www.bsc.es

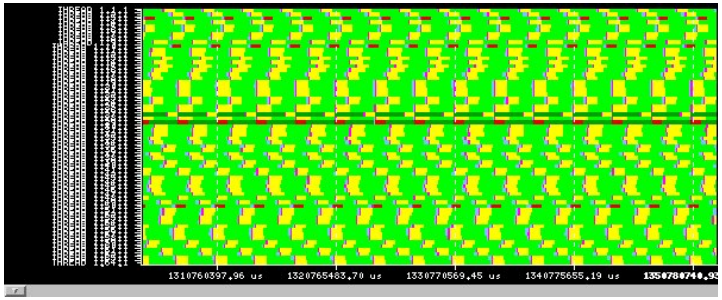
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Understanding applications performance with Paraver

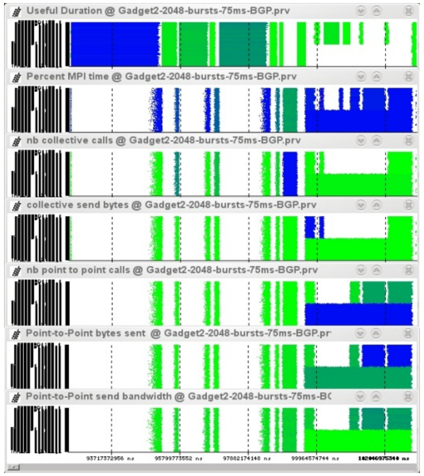
Judit Gimenez
BSC

PATC Parallel Programing Workshop
Oct 14-18 2013

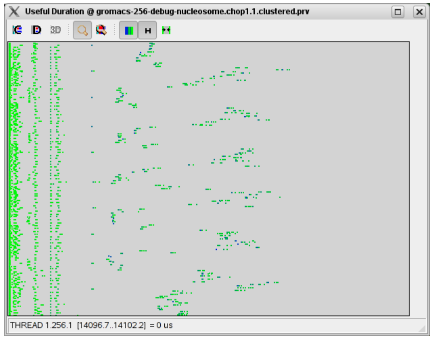
Our Tools



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



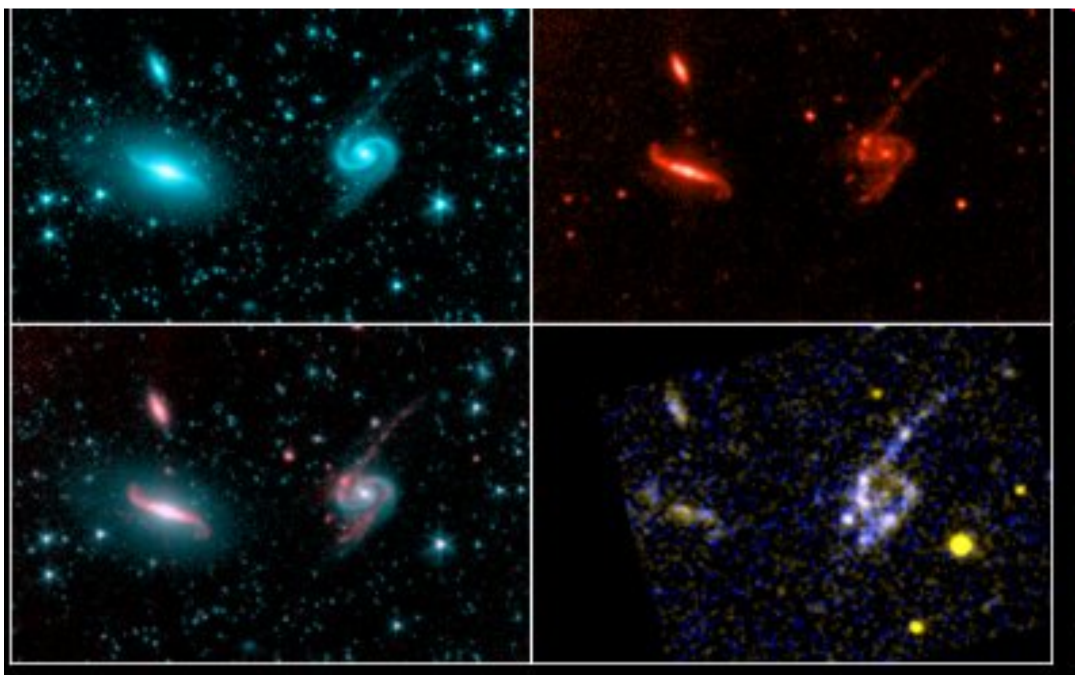
- Core tools:
  - Paraver (paramedir) – offline trace analysis
  - Dimemas – message passing simulator
  - Extrae – instrumentation

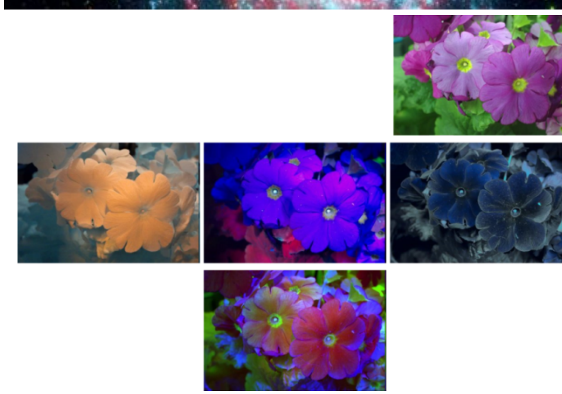


- Focus
  - Detail, flexibility, intelligence

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

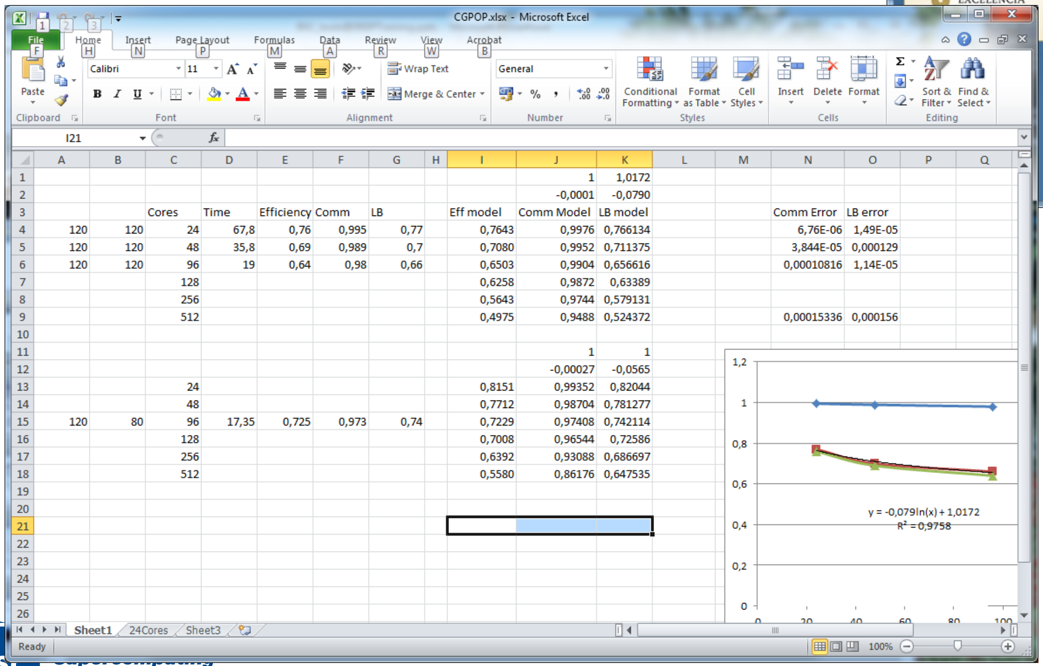




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Spreadsheets and browsers

- Display, manipulate data
  - Dynamic content
  - User defined operations



A “different” view on performance analysis and tools

- **Behavioral structure** vs. syntactic structure
  - Algorithmic and performance
  - In space and time

- **Variability**
  - Multimodal distributions
  - Variability + synchronization → critical non linear effects

- **Flexibility** to let analyst navigate the captured data and gain as much **insight** as possible from as **few application runs** as possible.

“That what is simple is rarely understood”
my iPads Shangai cookies

www.bsc.es

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Paraver

PATC Parallel Programing Workshop
Oct 14-18 2013

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**

Trace

- Sequence of time stamped records
  - Punctual events
    - Something happened: when and where (object/entity: … thread)
    - One record per specific information (encoded as a type and a value)
      - About the event
      - About the interval from the previous event till this one (i.e. hardware counts,…)
  - Relations between objects (… communications)
    - Source and sink
    - Attributes (… size, tag)
  - Separate numeric (.prv) and symbolic (.pcf) files
- Only information derived from captured events and data can be reported.
  - Trivial but … often forgotten

Extrae

- Major BSC instrumentation package

- When / where
  - Parallel programming model runtime
    - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, MIC…
    - API entry/exit, OpenMP outlined routines
  - Selected user functions
  - Periodic samples
  - User events

- Additional information
  - Counters
    - PAPI
    - Network counters
    - OS counters
  - Link to source code
    - Callstack

How does Extrae intercept your app?

- LD_PRELOAD
  - Works on production binaries
  - Specific library for each combination of runtimes
  - Does not require knowledge on the application

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

- Dynamic instrumentation
  - Works on production binaries
  - Just specify functions to be instrumented.

Based on DynInst
U.Wisconsin/U.Maryland

- Other possibilities
  - Link instrumentation library statically (i.e., PMPI @ BG/Q, …)
  - OmpSs (instrumentation calls injected by compiler + linked to library)

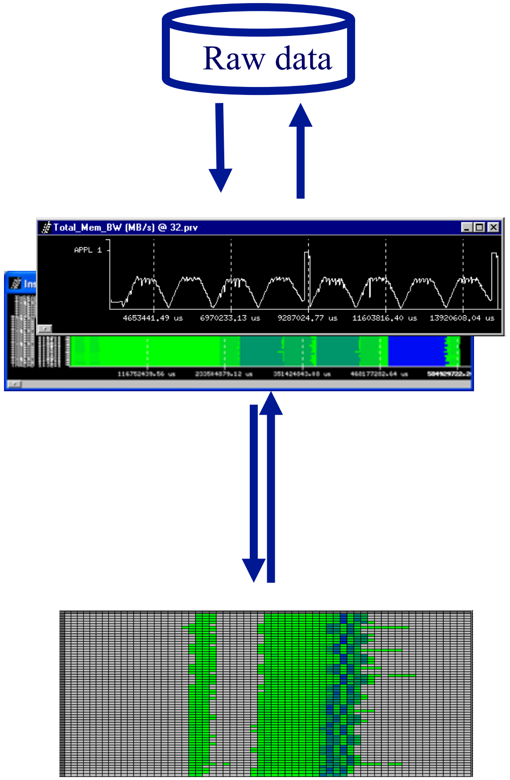
How to use Extrae?

- Adapt job submission script
  - Specify LD_PRELOAD library and xml instrumentation control file
- Specify the data to be captured in the .xml instrumentation control file
- Run and get the trace …

Extrae 2.3.4 User's Guide available in
http://www.bsc.es/computer-sciences/performance-tools/documentation

Default control files and further examples within installation in
$EXTRAE_HOME/share/example

Paraver – Performance data browser



Timelines

2/3D tables (Statistics)

Trace visualization/analysis

+ trace manipulation

Goal = Flexibility

No semantics

Programmable

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)
  - Language to describe how to compute such functions of time (GUI)
    - Basic operators (from) trace records
    - Ways of combining them

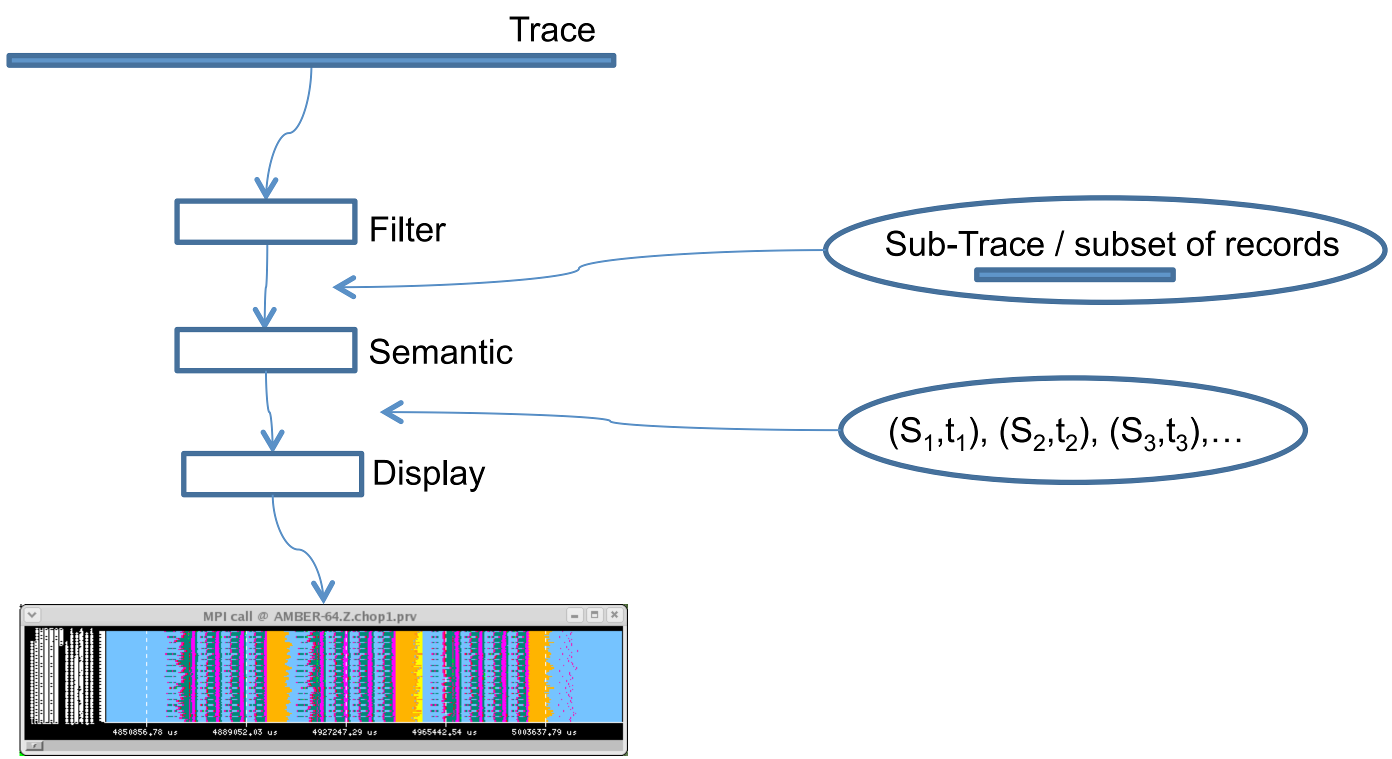
- Those functions of time can be rendered into a 2D image
  - Timeline

- Statistics can be computed for each possible value or range of values of that function of time
  - Tables: profiles and histograms

Paraver mathematical foundation

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

Function of time   Series of values



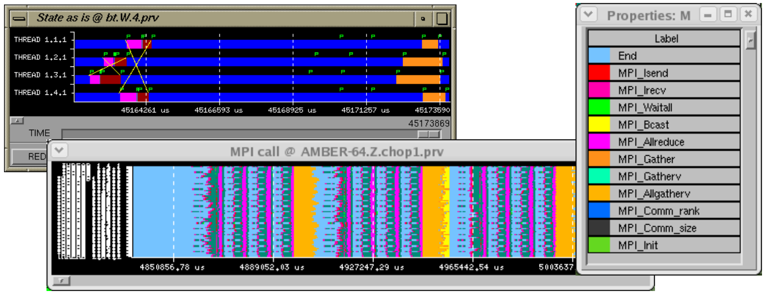
Timelines

- 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

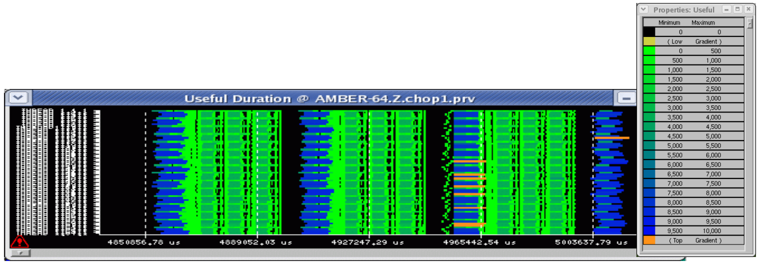
Timelines

- Representation
  - Function of time

  - Colour encoding



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



- Non linear rendering to address scalability

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

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

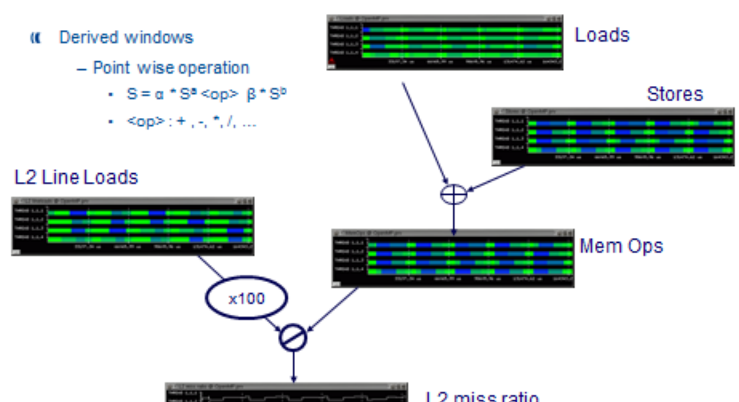
- From communication records to functions of time
  - Send Bytes $s(t)=\sum_j Sz(C_j),j \mid (T_S(C_j)<t)\wedge(T_R(Cj)>t)\wedge(Dir(Cj)==send)$
  - Send Bandwidth $s(t)=\sum_j \frac{Sz(C_j)}{T_R(C_j)-T_S(C_j)},j \mid (T_S(C_j)<t)\wedge(T_R(Cj)>t)\wedge(Dir(Cj)==send)$
  - Msgs in transit $s(t)=\sum_j sign(j),j \mid (T_S(C_j)<t)\wedge(T_R(Cj)>t)\wedge(Dir(Cj)==send)$
  - 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

- $S'(t) = f(S(t))$ $S' = f \circ S$
  - 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^a <op> β * S^b
    - <op> : + , -, *, /, …

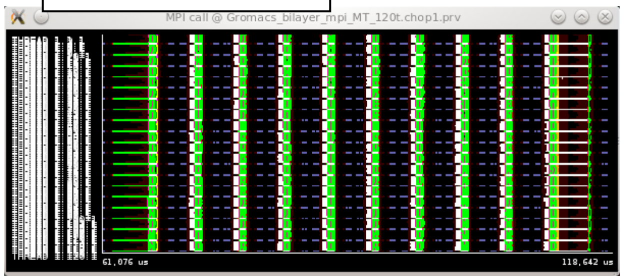


See slides at end of presentation for details

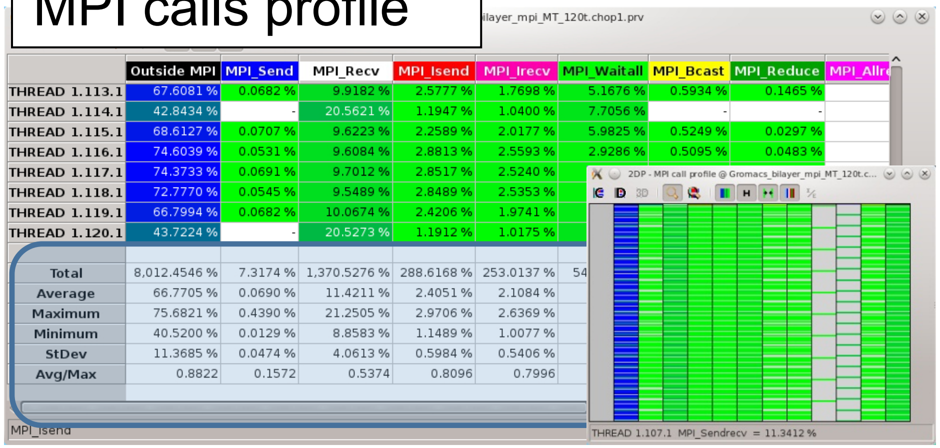
Tables: Profiles, histograms, correlations

From timelines to tables

MPI calls

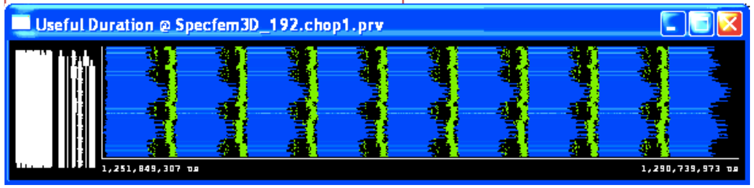


MPI calls profile

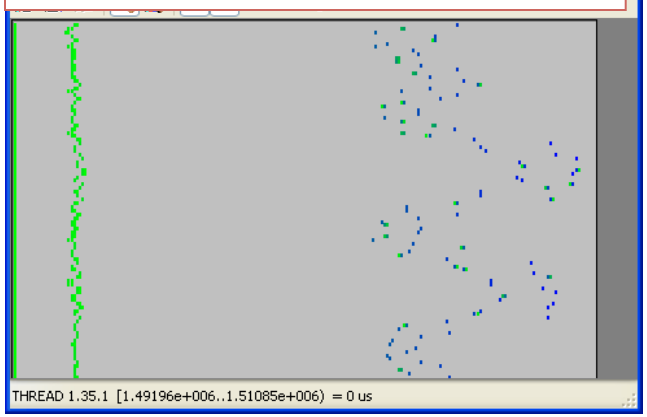


	Outside MPI	MPI_Send	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Bcast	MPI_Reduce	MPI_Allre
THREAD 1.113.1	67.6081 %	0.0682 %	9.9182 %	2.5777 %	1.7698 %	5.1676 %	0.5934 %	0.1465 %	
THREAD 1.114.1	42.8434 %	-	20.5621 %	1.1947 %	1.0400 %	7.7056 %	-	-	
THREAD 1.115.1	68.6127 %	0.0707 %	9.6223 %	2.2589 %	2.0177 %	5.9825 %	0.5249 %	0.0297 %	
THREAD 1.116.1	74.6039 %	0.0531 %	9.6084 %	2.8813 %	2.5593 %	2.9286 %	0.5095 %	0.0483 %	
THREAD 1.117.1	74.3733 %	0.0691 %	9.7012 %	2.8517 %	2.5240 %				
THREAD 1.118.1	72.7770 %	0.0545 %	9.5489 %	2.8489 %	2.5353 %				
THREAD 1.119.1	66.7994 %	0.0682 %	10.0674 %	2.4206 %	1.9741 %				
THREAD 1.120.1	43.7224 %	-	20.5273 %	1.1912 %	1.0175 %				
Total	8,012.4546 %	7.3174 %	1,370.5276 %	288.6168 %	253.0137 %	54			
Average	66.7705 %	0.0690 %	11.4211 %	2.4051 %	2.1084 %				
Maximum	75.6821 %	0.4390 %	21.2505 %	2.9706 %	2.6369 %				
Minimum	40.5200 %	0.0129 %	8.8583 %	1.1489 %	1.0077 %				
StDev	11.3685 %	0.0474 %	4.0613 %	0.5984 %	0.5406 %				
Avg/Max	0.8822	0.1572	0.5374	0.8096	0.7996				

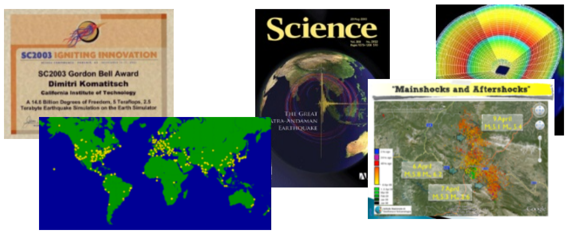
Useful Duration

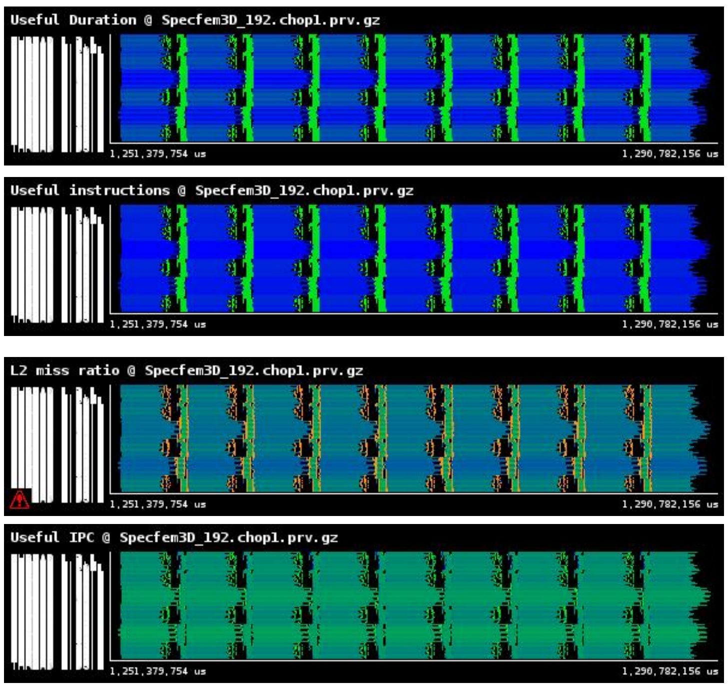


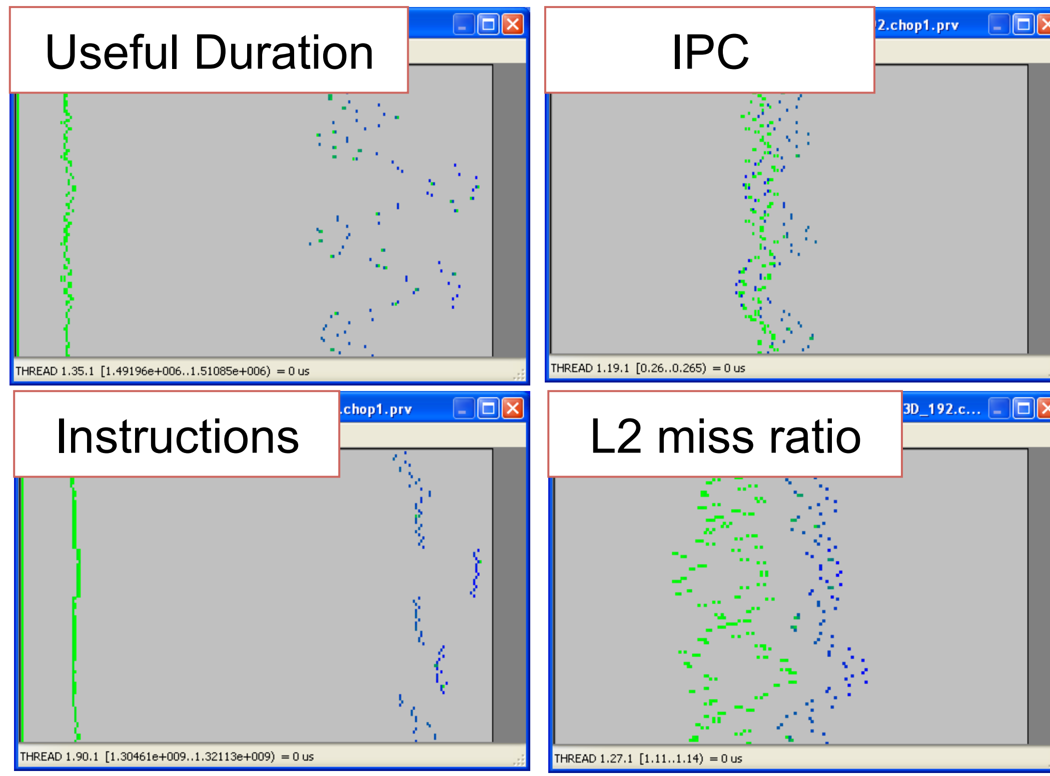
Histogram Useful Duration



Analyzing variability through histograms and timelines



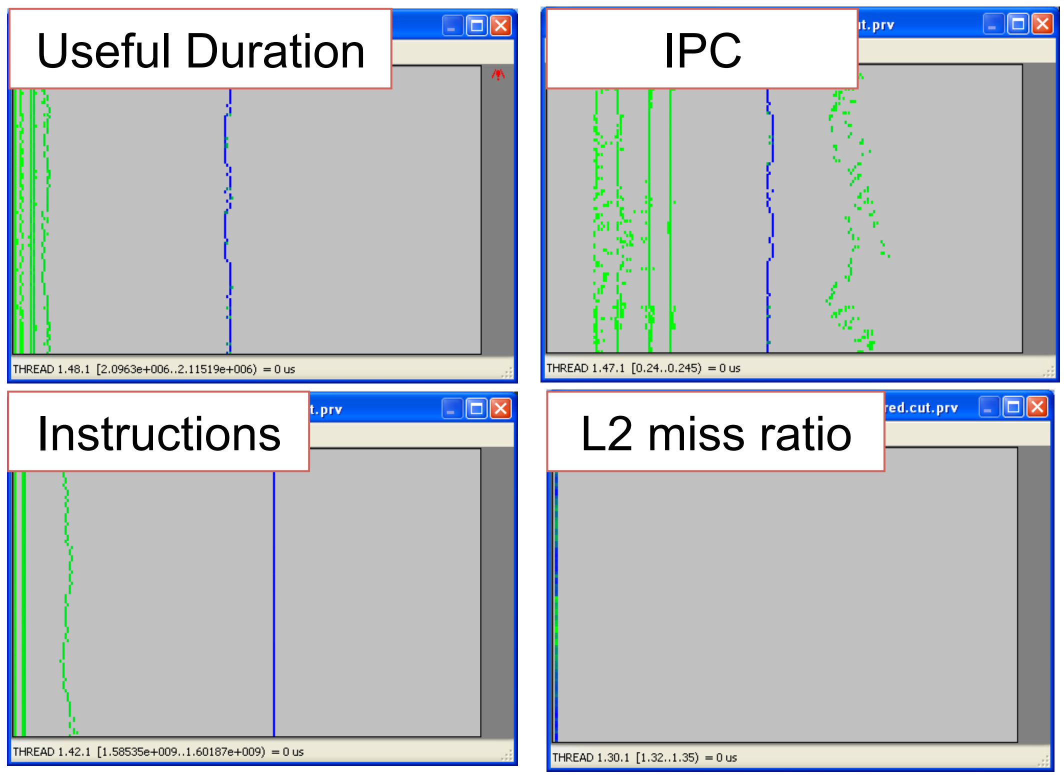




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

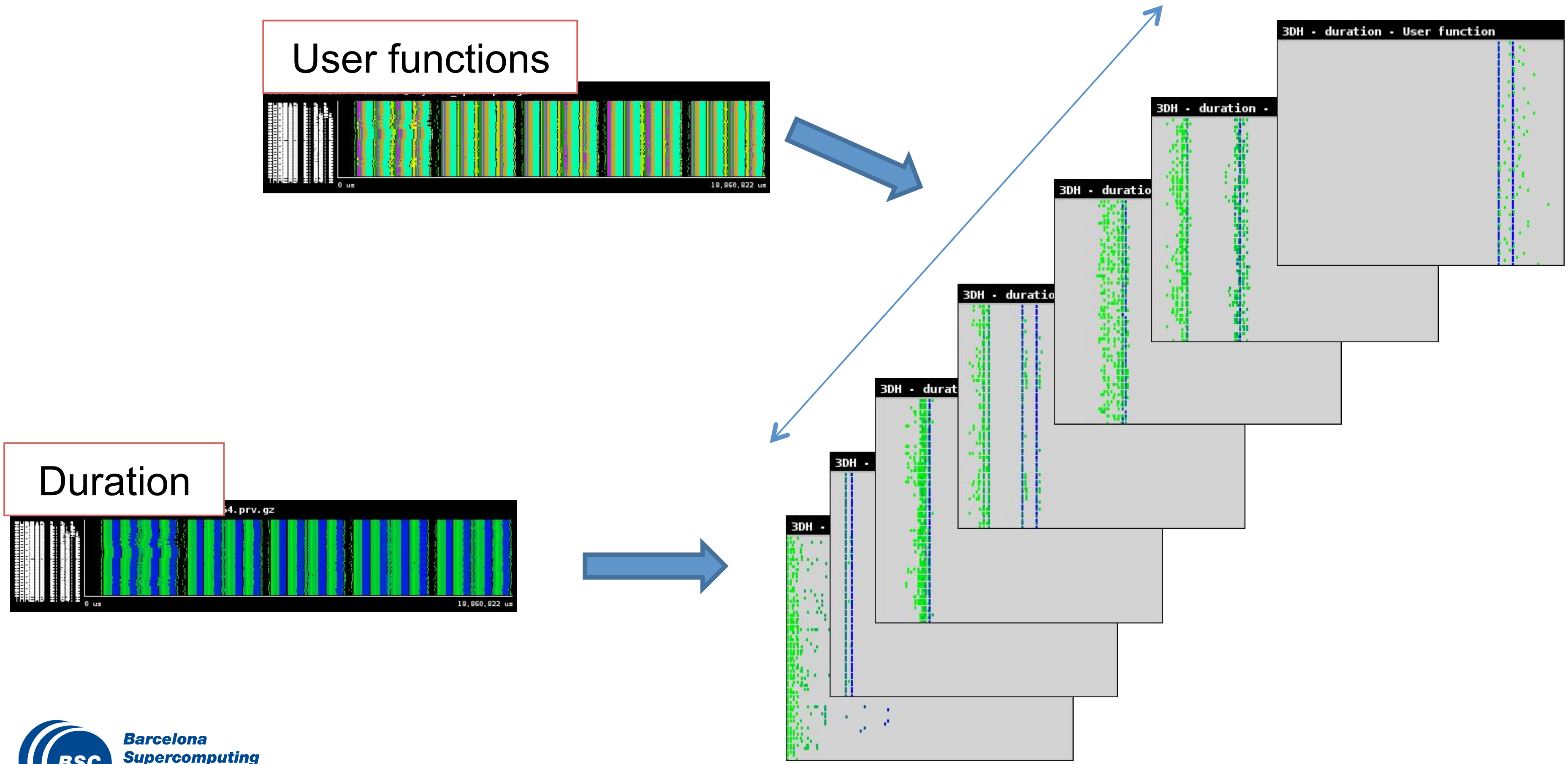
Analyzing variability through histograms and timelines

- By the way: six months later ….



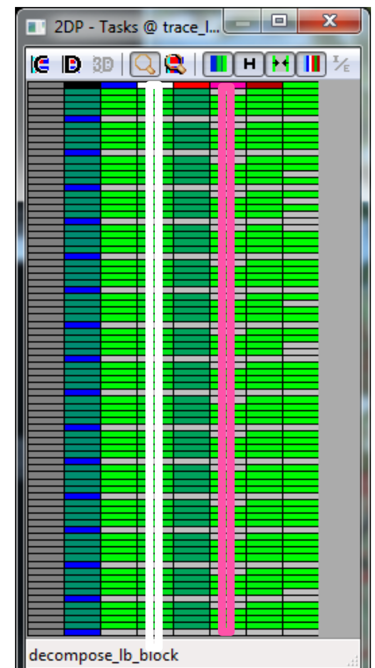
3D Tables

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

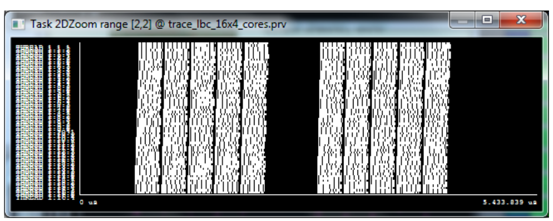


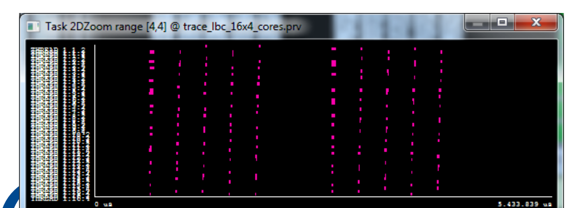
From tables to timelines

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



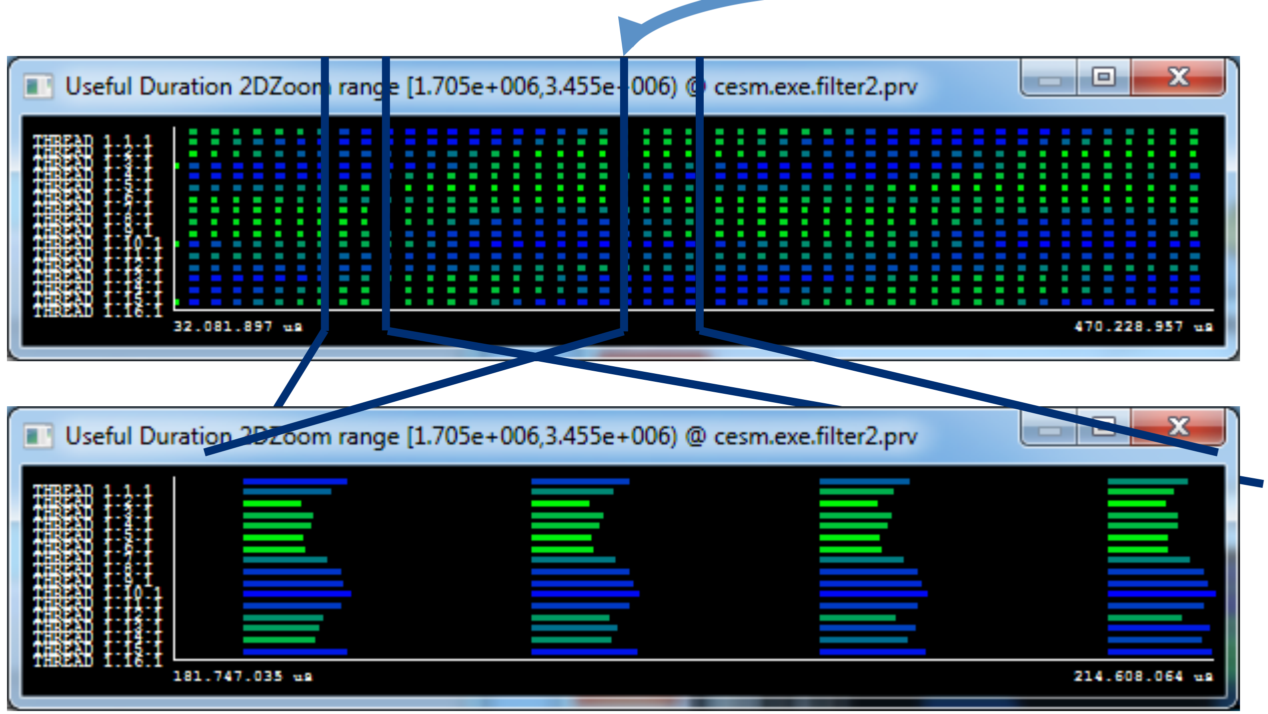
Only showing when a given value happens





Variability … is everywhere

- CESM: 16 processes, 2 simulated days
- Histogram useful computation duration shows high variability
- How is it distributed?
- Dynamic imbalance
  - In space and time
  - Day and night.
  - Season ? ☺



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Other mechanisms integrated in the GUI

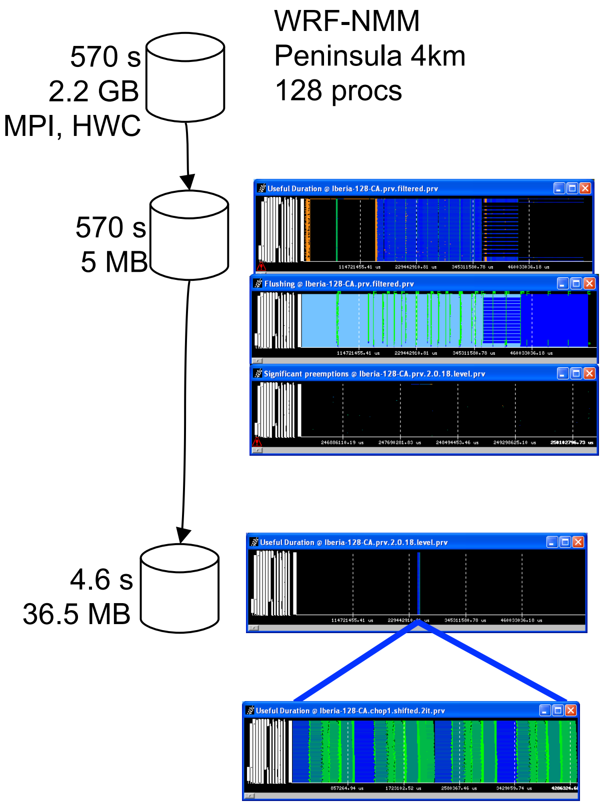
- Trace manipulation
  - Cut
  - Filter

- Performance analytics
  - Clustering
  - Folding
  - Tracking

- Executing external commands and tools
  - BSC Tools
  - Scripts
  - External tools

Trace manipulation

- Data handling/summarization capability
  - Filtering
    - Subset of records in original trace
    - By duration, type, value,…
    - Filtered trace IS a paraver trace and can be analysed with the same cfgs (as long as needed data kept)
  - Cutting
    - All records in a given time interval
    - Only some processes
  - Software counters
    - Summarized values computed from those in the original trace emitted as new even types
    - #MPI calls, total hardware count,…



See slides at end of presentation for details

External commands and tools

- Execute external commands …
  - Predefined: Dimemas, Stats,…
  - User specified binaries or scripts
- … specifying arguments
  - Trace
  - Command specific arguments

- Paramedir
  - Non graphical version of Paraver
  - Reads trace, applies standard cfgs, writes ASCII output (table,…)

- Scripts can use paramedir, Dimemas, clustering … in parametric sweeps, search/optimization loops,…

Executing external commands

Example: basic_analysis.py

```
==============================================================
Analysis of  XXXXX.prv
==============================================================
Timing:
   Elapsed duration              = 3.887  s
   Ideal time                    = 3.223  s
   Compute time                  = 2.637952  s
   MPI time                      = 1.24930663  s

Parallel Efficiency:
   Total Efficiency              = 0.679
       Load Balance              = 0.950
       Micro Load Balance        = 0.859
       Transfer                  = 0.829
       Bweff                     = 0.814
       Leff                      = 1.000

Load balance:
       Time Load Balance         = 0.950
       Instructions Load Balance = 0.990
       Cycles load balance       = 0.950
       IPC Load Balance          = 0.960

Computational analysis:
   Total useful instructions     = 5.399636e+11
   Average useful instructions per process       = 8.436931e+09
   Instructions based microload balance/sync     = 0.924

sequential performance:
   Average MIPS                  = 3199
   Average IPC                   = 1.050
```

```
Basic communication statistics:
   Point to point:
      Average number of calls      = 554.0
      Balance in number of calls   = 1.0
      Average bytes per process    = 728596480.0
      Balance in bytes             = 0.990
   Collectives:
      Num collective calls         = 6.0
      Average bytes per call       = 6.670
      Max bytes per call           = 8.000
      Balance in bytes             = 1.000

Inter - Intra node communication statistics:
   Bytes sent (MB):
      Locally                      = 22229.811
      Remotely                     = 24645.730
   Number of sends:
      Locally                      = 3520.000
      Remotely                     = 6592.000
```

64Proc – 8 MPI by node

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Performance Analytics

PATC Parallel Programing Workshop
Oct 14-18 2013

Performance Analytics

- 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
- Some techniques worked on at BSC
  - Spectral analysis
  - Clustering
  - Folding
  - Simulation (Dimemas)

Spectral analysis

Repetitive behavior

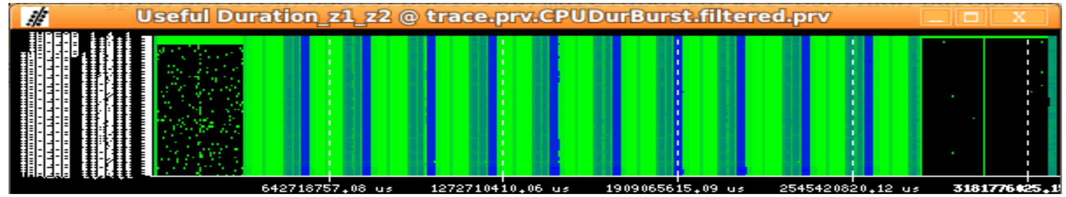
- Applications tend to have Iterative behavior
  - Detailed analysis can be applied to a few such iterations

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

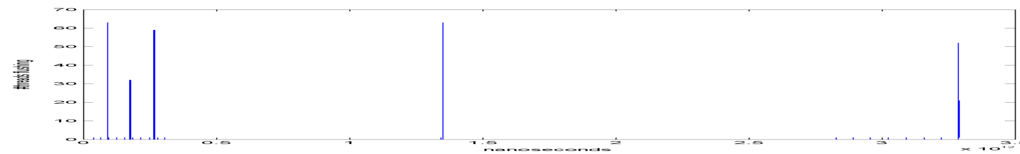
Signal processing applied to performance analysis

- **Techniques**
  - Mathematical morphology
    - clean up perturbed regions
  - Wavelet transform
    - identify coarse regions
  - Spectral analysis
    - detailed periodic pattern

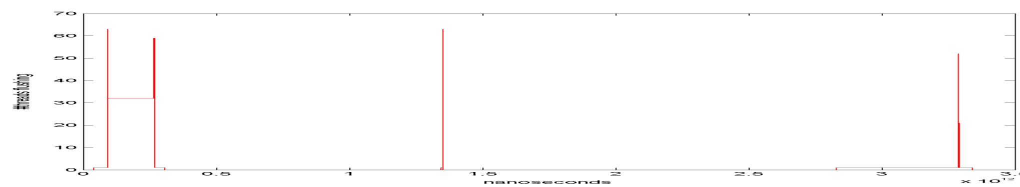
- **Useful**
  - Identify structure (periodicity)
  - Reduce trace sizes
  - Increase precision of profiles (report non perturbed stats)



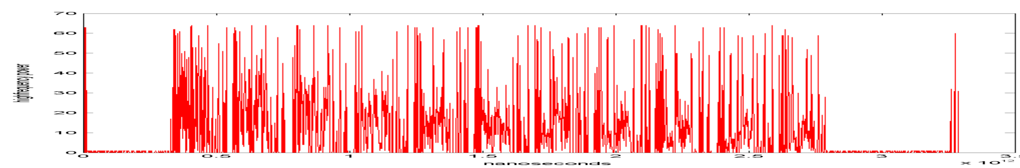
Flushing



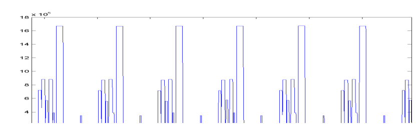
Flushing filtered



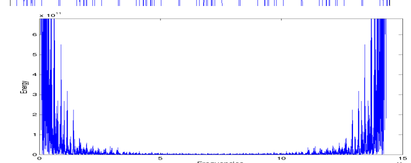
Wavelet High frequency



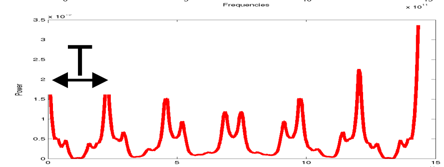
Σ Useful Duration



Spectral density



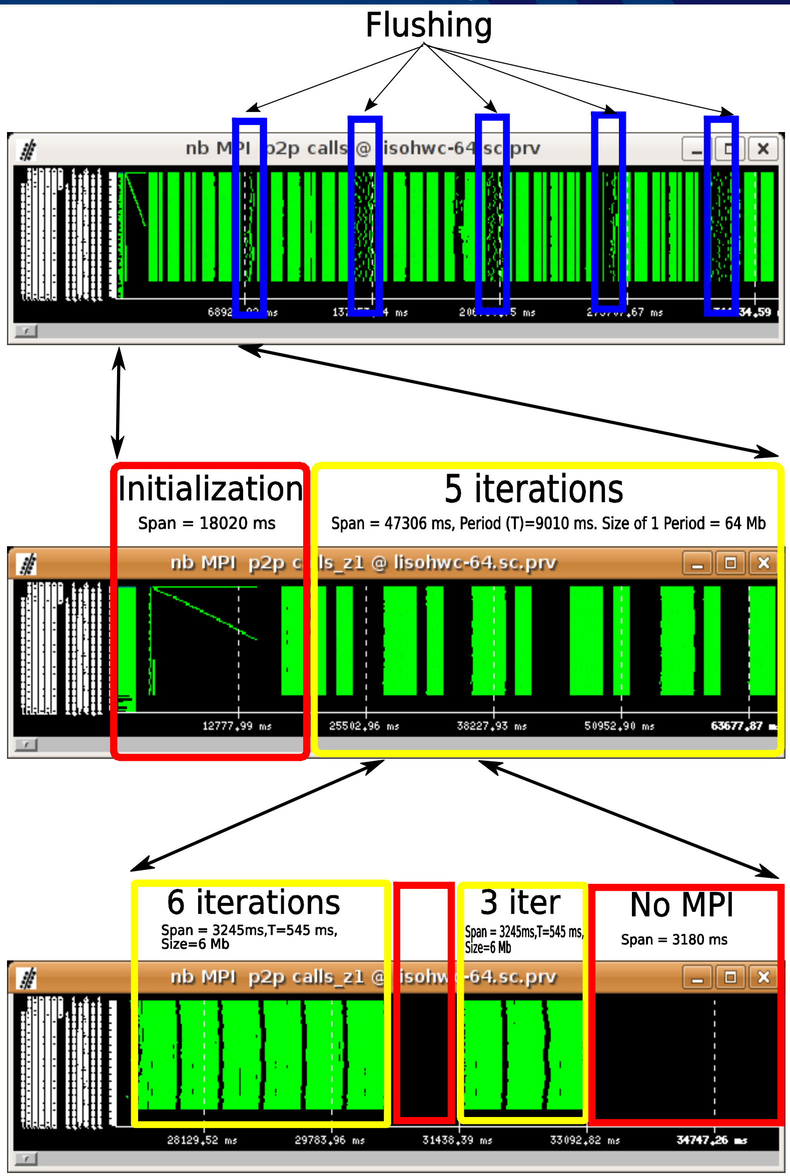
Autocorrelation



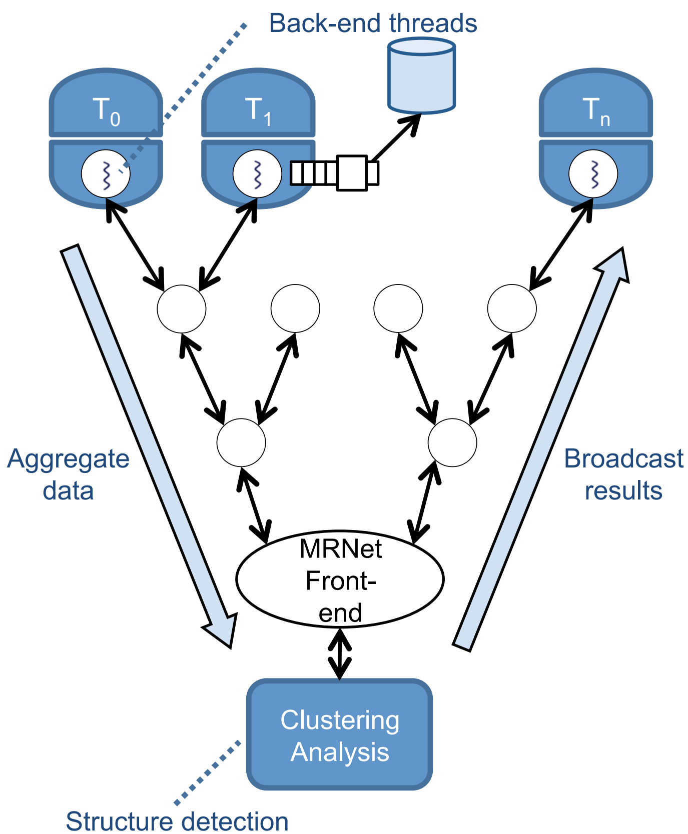
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

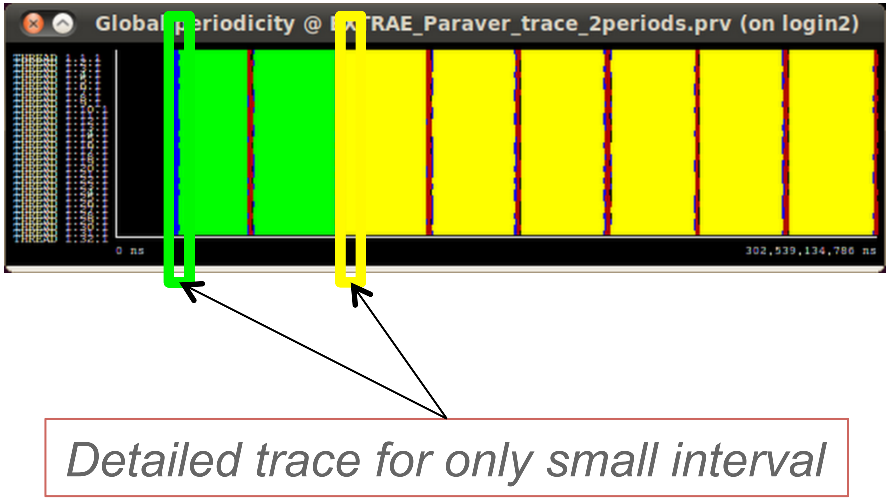
Signal processing applied to performance analysis

- Hierarchical structure identification



Scalability: online automatic interval selection





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

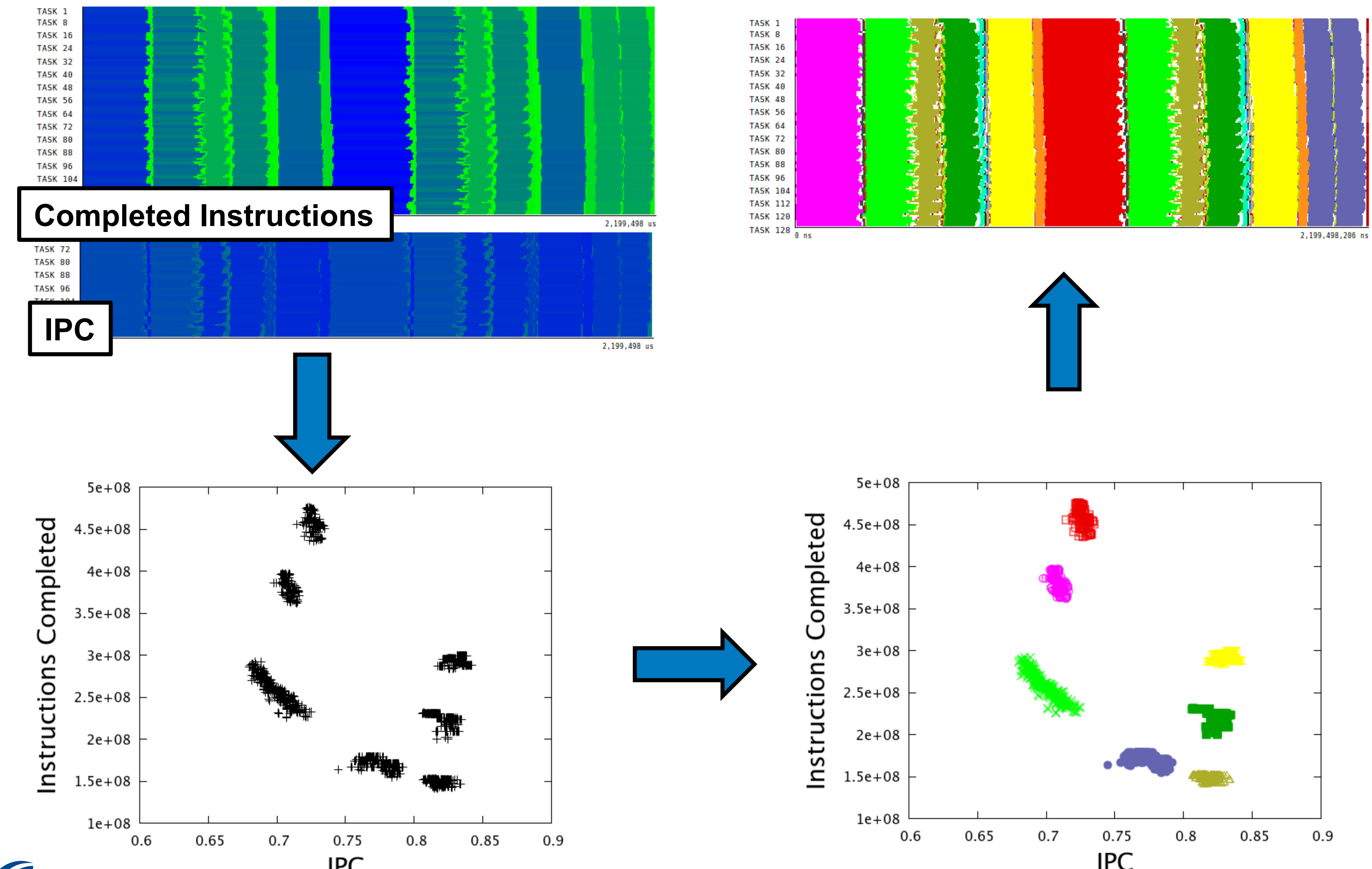
Clustering

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

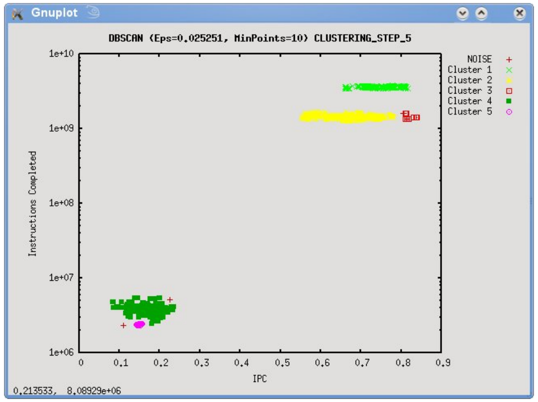
- Scatter plot on some relevant metrics
  - Instructions: idea of computational complexity, computational load imbalance,…
  - IPC: Idea of absolute performance and performance imbalance
  - Automatically Identify clusters

Using Clustering to identify structure

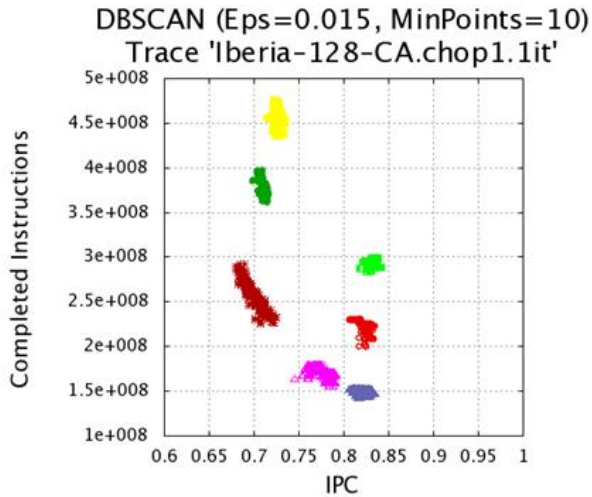


J. Gonzalez et al, "Automatic Detection of Parallel Applications Computation" Phases. (IPDPS 2009)

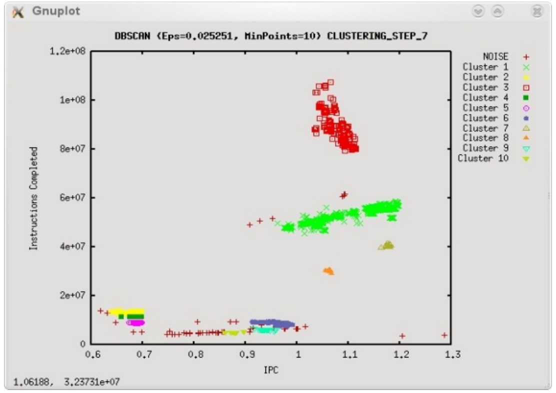
Performance @ serial computation bursts



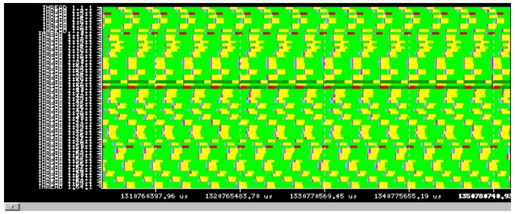
SPECFEM3D

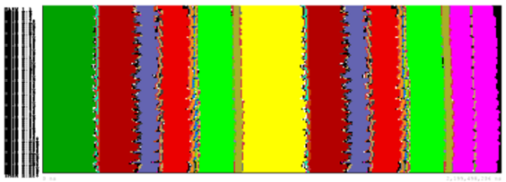


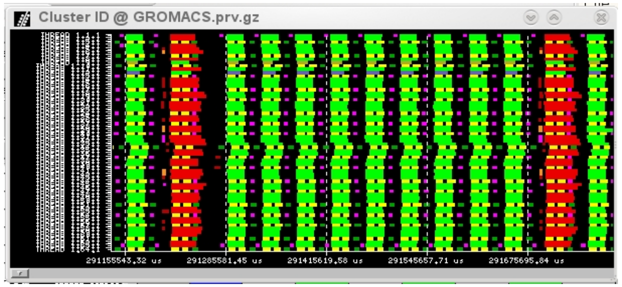
WRF 128 cores



GROMACS







Asynchronous SPMD

Balanced #instr
variability in IPC

SPMD
Repeated substructure

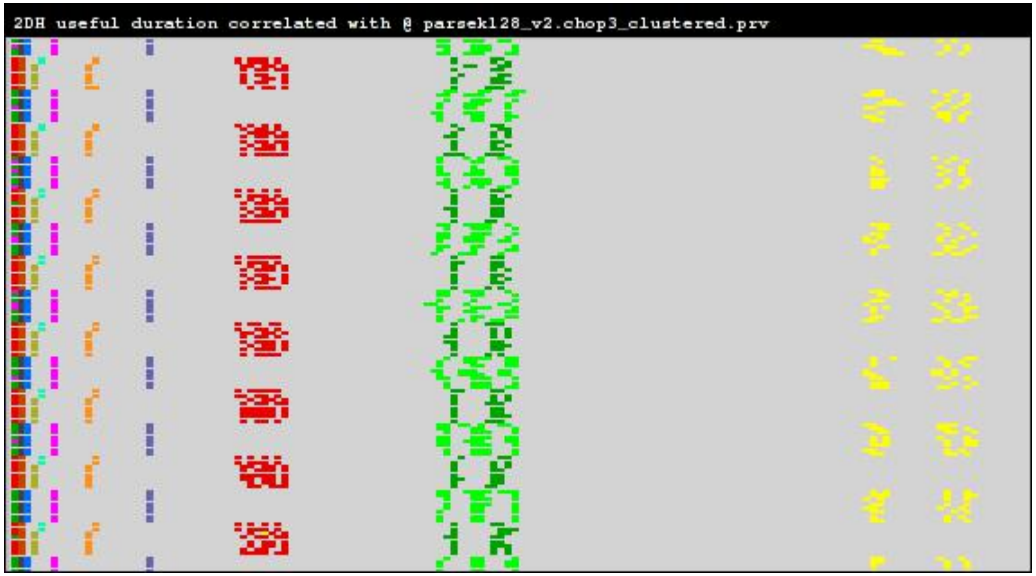
Coupled imbalance

MPMD structure

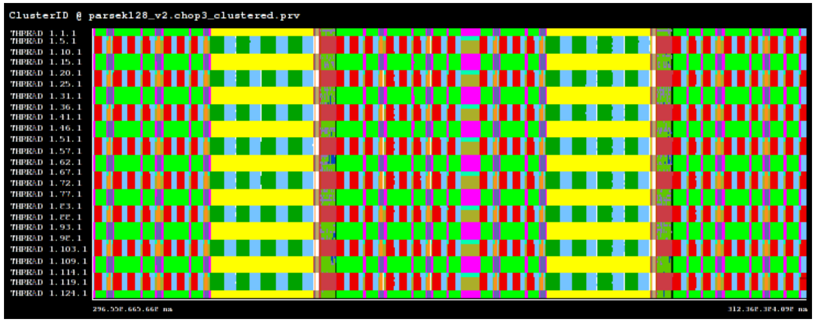
Different coupled
imbalance trends

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

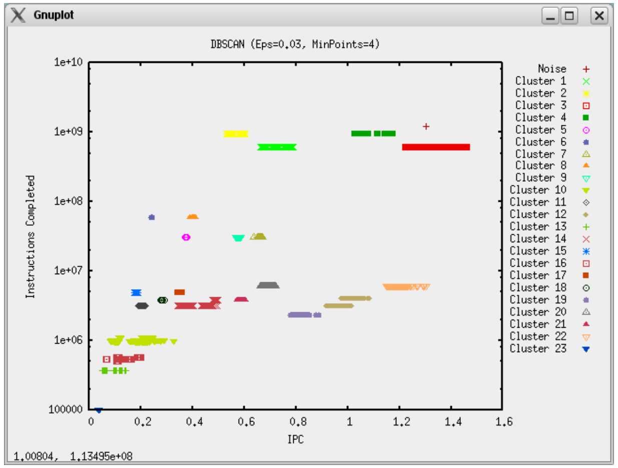
Example PARSEK (DEEP)

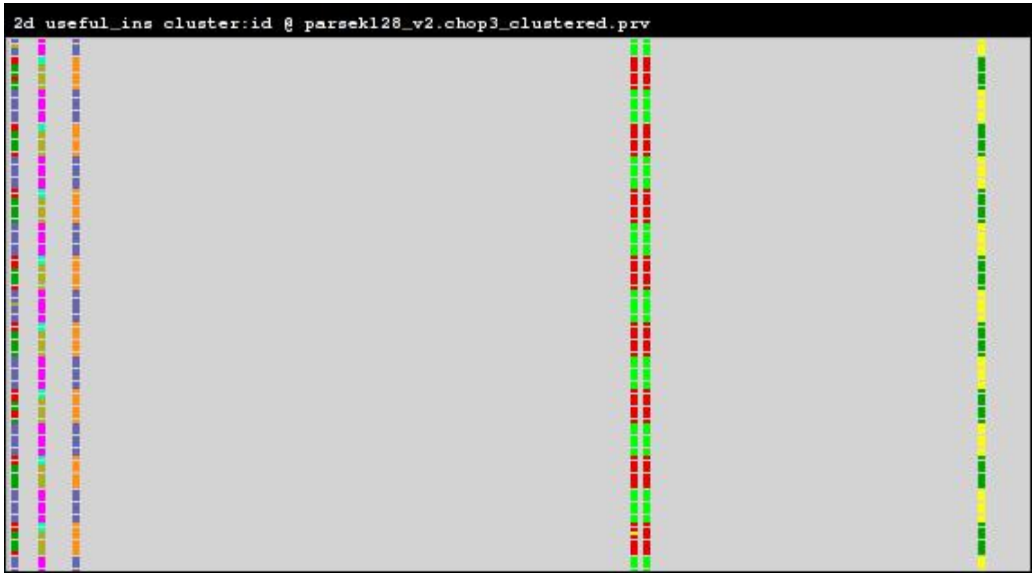


duration vs. cluster

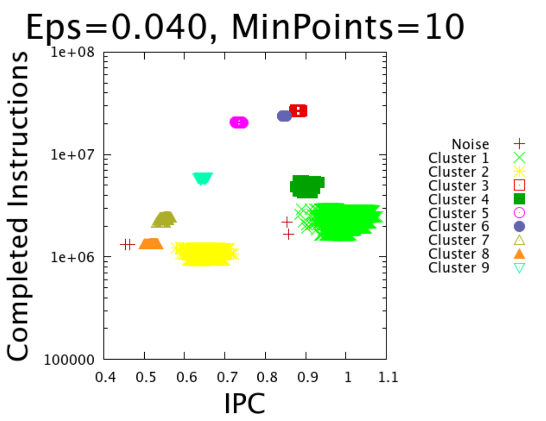


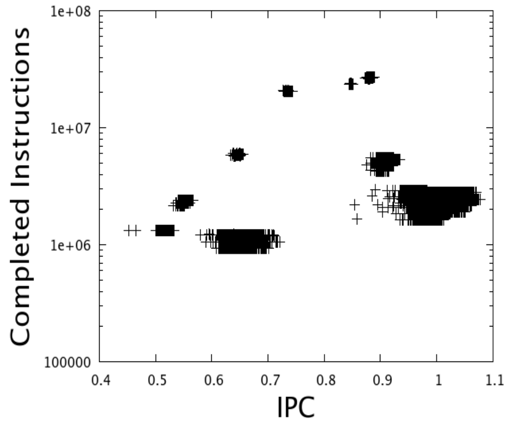
instr. vs. cluster

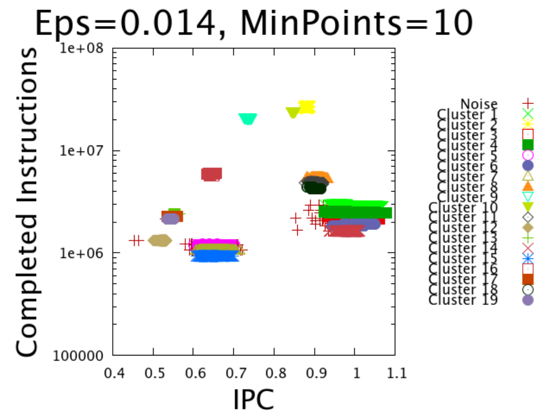


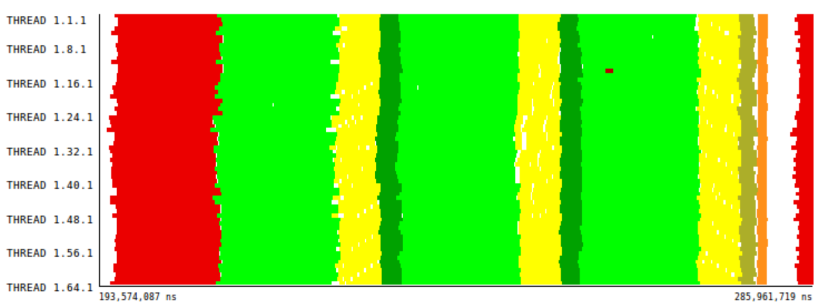


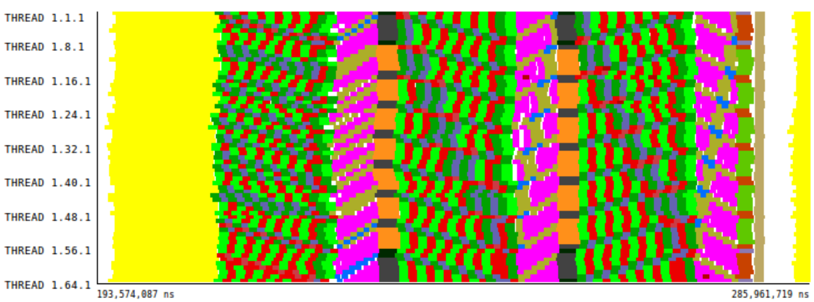
Structure quality?











- How many clusters?
- Which is better?
  - The two describe interesting structure
  - Typically SPMD would be a good first level of description for most apps

Using clustering

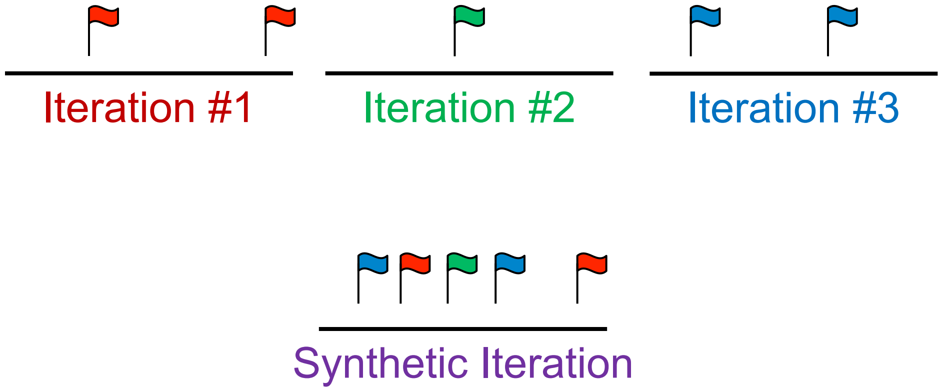
- Clustering enables focusing the analysis and opens many different uses
  - Analysis
    - Detection of application structure
  - Precise instantaneous metrics
    - correlation of sampled data to generate instantaneous metric evolution
  - Dimemas:
    - Separate speed factors per cluster on predictive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

Folding

Mixing instrumentation and sampling …

- … to get extreme detail with minimal overhead

- Different roles
  - Instrumentation delimits regions
  - Sampling report progress within region



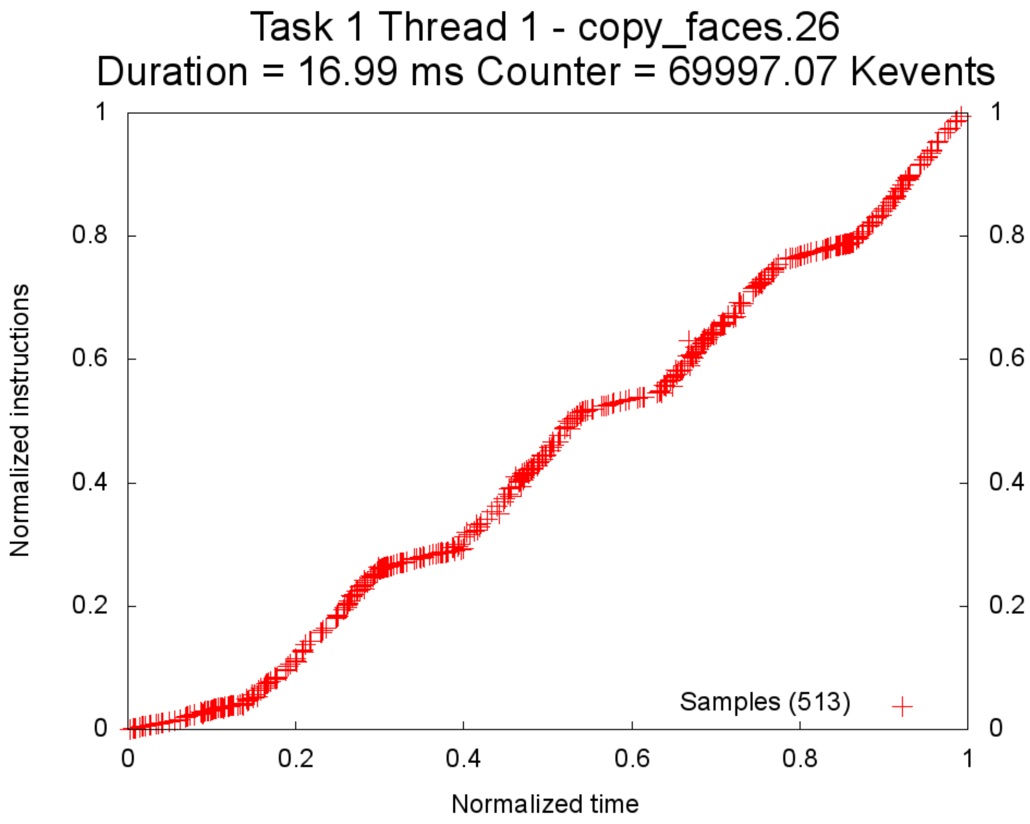
Harald Servat et al. "Detailed performance analysis using coarse grain sampling" PROPER@EUROPAR, 2009

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

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Folding hardware counters

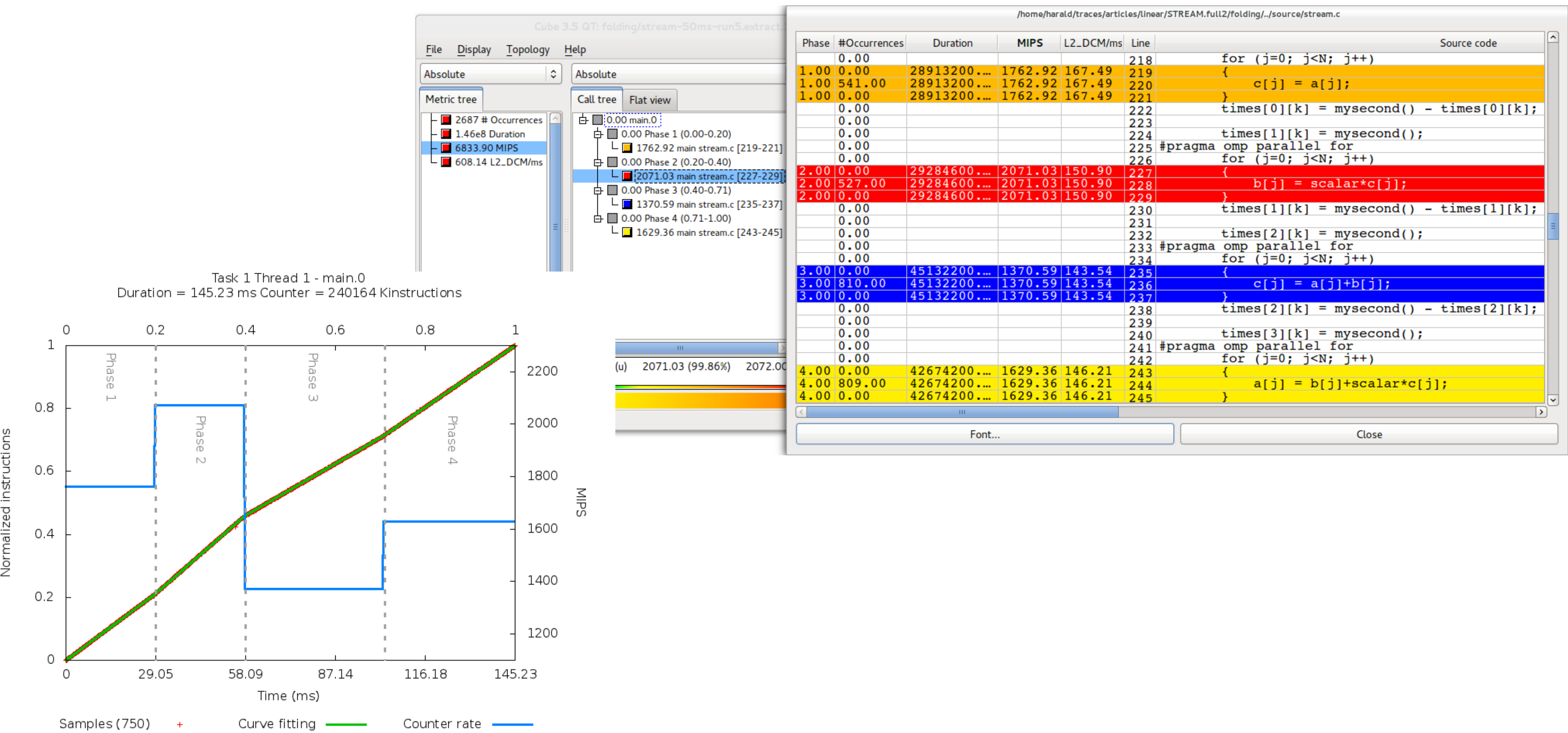
- Instructions evolution for routine copy_faces of NAS MPI BT.B



- Red crosses represent the folded samples and show the completed instructions from the start of the routine
- Green line is the curve fitting of the folded samples and is used to reintroduce the values into the tracefile
- Blue line is the derivative of the curve fitting over time (*counter rate*)

Folding → profiles of rates and ratios

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

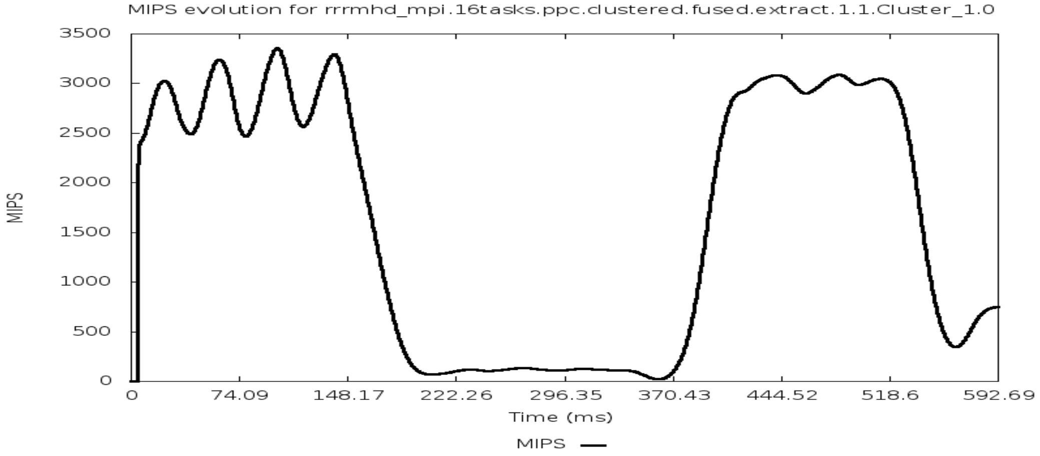


What is a good performance?

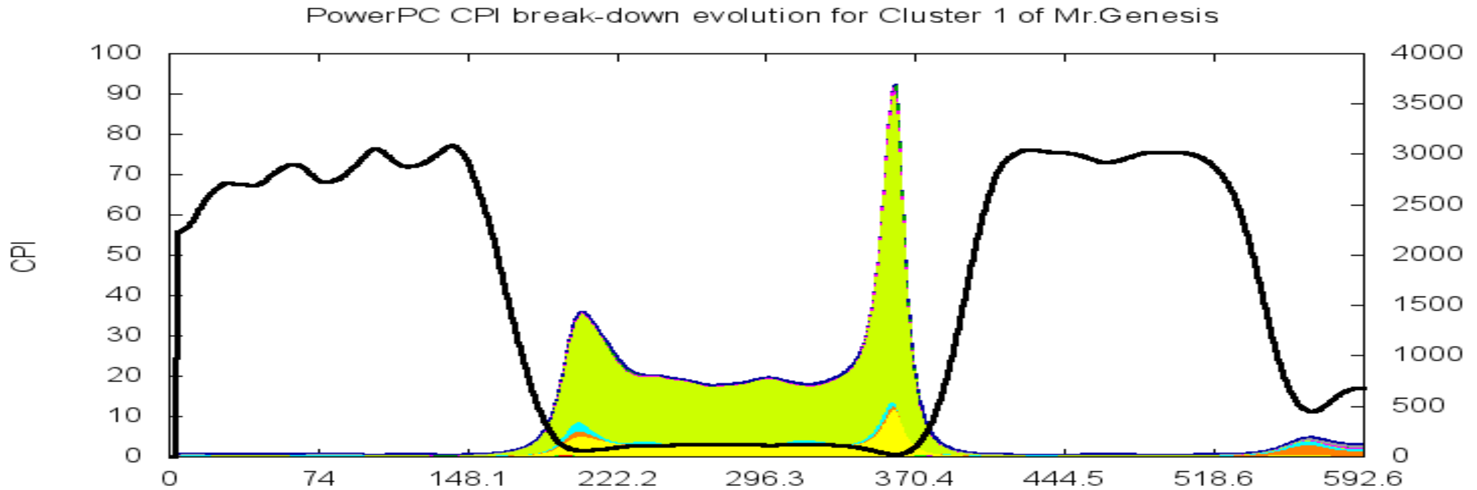
- Performance of a sequential region = 2000 MIPS

  Is it good enough?

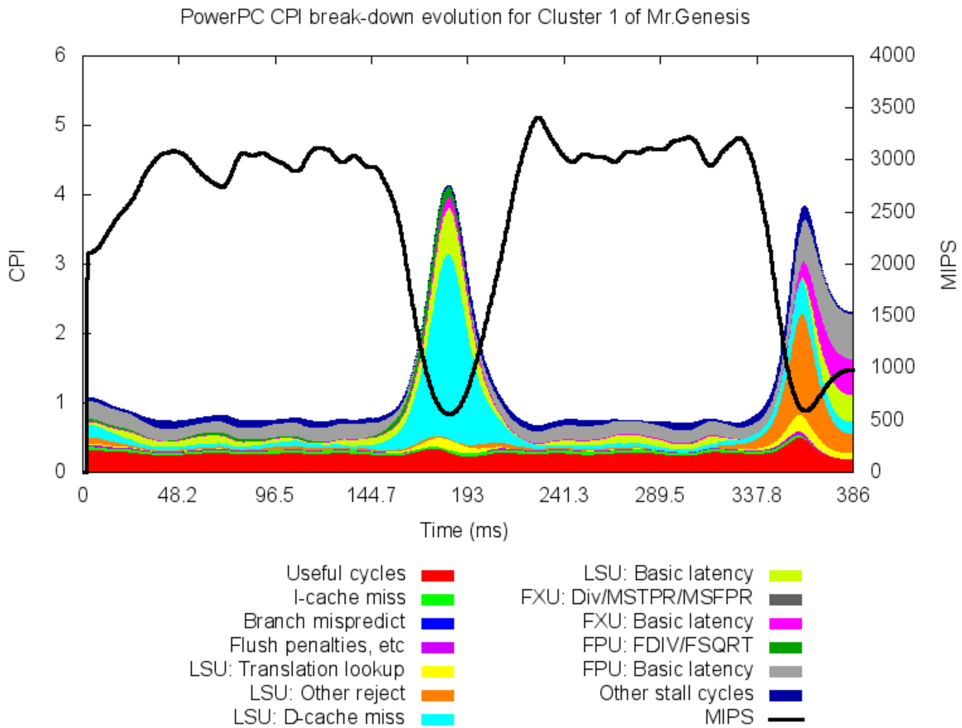
  Is it easy to improve?



Instantaneous CPI stack

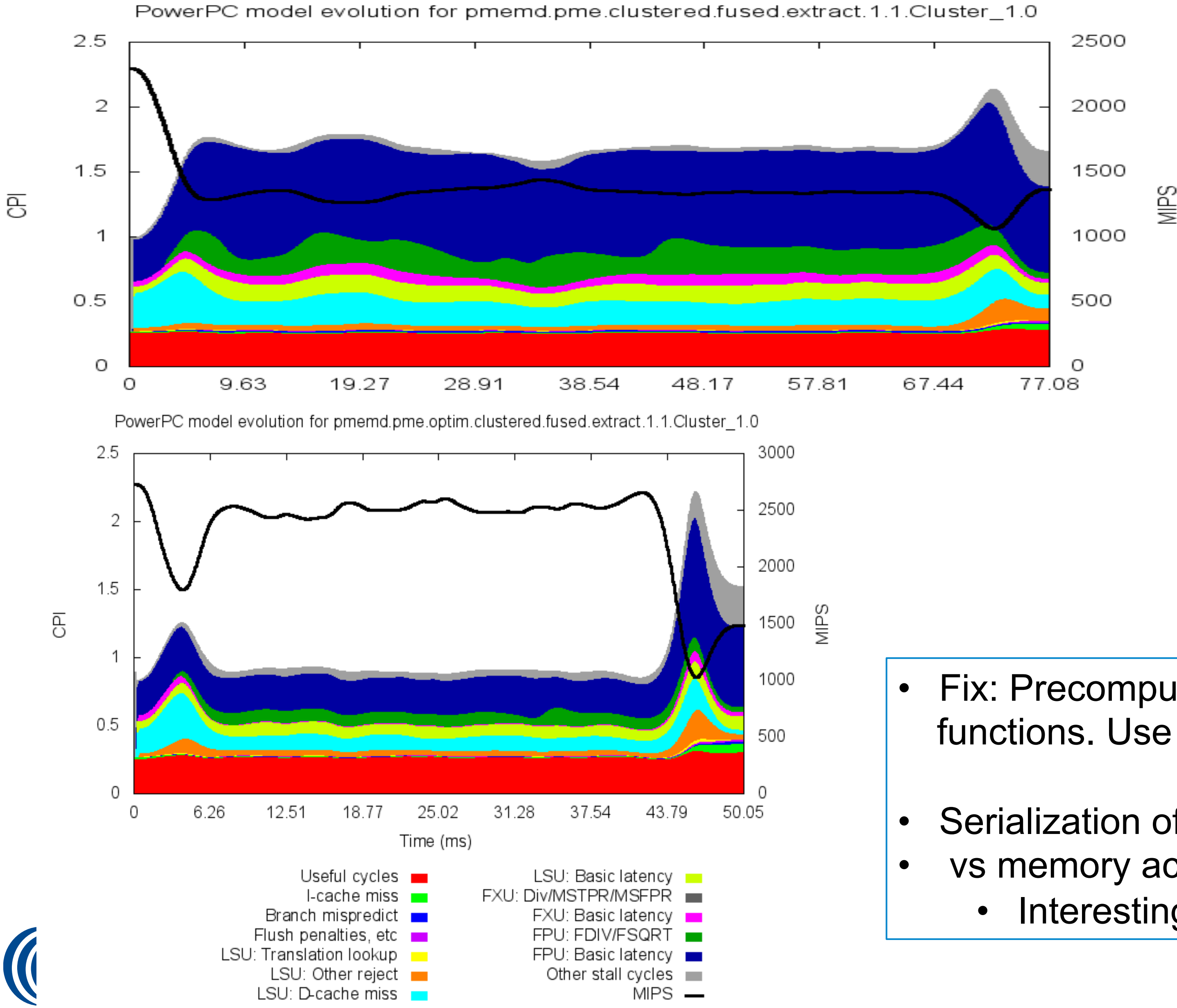


MRGENESIS



- Trivial fix.(loop interchange)
- Easy to locate?
- Next step?
- Availability of CPI stack models for production processors?
  - Provided by manufacturers?

Instantaneous CPI stack

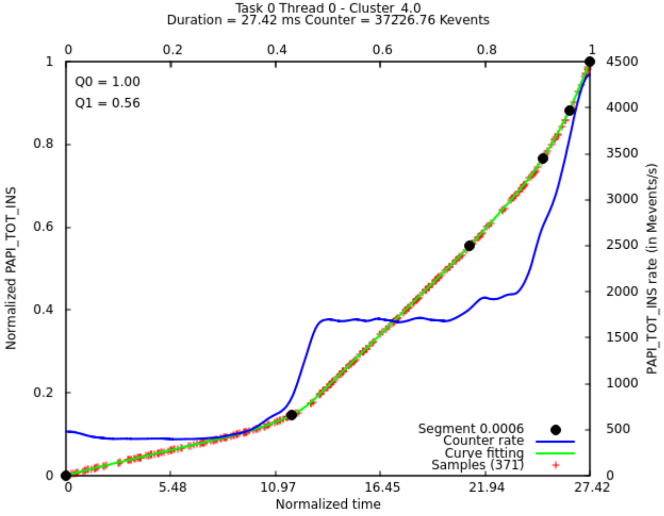


PMEMD

- Fix: Precompute transcendental functions. Use stored value.

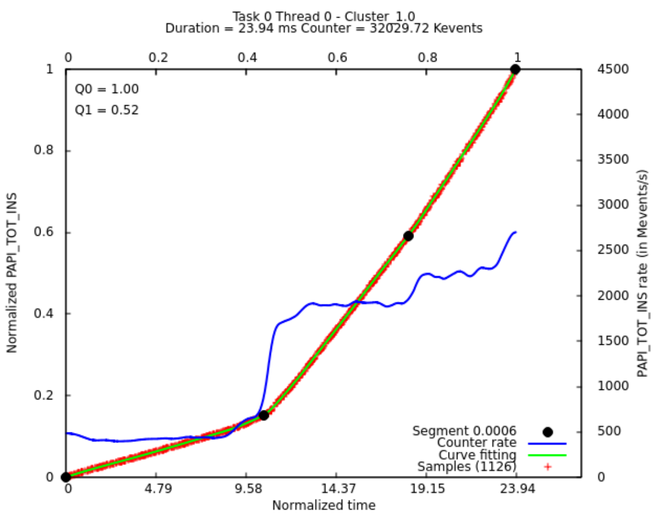
- Serialization of computation
- vs memory access
  - Interesting tradeoff

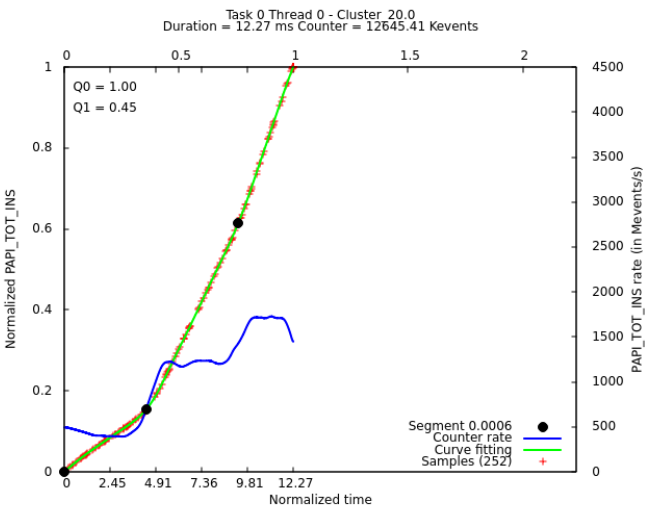
Correlating counters



CG-POP

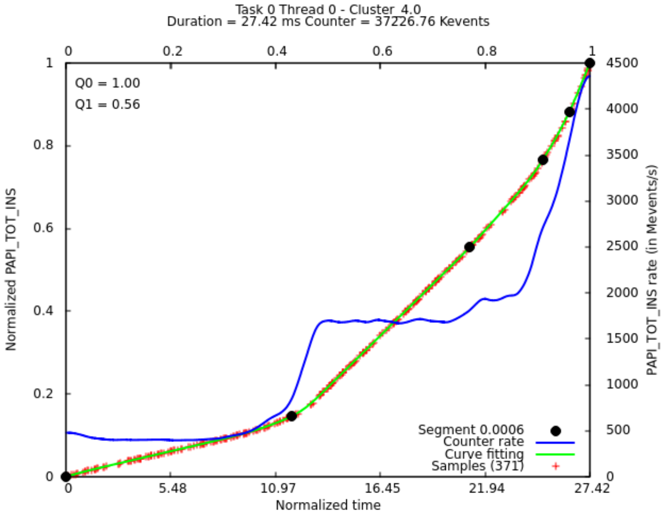
- Between processes
- 3 Algorithmic phases
- Impact of multicore sharing





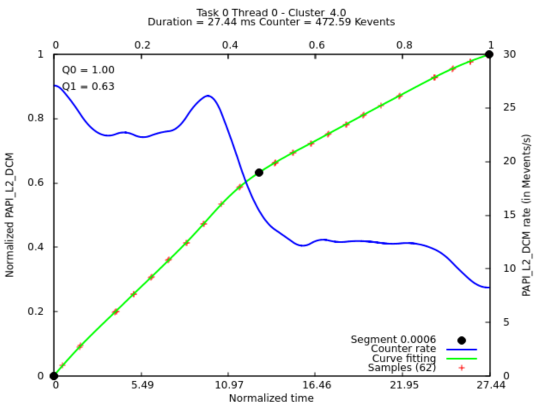
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

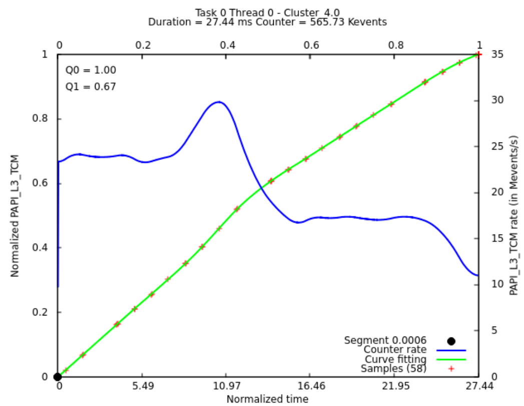
Correlating counters

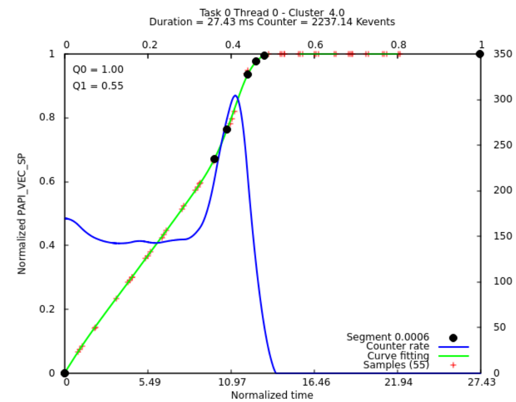


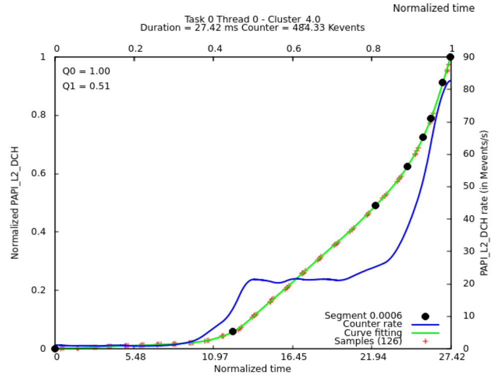
CG-POP

- Within a process
- 3 algorithmic phases
- Impact of multicore sharing









Barcelona Supercomputing Center
Centro Nacional de Supercomputación

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Methodology

PATC Parallel Programing Workshop
Oct 14-18 2013

Performance analysis tools objective

Help generate hypotheses

Help validate hypotheses

Qualitatively

Quantitatively

Tools: mechanisms and navigation

- The tools are instruments to address the questions

- Need to know how to use
  - 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.

Paraver Tutorial:

Introduction to Analysis with Paraver (MPI)

First steps

- Parallel efficiency – percentage of time invested on computation
  - Identify sources for “inefficiency”:
    - load balance
    - Communication /synchronization
- Serial efficiency – how far from peak performance?
  - IPC
- Scalability – code replication?
  - Total #instructions
- Behavioral structure? Variability?

Paraver Tutorial:
Introduction to Paraver and Dimemas methodology

Presenting application performance

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

$$\eta_{\|} = LB * \mu LB * Transfer$$

CommEff

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

$$\eta_{instr} = \frac{\#instr_0}{\#instr_P}$$

$$\eta_{IPC} = \frac{IPC_P}{IPC_0}$$

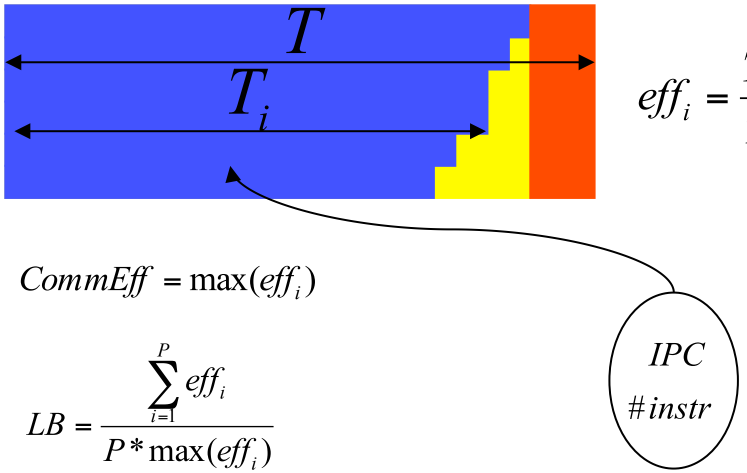
- Overall Scaling model

$$\eta = \eta_{\|} * \eta_{instr} * \eta_{IPC}$$

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

Scaling model

$$\eta_{\parallel} = LB * CommEff$$



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

$$CommEff = \max(eff_i)$$

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

Directly from real execution metrics

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

	Outside MPI	MPI_Recv	MPI_Isend	MPI_Irecv
THREAD 1.130.1	87,95 %	9,31 %	0,01 %	0,02 %
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

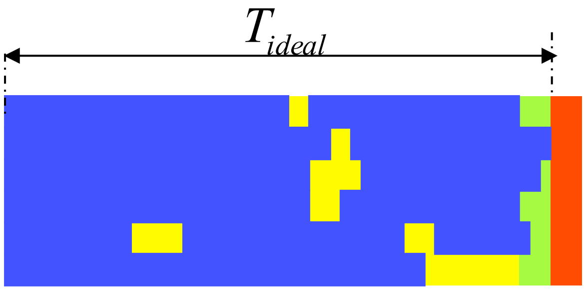
η

CommEff

LB

Scaling model

- Dimemas simulation with ideal target
  - Latency =0; BW = ∞



Migrating/local load imbalance
Serialization

$$CommEff = \mu LB * Transfer$$

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

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

	Outside MPI	MPI_Recv	MPI_Isend	MPI_Irecv
THREAD 1.130.1	[illegible]	[illegible]	0,01 %	0,02 %
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

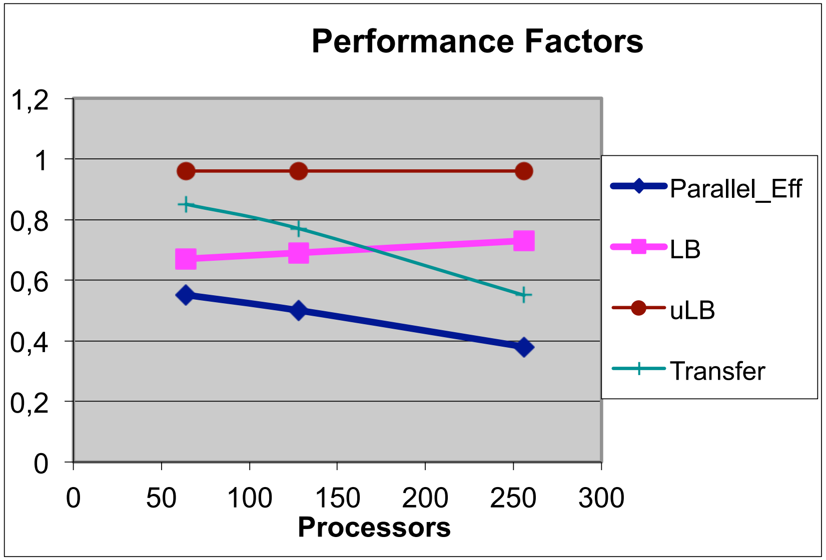
µLB

Scaling model

- Fundamental behavior
- Explains bottleneck …
- …how they migrate …
- … and combined effect

$$\eta_{\parallel} = LB * \mu LB * Transfer$$

GROMACSv4.5

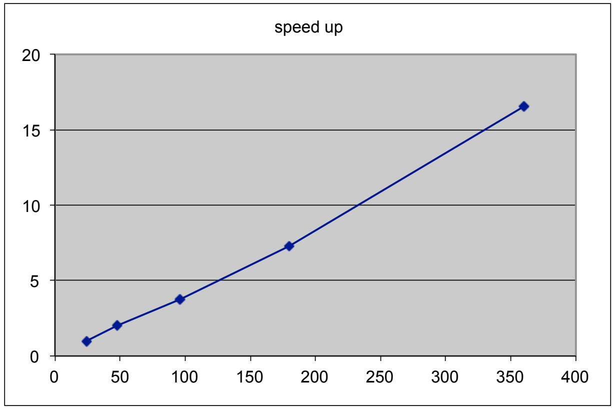


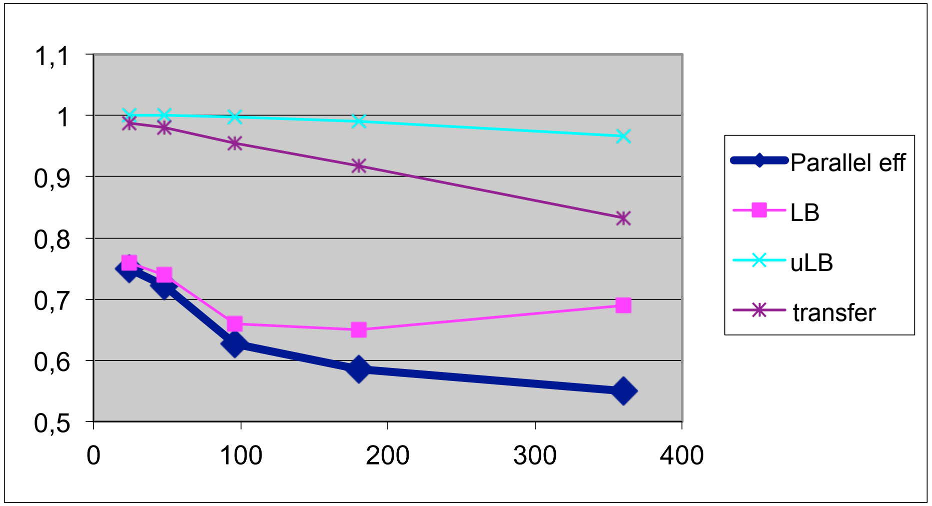
Modelling efficiency

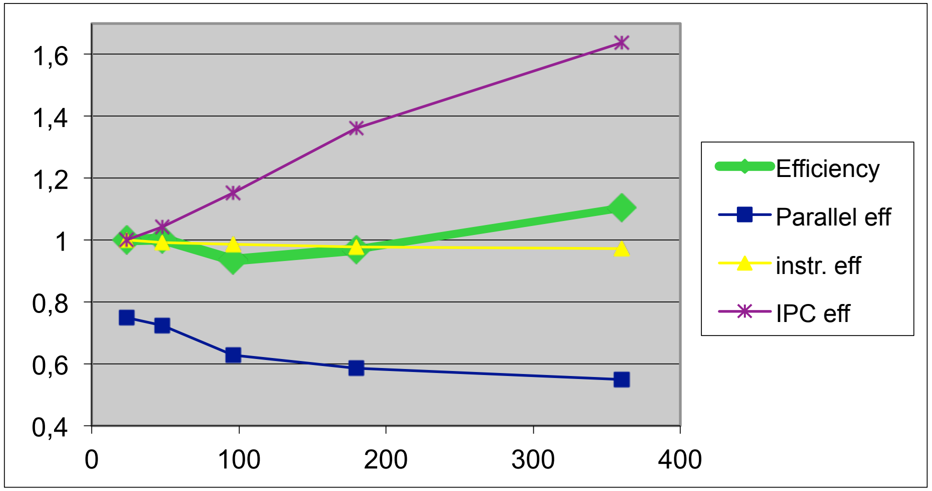
$$\eta_{\parallel} = LB * \mu LB * Transfer$$

CG-POP mpi2s1D - 180x120

Good scalability !!
Should we be happy?







$$\eta = \eta_{\parallel} * \eta_{instr} * \eta_{IPC}$$

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

BSC Tools web site

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

- Getting started
  - Start wxparaver
  - Help → tutorials and follow instructions
  - Follow training guides
    - Paraver introduction (MPI): Navigation and basic understanding of Paraver operation

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

THANKS

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Detailed material

PATC Parallel Programing Workshop
Oct 14-18 2013

Semantic Module

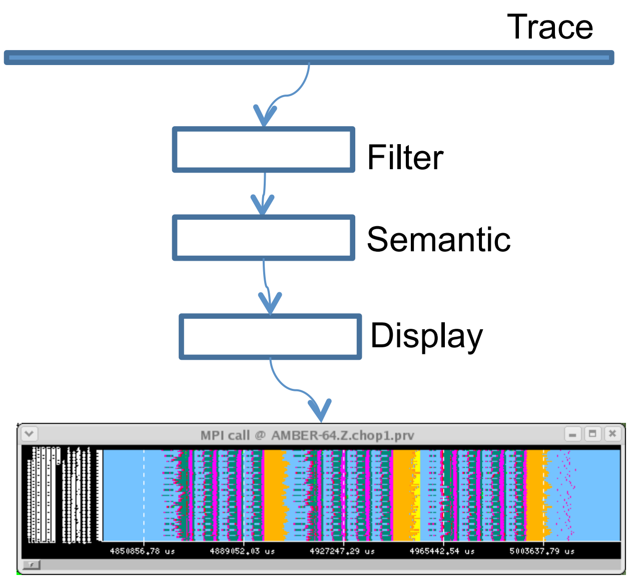
Basic functions of time

- 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

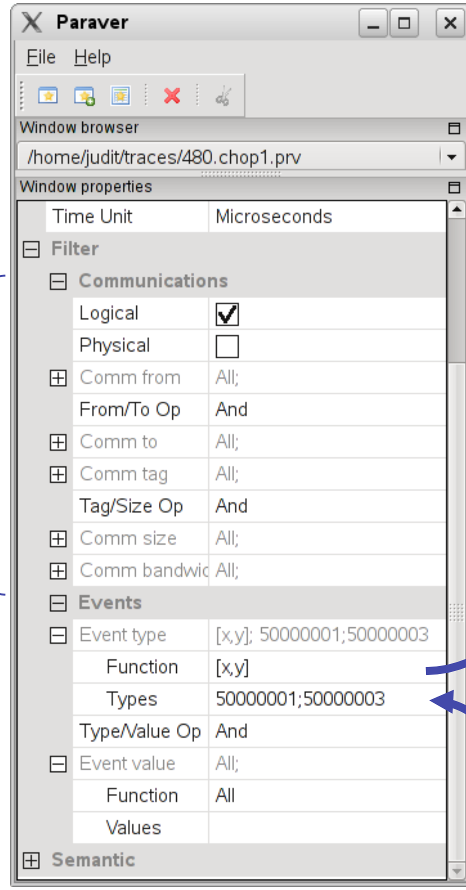
- **Semantic module** builds

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

Function of time   Series of values

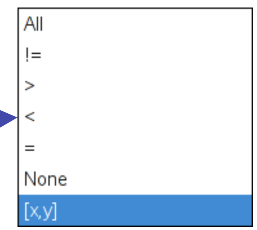


Filter module



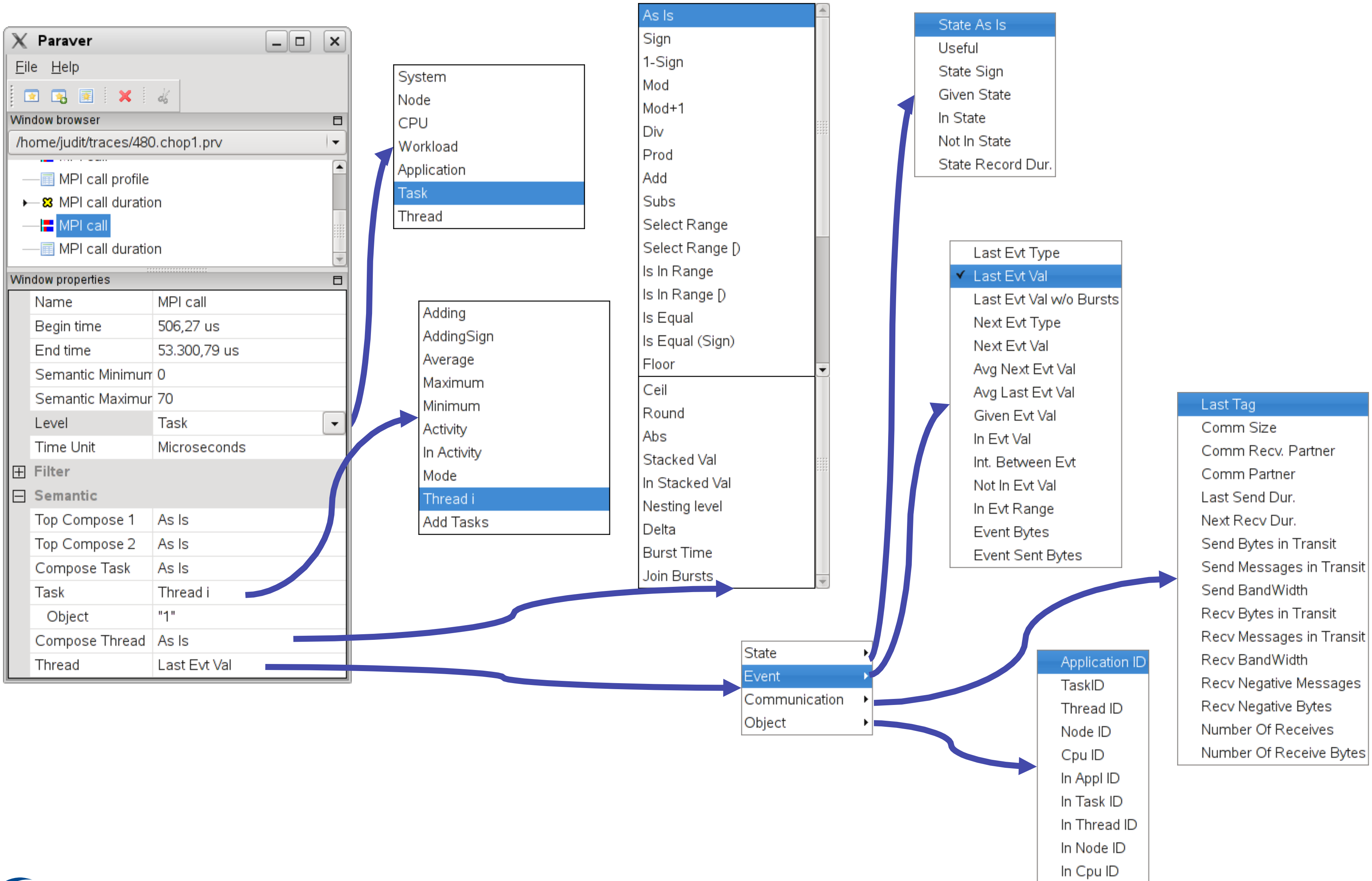
Communications that pass through the filter

Events that pass through the filter



Show list of event types

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

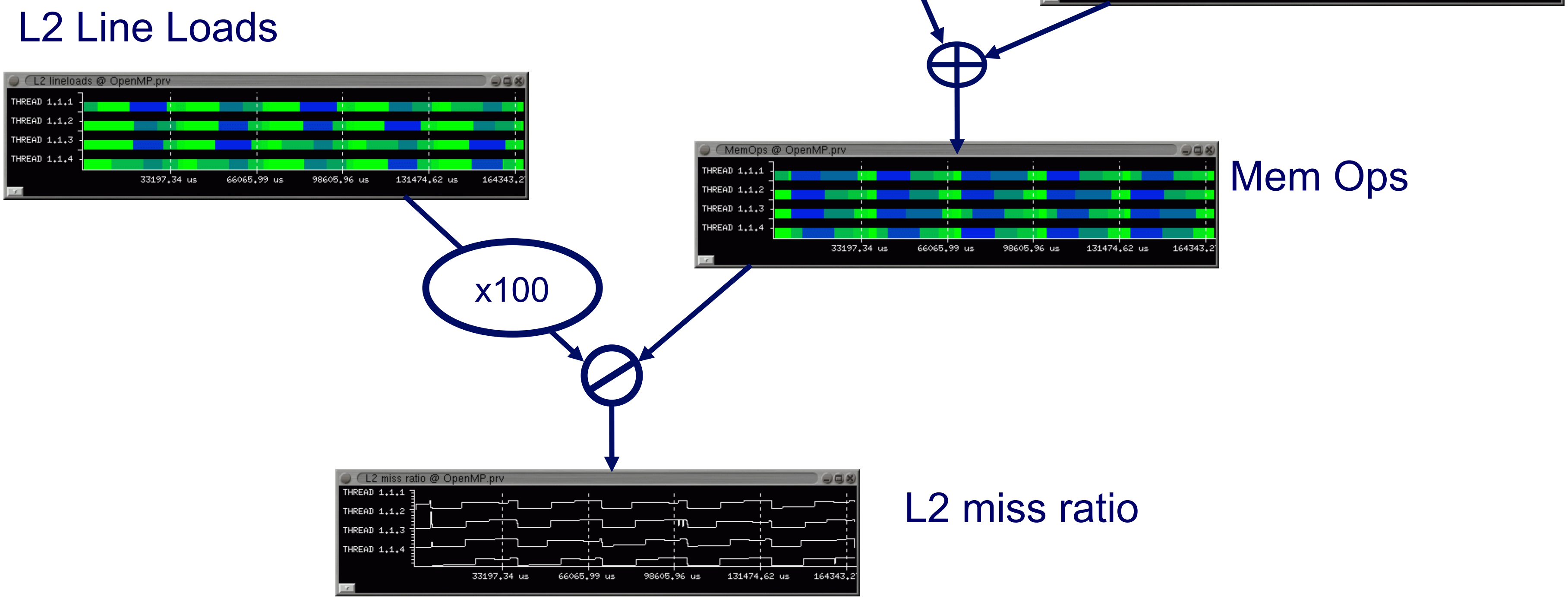
- From communication records to functions of time
  - Send Bytes $$s(t) = \sum_j Sz(C_j), j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$$
  - Send Bandwidth $$s(t) = \sum_j \frac{Sz(C_j)}{T_R(C_j) - T_S(C_j)}, j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$$
  - 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

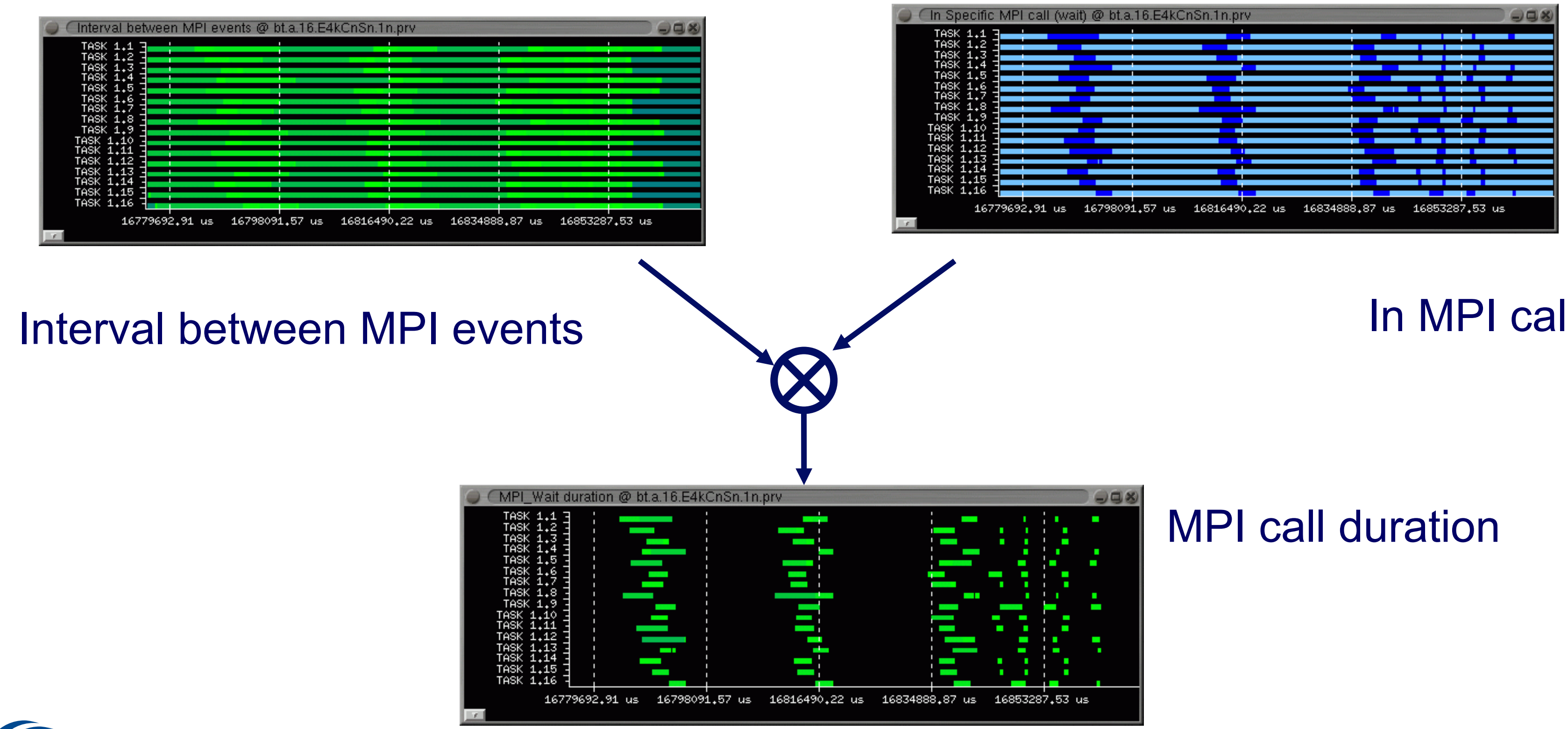
Semantic module

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

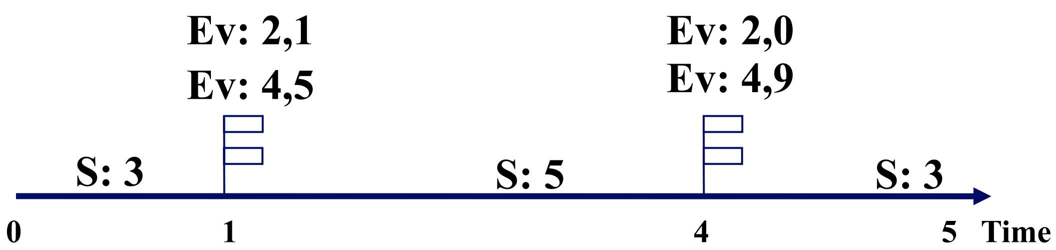


Semantic module

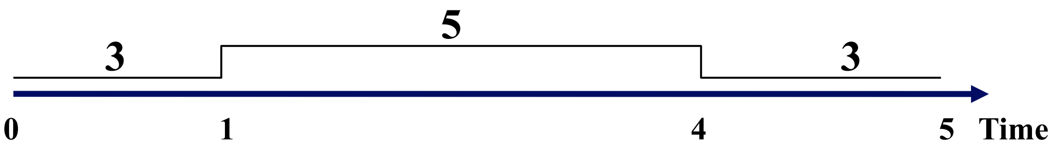
- Derived windows
  - Point wise operation
    - $S = \alpha * S^a \text{<op>} \beta * S^b$
    - <op> : + , -, *, /, …



Semantic module: Examples

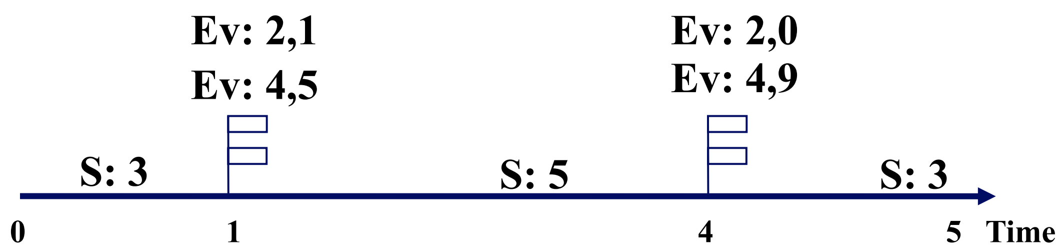


Thread function: State as is

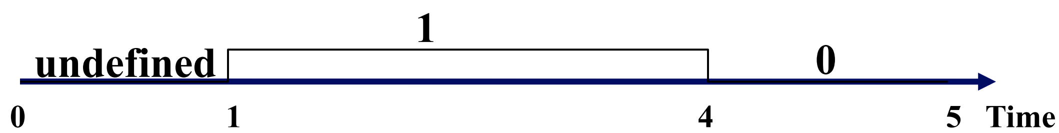


- Useful for
  - Global thread activity: computing, idle, fork/join, waiting,…..

Semantic module: Examples

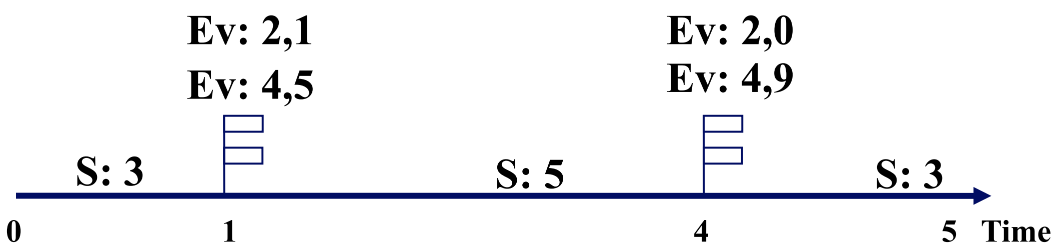


- Filter: type == 2
  - Thread function: Last event value

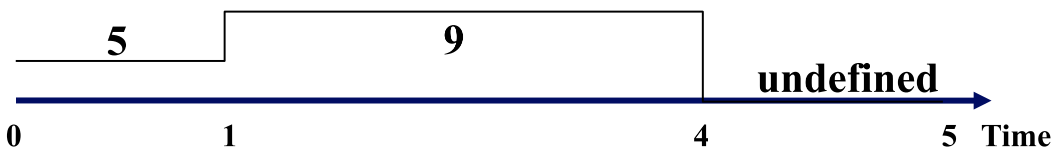


- Useful for
  - In parallel region
  - Mutual exclusion
  - Variable values: iteration,….

Semantic module: Examples

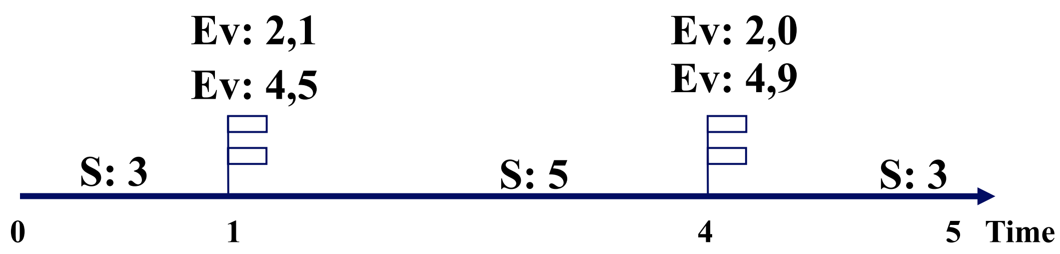


- Filter: type == 4
  - Thread function: Next event value

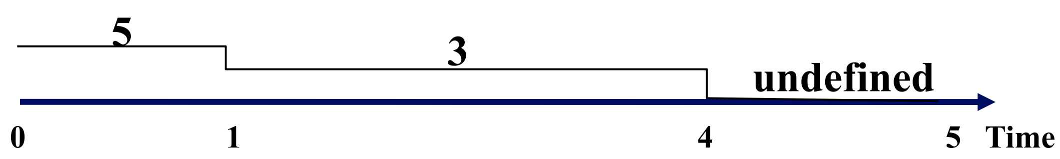


- Useful for
  - Hwc events (TLB, L1 misses,...) within interval

Semantic module: Examples



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

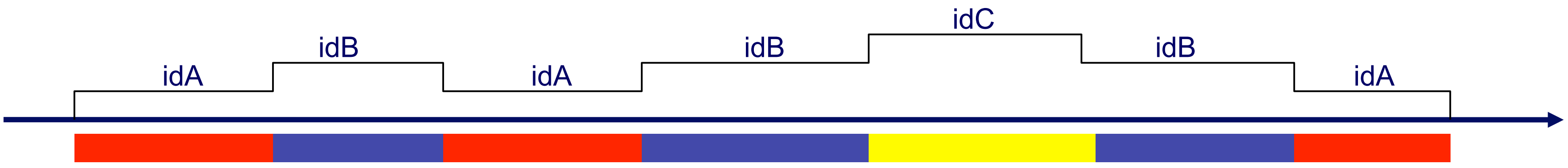


- Useful for
  - Hwc events (TLB, L1 misses,...) per time unit within interval

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

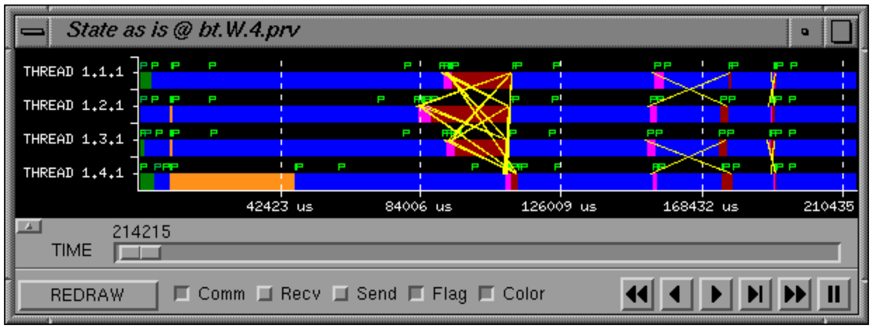
Semantic module: Examples

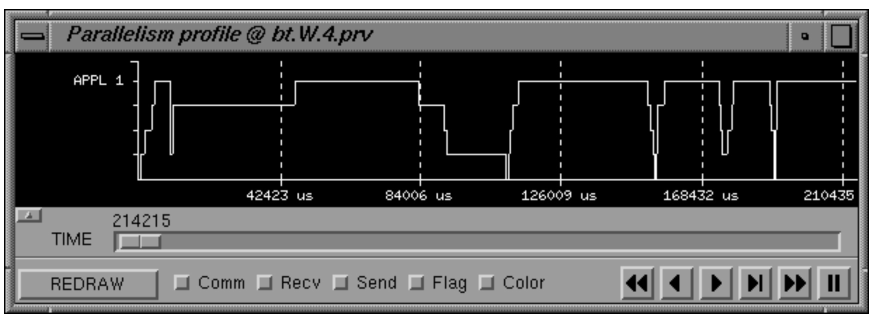
- Filter: type == USR_FCT
  Thread function: Last event value
  Compose: Stacked value



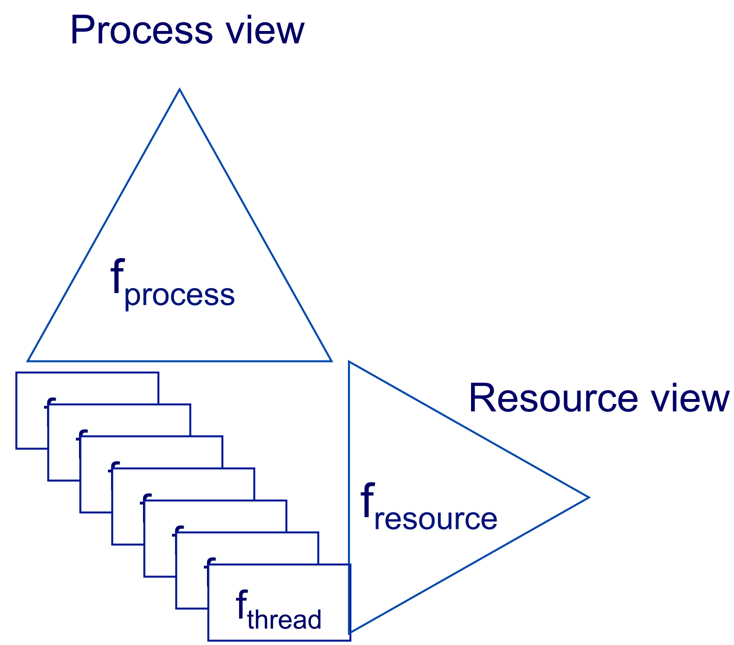
- Useful for
  - Routine

Semantic module perspective



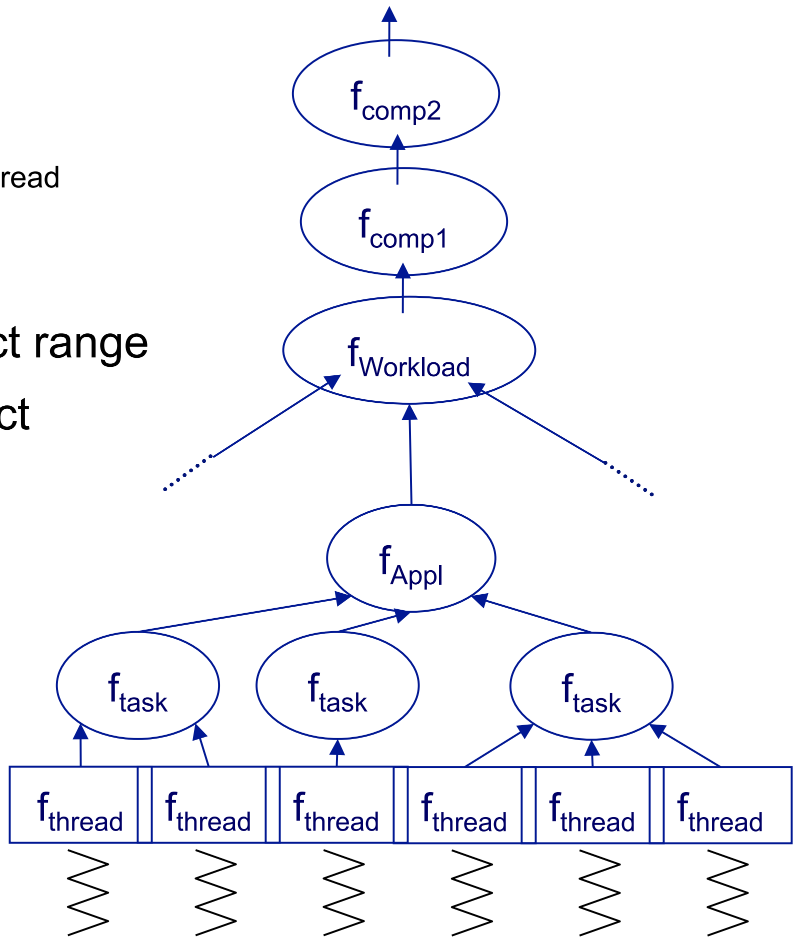


- Process model
  - Thread, task, application, workload
- Resource model
  - CPU, node, system



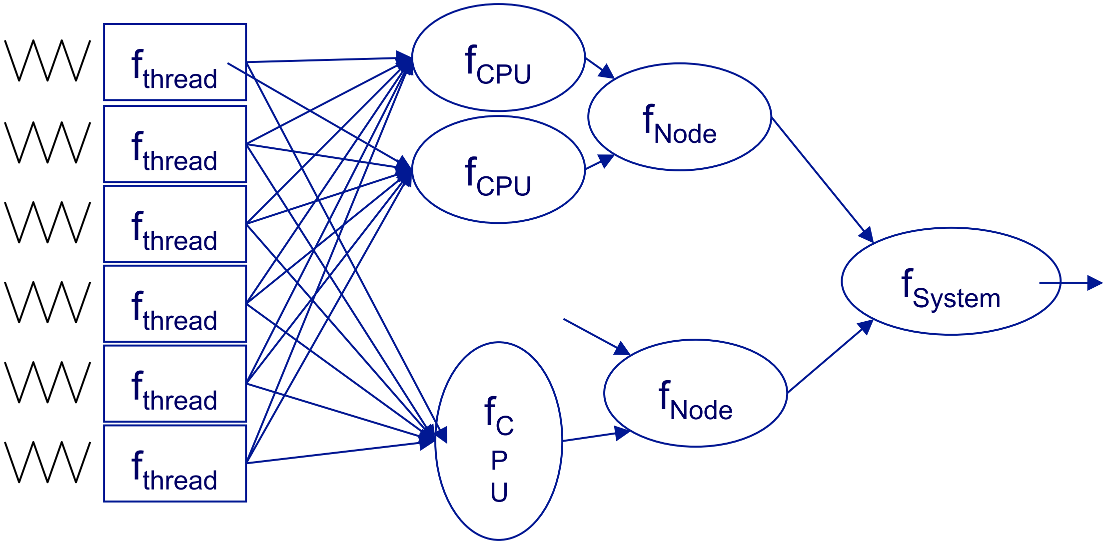
Process model perspective

- 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, …



Resource model perspective

- $Sf_{resource} = f_{comp2} \circ f_{comp1} \circ f_{System} \circ f_{Node} \circ f_{CPU} \circ S_{thread}$
- 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

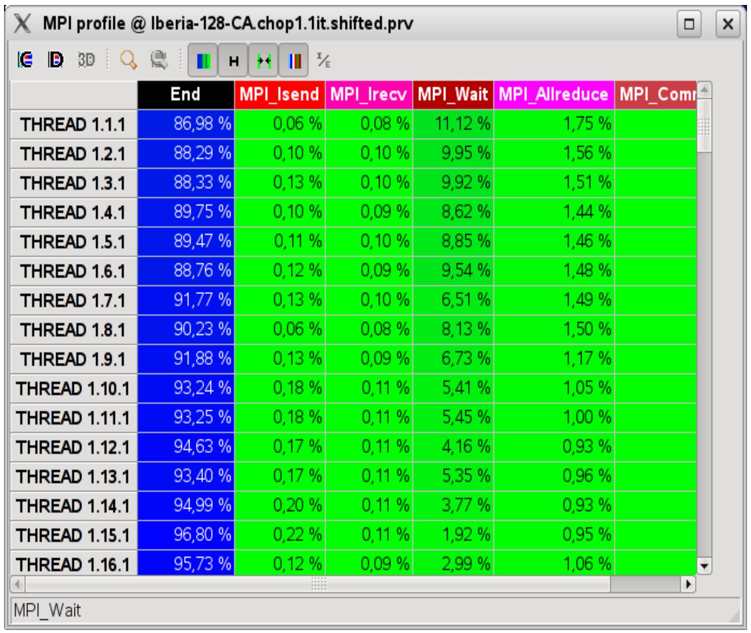


Analysis Module

How to read profiles

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

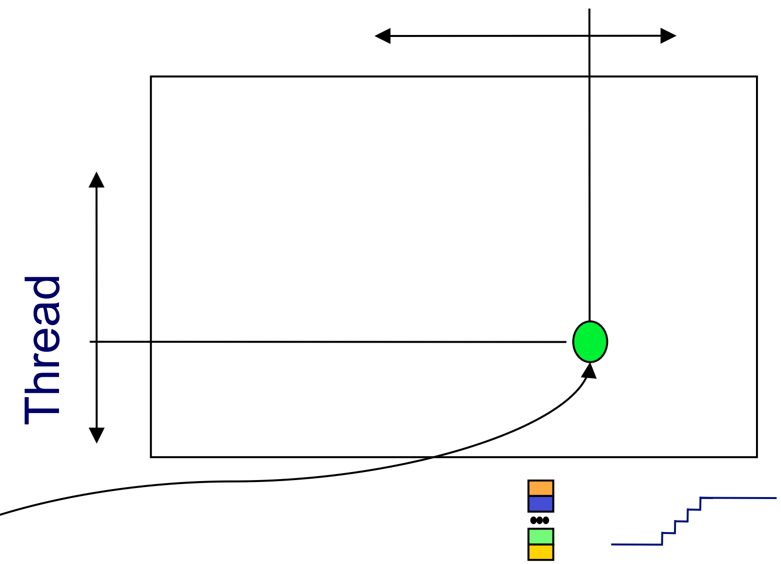
MPI call, user function,…



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

	End	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Allreduce	MPI_Comr
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



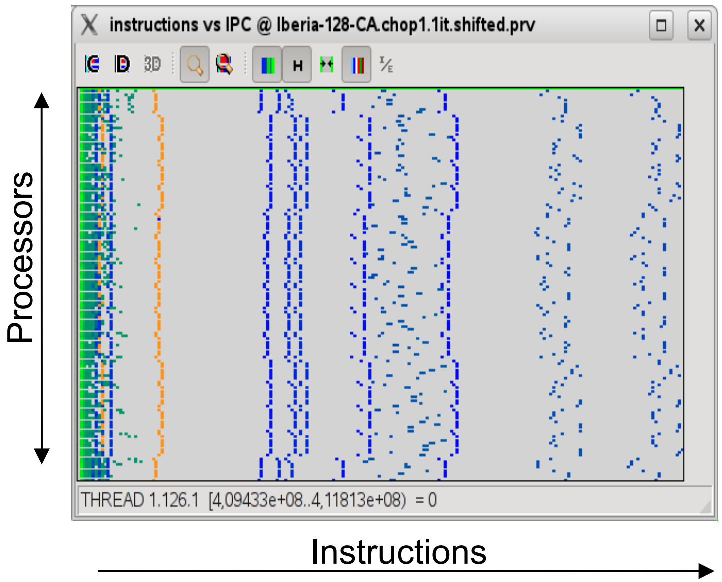
Value/color is a statistic computed for the specific thread
when control window had the value corresponding to the column

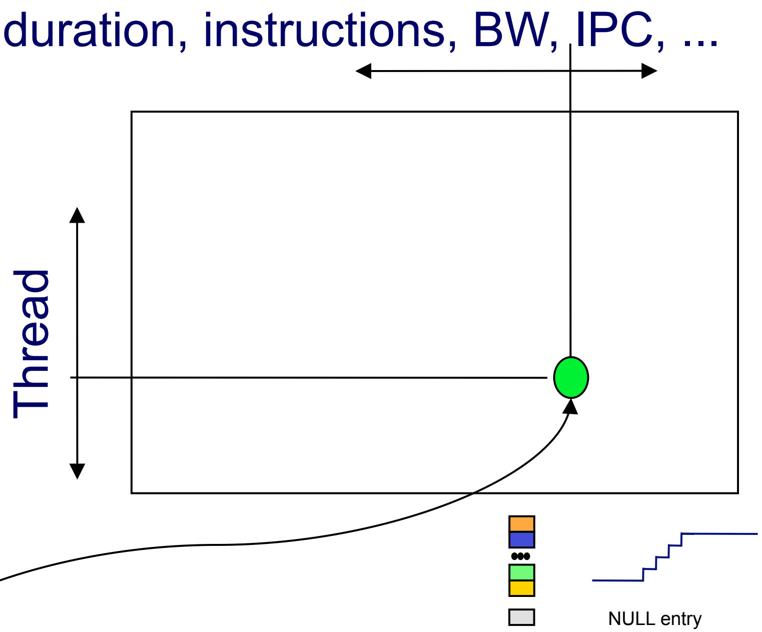
Relevant statistics:
Time, %time, #bursts, Avg. burst time
Average of **Data window**

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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**

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

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

- For each control window we define a set of bins

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

- And the discriminator functions

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

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

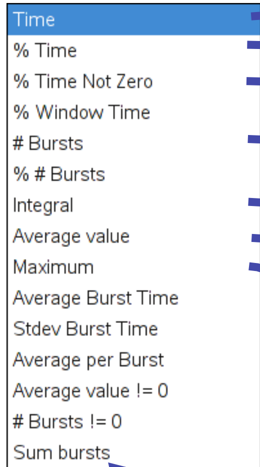
Identify regions with cw's within the (j,k) bin

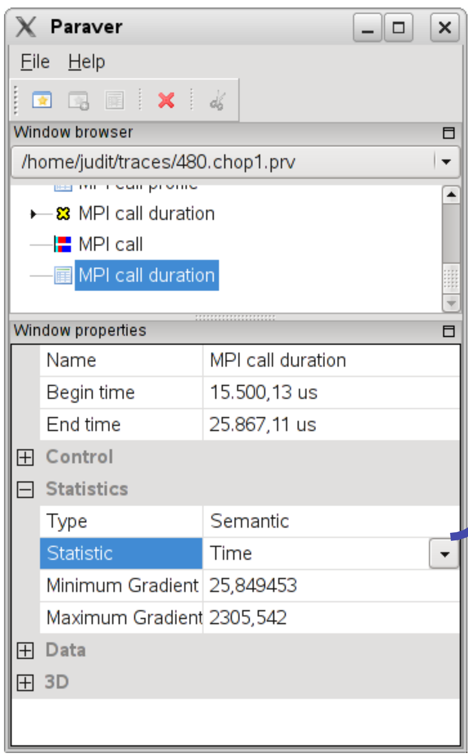
- The 3D analysis module computes a cube (or plane in the case of 2D) of statistics

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

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

2D analysis module





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

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

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

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

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

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

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

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

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

Distributed Configurations

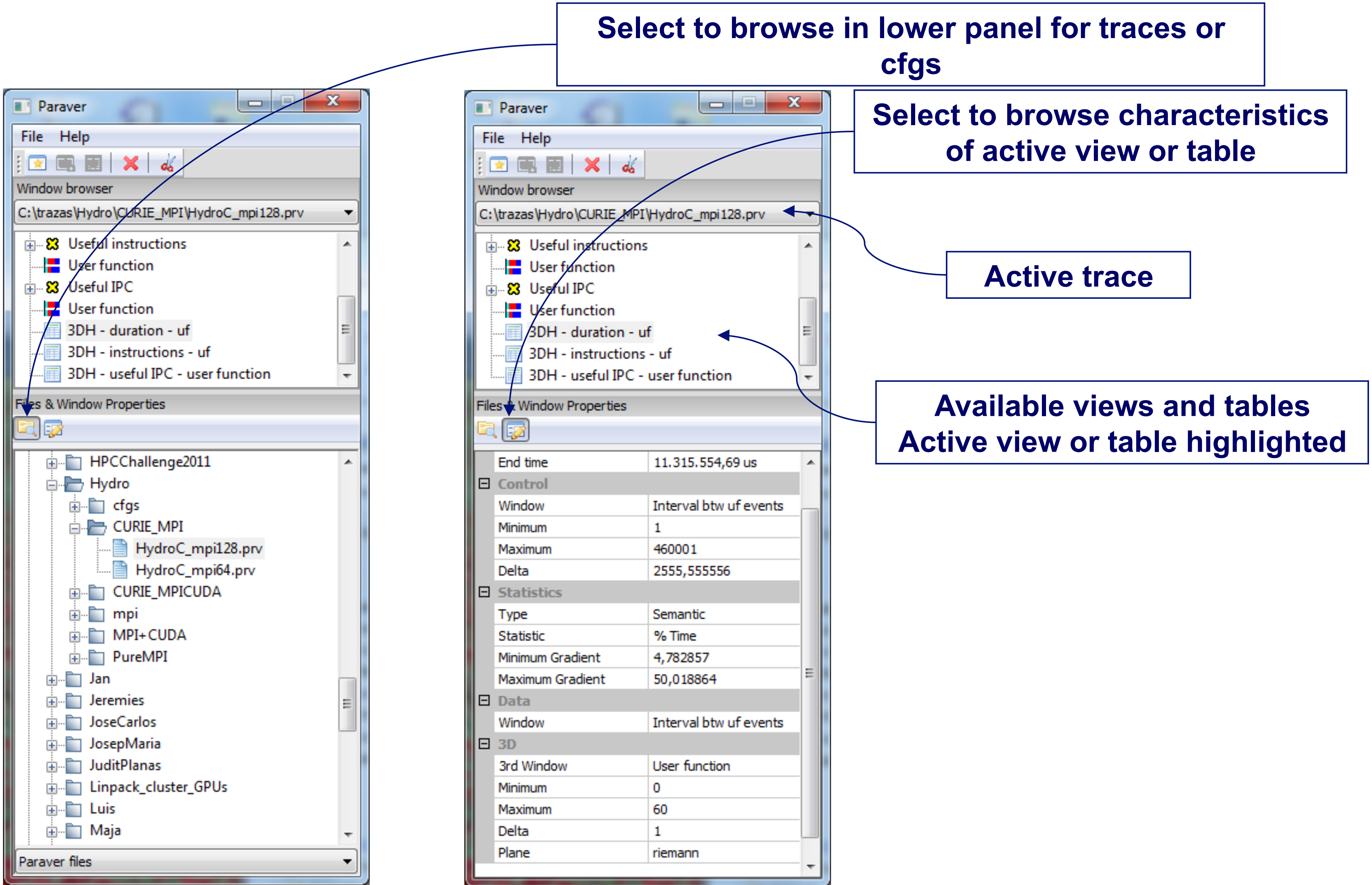
Distribution of cfg directories

$PARAVER_HOME/cfgs

- CFG
  - 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

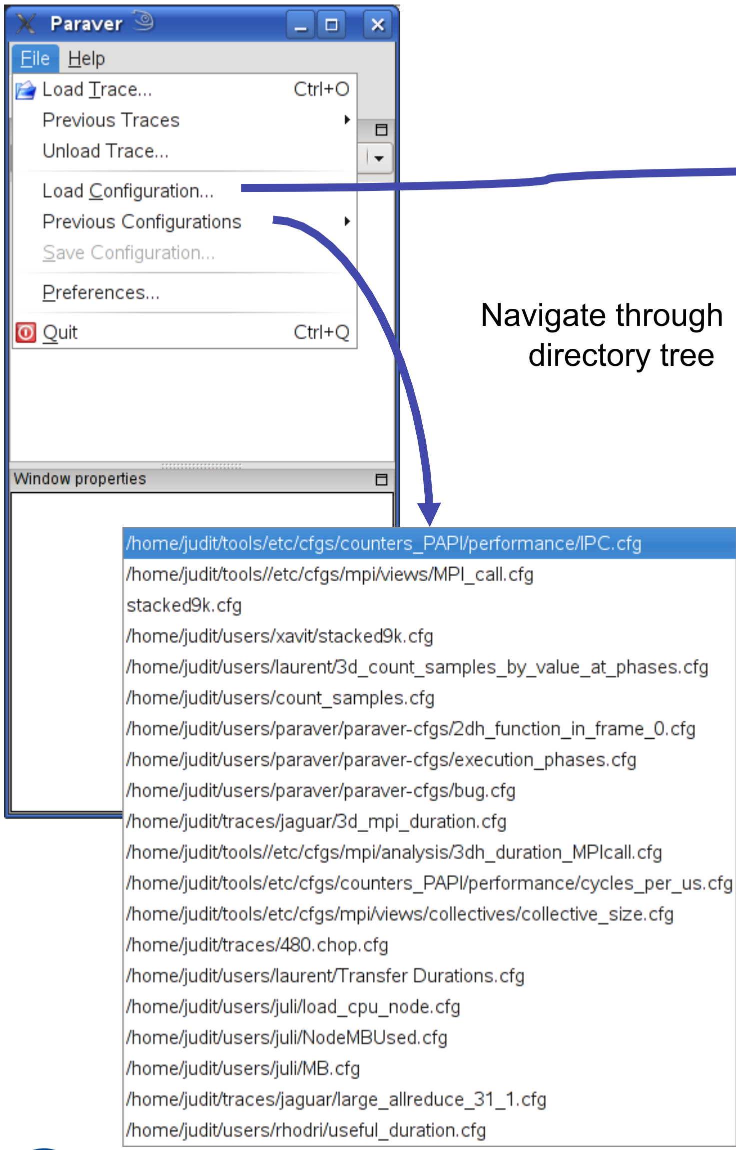
How to …

Main Paraver window



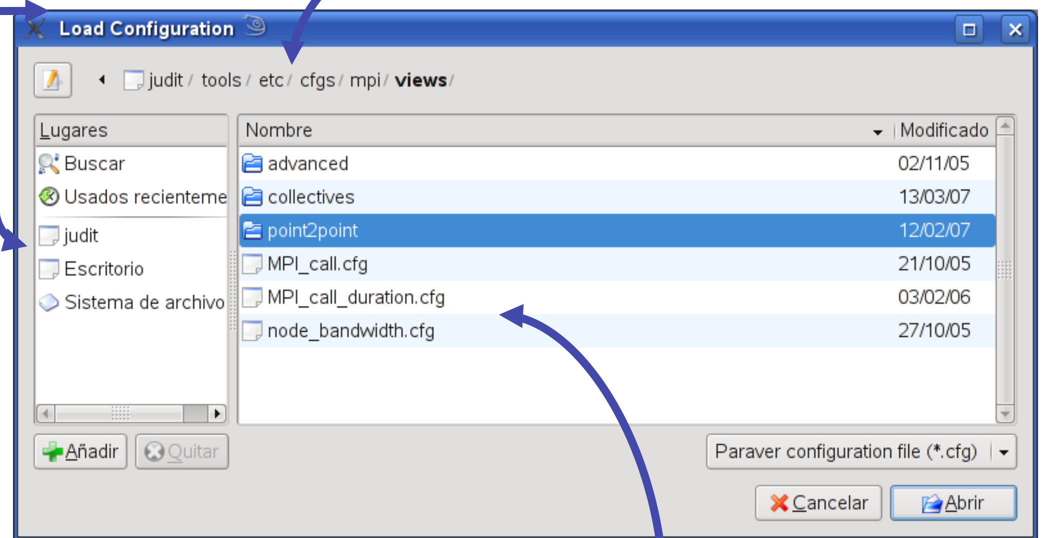
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Load configuration files



Select directory

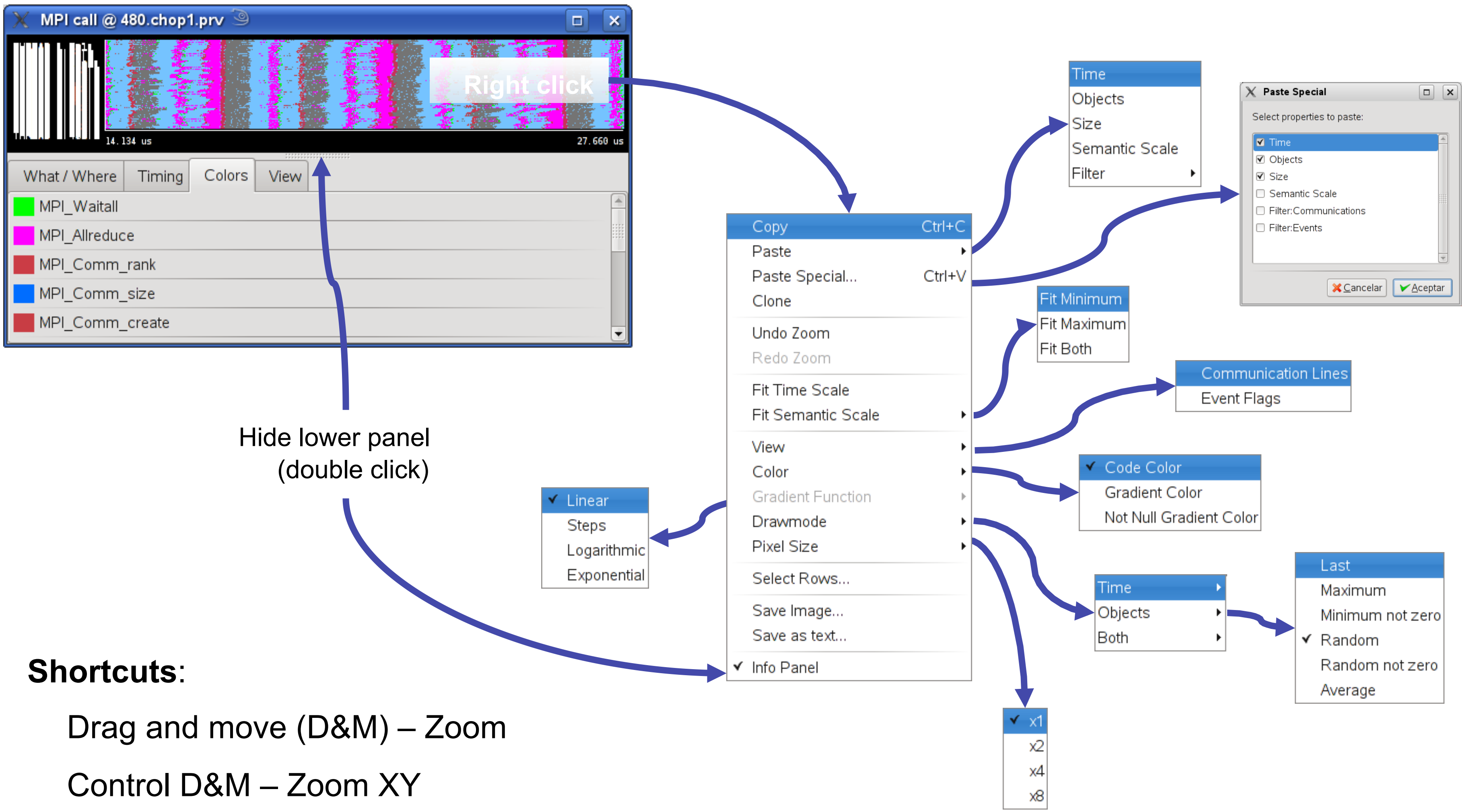
Navigate through directory tree



List of directories and configuration files in current directory

APPLIED TO THE CURRENT TRACEFILE

Navigation



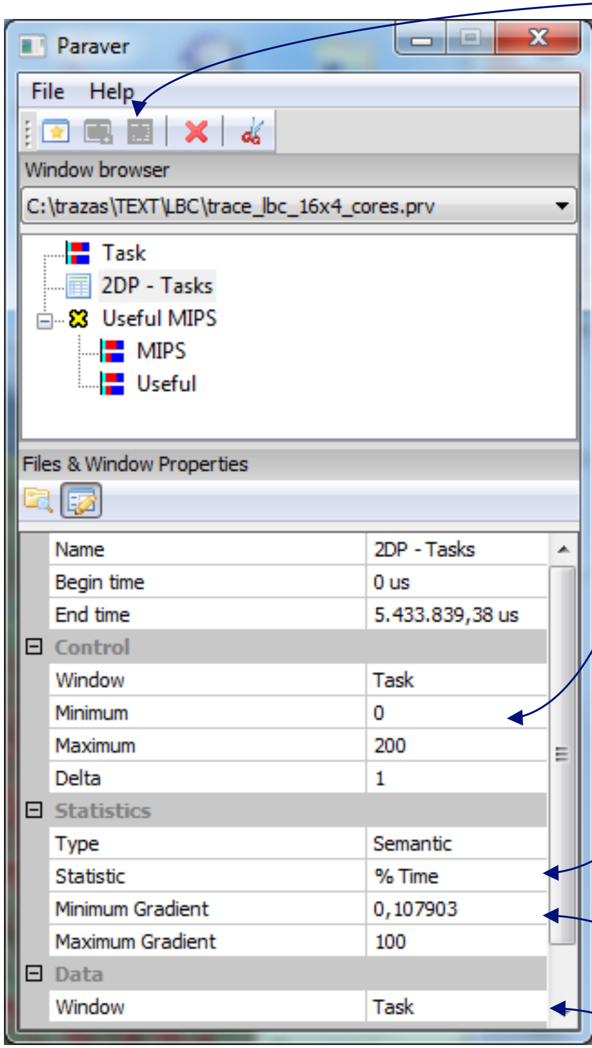
Shortcuts:

Drag and move (D&M) – Zoom

Control D&M – Zoom XY

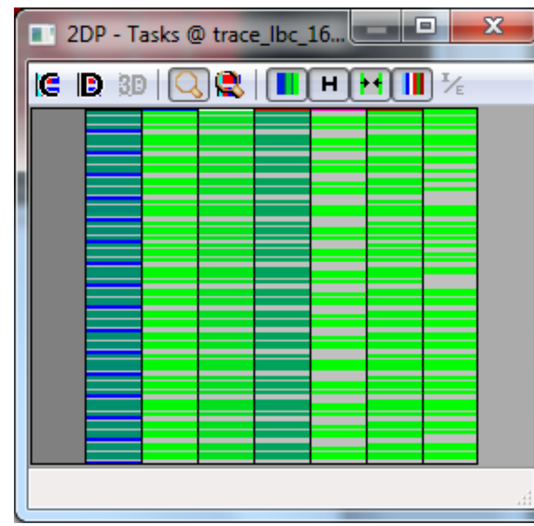
Shift D&M – Timing

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

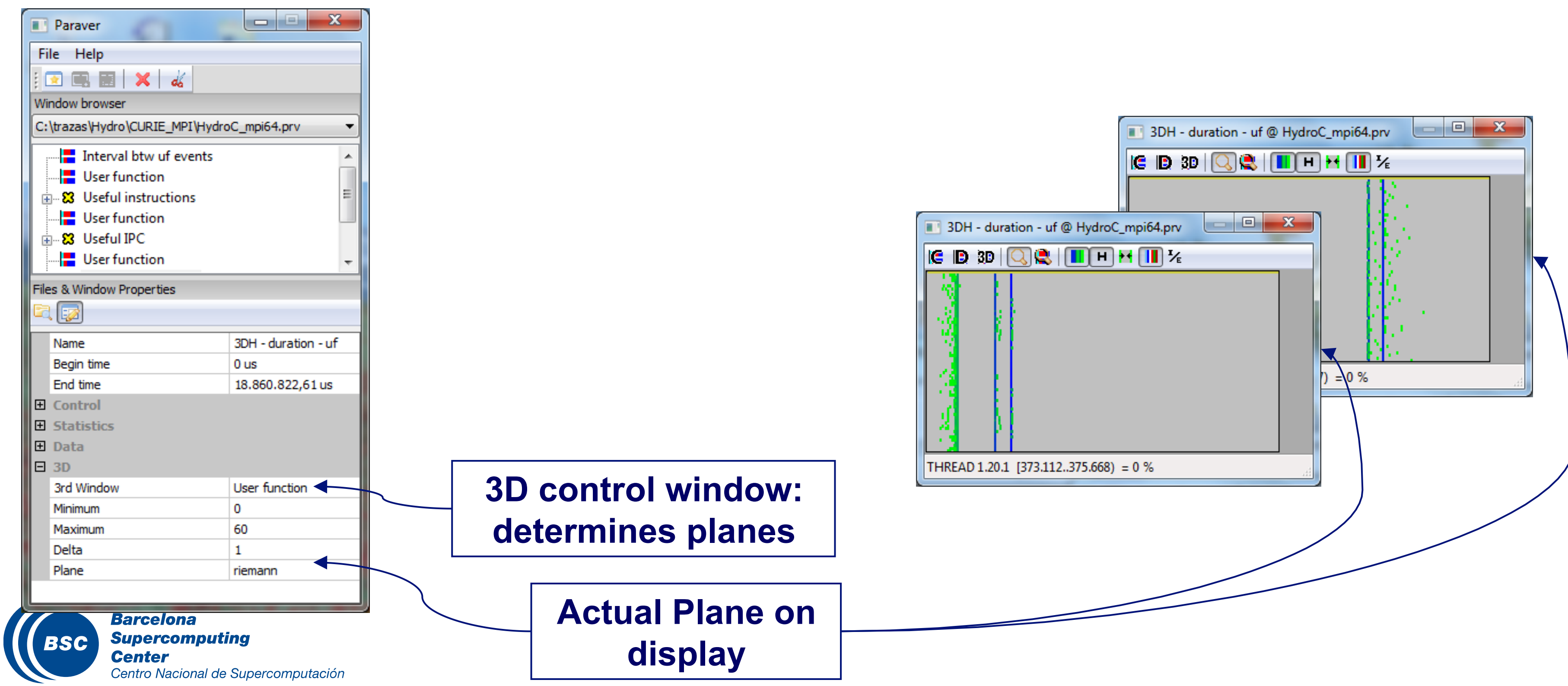
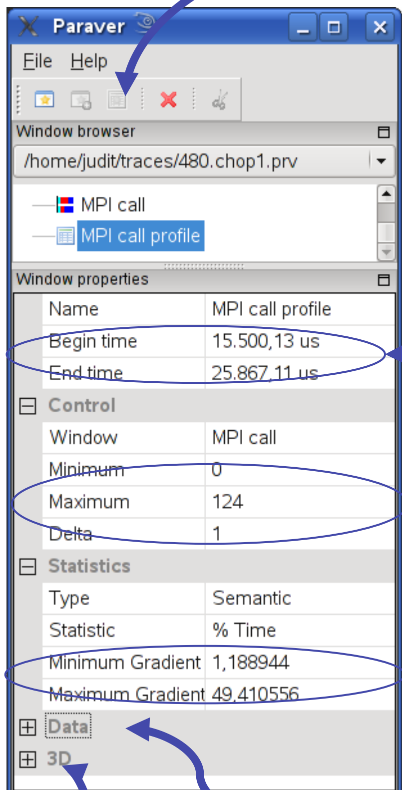


Table information and control

Create a new table



Display whole table / cell text

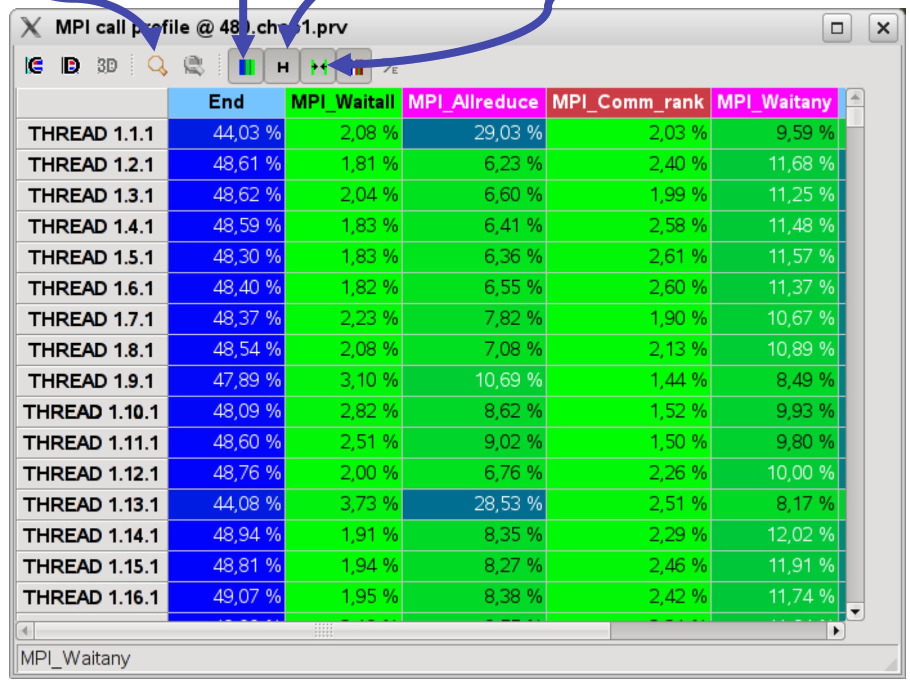
Color/not cells

Transpose

Hide null columns

Region analyzed

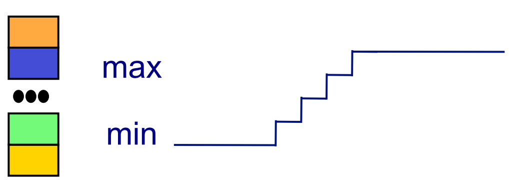
Bin definition



	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 %

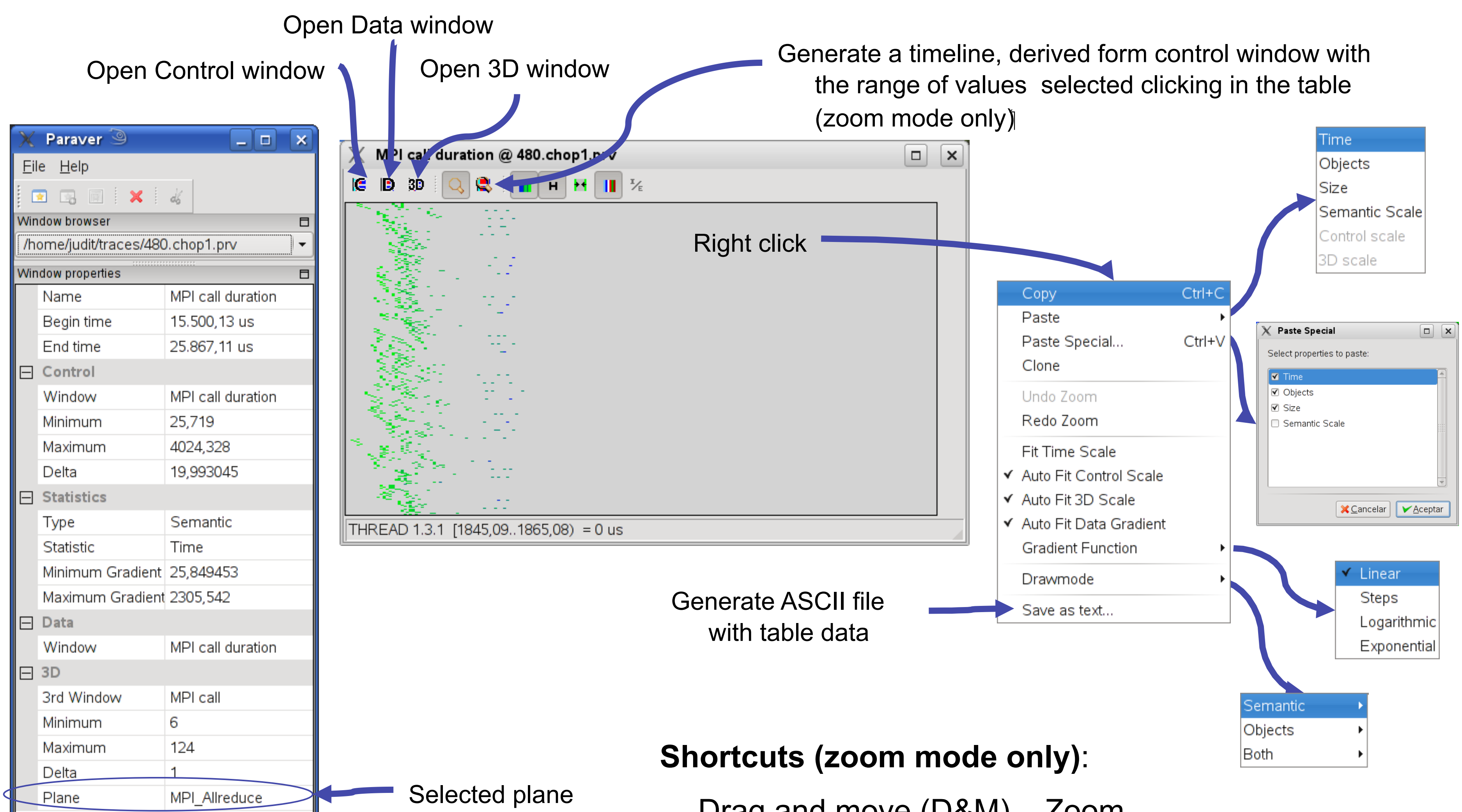
Change Data window

Color encoding



Activate 3D analysis

Table information and control



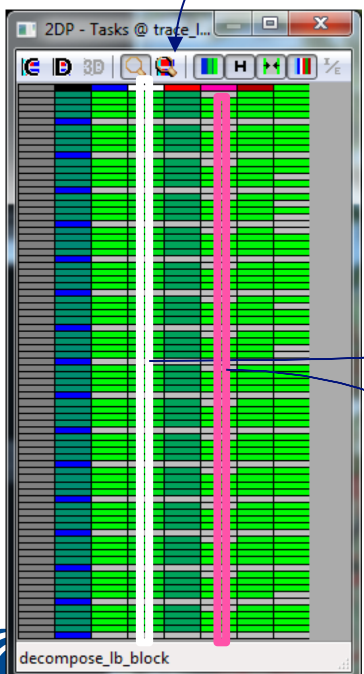
Shortcuts (zoom mode only):

Drag and move (D&M) – Zoom

Control D&M – Zoom XY

From tables to timelines

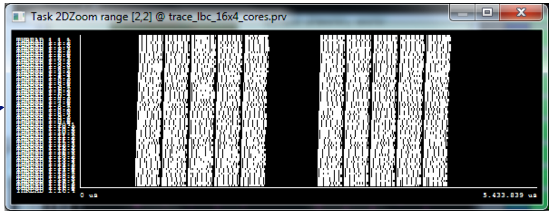
- Where in the timeline do the values in certain table columns appear?
  – ie. want to see the time distribution of a given routine?



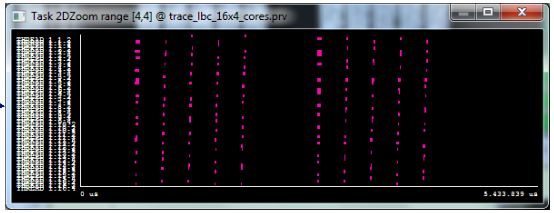
Click button and select column(s)

Will automatically generate derived views from the global view

Only showing when is routine white executing



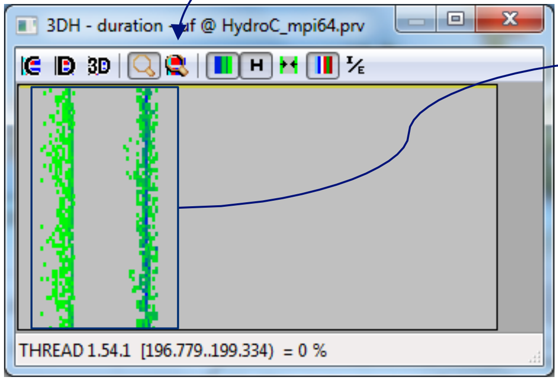
Only showing when is routine pink executing



From tables to timelines

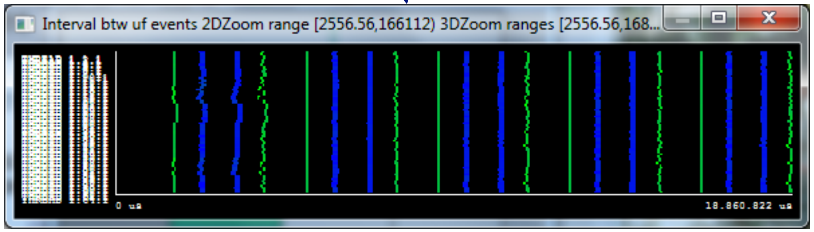
- Where in the timeline do the values in certain table columns appear?
  - ie. want to see where the timeline happen computation bursts of a given length?

Click button and select column(s)



3D histogram of duration of routine foo

Will automatically generate



Only showing duration of routine foo

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Trace manipulation

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

Guided hands-on available in
http://www.bsc.es/computer-sciences/performance-tools/documentation → Trace Preparation

Filtering very large traces

Trace to which it will be applied

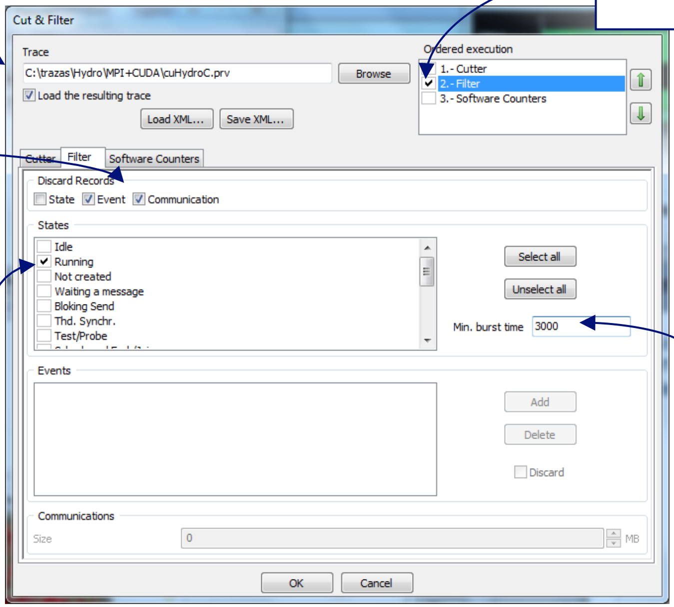
A trace with basename.filter1.prv will be generated

Select filtering option

Discard events and communications

Keep only Running bursts ….

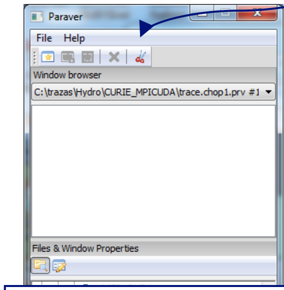
--- longer than 3000 ns



Cutting very large traces

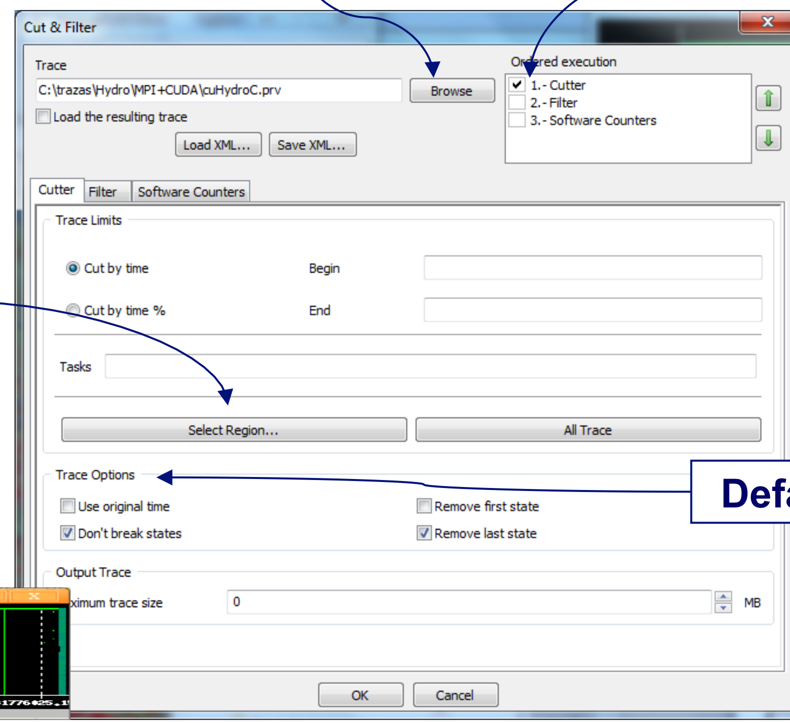
- Load a filtered trace and use the scissors tool

Scissors tool



Browse to select file from which the cut will be obtained

Select cutter

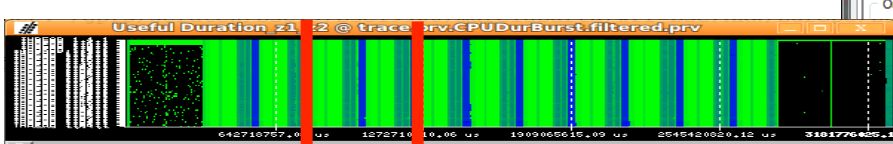


Click to select region

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

Recommended cuts within long computation bursts

Default setups



Extrae

Adapt job submission script

```
#!/bin/bash

export NP=8
export INPUT=$1

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

appl.job

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

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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

…

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

LD_PRELOAD library selection

- Library depends on programming model

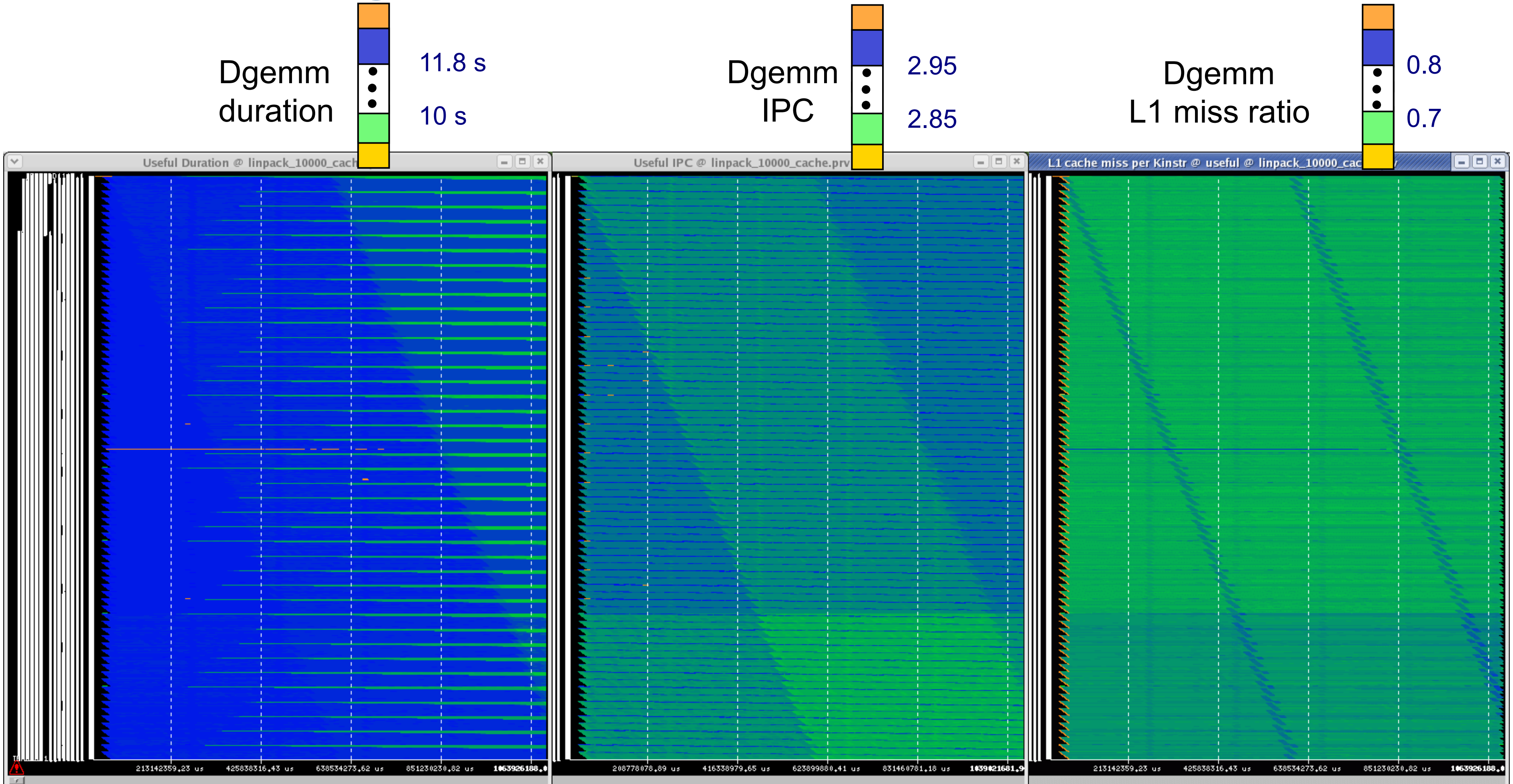
Programming model	Library
Serial	libseqtrace
Pure MPI	libmpitrace[f][1]
Pure OpenMP	libomptrace
Pure Pthreads	libpttrace
CUDA	libcudatrace
MPI + OpenMP	libompitrace[f] [1]
MPI + Pthreads	libptmpitrace[f] [1]
Mpi + CUDA	libcudampitrace[f] [1]

[1] for Fortran codes

Scalability

Scalability of Presentation

- Linpack @ Marenostrum: 10k cores x 1700 s



Scalability of analysis

Jugene

Jaguar

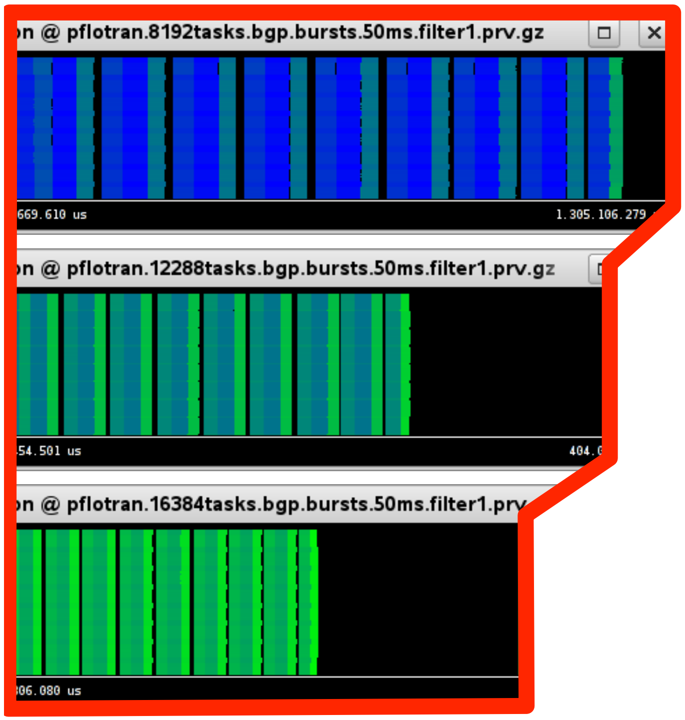
~ 105 seconds

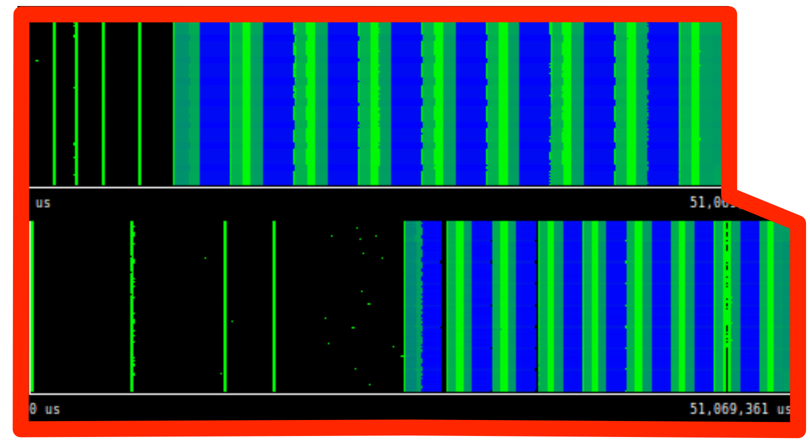
~ 47 seconds

8K cores

12K cores

16K cores





Flow Tran

Tran Flow

PFLOTRAN

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Data reduction techniques

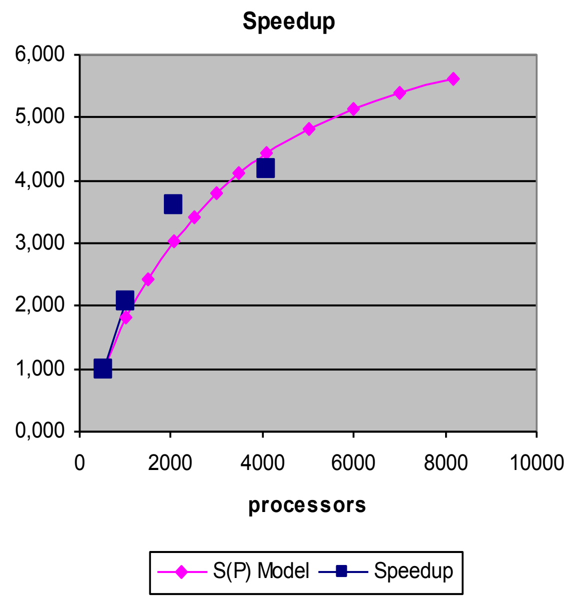
- Software counters
  - Summarize information of some event types (ie. MPI calls) by emitting aggregate counts
  - Emit counts at structurally relevant points (i.e. begin and end of long computation phases)

- 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

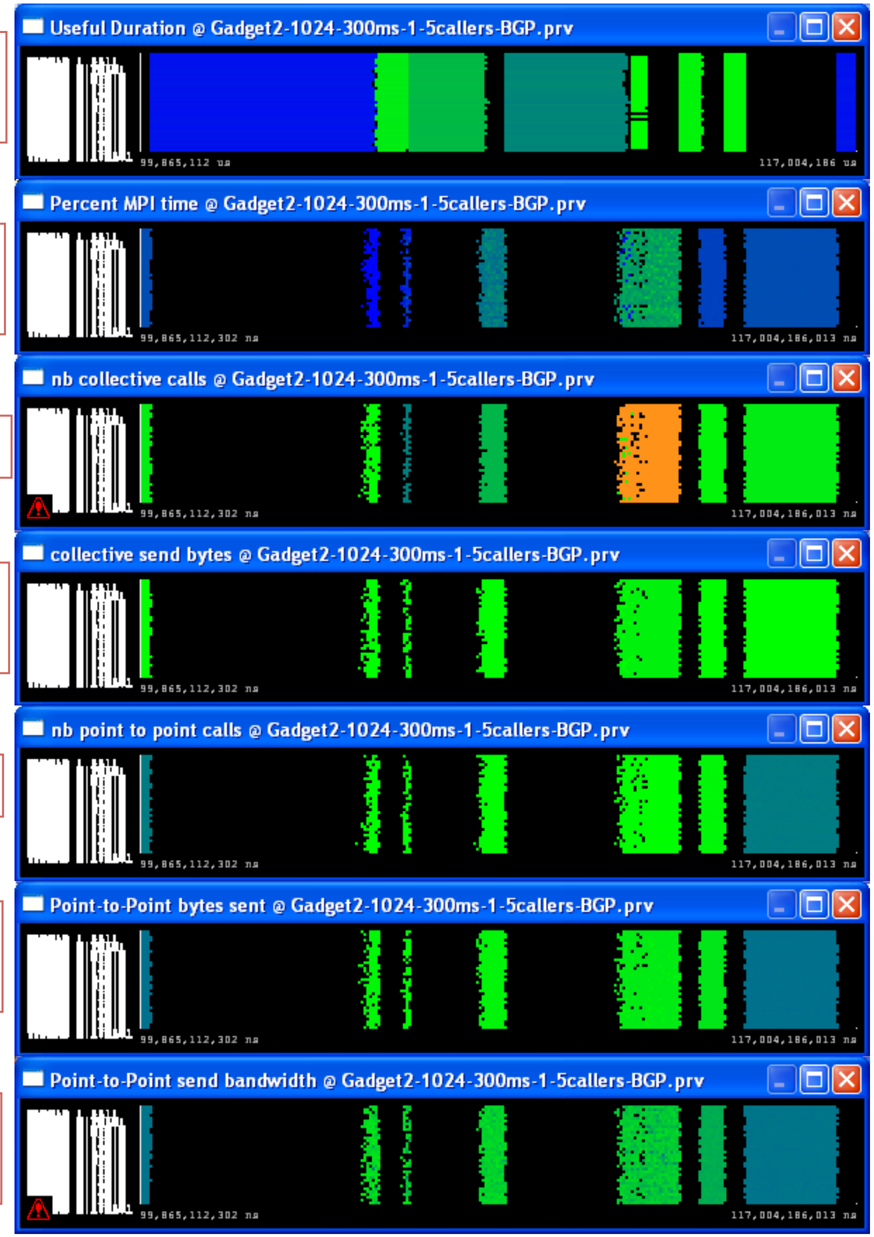
collectives

Collective bytes

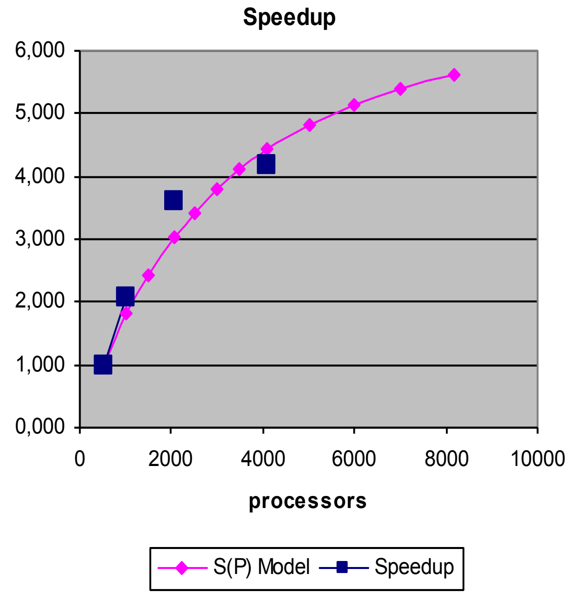
p2p

p2p bytes

p2p BW



Software counters



GADGET, PRACE Case A, 2048 procs

Useful duration

% MPI time

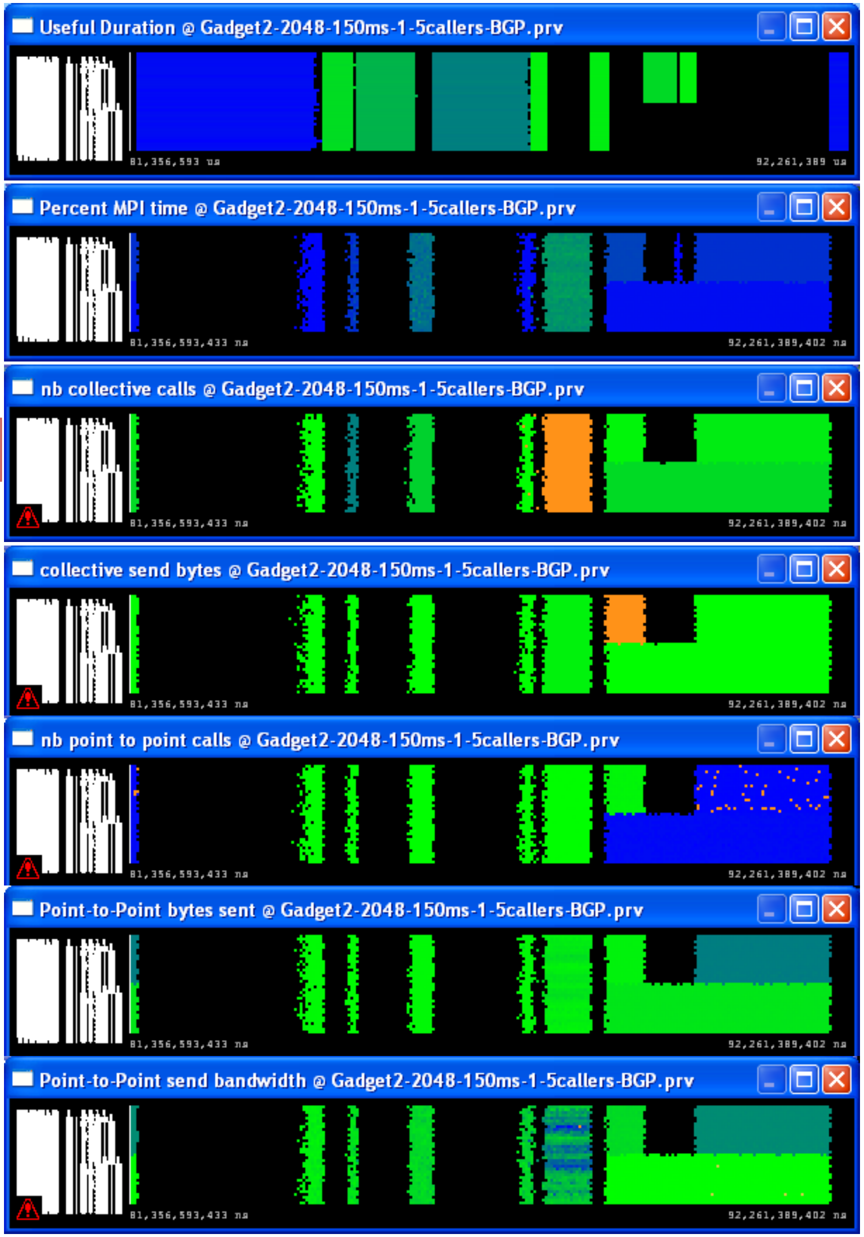
collectives

Collective bytes

p2p

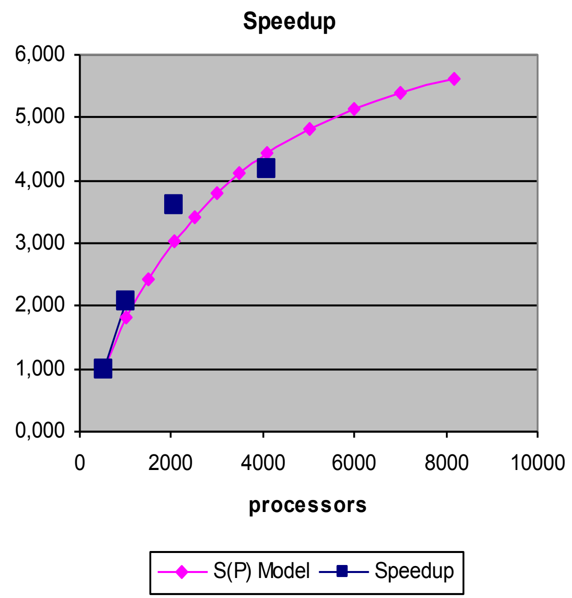
p2p bytes

p2p BW



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Software counters



GADGET, PRACE Case A, 4096 procs

Useful duration

% MPI time

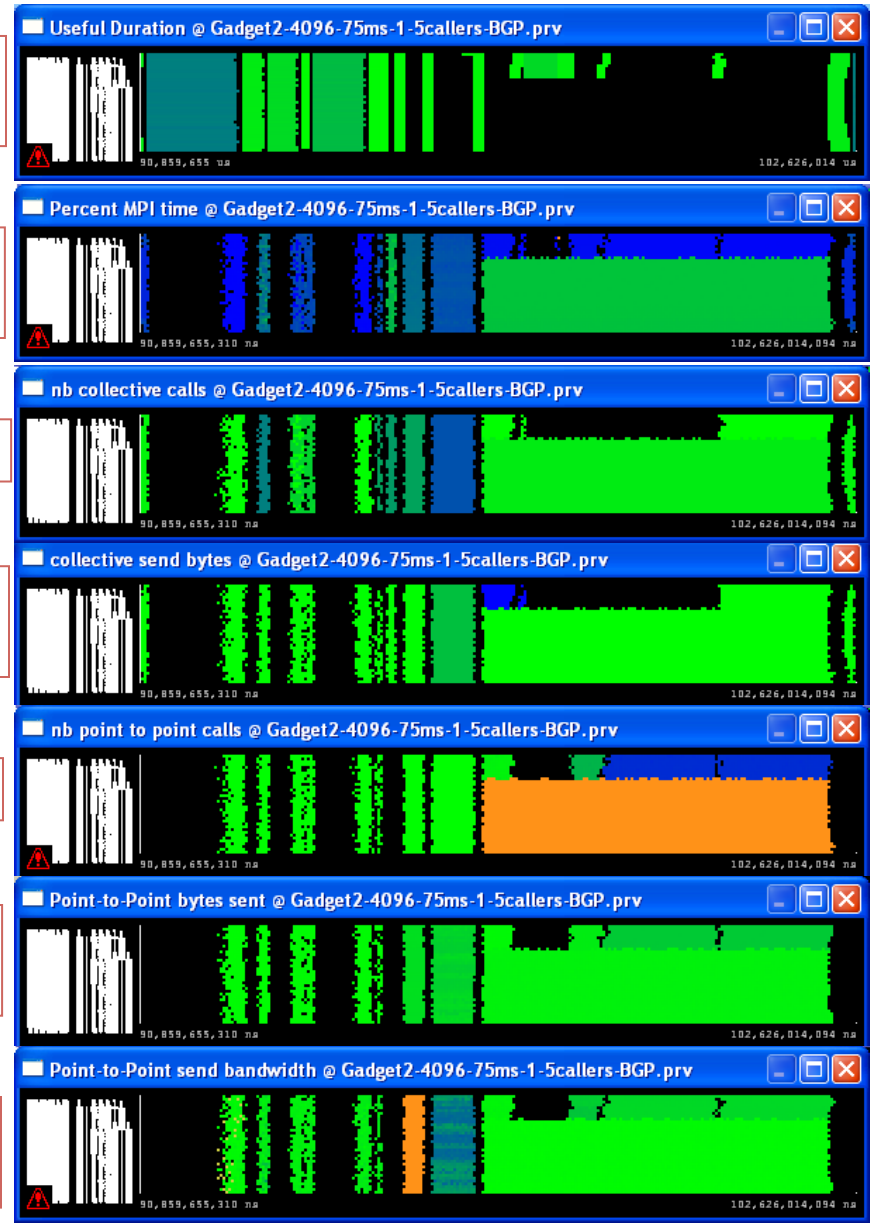
collectives

Collective bytes

p2p

p2p bytes

p2p BW



Barcelona Supercomputing Center
Centro Nacional de Supercomputación