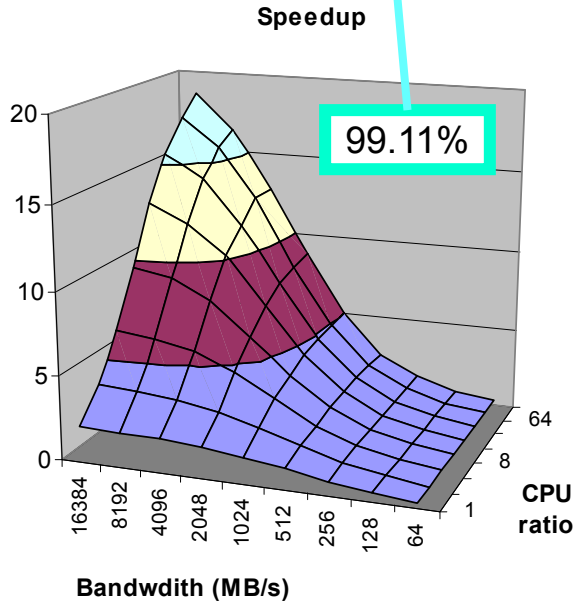
The potential of hybrid/accelerator parallelization

- **Hybrid parallelization**
  - Speedup SELECTED regions by the CPUratio factor
- We do need to overcome the **hybrid Amdahl's law**
  - **→ asynchrony + Load balancing mechanisms !!!**



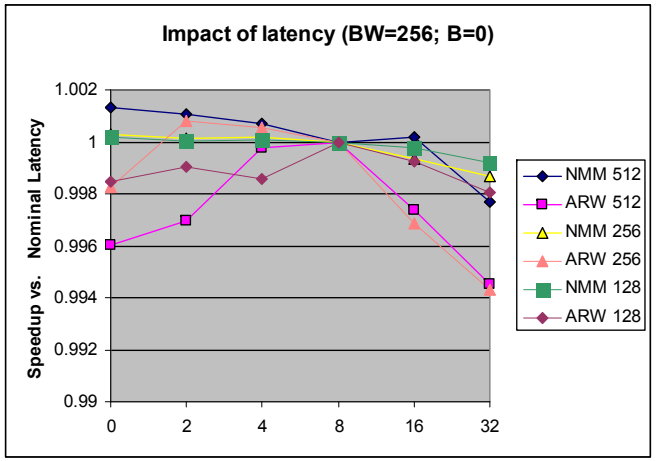


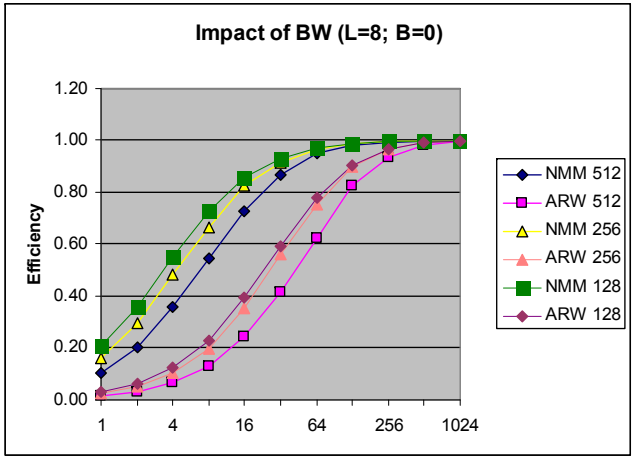


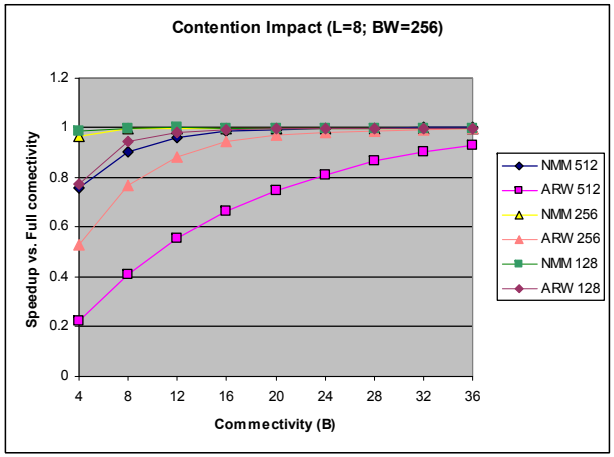


More parametric studies – network sensitivity

- WRF, Iberia 4Km, 4 procs/node
  - No sensitive to latency
  - NMM
    - BW – 256MB/s
    - 512 – sensitive to contention
  - ARW
    - BW - 1GB/s
    - Sensitive to contention



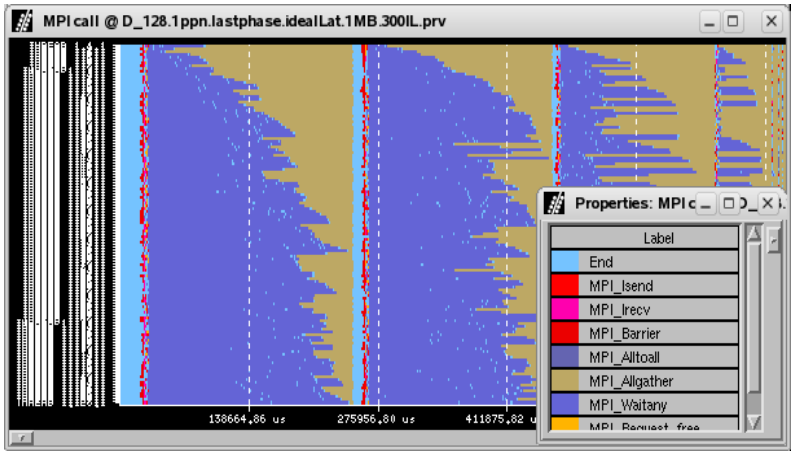


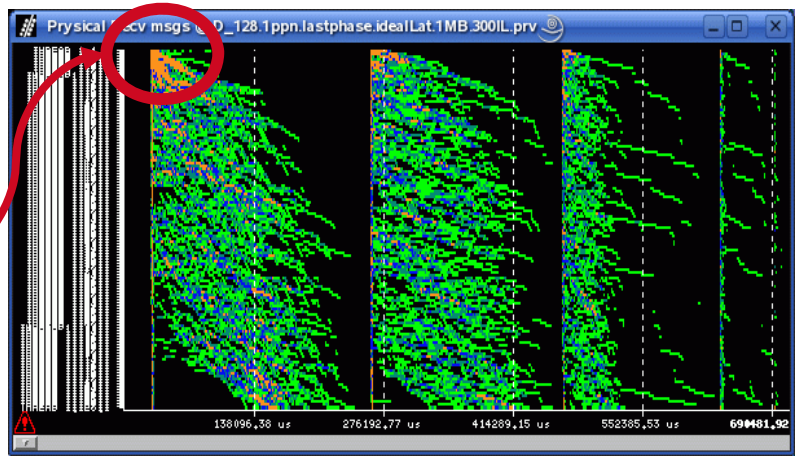


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Intrinsic application behavior

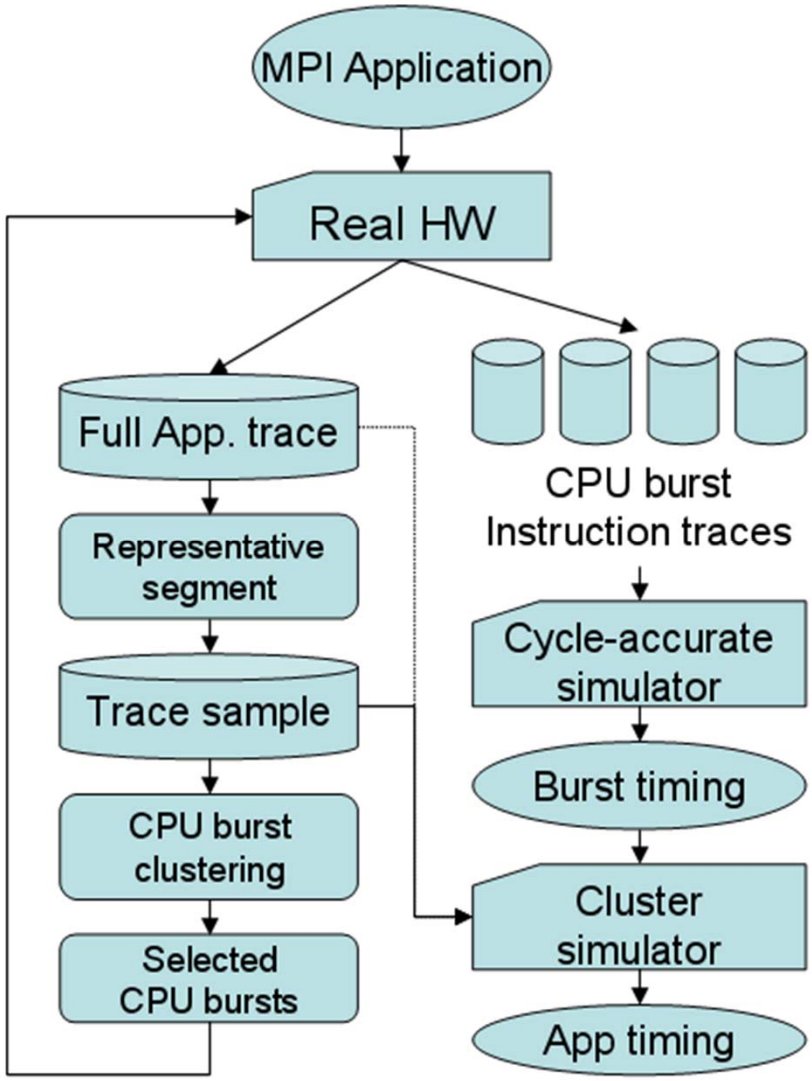
- End point contention
  - Simulation with Dimemas
    - Very low BW
    - 1 output link, ∞ input links
- Recommendation:
  - Important to schedule communications.

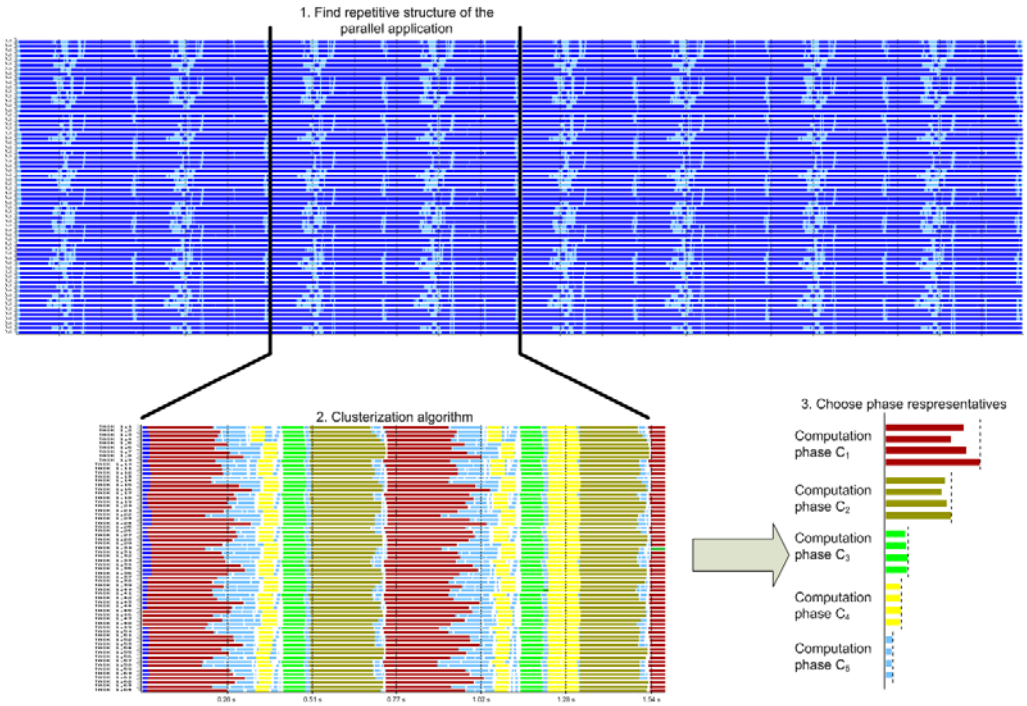




Everybody sending by destination rank order
Endpoint contention at low ranked processes

Multiscale Simulation

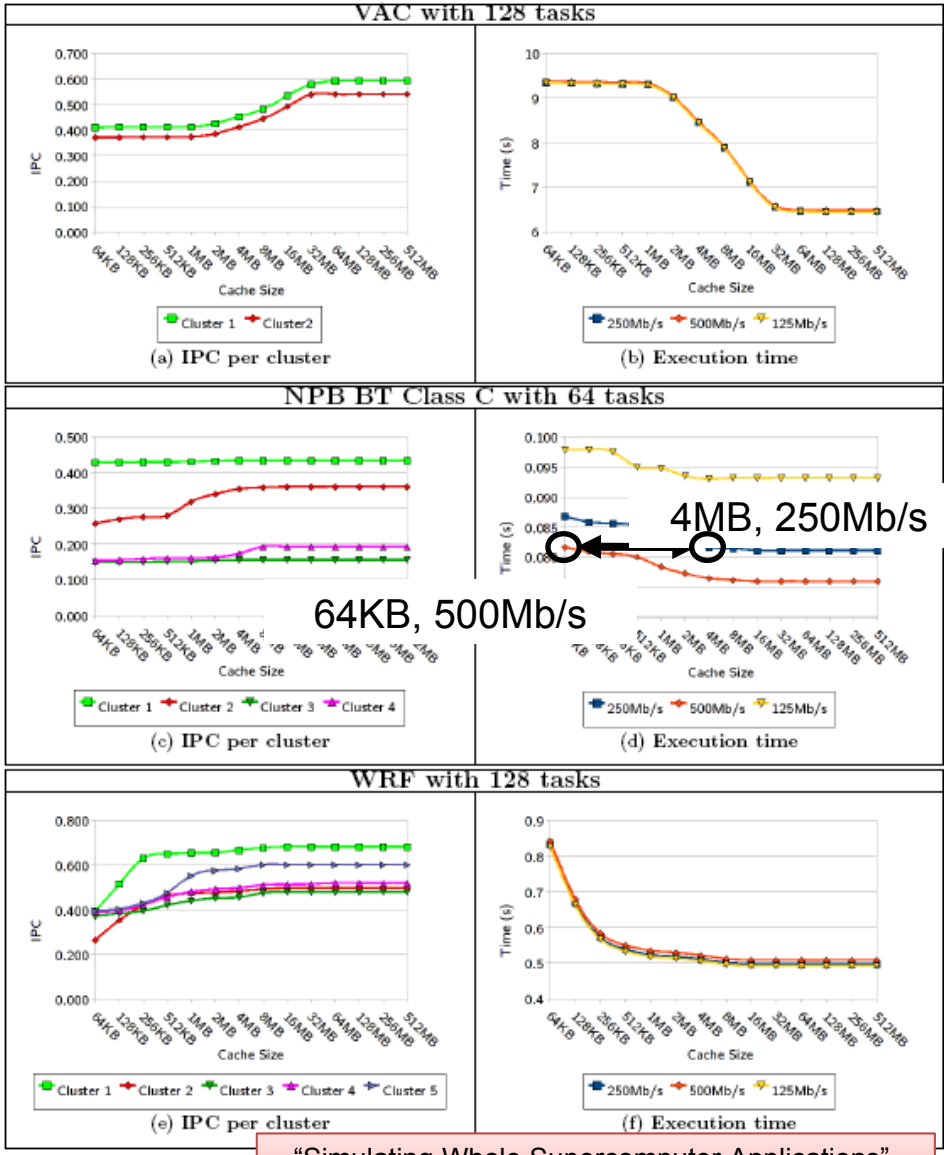




"Simulating Whole Supercomputer Applications". González, J, et all. IEEE MICRO May 2011

Multiscale simulation: L2 size vs network bw

- Left: clusters IPC with different cache sizes
  - 64KB - 512MB
- Right: execution time with different network bw and cache size
  - 125Mb/s - 500Mb/s
- NAS BT
  - Can compensate cache reduction with more network bw
- VAC, WRF
  - Dominated by comp.



"Simulating Whole Supercomputer Applications". González, J, et all. IEEE MICRO May 2011

Using Dimemas

Dimemas trace generation

Paraver → Dimemas trace Generation

- prv2dim original.prv dimemas.dim
- Default: duration of each computation region taken from .prv computation duration

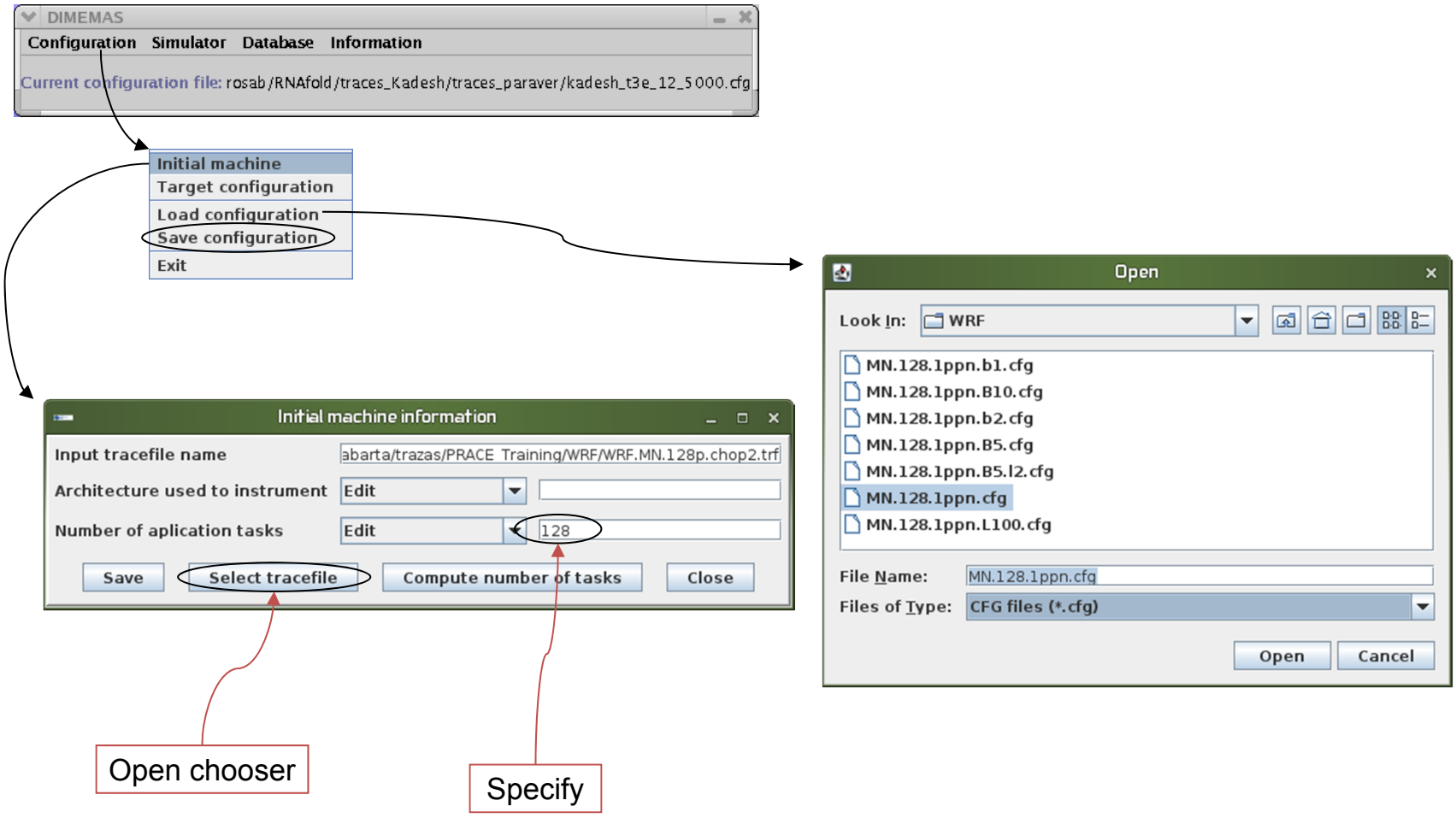
```
Usage:
  prv2dim -i <iprobe_miss_threshold> -b <hw_counter_type>,<factor>
  <paraver_trace> <dimemas_trace>

  -h                                This help
  -n                                No generate initial idle states
  -i <iprobe_miss_threshold>        Maximun MPI_Iprobe misses to discard Iprobe
                                    area CPU burst
  -b <hw_counter_type>,<factor>     Hardware counter type and factor used to
                                    generate burst durations
…
```

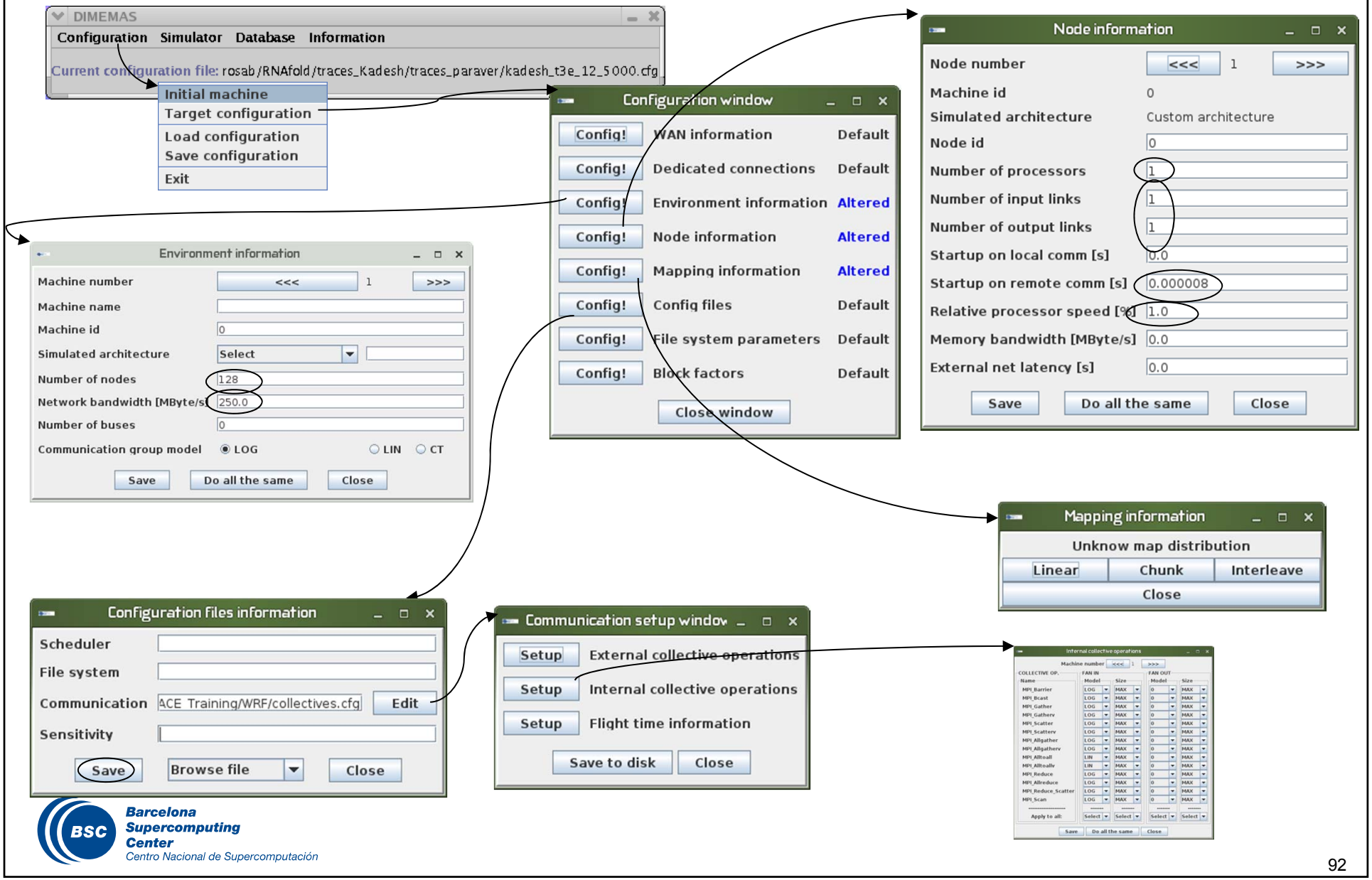
Force synchronized start of all threads

Computation region duration derived from hardware counters assuming/modeling a given performance (<factor>)
Ie. estimate impact if we get rid of IPC imbalances

Dimemas GUI – Specify trace to simulate

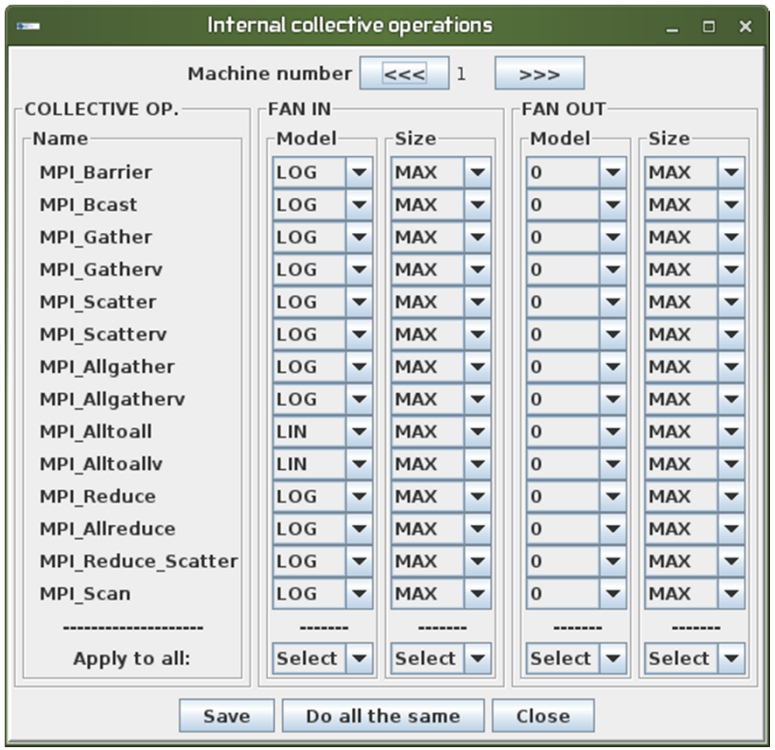


Dimemas GUI – Specify target machine



Collective Communication Model

- Per call model
  - Model factor
    - Lin
    - Log
    - Const
  - Size of message
    - Min over all processes
    - Mean over all processes
    - Max over all processes
  - Specified in input file



Scaling model

Presenting application performance

- Factors modeling parallel efficiency
  - **Load balance** (LB)
  - **Micro load balance** (μLB) or **serialization**
  - **Transfer**

$$\eta = LB * \mu LB * Transfer$$

$CommEff$

- Factors describing serial behavior
  - Computational complexity: **#instr**
  - Performance: **IPC**

$$T_{Comp} = \frac{\#instr}{IPC}$$

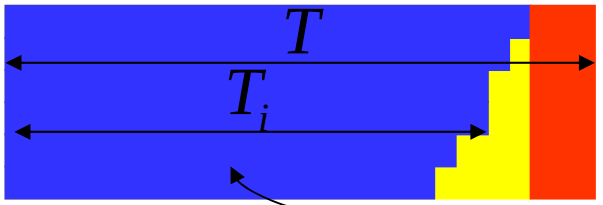
- Scaling model

$$Sup = \frac{P}{P_0} * \frac{\eta}{\eta_0} * \frac{IPC}{IPC_0} * \frac{\#instr_0}{\#instr}$$

M. Casas et al, "Automatic analysis of speedup of MPI applications". ICS 2008.

Scaling model

$$\eta = LB * CommEff$$



$$eff_i = \frac{T_i}{T}$$

$$CommEff = \max(eff_i)$$

$$LB = \frac{\sum_{i=1}^{P} eff_i}{P * \max(eff_i)}$$

IPC

$\#instr$

Directly from real execution metrics

2DP - MPI call profile @ trace_24h_atmos_symbols.cho...

	Outside MPI	MPI_Recv	MPI_Isend	MPI_Irecv
THREAD 1.131.1	88,16 %	9,09 %	0,00 %	0,02 %
THREAD 1.132.1	88,18 %	9,09 %	0,00 %	0,02 %
THREAD 1.133.1	88,18 %	9,09 %	0,00 %	0,02 %
Total	9.309,74 %	306,53 %	1.411,18 %	3,83 %
Average	69,00 %	2,30 %	10,69 %	0,03 %
Maximum	88,18 %	67,62 %	54,97 %	0,04 %
Minimum	30,67 %	0,00 %	0,00 %	0,02 %
StDev	15,27 %	6,06 %	21,40 %	0,00 %
Avg/Max	0,79	0,03	0,19	0,81

η

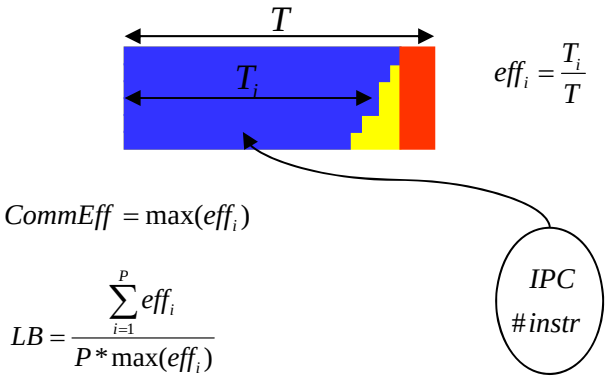
CommEff

LB

direct computation and presentation of model values to appear in future distributions of Paraver

Scaling model

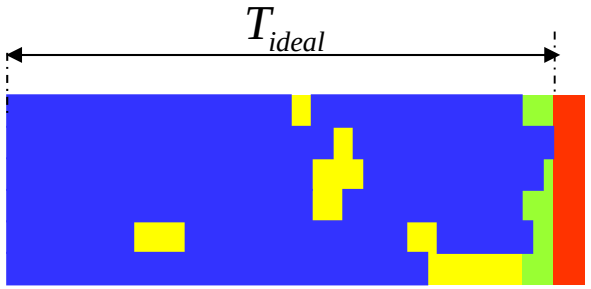
$$Sup = \frac{P}{P_0} * \frac{LB}{LB_0} * \frac{CommEff}{CommEff_0} * \frac{IPC}{IPC_0} * \frac{\#instr_0}{\#instr}$$



Directly from real execution metrics

$$Sup = \frac{P}{P_0} * \frac{macroLB}{macroLB_0} * \frac{microLB}{microLB_0} * \frac{Transfer}{Transfer_0} * \frac{IPC}{IPC_0} * \frac{\#instr_0}{\#instr}$$

$$microLB = \frac{\max(T_i)}{T_{ideal}} \qquad Transfer = \frac{T_{ideal}}{T}$$



Requires Dimemas simulation

Migrating/local load imbalance
Serialization

www.bsc.es

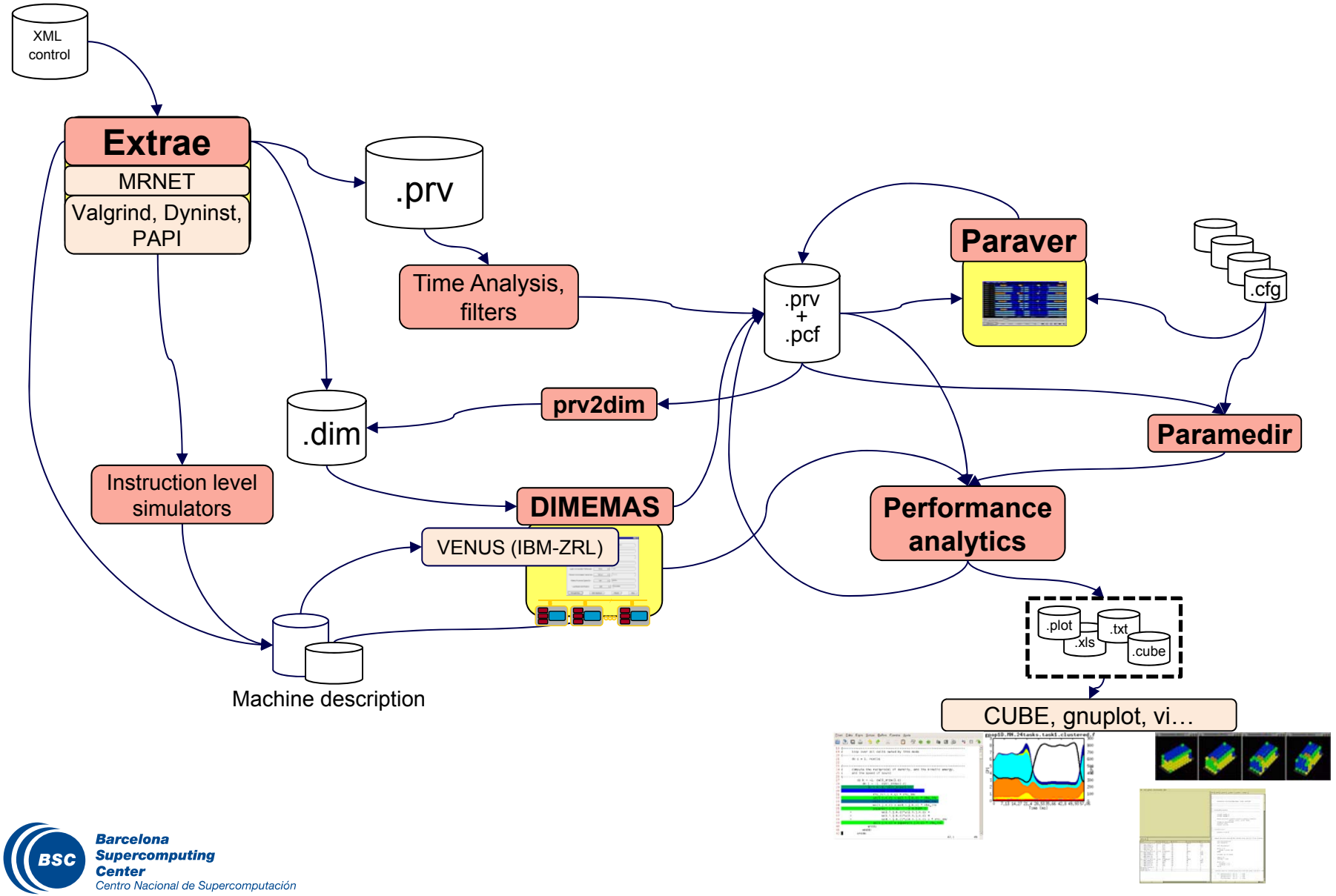
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Performance Analytics

Jesús Labarta, Judit Gimenez
BSC

Moscow, Nov 27th 2012

BSC – tools framework



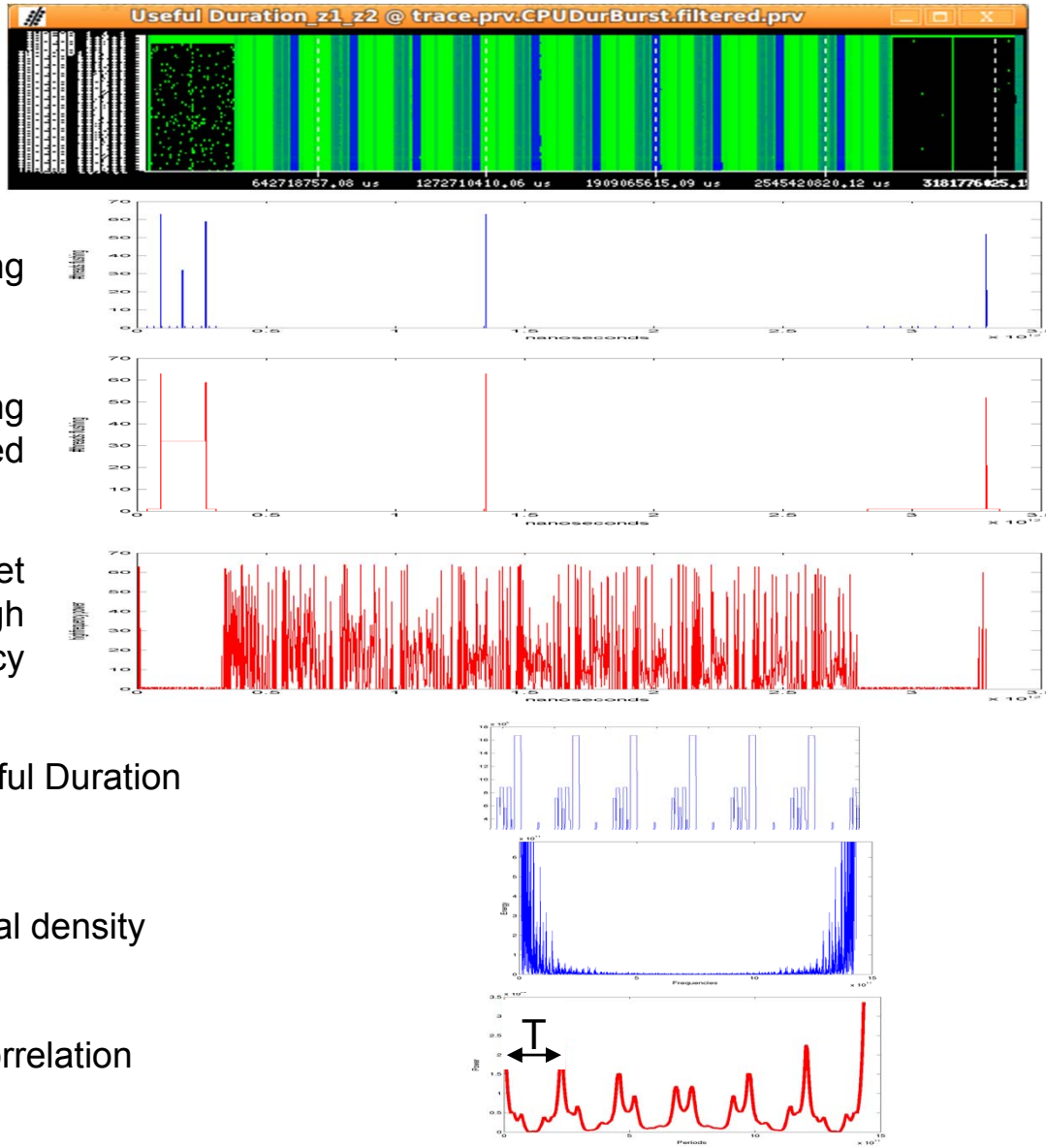
Spectral analysis

Metrics …

- … in Paraver are functions of time
- Natural target for signal processing techniques
- Relevant functions of time at global application level
  - # processes in MPI, outside MPI, …
  - Sum (useful burst duration)
    - Semantic: high when many processes are in the middle of very long computation bursts
    - Does capture repetitive structure of application

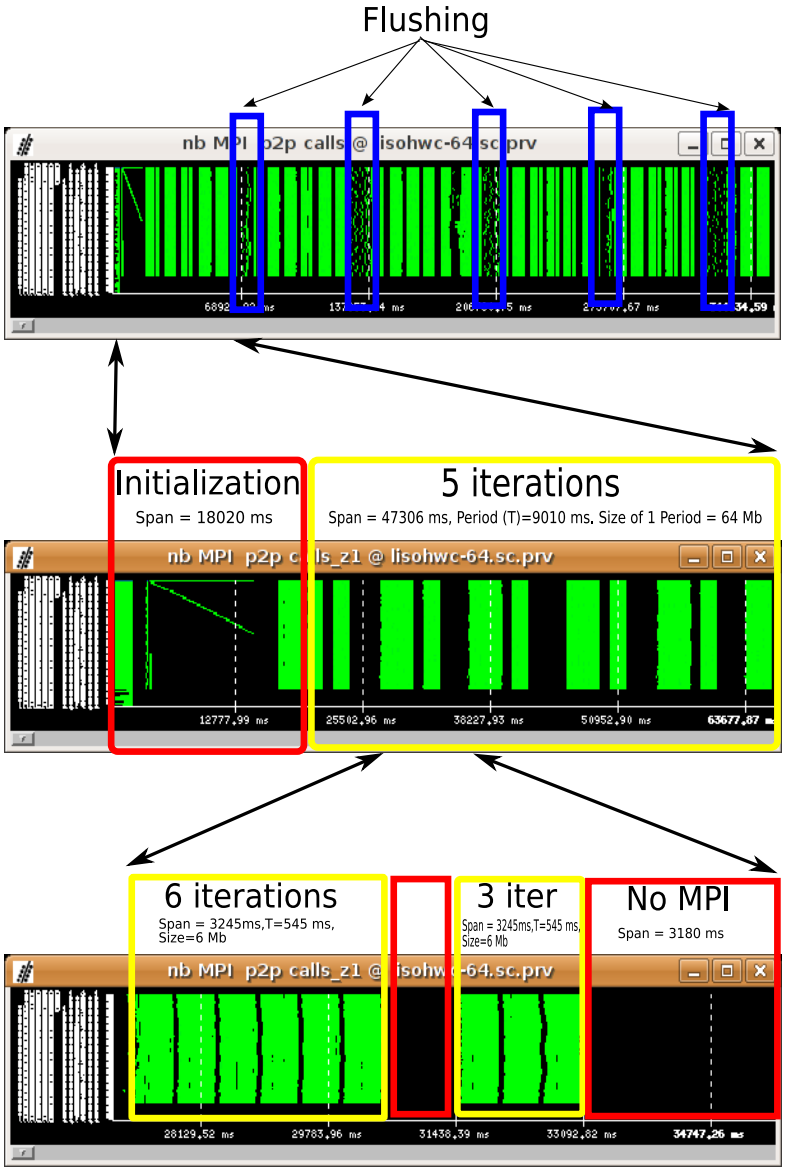
Signal processing applied to performance analysis

- Techniques
  - Mathematical morphology
    - clean up perturbed regions
  - Wavelet transform
    - identify coarse regions
  - Spectral analysis
    - detailed periodic pattern
- Useful
  - Identify structure (periodicity)
  - Reduce trace sizes
  - Increase precision of profiles (report non perturbed stats)

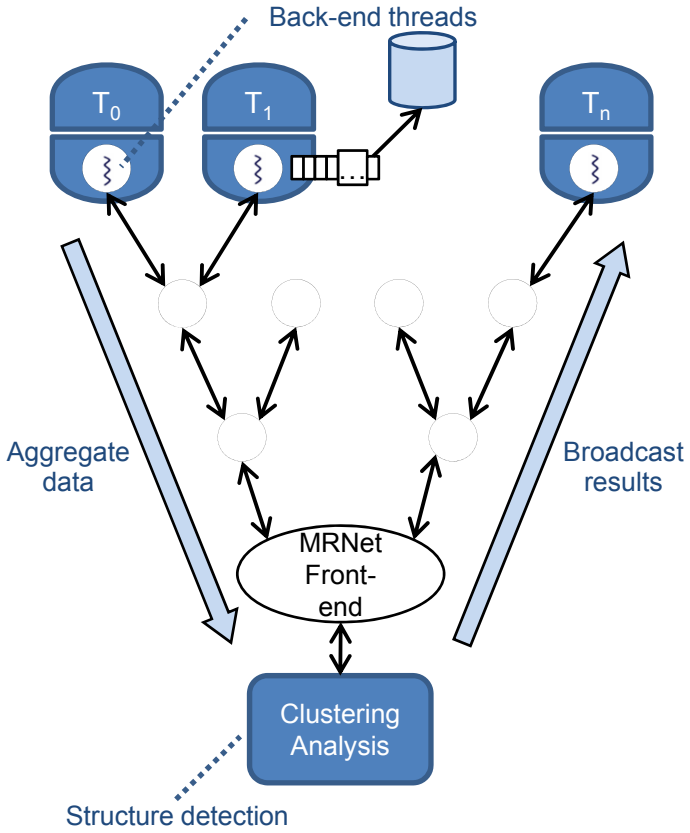


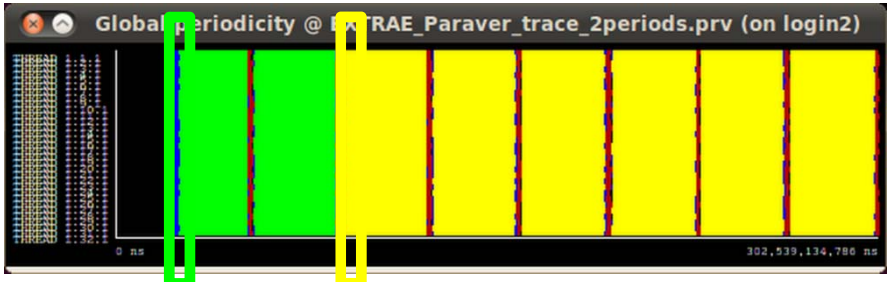
Signal processing applied to performance analysis

- Hierarchical structure identification



Scalability: online automatic interval selection





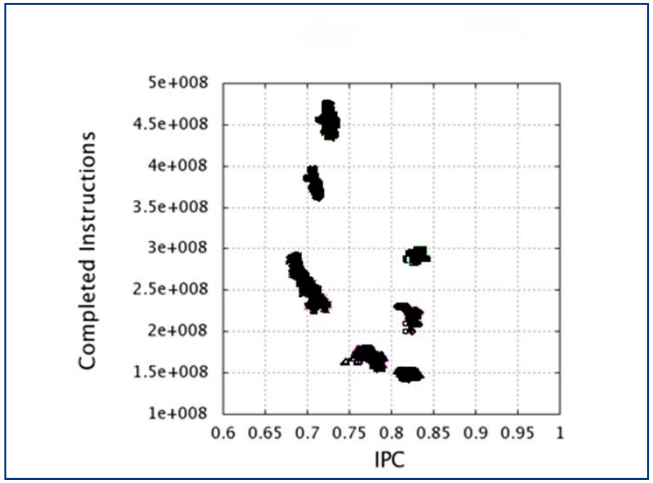
Detailed trace for only small interval

" G. Llort et all, "Scalable tracing with dynamic levels of detail" ICPADS 2011

Clustering

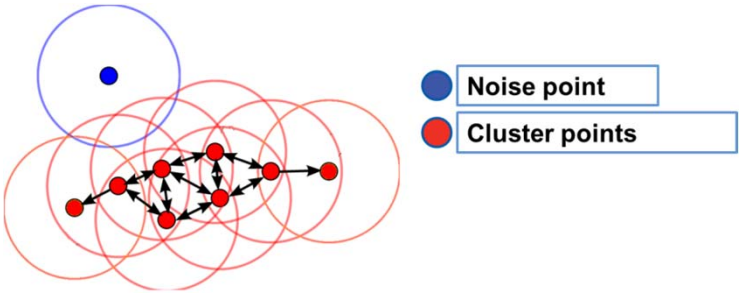
Clustering: analysis of performance @ serial computation bursts

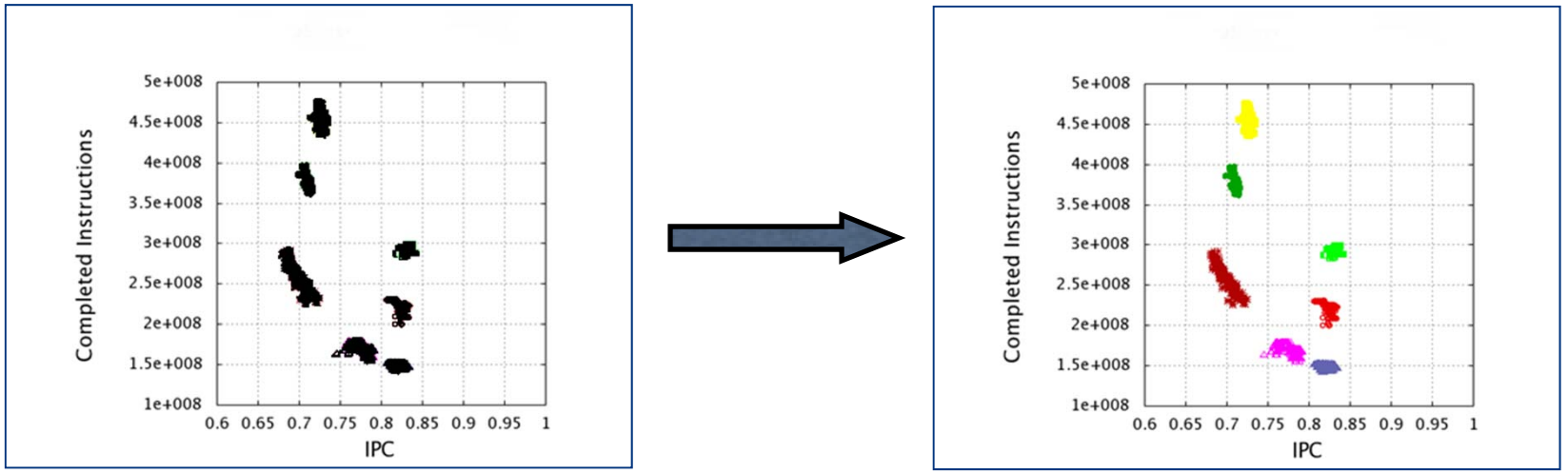
- Identification of computation structure
  - CPU burst = region between consecutive runtime (MPI, OpenMP) calls
    - Described with performance hardware counters
    - Associated with call stack data
- Scatter plot on some relevant metrics
  - Instructions: idea of computational complexity, computational load imbalance,…
  - IPC: Idea of absolute performance and performance imbalance
  - Automatically Identify clusters



Clustering: analysis of performance @ serial computation bursts

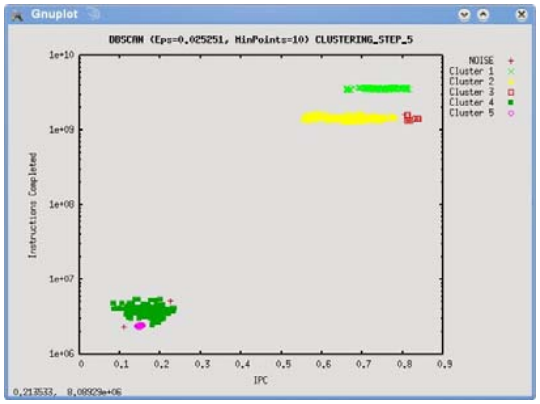
- "Good" clustering algorithms ?
  - Data not necessarily Gaussian → K-means type of algorithms not very good
  - Density based algorithms "better"
    - DBSCAN



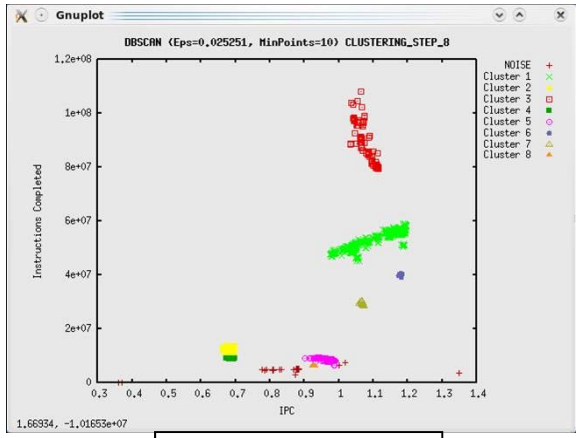


J. Gonzalez et al, "Automatic Detection of Parallel Applications Computation" Phases. (IPDPS 2009)

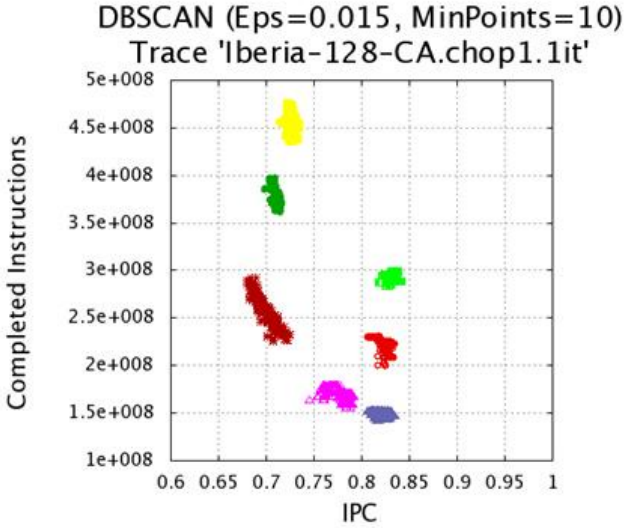
Performance @ serial computation bursts



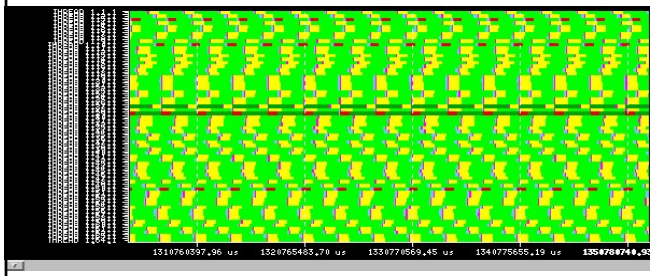
SPECFEM3D

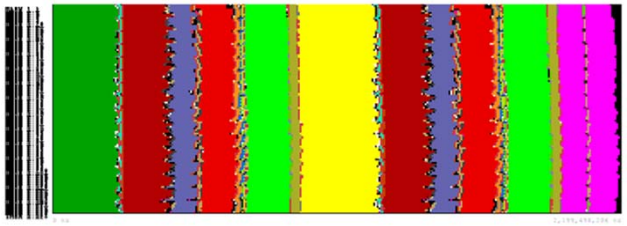


GROMACS



WRF 128 cores

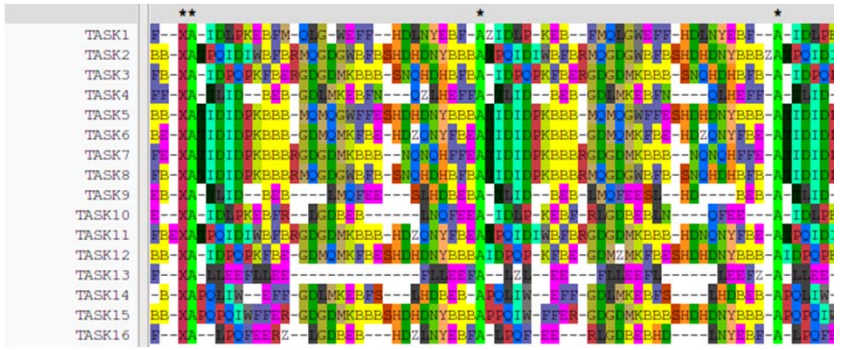




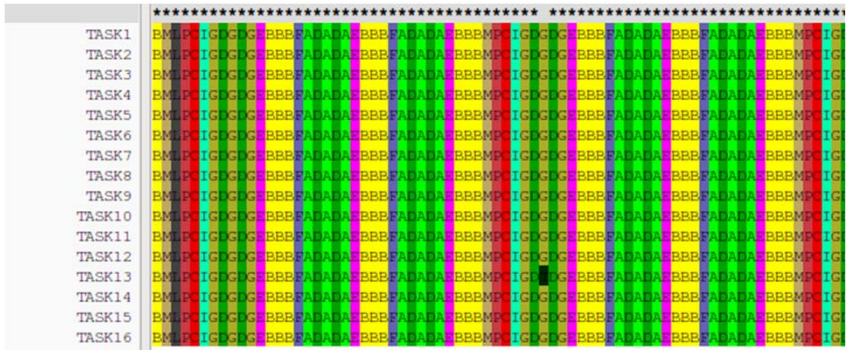
Automatic clustering quality assessment

- Leverage Multiple Sequence alignment tools from Life Sciences

- Process == Sequence of clusters ↔ sequence of amino acids == DNA
- CLUSTAL W, T-Coffee, Kalign2
- Cluster Sequence Score (0..1)
- Per cluster / Global
  - Weighted average

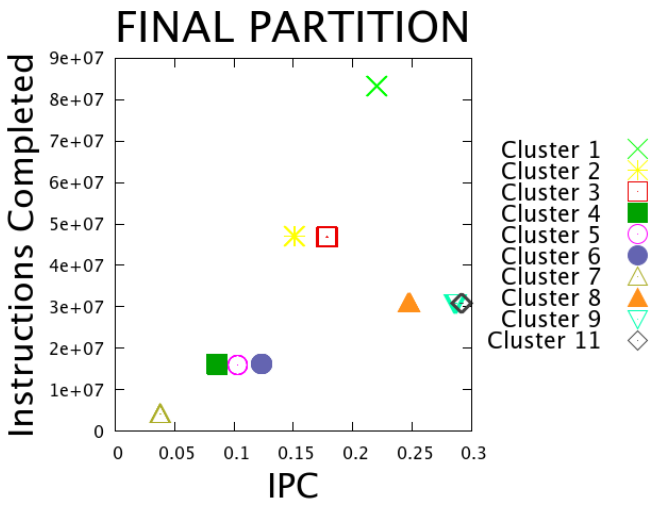


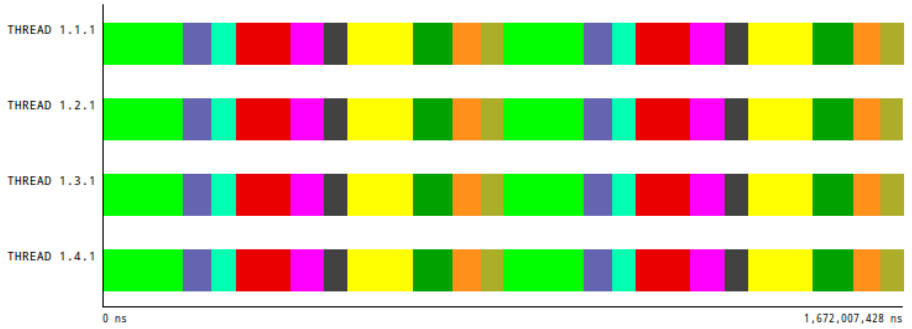
BT.A 0022

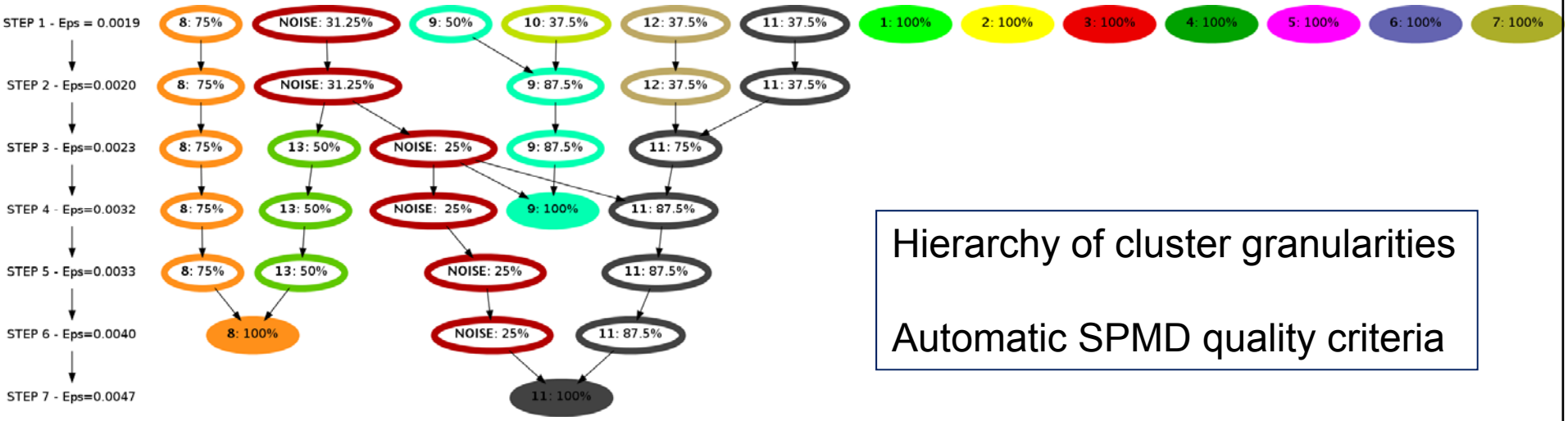


BT.A 0043

Quality driven clustering algorithms







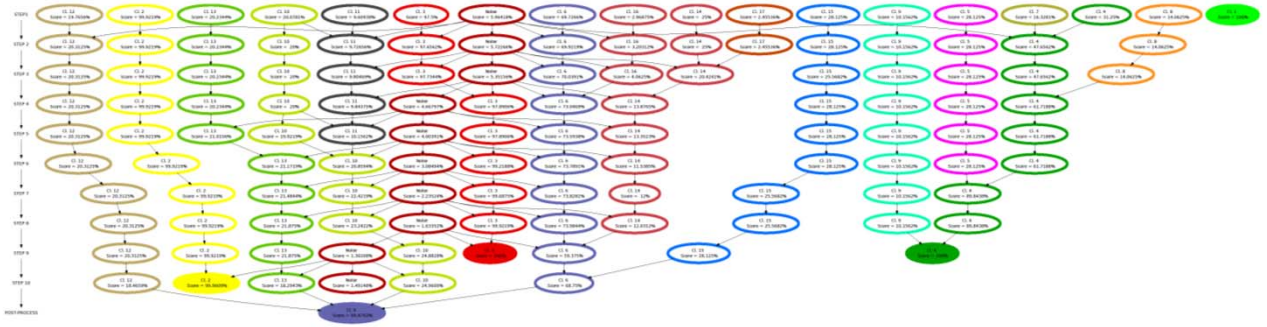
Hierarchy of cluster granularities

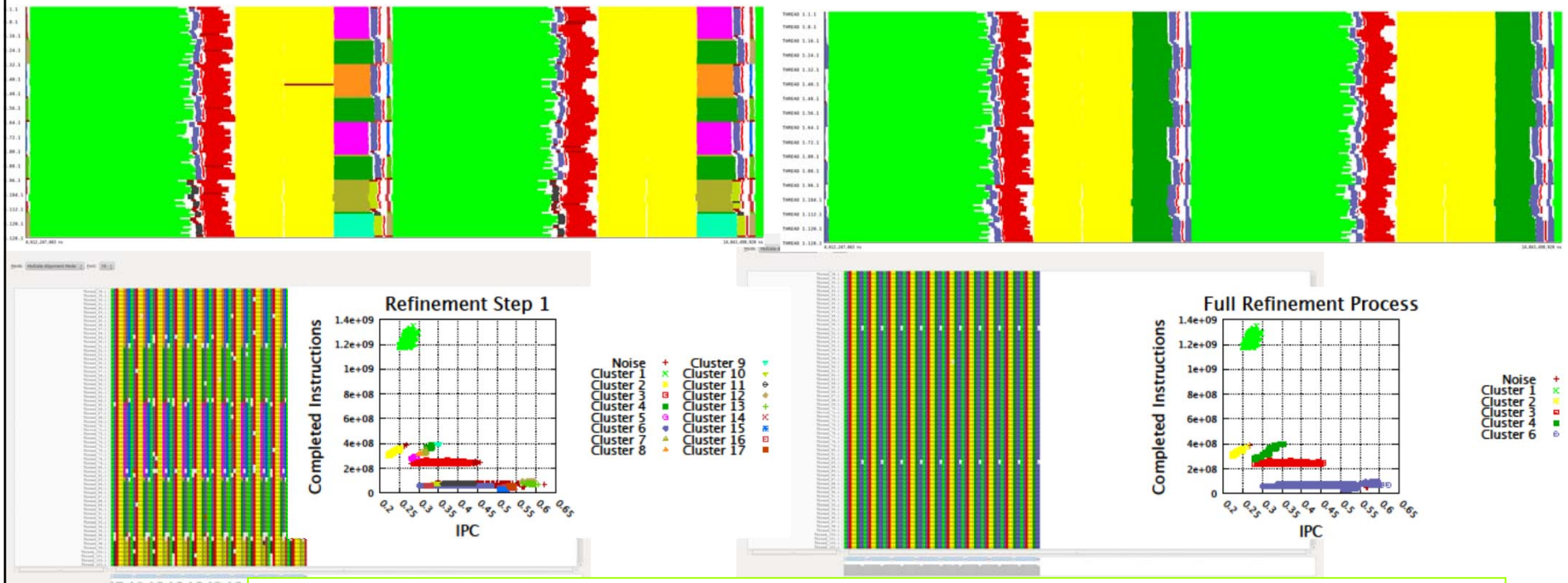
Automatic SPMD quality criteria

J. Gonzalez et al, "Automatic Refinement of Parallel Application Structure Detection" (LSPP 2012)

Hierarchical clustering guided by SPMDness

- Automatic improvement of clustering "quality"



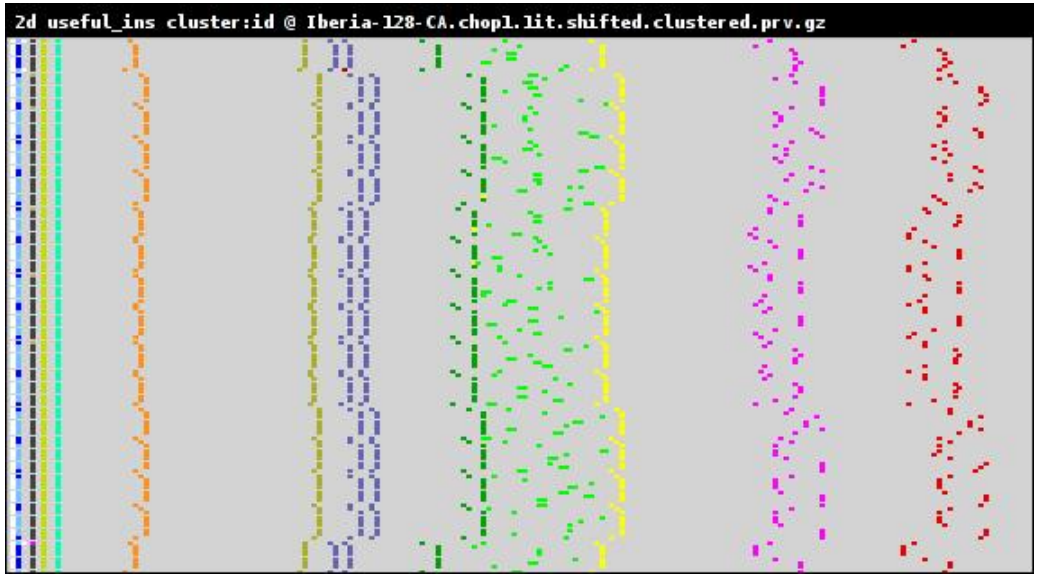


J. Gonzalez et al, "Automatic Refinement of Parallel Application Structure Detection" (LSPP 2012)

Using clustering

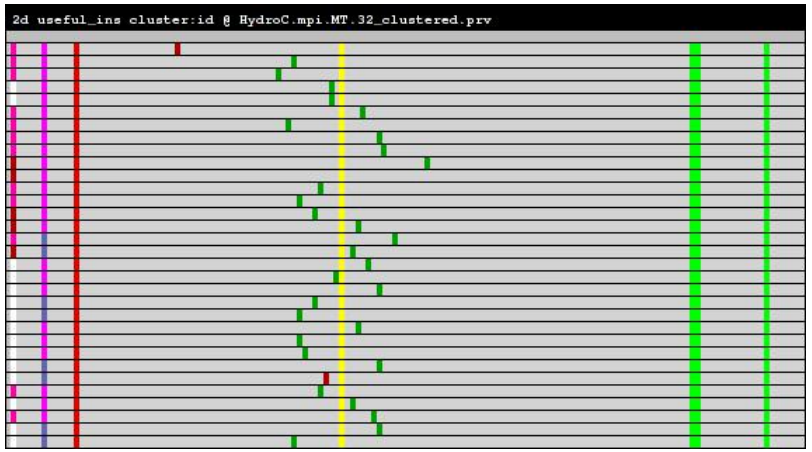
- Clustering enables focusing the analysis and opens many different uses
  - Analysis
    - Detection of application structure
  - Precise instantaneous metrics
    - correlation of sampled data to generate instantaneous metric evolution
  - Dimemas:
    - Separate speed factors per cluster on predictive simulations
  - Track the evolution of application behaviour effects
  - …

Using clustering: Identifying code regions and variability



WRF

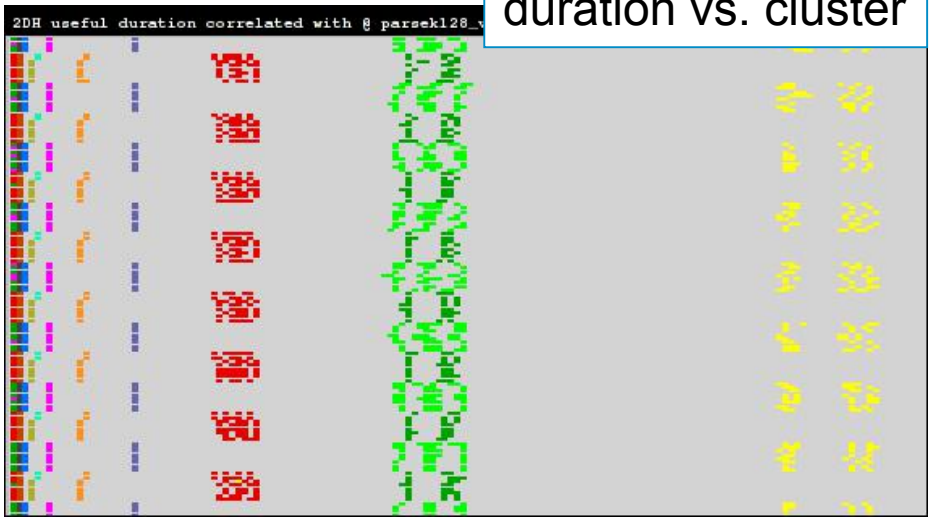
instr. vs. cluster



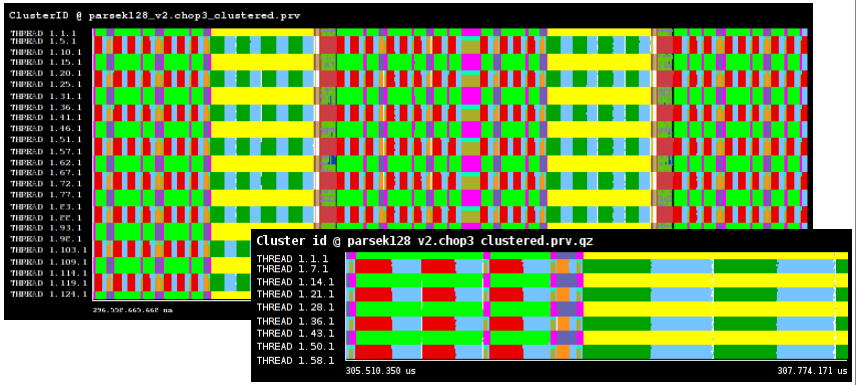
HydroC

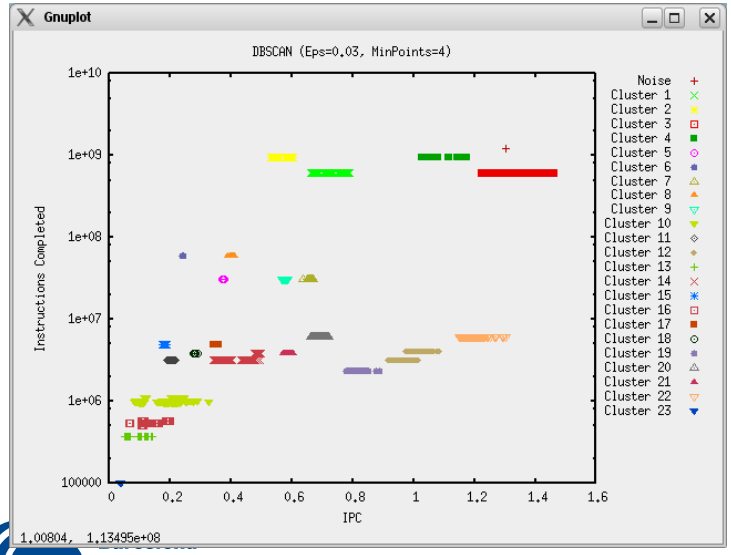
Identifying variability in space and time

duration vs. cluster

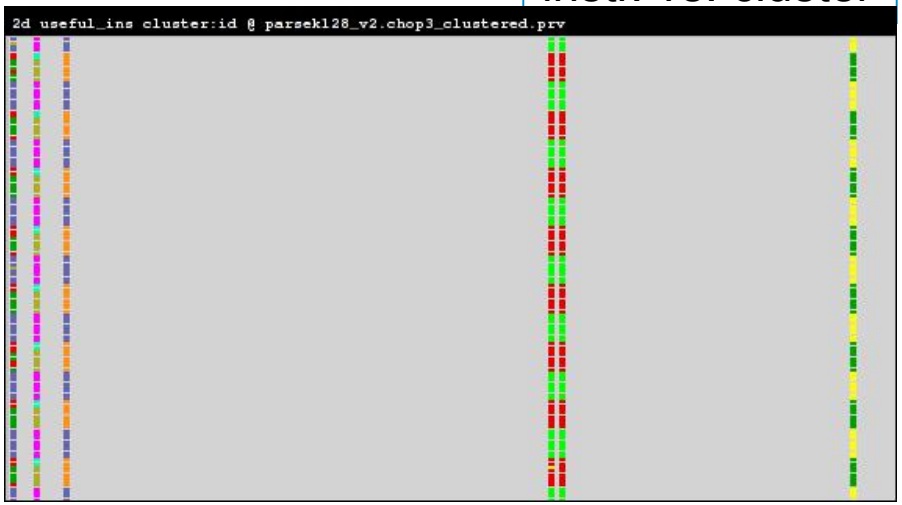


PARSEK





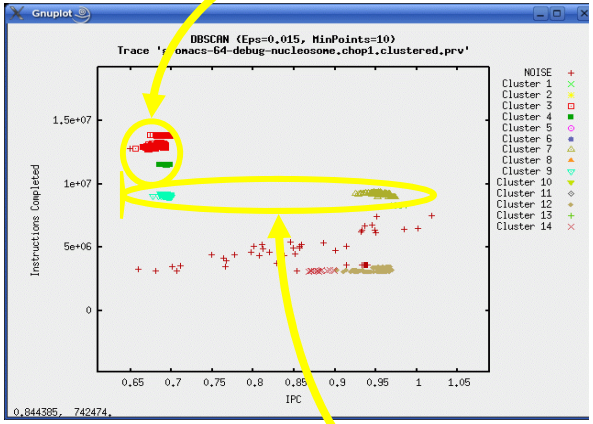
instr. vs. cluster

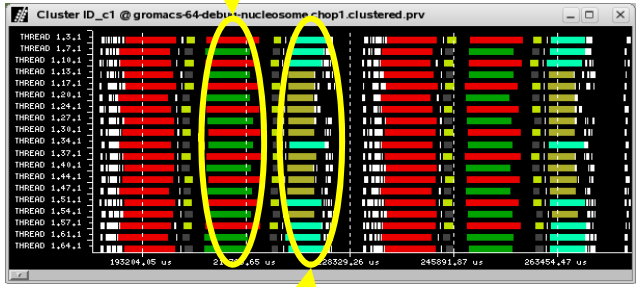


Using clusters: to understand apps behavior

Instructions imbalance

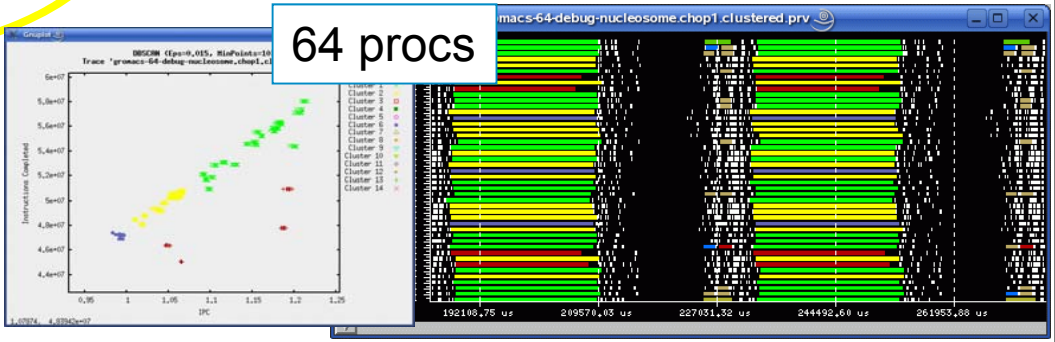
GROMACS



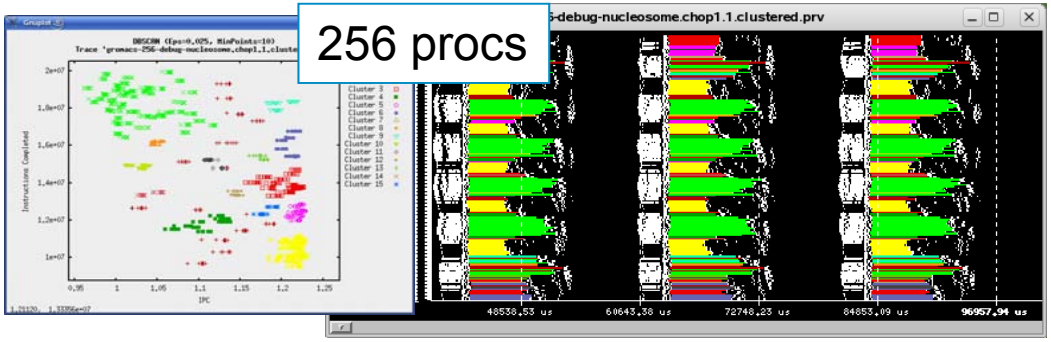


IPC Imbalance

64 procs



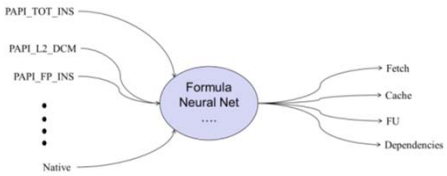
256 procs



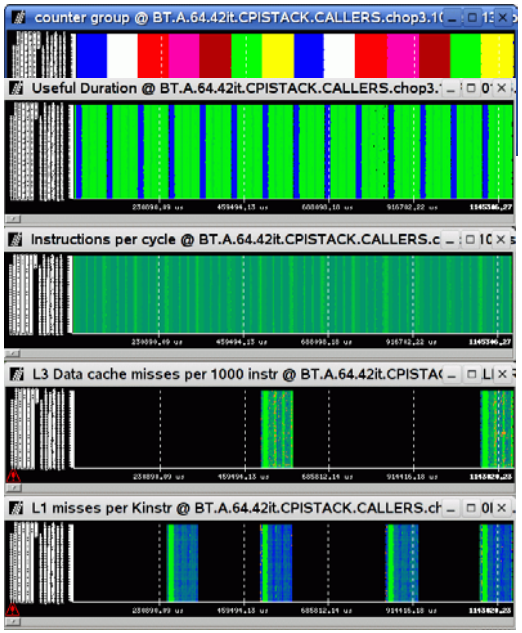
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Using clusters: HWC projection

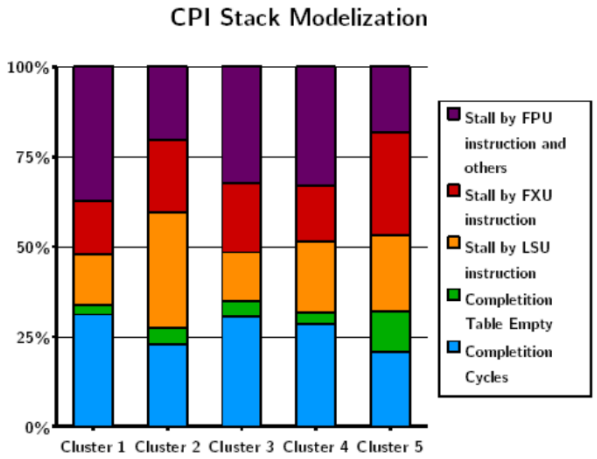
- Complete hardware counter characterization
  - Precise Statistics (>> multiplexing)
  - CPI stack model



- From a single run



Cluster	1	2	3	4	5
%Time	54.88	17.96	16.90	6.44	1.42
Avg. Burst Dur. (ms)	1.02	0.78	13.14	2.50	1.11
IPC	1.02	0.65	0.89	0.91	0.53
MIPS	2231.8	1423.3	1966.5	2001.8	1163.0
MFLOPS	339.2	46.3	191.6	269.2	23.6
L1M/KInstr	0.92	1.53	1.19	1.17	2.88
L2M/KInstr	0.06	1.26	0.06	0.35	0.21
Mem.BW (MB/s)	16.79	218.47	13.87	85.77	29.76

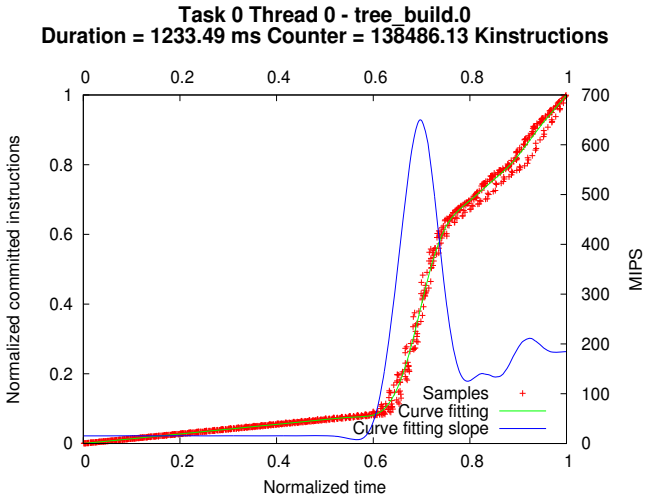


J. Gonzalez et all, "Performance Data Extrapolation in Parallel Codes", ICPADS 2010

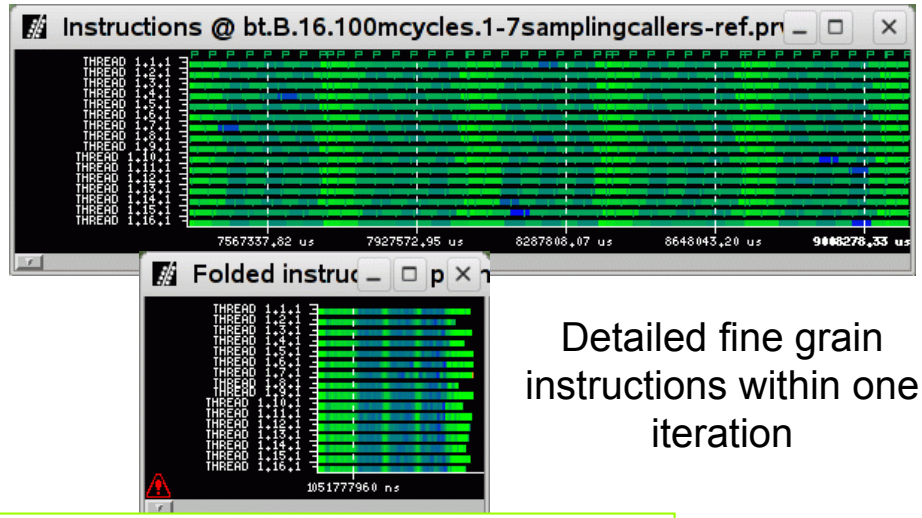
Folding

Folding: Instrumentation + sampling

- Extremely detailed time evolution of hardware counts, rates and callstack
- Minimal overhead
- Based on
  - Instrumentation events (iteration, MPI, …) and periodic samples.
  - Application structure: manual iteration instrumentation, routines, clusters
- Folding
  - Post processing to project all samples into one instance



Original sampled instructions

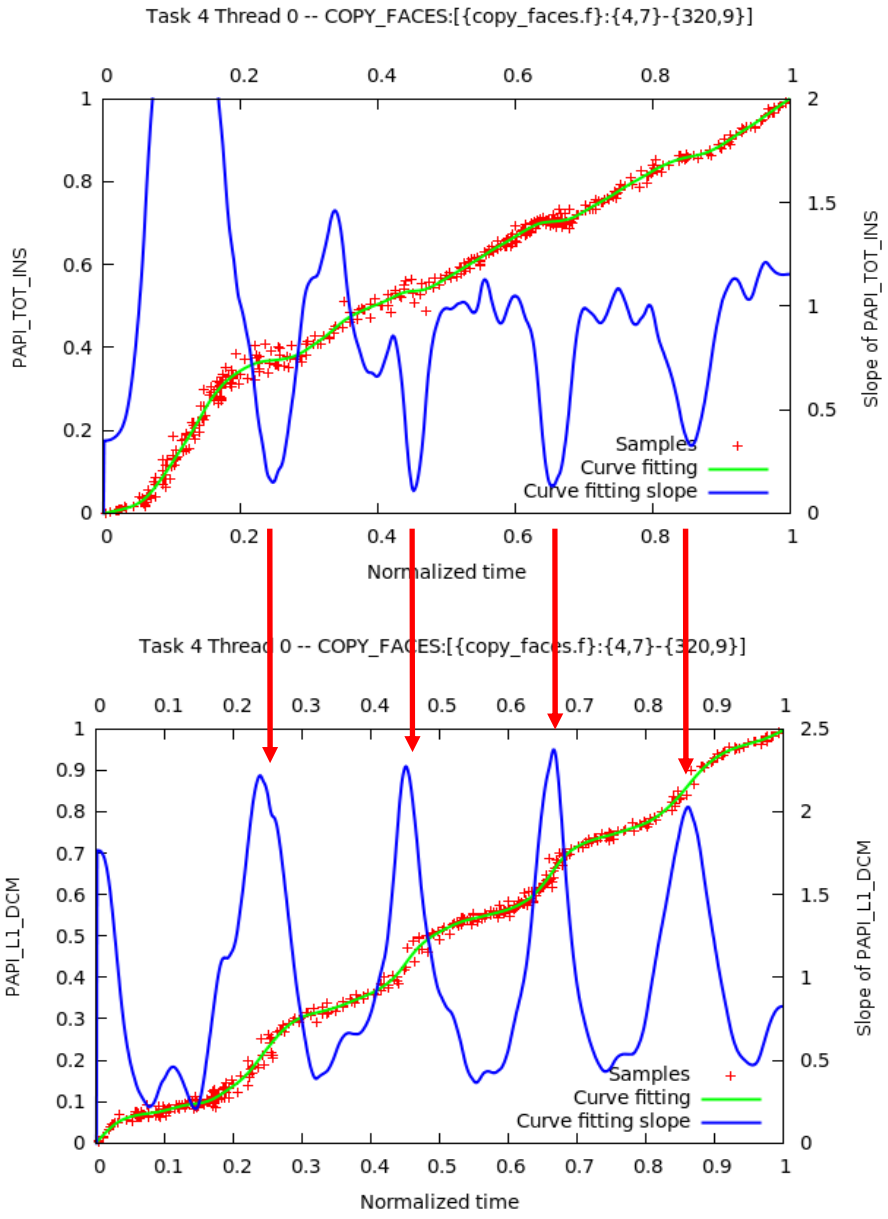


Detailed fine grain instructions within one iteration

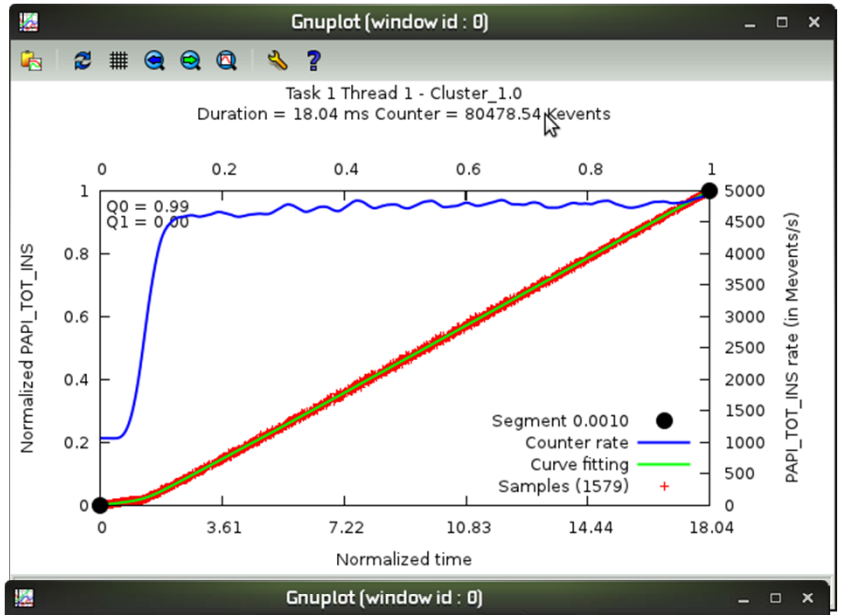
Harald Servat et al. "Detailed performance analysis using coarse grain sampling" PROPER@EUROPAR, 2009

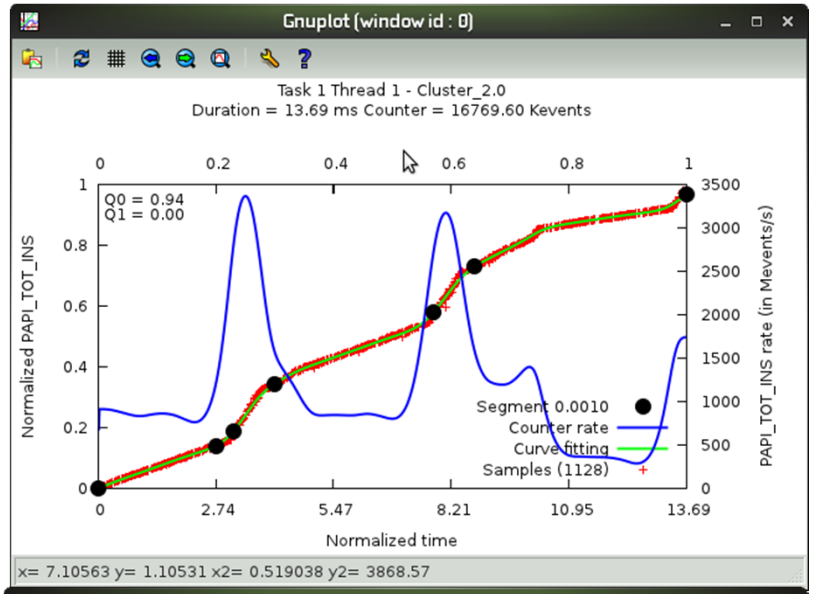
Sampling and Folding

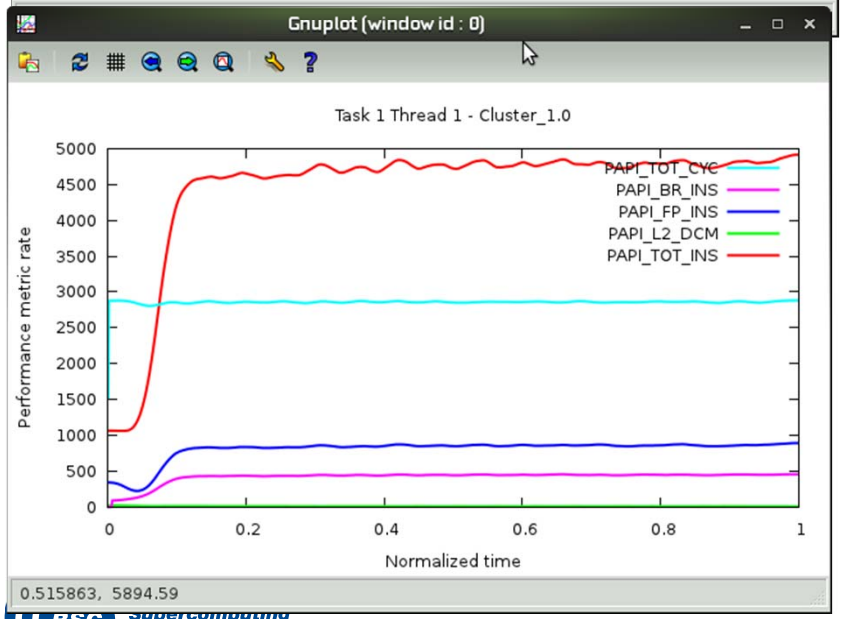
- Instructions
- L1 data cache misses

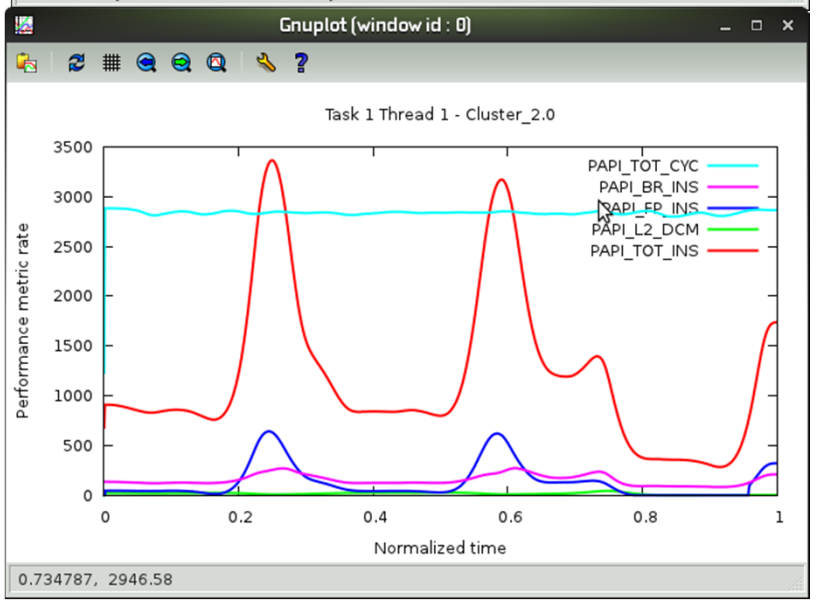


Sampling and Folding



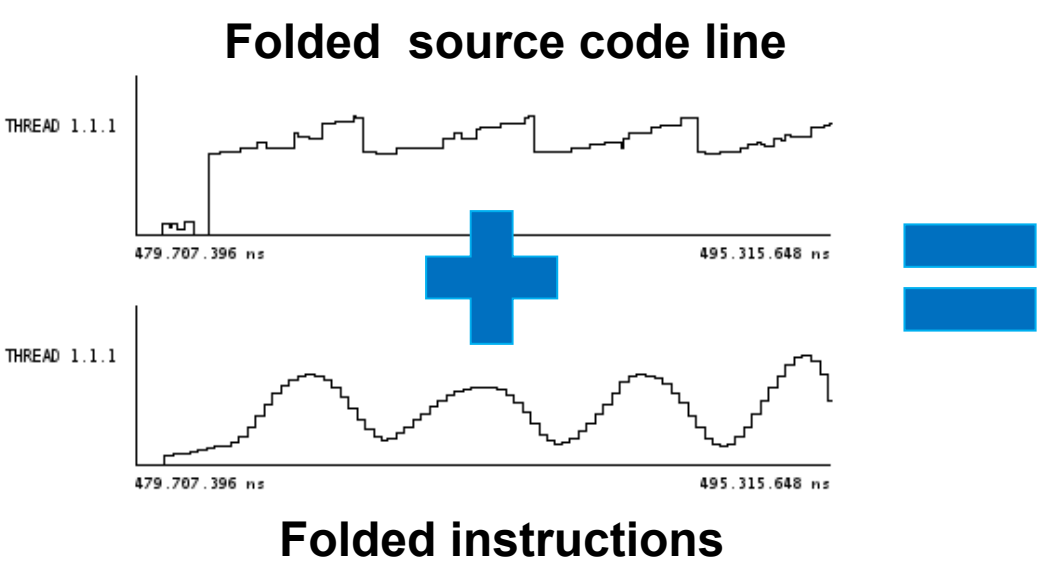


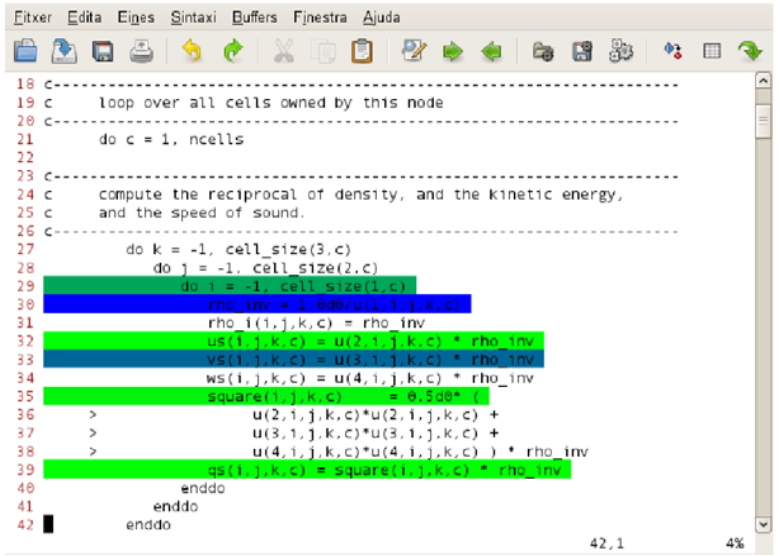




Folding → profiles of rates and ratios

- 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





H. Servat et al. "Unveiling Internal Evolution of Parallel Application Computation Phases" ICPP 2011

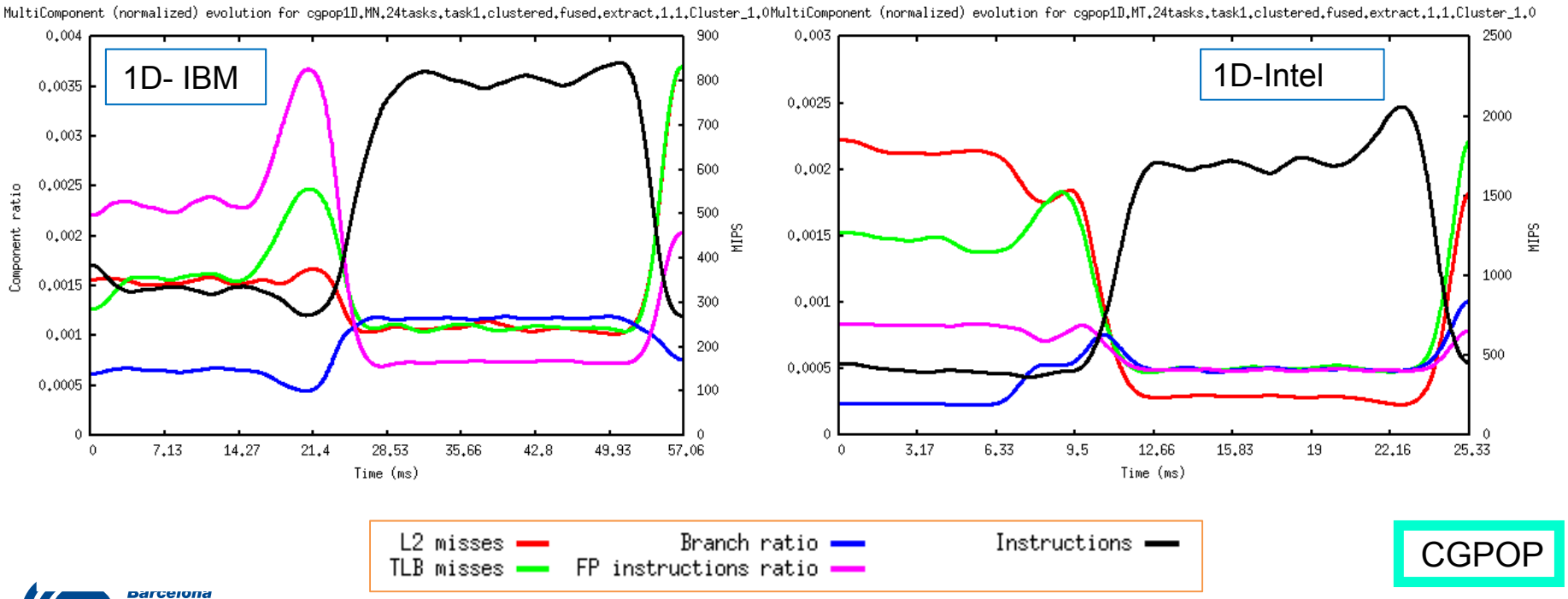
Examples

Instantaneus hardware counter rates & CPI stack models

- Normalized rates

$$M(c,t) = \frac{Counter\_rate(c,t)/\max(Counter\_rate(c,t))}{MIPS(t)}$$

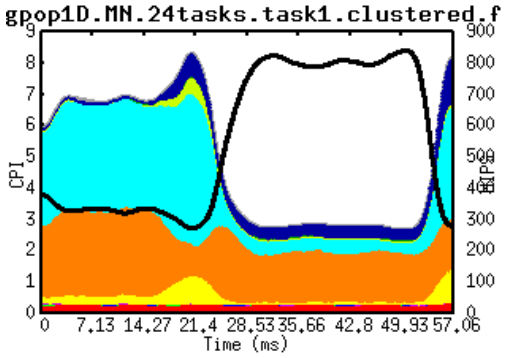
- Different compilers and architectures → slightly different characteristics

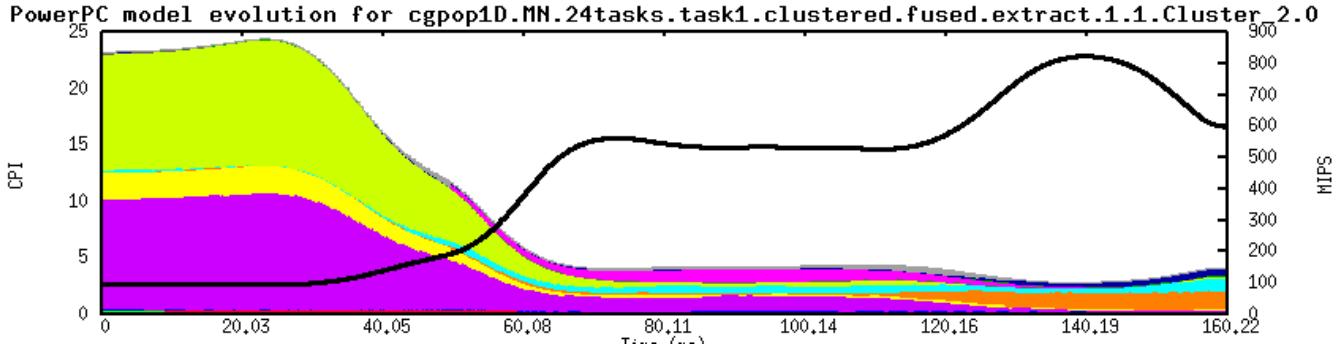


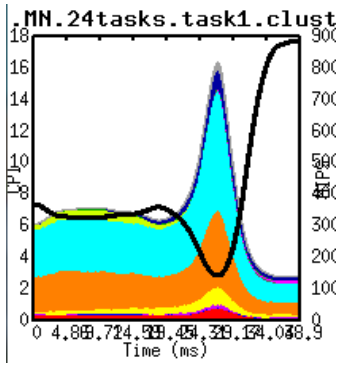
Instantaneus hardware counter rates & CPI stack models

- 1D
- 180 x 120
- 24 processes
- IBM PPC-970
- Three main program regions

CGPOP







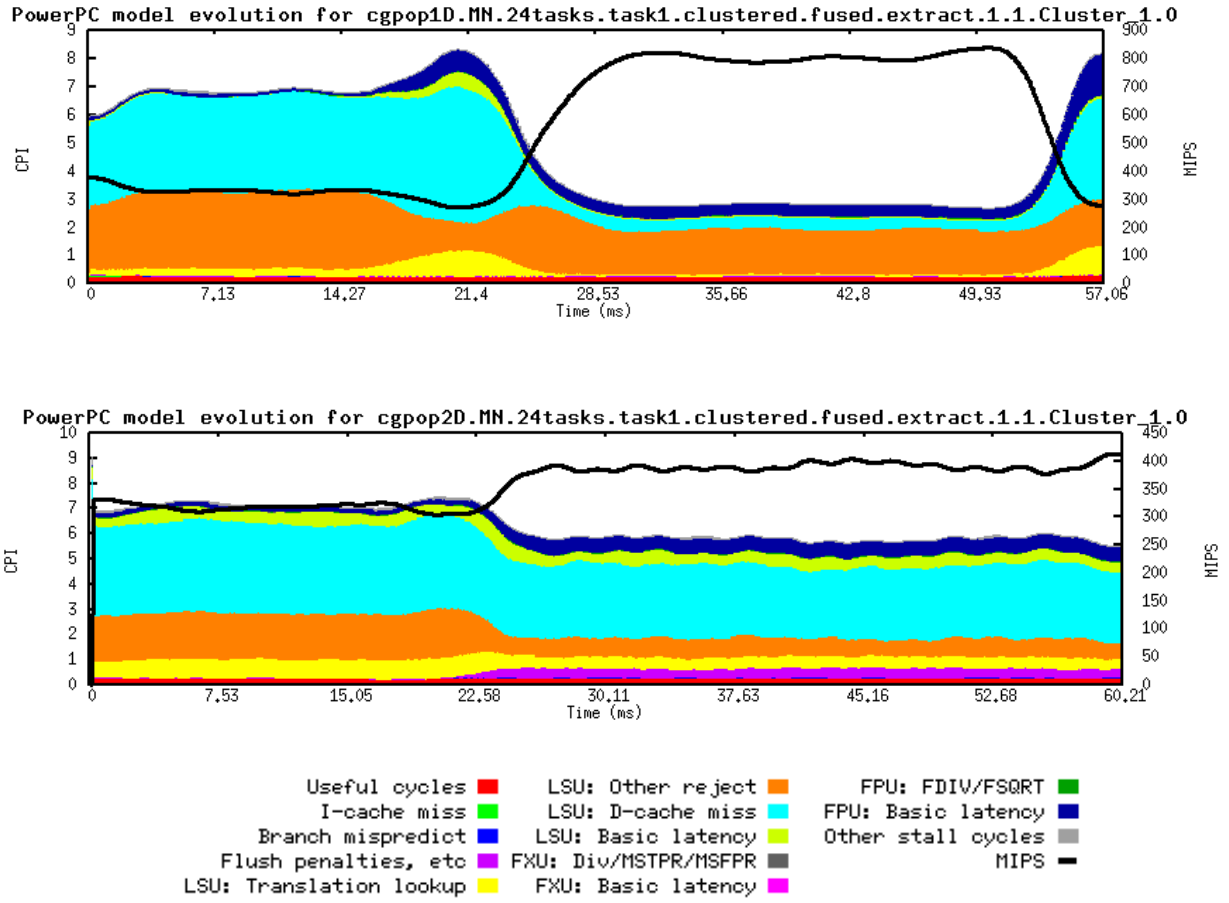
Useful cycles
I-cache miss
Branch mispredict
Flush penalties, etc
LSU: Translation lookup
LSU: Other reject
LSU: D-cache miss
LSU: Basic latency
FXU: Div/MSTPR/MSFPR
FXU: Basic latency
FPU: FDIV/FSQRT
FPU: Basic latency
Other stall cycles
MIPS

H. Servat et al. "Unveiling Internal Evolution of Parallel Application Computation Phases" ICPP 2011

H. Servat et al. "Methodology and framework for a productive performance optimization" Submitted

1D vs. 2D

CGPOP



Sequential code optimization

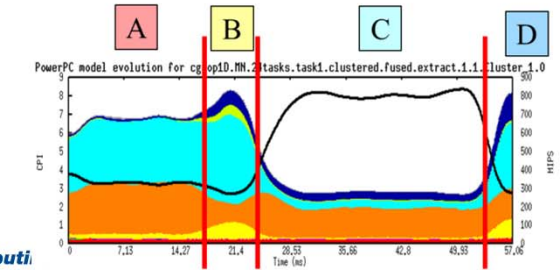
```
iter_loop: do m = 1, solv_max_iters
  sumN1=c0
  sumN3=c0
  do i=1,nActive
    Z(i) = Minv2(i)*R(i)
    sumN1 = sumN1 + R(i)*Z(i)
    sumN3 = sumN3 + R(i)*R(i)
  enddo
  do i=iptrHalo,n
    Z(i) = Minv2(i)*R(i)
  enddo
  call matvec(n,A,AZ,Z)
  sumN2=c0
  do i=1,nActive
    sumN2 = sumN2 + AZ(i)*Z(i)
  enddo
  call update_halo(AZ)
  ...
  do i=1,n
    stmp = Z(i) + cg_beta*S(i)
    qtmp = AZ(i) + cg_beta*Q(i)
    X(i) = X(i) + cg_alpha*stmp
    R(i) = R(i) - cg_alpha*qtmp
    S(i) = stmp
    Q(i) = qtmp
  enddo
end do iter_loop
```

B C D A

```
sumN1=c0
sumN3=c0
do i=1,nActive
  Z(i) = Minv2(i)*R(i)
  sumN1 = sumN1 + R(i)*Z(i)
  sumN3 = sumN3 + R(i)*R(i)
enddo
do i=iptrHalo,n
  Z(i) = Minv2(i)*R(i)
enddo

iter_loop: do m = 1, solv_max_iters
  sumN2=c0
  call matvec r(n,A,AZ,Z,nActive,sumN2)
  call update_halo(AZ)
  ...
  sumN1=c0
  sumN3=c0
  do i=1,n
    stmp = Z(i) + cg_beta*S(i)
    qtmp = AZ(i) + cg_beta*Q(i)
    X(i) = X(i) + cg_alpha*stmp
    R(i) = R(i) - cg_alpha*qtmp
    S(i) = stmp
    Q(i) = qtmp
    Z(i) = Minv2(i)*R(i)}
    if (i <= nActive) then}
      sumN1 = sumN1 + R(i)*Z(i)
      sumN3 = sumN3 + R(i)*R(i)
    endif
  enddo
end do iter_loop
```

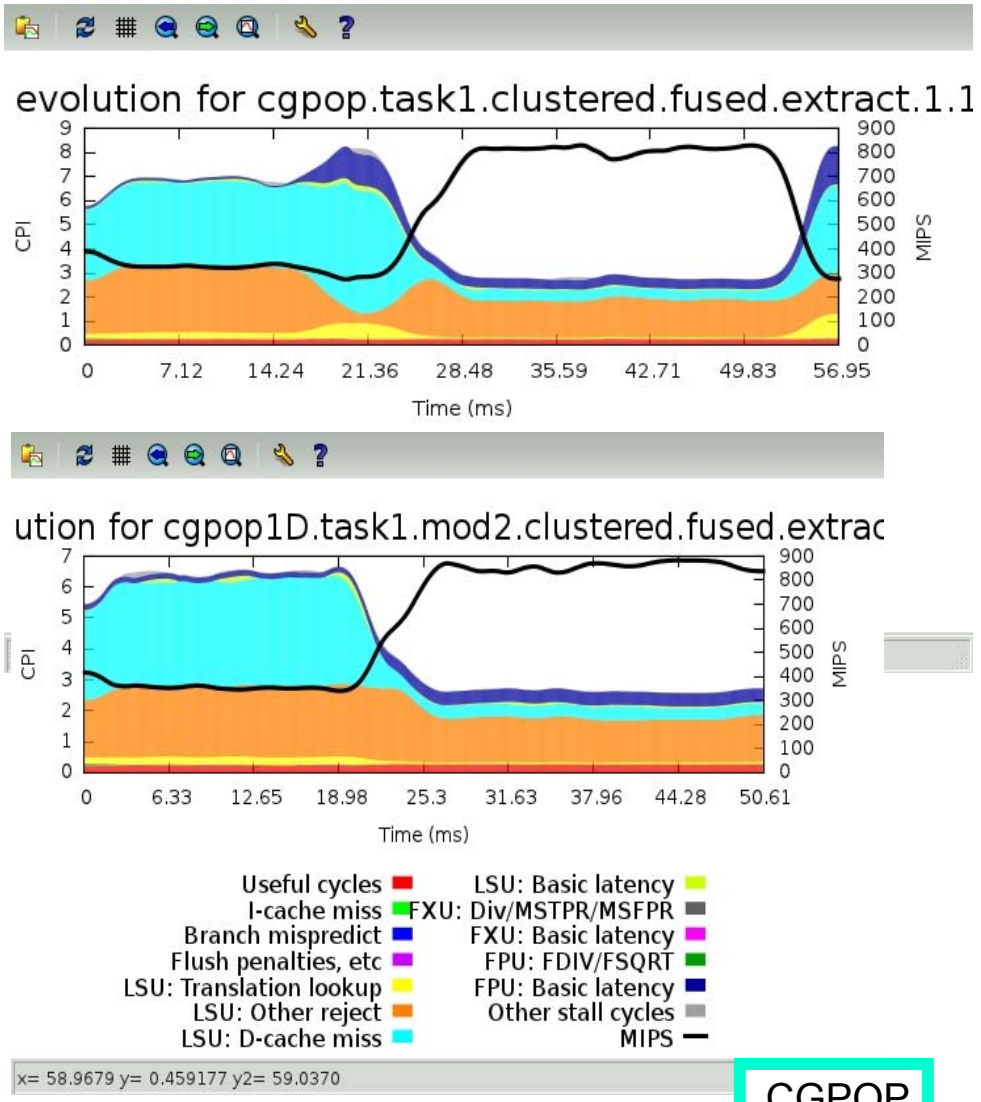
B A



CGPOP

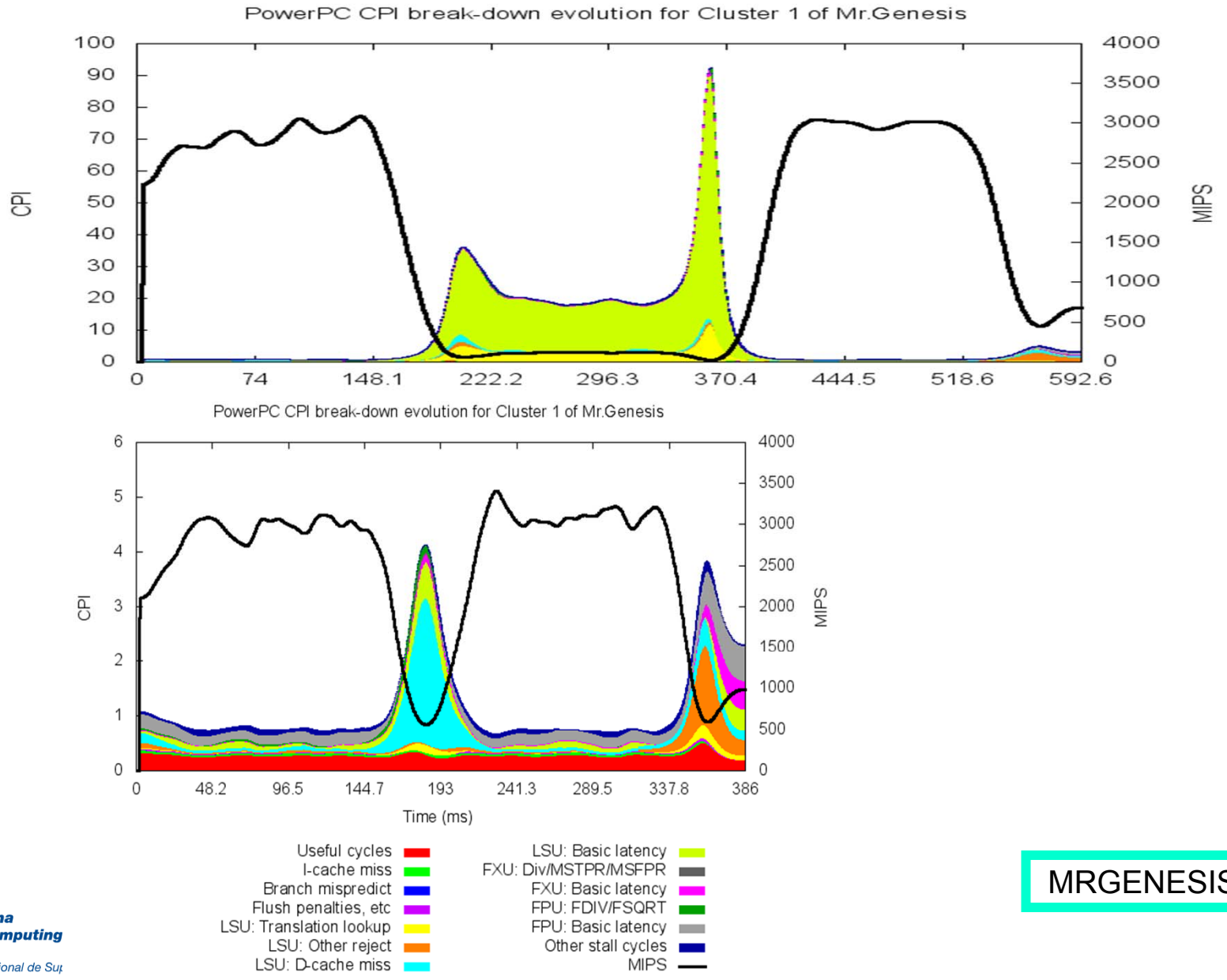
Clustering + Folding + CPI Stack

- Unmodified production binary
  - Instrumentation of MPI calls plus periodic samples. Rotating hardware counters.
  - Clustering detects structure to enable folding
  - Folding of multiple hardware counters to compute CPI stack model
- Instantaneous MIPS and CPI Stack model
  - Clean abstract identification of performance problem
  - Good identification of phases



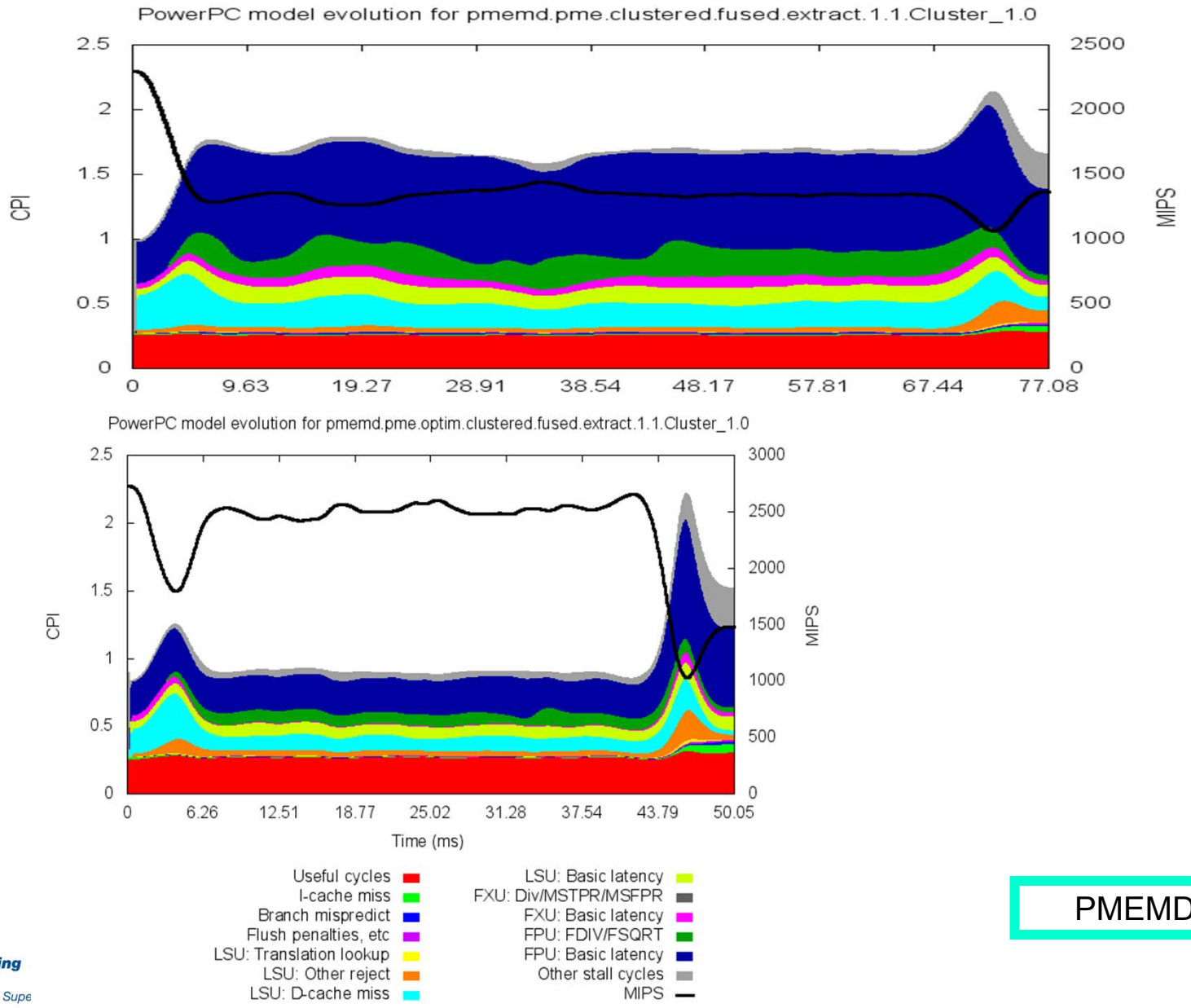
CGPOP

Instantaneous CPI stack



MRGENESIS

Instantaneous CPI stack

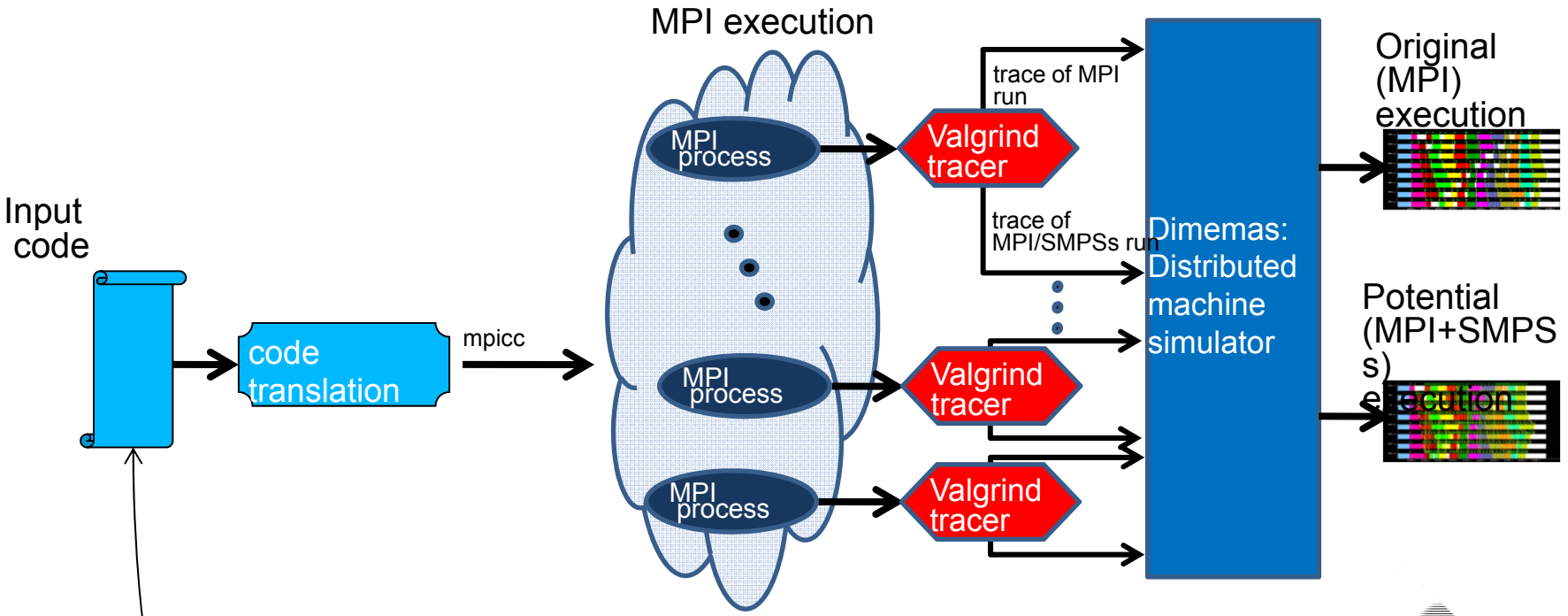


PMEMD

Tareador

Predicting performance of task based models

- Performance prediction
  - Predict MPI/StarSs multithreaded from a pure MPI run
  - Leveraging other tools in environment



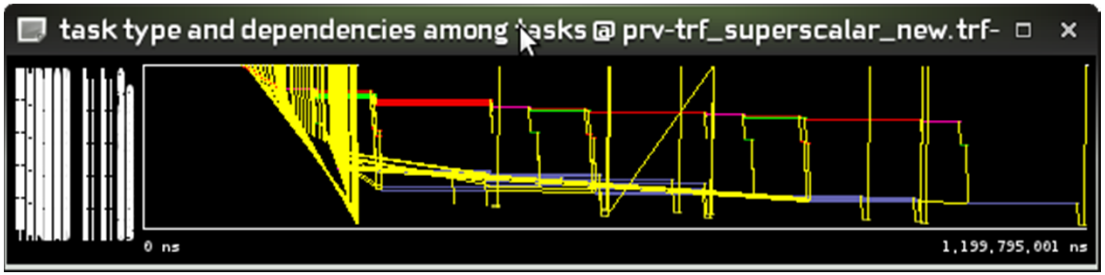
Incomplete/suggested taskification
- pragmas
- runtime calls (even on not well structured code)

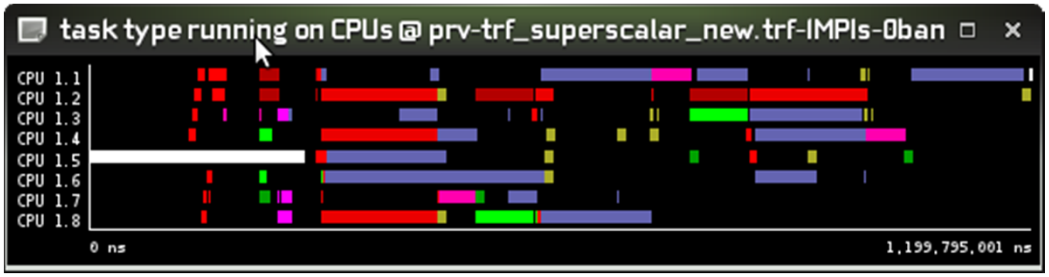
bscgrid06.bsc.es/~tareador

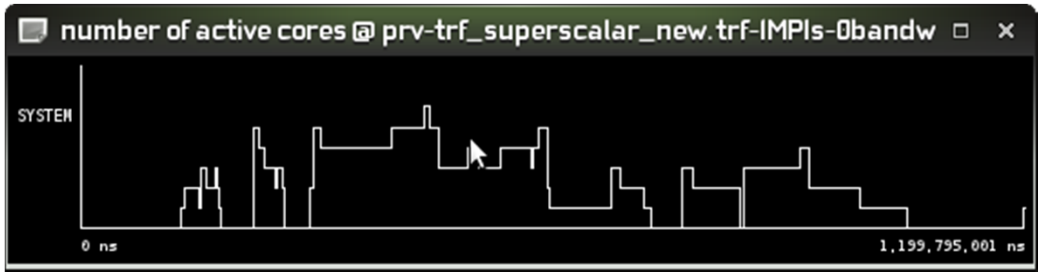
Predicting performance

- Potential concurrency between task (for a suggested taskification)…
- … as number of cores increases.

Predicting performance







Conclusion

Conclusion

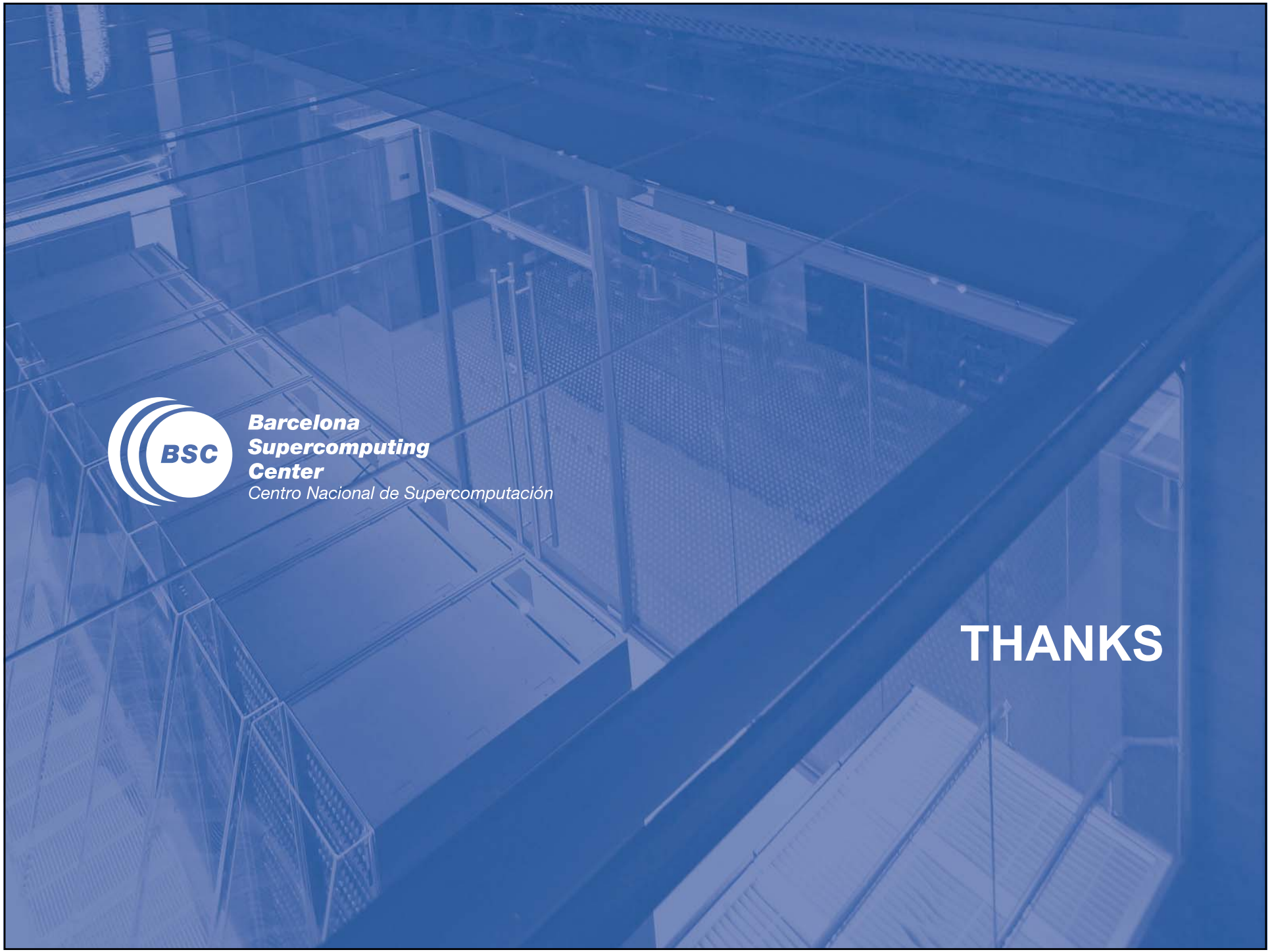
- Dominant practice
  - We focus a lot on capturing a lot of data
  - but we present either everything or first order statistics
  - and require new experiments without squeezing the potential information from the previous one
- Need for performance analytics
  - Leveraging techniques from data analytics, mining, signal processing, life sciences,…
  - towards insight
  - and models

Conclusion

- BSC tools: Extreme flexibility and analysis power
  - Maximize insight obtained form single experiment
  - Learning curve
  - “Don’t ask whether something can be done, ask how can it be done”
- Huge:
  - Needs for such detailed and precise analysis
    - Systems are complex, variability is everywhere, …
    - Insight, not speculation to fly in the midst and “maximize” productivity
  - Potential of data analysis techniques applied to performance data

« Use them, stay tuned:

www.bsc.es/paraver


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
THANKS

Detailed material

Semantic Module

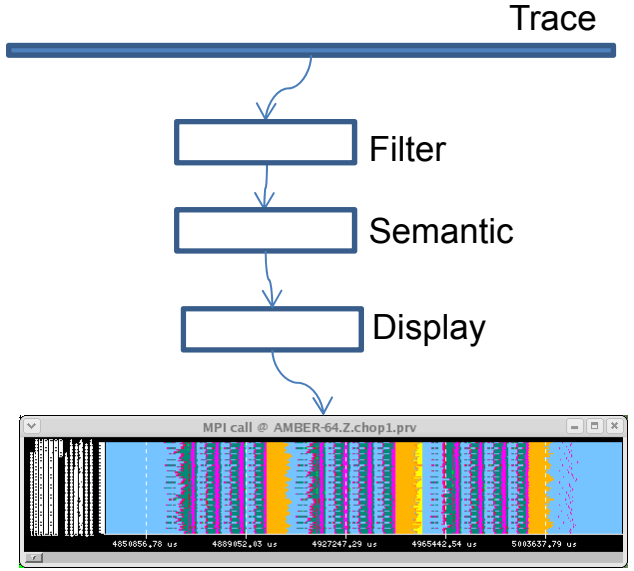
Basic functions of time

- The filter module presents a subset of the trace to the semantic module. Each thread th is described by
  - A sequence of events $Ev_i, i \in N$, states $St_i, i \in N$ and communications $C_i, i \in N$
  - For each event let $T(Ev_i)$ be its time and $V(Ev_i)$ its value
  - For each state let $T_s(St_i)$ be its start time $T_e(St_i)$ its stop time and $V(St_i)$ its value
  - For each Communication let $T_S(C_i)$ be its send time, $T_R(C_i)$ its receive time, $Sz(C_i)$ its size.
  - $Partner(C_i)$ and $Dir(C_i) \in \{send, recv\}$ identify the partner process and direction of the transfer

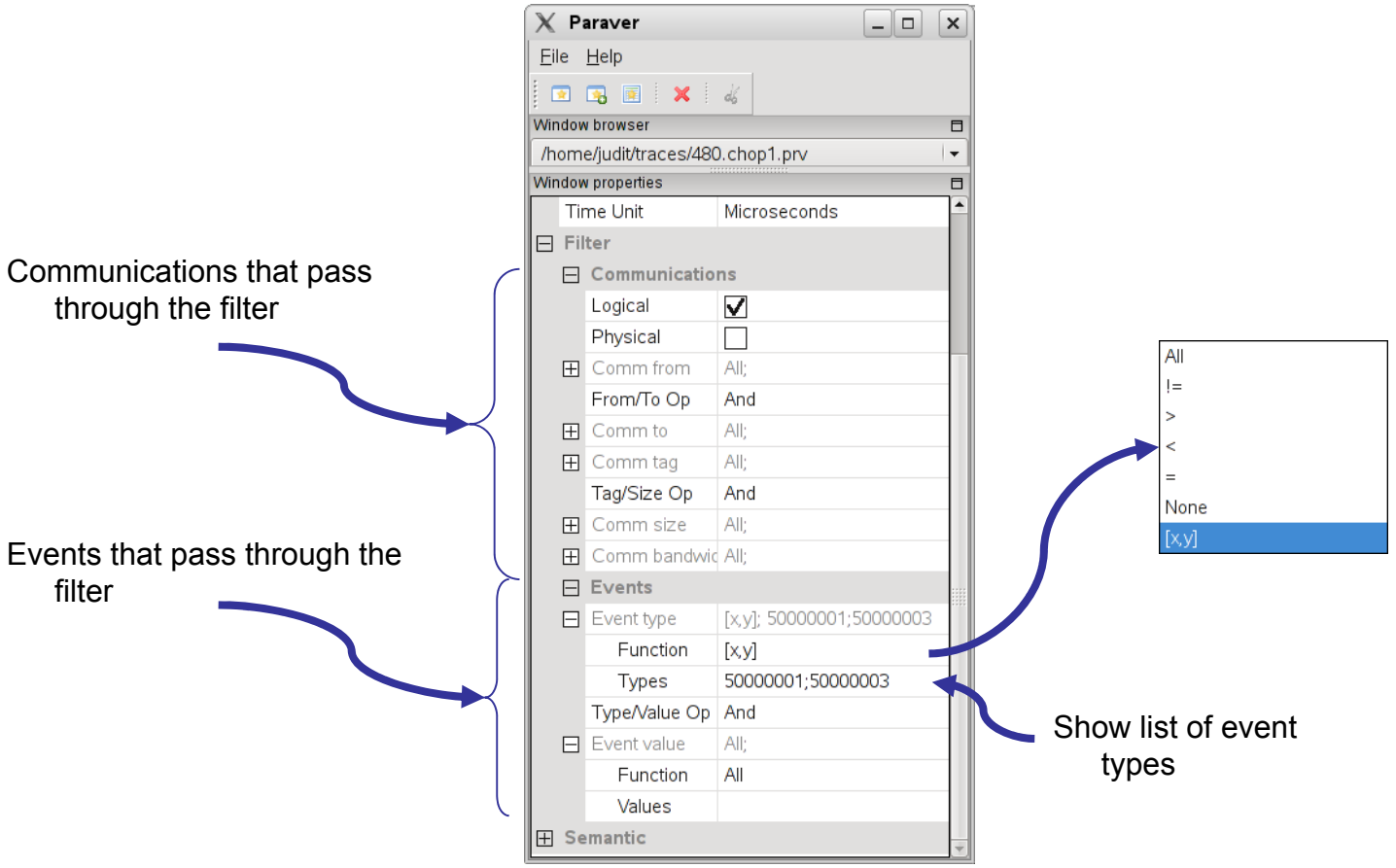
- Semantic module builds

$$s(t) = S(i), t \in [t_i, t_{i+1}), i \in \mathrm{N}$$

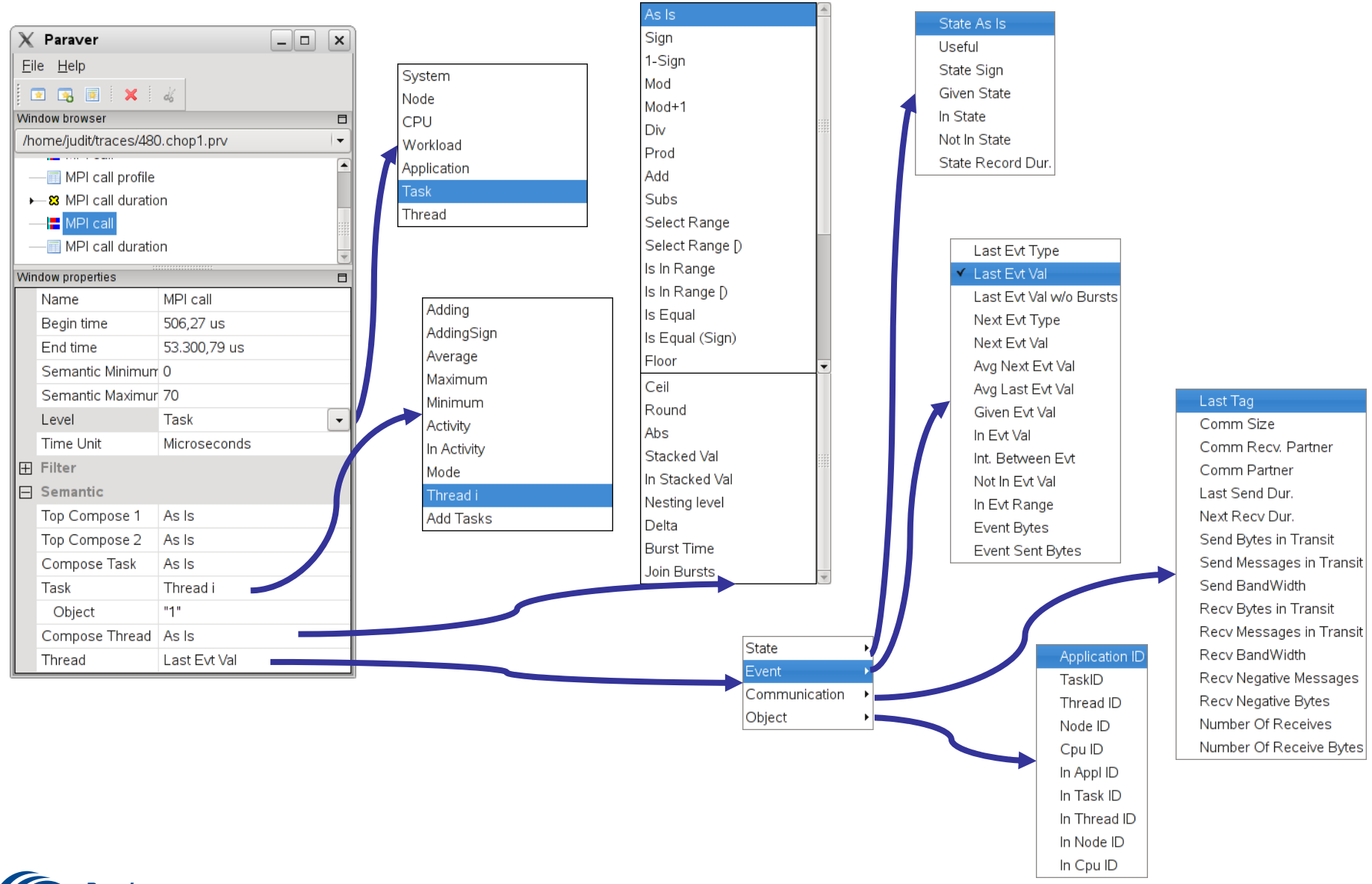
Function of time Series of values



Filter module



Semantic module: Control



Semantic module

- From Events to functions of time
  - Last event value $S(i) = V(Ev_i)$
  - Next event value $S(i) = V(Ev_{i+1})$
  - Average Next Event Value $S(i) = \frac{V(Ev_{i+1})}{T(Ev_{i+1}) - T(Ev_i)}$
  - Interval btw. Events $S(i) = T(Ev_{i+1}) - T(Ev_i)$

Semantic module

- From communication records to functions of time
  - Send Bytes

$$s(t) = \sum_j Sz(C_j), j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$$

  - Send Bandwidth

$$s(t) = \sum_j \frac{Sz(C_j)}{T_R(C_j) - T_S(C_j)}, j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$$

  - Msgs in transit

$$s(t) = \sum_j sign(j), j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$$

  - Recv. Bandwidth

$$s(t) = \sum_j \frac{Sz(C_j)}{T_R(C_j) - T_S(C_j)}, j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(C_j) == recv)$$

  - Rec. Negative Msgs

$$s(t) = \sum_j sign(j), j \mid (T_R(C_j) < t) \wedge (T_S(Cj) > t) \wedge (Dir(Cj) == recv)$$

  - Comm. Partner

$$s(t) = Partner(C_j), j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t)$$

  - Bytes btw. Events

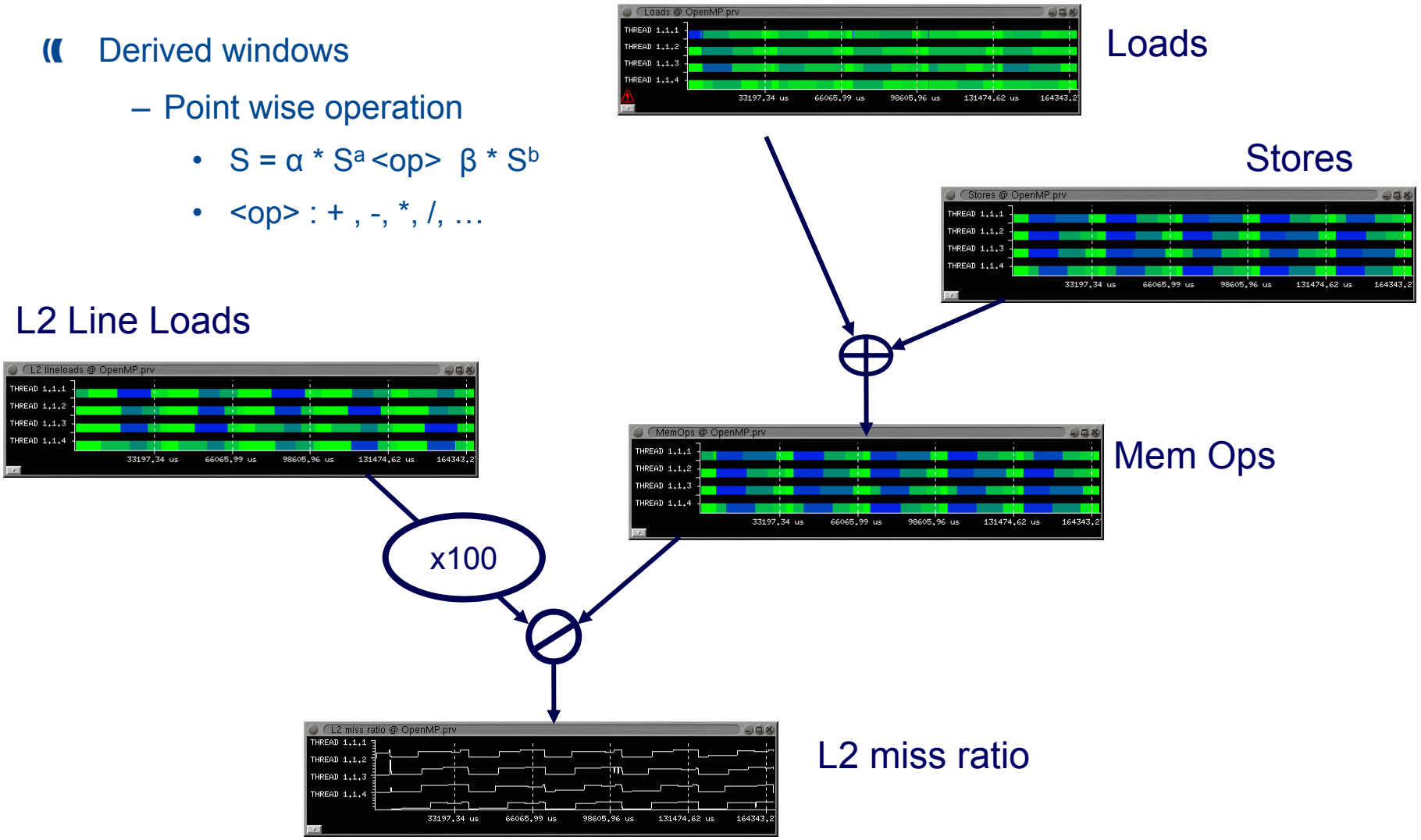
$$S(i) = \sum_j Sz(C_j), j \mid T_S(C_j) \in \left[T(Ev_i), T(Ev_{i+1})\right) \vee T_R(C_j) \in [T(Ev_i), T(Ev_{i+1}))$$

Composition

- S’(t) = f(S(t)) S’ = f ° S
  - Sign $S'(t) = sign\,(S(t))$
  - 1-sign $S'(t) = 1 - sign\,(S(t))$
  - Select range $S'(t) = S(t) \in [a,b] ? S(t) : 0$
  - Sign ° Is equal $S'(t) = sign\,(S(t) = a ? S(t) : 0)$
  - Delta $S'(t) = S_{i+1} - S_i$
  - Stacked value

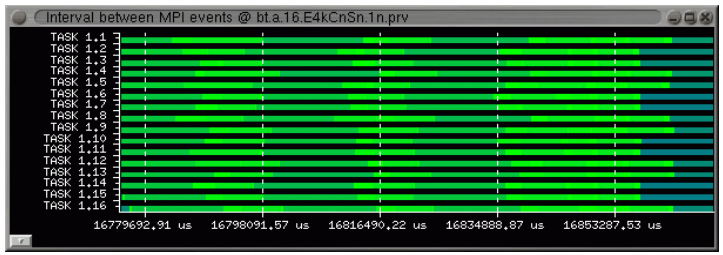
Semantic module

- Derived windows
  - Point wise operation
    - $S = \alpha * S^a \ <op> \ \beta * S^b$
    - <op> : + , -, *, /, …

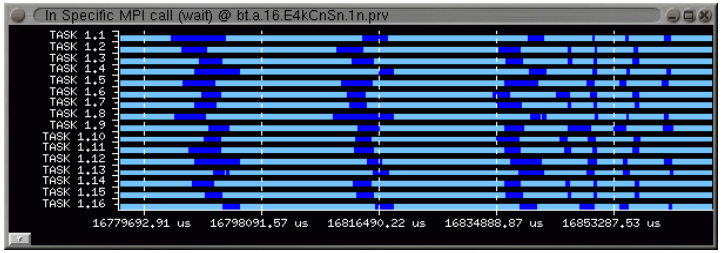


Semantic module

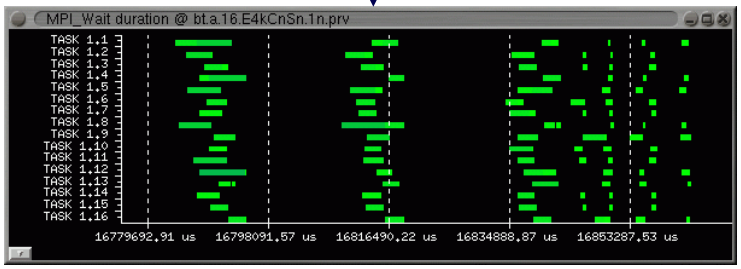
- Derived windows
  - Point wise operation
    - $S = \alpha * S^a \ <op> \ \beta * S^b$
    - <op> : + , -, *, /, …



Interval between MPI events

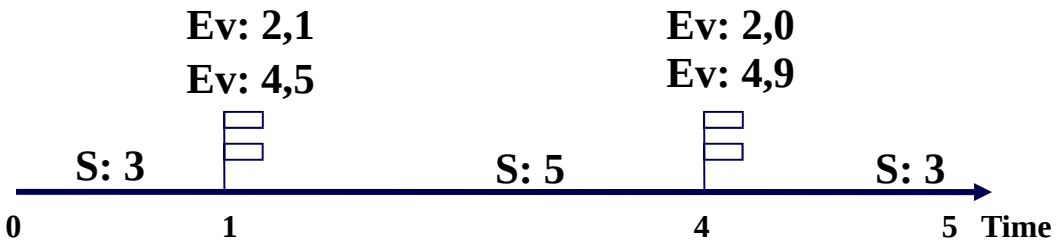


In MPI call

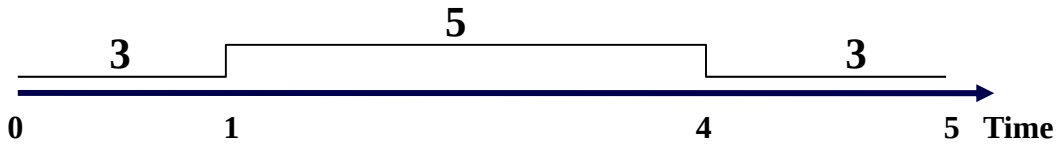


MPI call duration

Semantic module: Examples

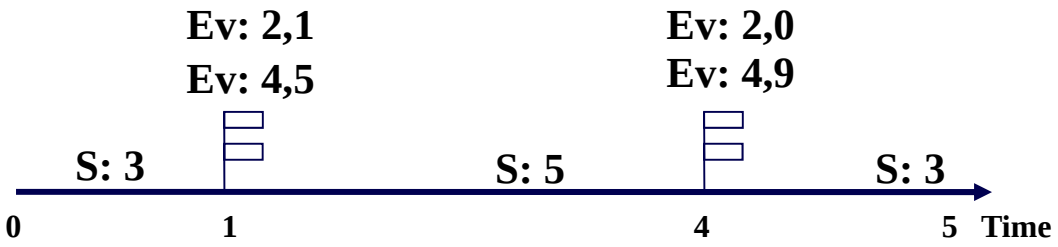


- Thread function: State as is



- Useful for
  - Global thread activity: computing, idle, fork/join, waiting,…..

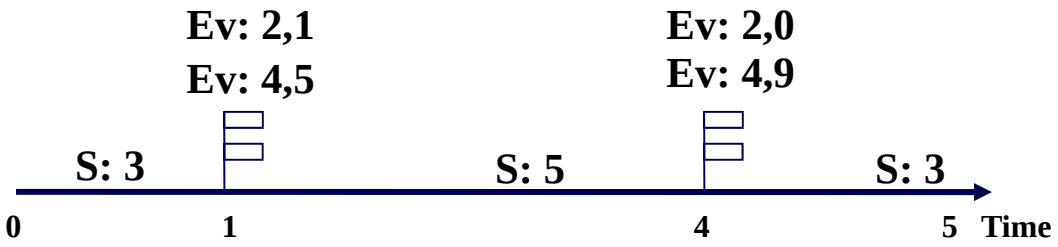
Semantic module: Examples



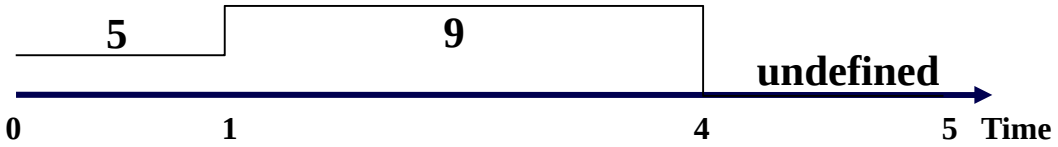
- Filter: type == 2
  - Thread function: Last event value

- Useful for
  - In parallel region
  - Mutual exclusion
  - Variable values: iteration,….

Semantic module: Examples

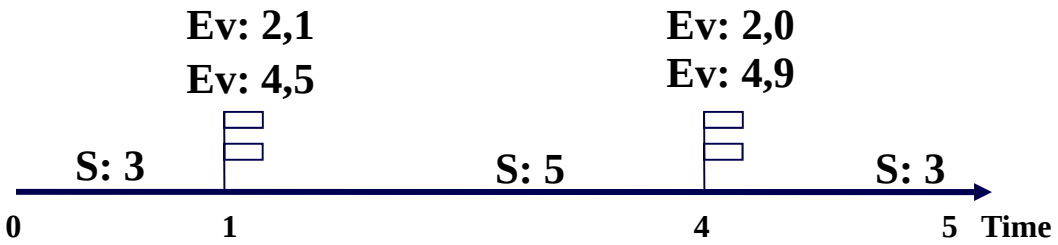


- Filter: type == 4
  - Thread function: Next event value

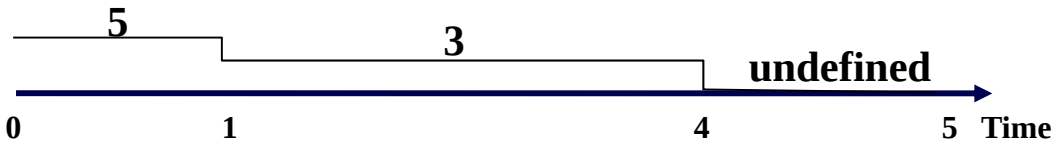


- Useful for
  - Hwc events (TLB, L1 misses,...) within interval

Semantic module: Examples



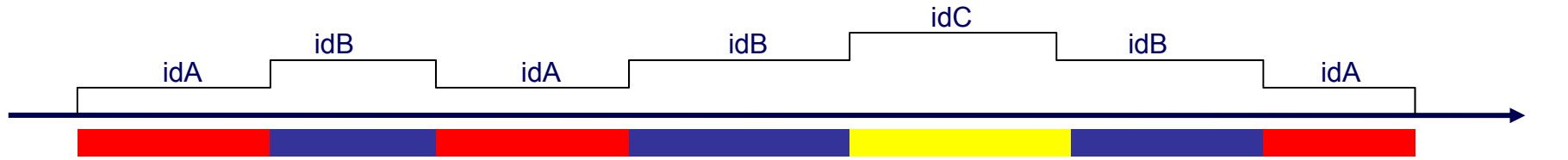
- Filter: type == 4
  - Thread function: Average next event value



- Useful for
  - Hwc events (TLB, L1 misses,...) per time unit within interval

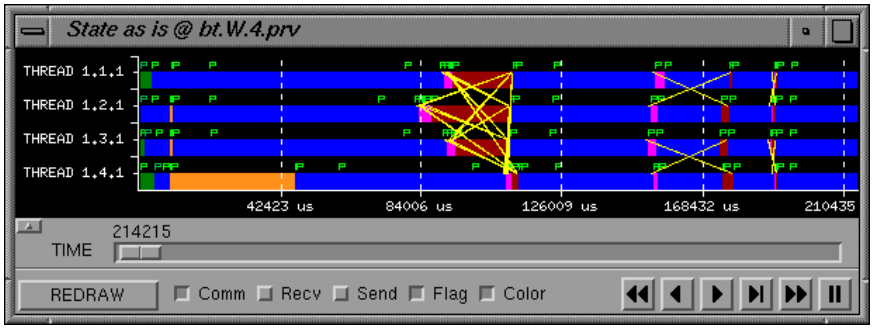
Semantic module: Examples

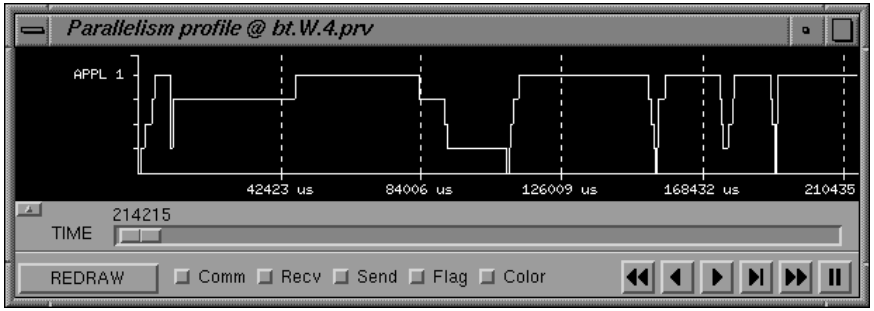
- Filter: type == USR_FCT
  Thread function: Last event value
  Compose: Stacked value



- Useful for
  - Routine

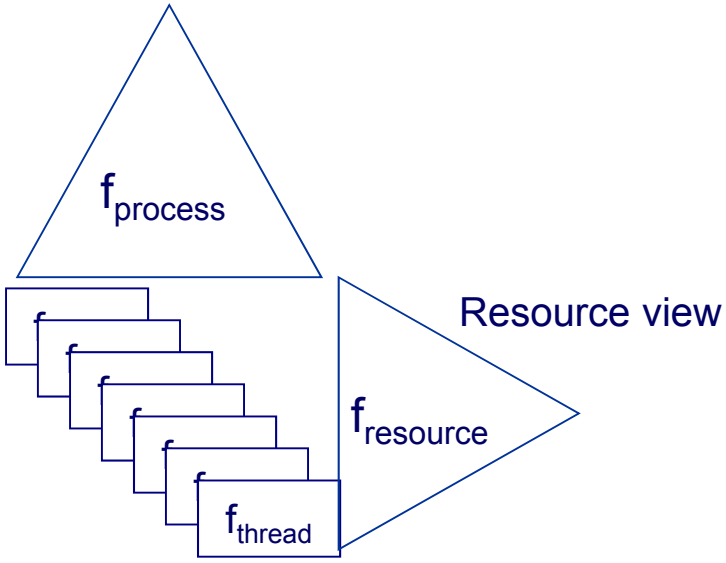
Semantic module





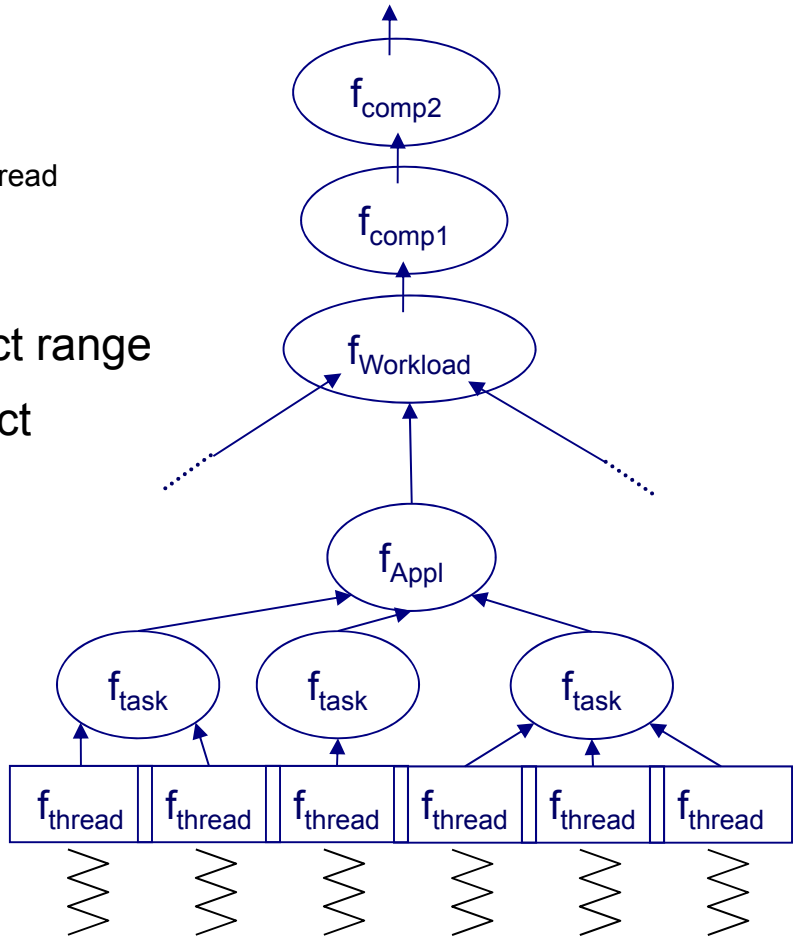
- Perspective:
  - Process model
    - Thread, task, application, workload
  - Resource model
    - CPU, node, system

Process view



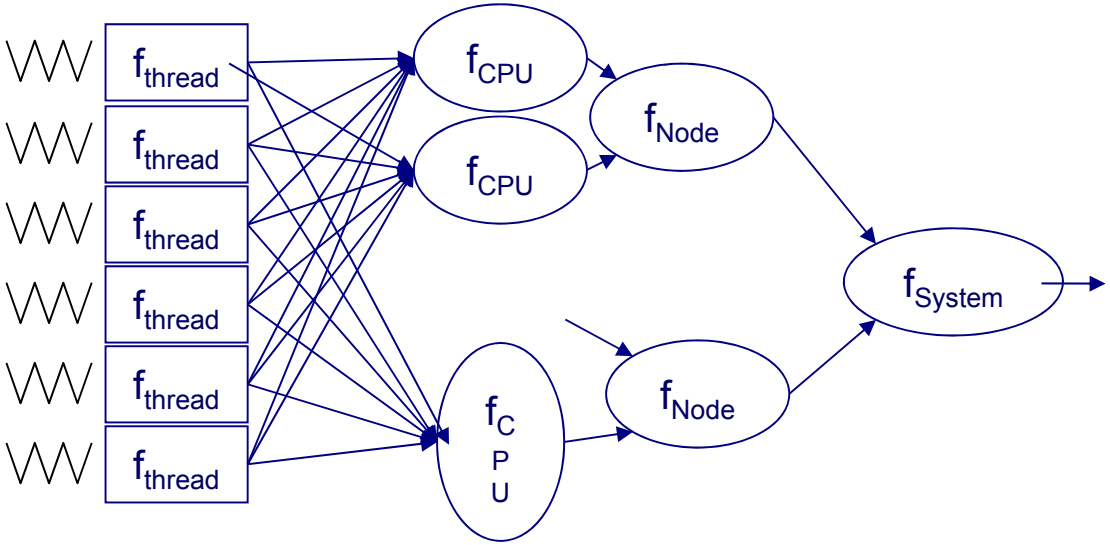
Semantic model: process model view

- Semantic value: S(t)
- $S = f_{comp2} \circ f_{comp1} \circ f_{Workload} \circ f_{Application} \circ f_{task} \circ S_{thread}$
- Semantic functions
  - f_{comp2}, f_{comp1}: sign, mod, div, in range, select range
  - $f_{Application}$, $f_{Workload}$: add, average, max, select
  - f_{task} : add, average, max, select
  - S_{thread}: in state, useful, given state,
  - last event value,
  - next event value,
  - average next event value
  - interval between events, …



Semantic module: resource view

- $Sf_{resource} = f_{comp2} \circ f_{comp1} \circ f_{System} \circ f_{Node} \circ f_{CPU} \circ S_{thread}$
- Semantic functions
  - f_{System} : add, average, max, select
  - f_{Node} : add, average, max, select
  - f_{CPU} : active thread, select
  - S_{thread}: in state, useful, given state, next event value, thread_id



Analysis Module

Tables

- Single flexible quantitative analysis mechanism
- Let
  - cw_1 and cw_2 two views we will call control views
  - dw a view we will call data window

For each window w

$$S_{th}^{w}(t) = S_{th}^{w}(i), t \in \left[t_i^w, t_{i+1}^w\right)$$

- For each control window we define a set of bins

$$bin_j^{cw} = \left[range_j^{cw}, range_{j+1}^{cw}\right) \qquad range_{j+1}^{cw} = range_j^{cw} + delta^{cw}$$

- And the discriminator functions

$$\delta_j^{cw}(t) = ((S^{cw}(t) \in bin_j^{cw})?1:0)$$

$$\delta_{j,k}(t) = \delta_j^{cw_1}(t) * \delta_k^{cw_2}(t)$$

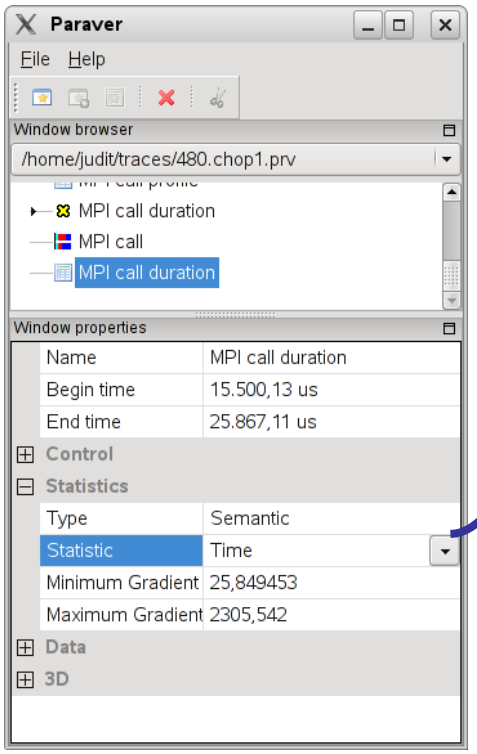
Identify regions with cw's within the (j,k) bin

- The 3D analysis module computes a cube (or plane in the case of 2D) of statistics

$$M(thread, j, k) = statistic(S_{th}^{dw}(t) * \delta_{th,j,k}(t))$$

- Where the statistic can represent the average value, the number of intervals,….

2D analysis module



- Time
- % Time
- % Time Not Zero
- % Window Time
- # Bursts
- % # Bursts
- Integral
- Average value
- Maximum
- Average Burst Time
- Stdev Burst Time
- Average per Burst
- Average value != 0
- # Bursts != 0
- Sum bursts

$$Time(th,j,k)=\int_{t_{start}}^{t_{end}}\delta_{th,j,k}(t)dt$$

$$\%Time(th,j,k)=\frac{\int_{t_{start}}^{t_{end}}\delta_{th,j,k}(t)dt}{t_{end}-t_{start}}$$

$$\%TimeNotZero(th,j,k)=\frac{\int_{t_{start}}^{t_{end}}\delta_{th,j,k}(t)dt}{\sum_{j}\int_{t_{sctart}}^{t_{end}}\delta_{tn,j,k}(t)dt}$$

$$NumBurst(th,j,k)=i_{end}-i_{start}+1$$

$$i_{start}=\min(i)\,|\,t_i>t_{start},i_{end}=\max(i)\,|\,t_i<t_{end}$$

$$Integral(th,j,k)=\int_{t_{start}}^{t_{end}}S_{th}^{dw}(t)\delta_{th,j,k}(t)dt$$

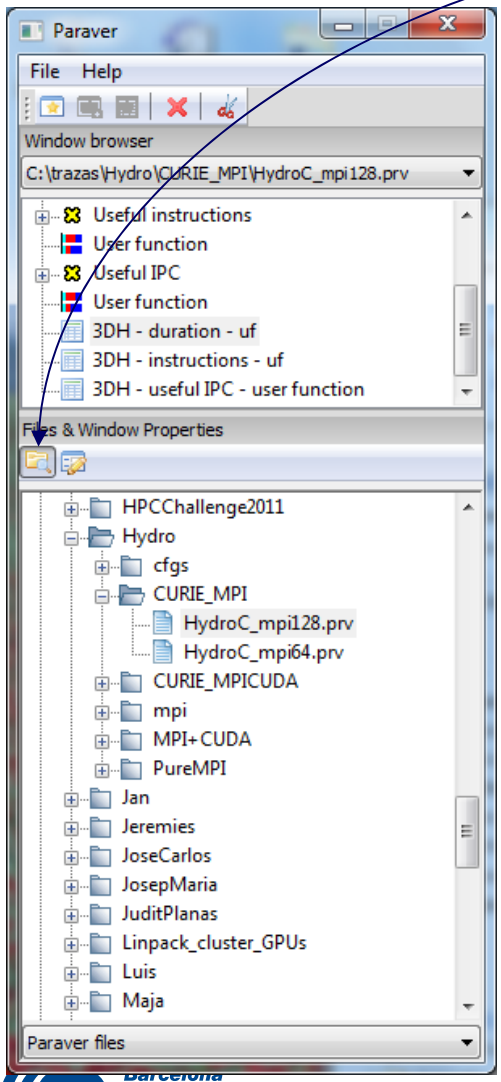
$$Average(th,j,k)=\frac{\int_{t_{start}}^{t_{end}}S_{th}^{dw}(t)\delta_{th,j,k}(t)dt}{\int_{t_{start}}^{t_{end}}\delta_{th,j,k}(t)dt}$$

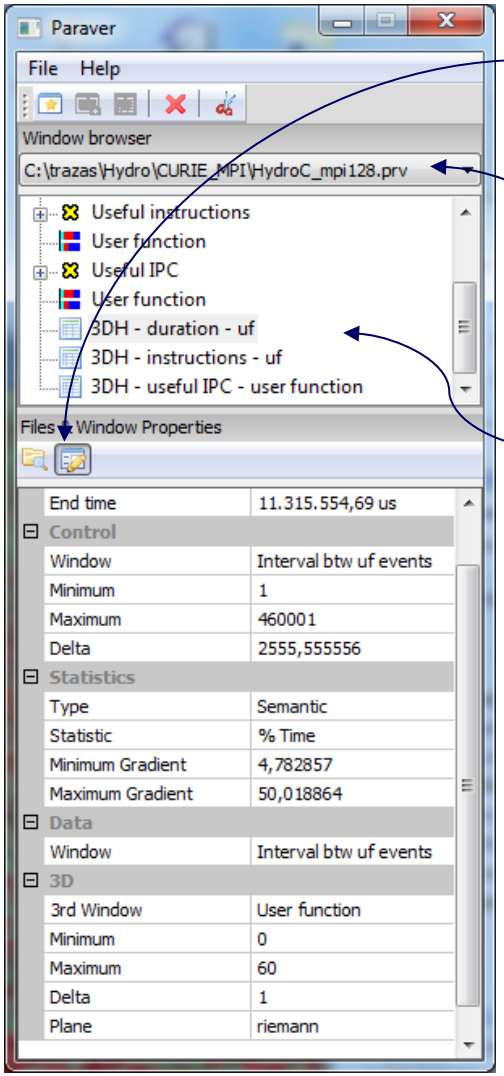
$$Maximum(th,j,k)=\max(S_{th}^{dw}(t)\delta_{th,j,k}(t)),t=[t_{start},t_{end})$$

$$SumBurst(th,j,k)=\sum_{i_{start}}^{i_{end}}S_{th}^{dw}(i)\delta_{th,j,k}(t_i),i_{start}=\min(i)\,|\,t_i>t_{start},i_{end}=\max(i)\,|\,t_i<t_{end}$$

How to …

Main Paraver window





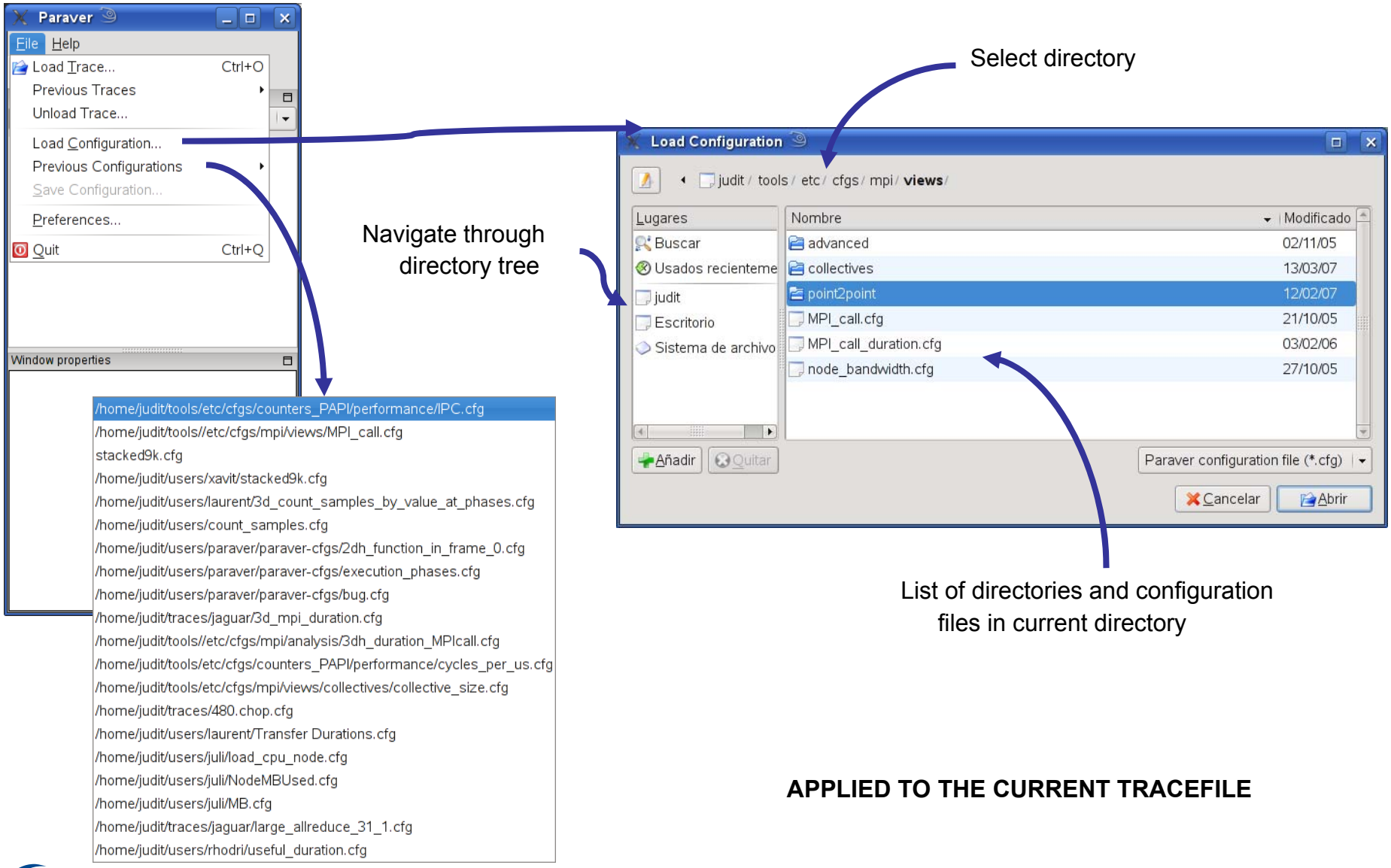
Select to browse in lower panel for traces or cfgs

Select to browse characteristics of active view or table

Active trace

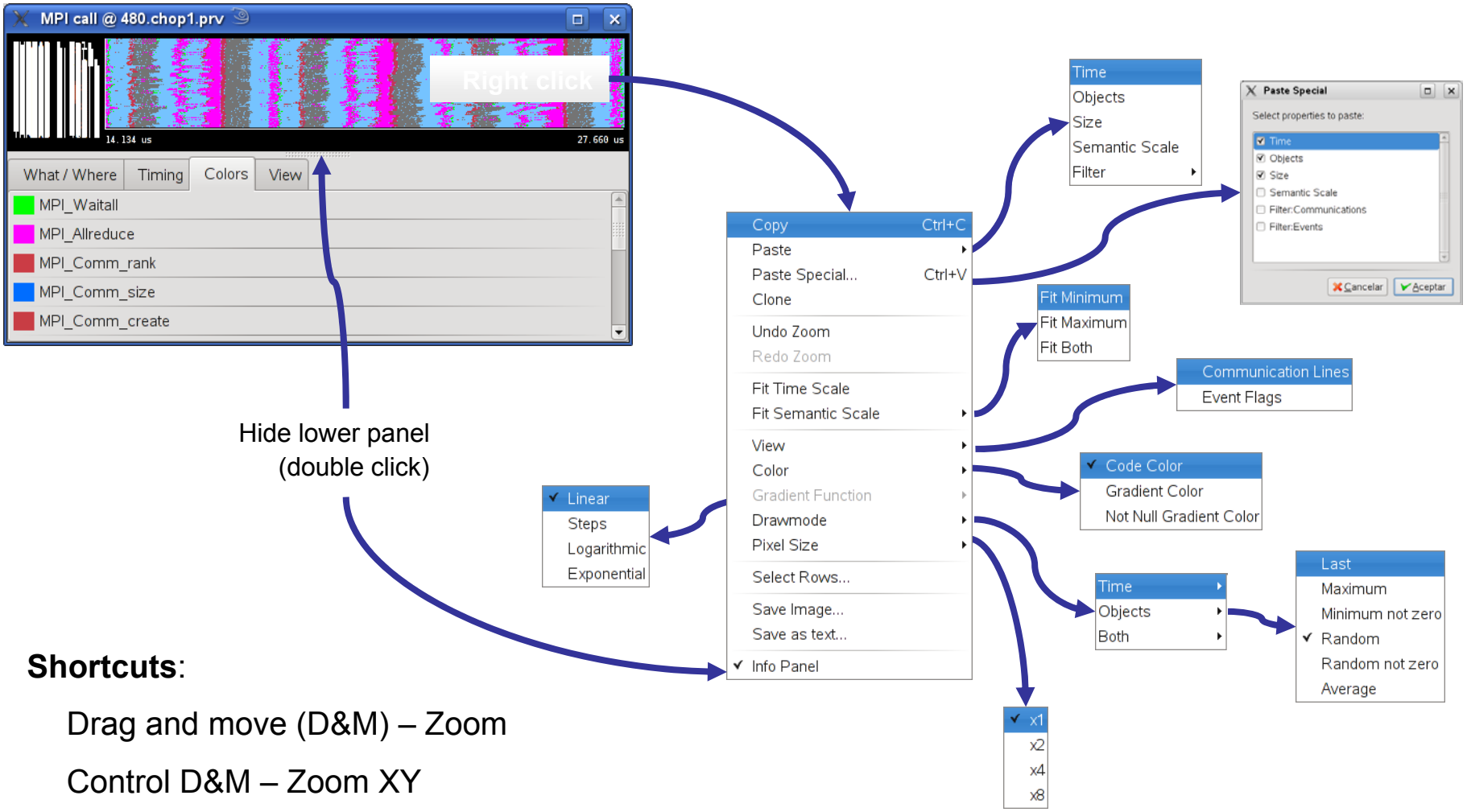
Available views and tables
Active view or table highlighted

Load configuration files



APPLIED TO THE CURRENT TRACEFILE

Navigation



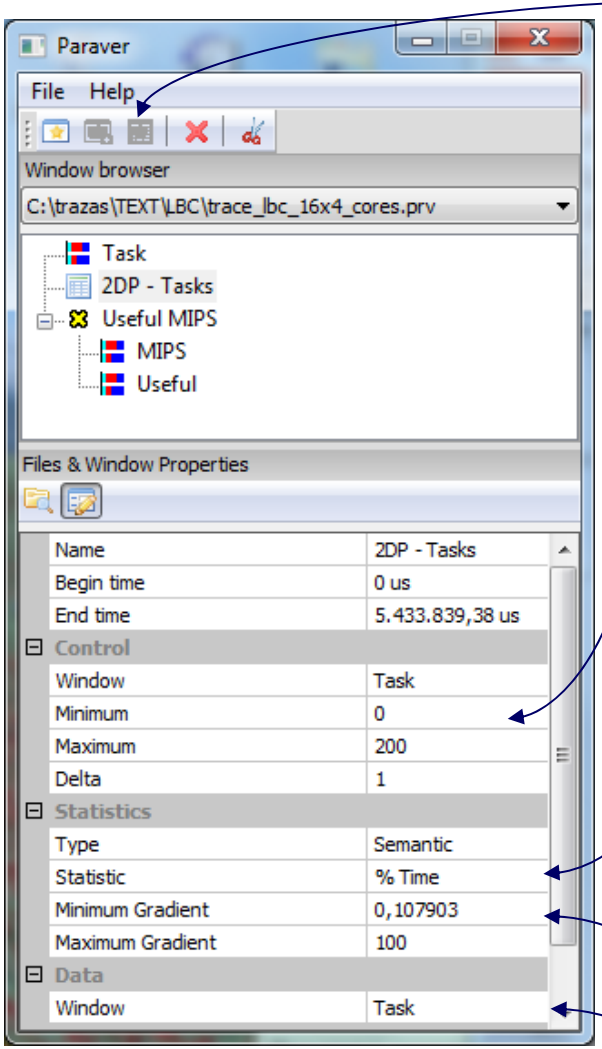
Shortcuts:

Drag and move (D&M) – Zoom

Control D&M – Zoom XY

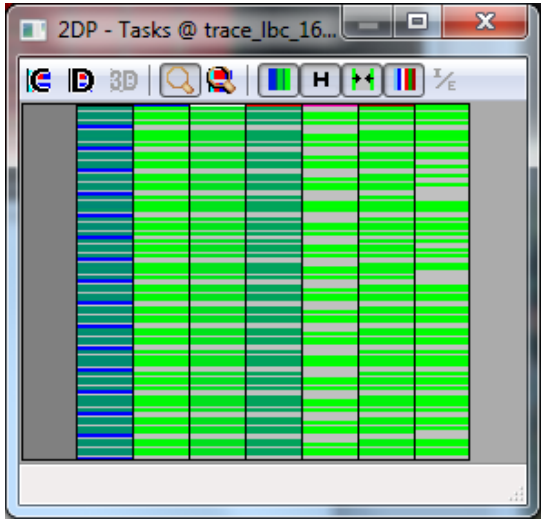
Shift D&M – Timing

How to generate table and change statistic



To generate table: click button and select region of the window whose values will determine the columns of the table

Range and bin width (delta) represented by each column. By default is automatically selected, but can be manually changed



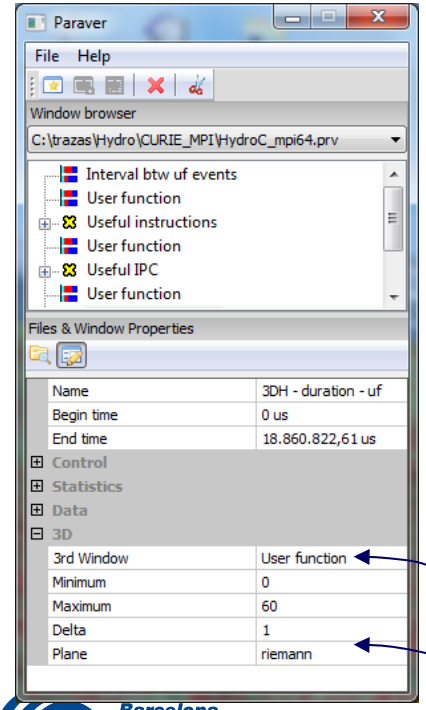
Selection of statistic to appear in each cell

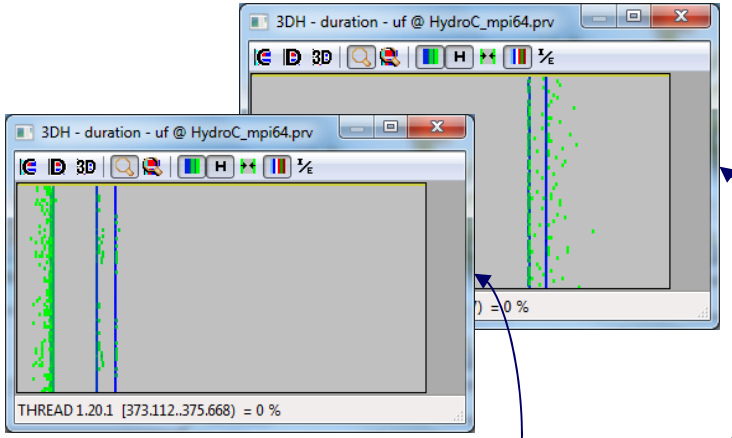
Cell coloring gradient control

Window used to compute statistic (only used by some statistics)

3D tables

- One additional dimension
  - One plane per value of a 3D control window
- Useful to categorize histograms
  - i.e. histogram of duration of specific user function

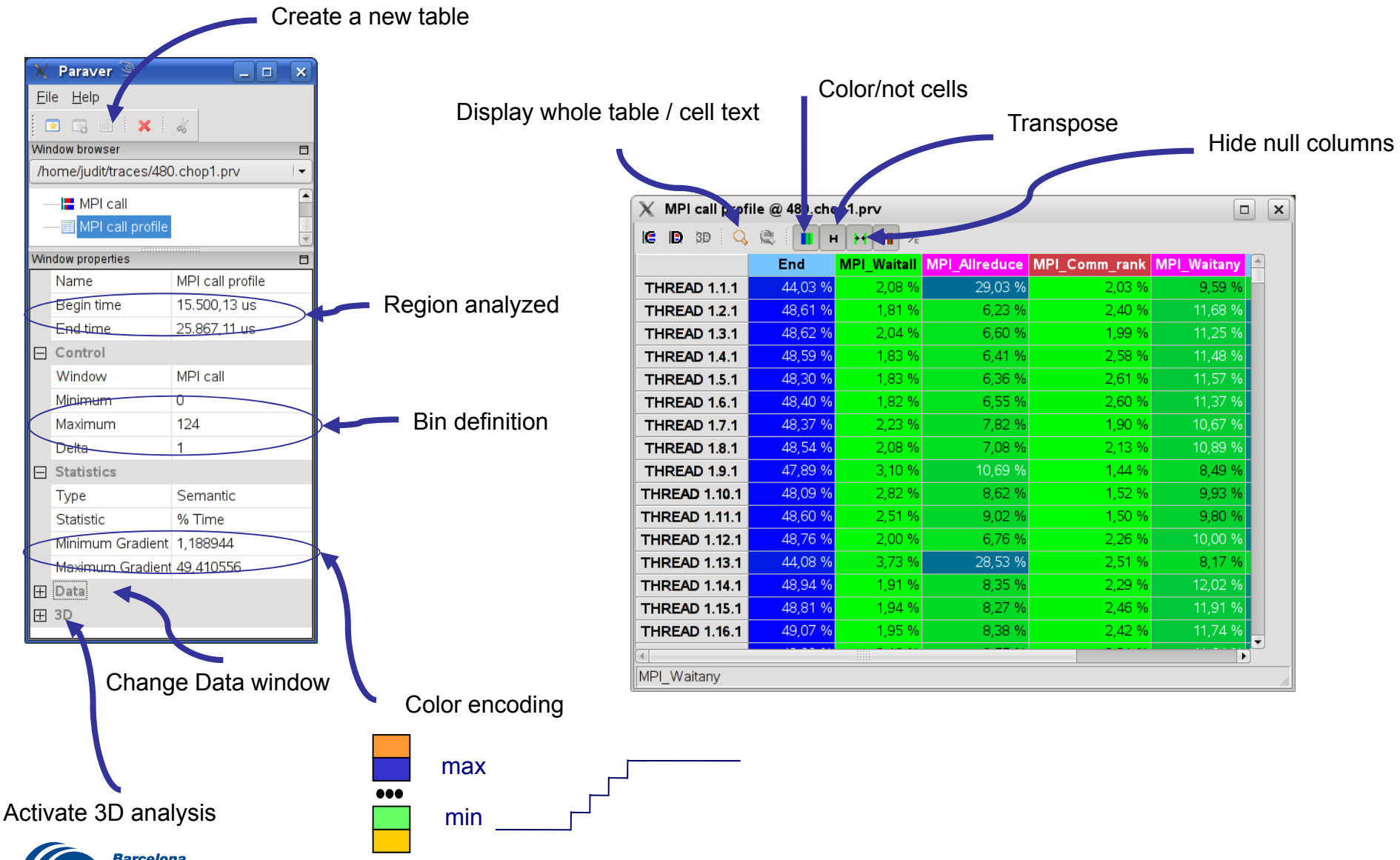




3D control window: determines planes

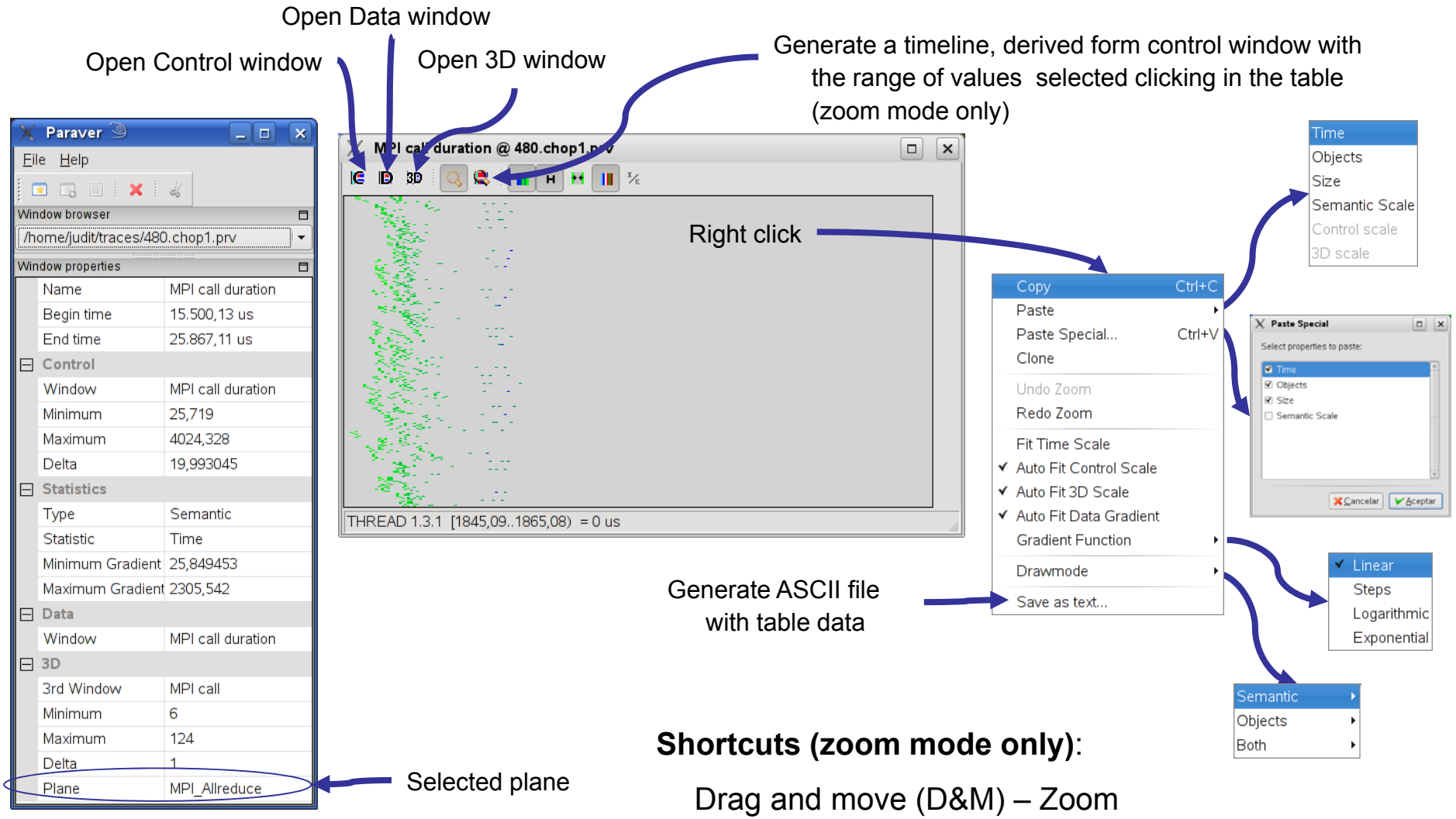
Actual Plane on display

Table information and control



	End	MPI_Waitall	MPI_Allreduce	MPI_Comm_rank	MPI_Waitany
THREAD 1.1.1	44,03 %	2,08 %	29,03 %	2,03 %	9,59 %
THREAD 1.2.1	48,61 %	1,81 %	6,23 %	2,40 %	11,68 %
THREAD 1.3.1	48,62 %	2,04 %	6,60 %	1,99 %	11,25 %
THREAD 1.4.1	48,59 %	1,83 %	6,41 %	2,58 %	11,48 %
THREAD 1.5.1	48,30 %	1,83 %	6,36 %	2,61 %	11,57 %
THREAD 1.6.1	48,40 %	1,82 %	6,55 %	2,60 %	11,37 %
THREAD 1.7.1	48,37 %	2,23 %	7,82 %	1,90 %	10,67 %
THREAD 1.8.1	48,54 %	2,08 %	7,08 %	2,13 %	10,89 %
THREAD 1.9.1	47,89 %	3,10 %	10,69 %	1,44 %	8,49 %
THREAD 1.10.1	48,09 %	2,82 %	8,62 %	1,52 %	9,93 %
THREAD 1.11.1	48,60 %	2,51 %	9,02 %	1,50 %	9,80 %
THREAD 1.12.1	48,76 %	2,00 %	6,76 %	2,26 %	10,00 %
THREAD 1.13.1	44,08 %	3,73 %	28,53 %	2,51 %	8,17 %
THREAD 1.14.1	48,94 %	1,91 %	8,35 %	2,29 %	12,02 %
THREAD 1.15.1	48,81 %	1,94 %	8,27 %	2,46 %	11,91 %
THREAD 1.16.1	49,07 %	1,95 %	8,38 %	2,42 %	11,74 %

Table information and control



Shortcuts (zoom mode only):

Drag and move (D&M) – Zoom

Control D&M – Zoom XY