Understanding applications with Paraver and Dimemas

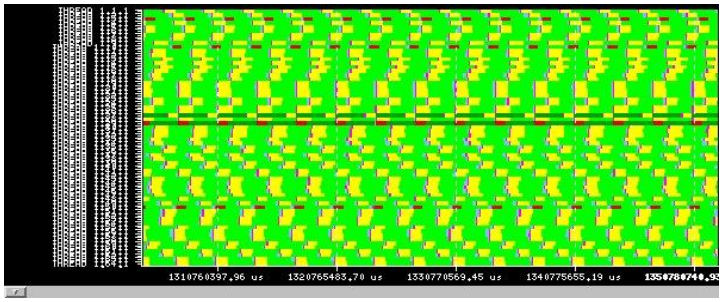
Judit Giménez, Harald Servat

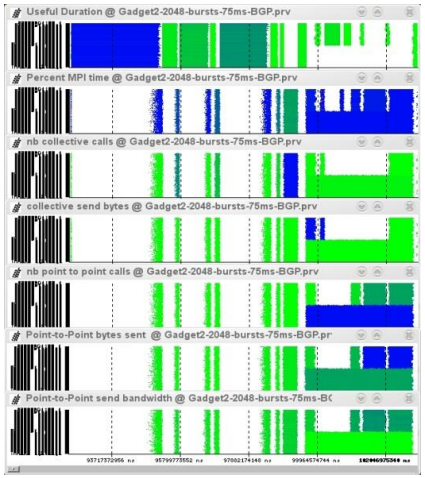
tools@bsc.es

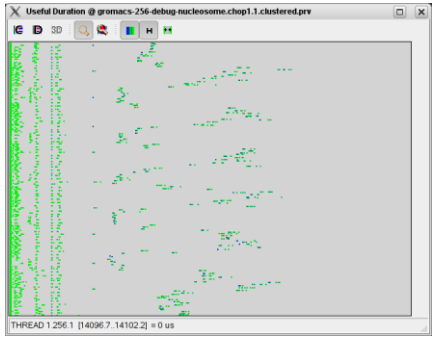
Oct 10th 2013

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

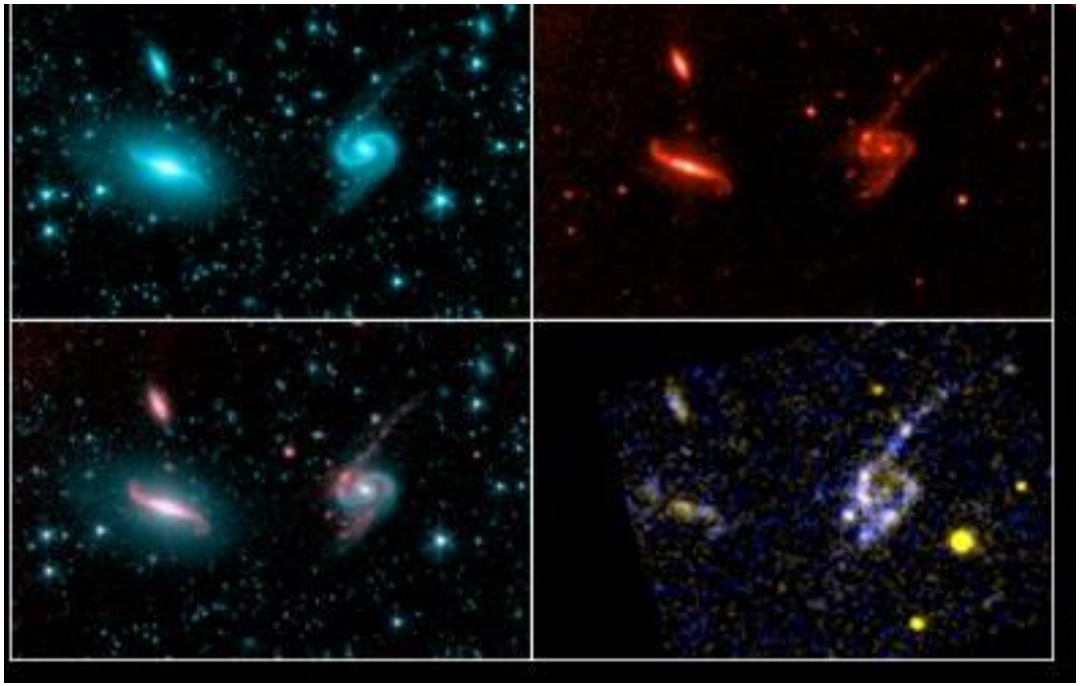


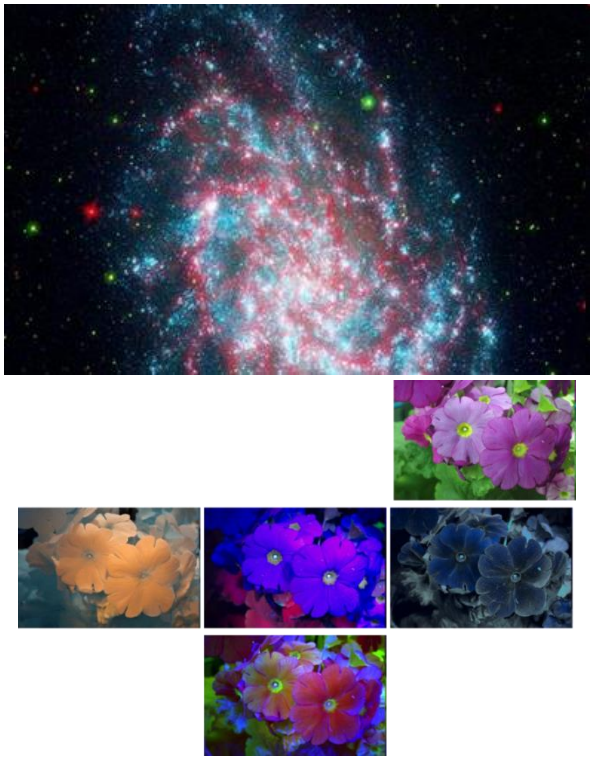




Multispectral imaging

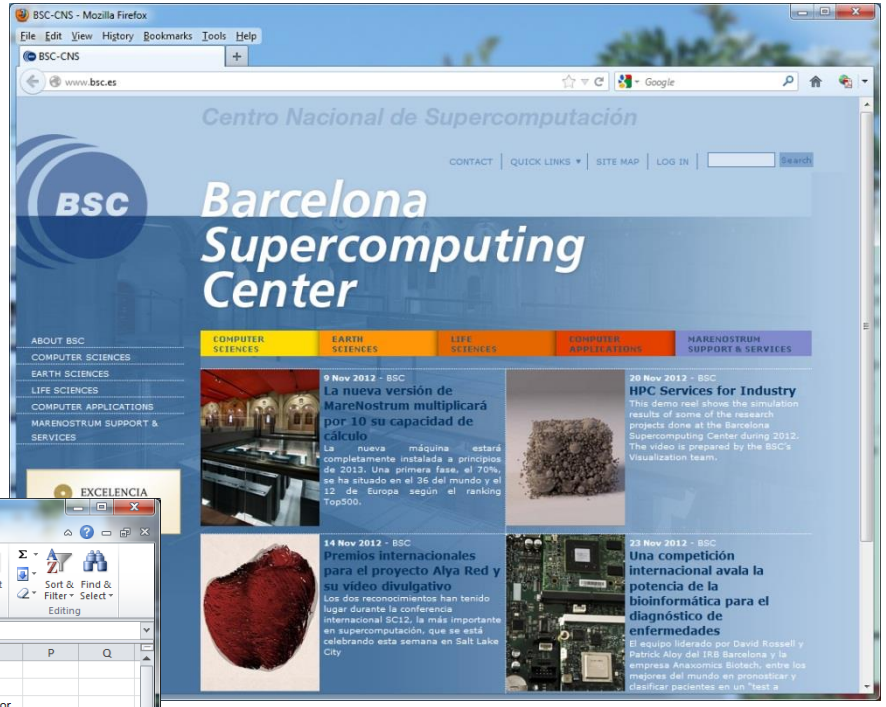
- Different looks at one reality
  - Different spectral bands (light sources and filters)
- Highlight different aspects
  - Can combine into false colored but highly informative images

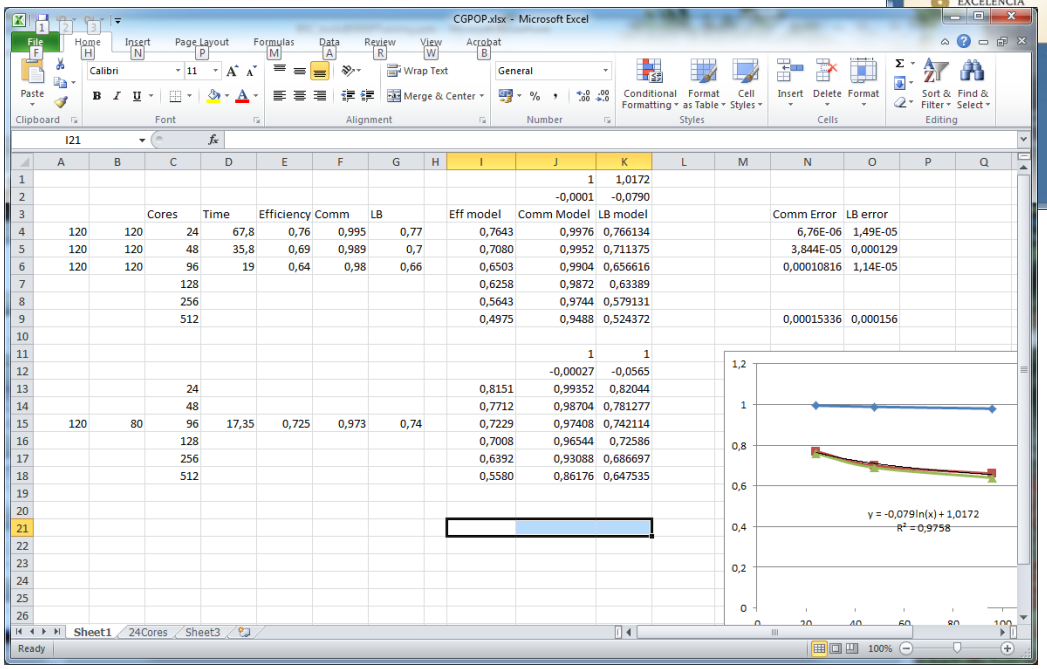




Spreadsheets and browsers

- 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.


VI-HPS
SOFTWARE
0.00 updatedt
6.62 updatex
372.85 updatelen
0.00 gene
0.00 <<iteration loop>>
293.65 genbc
PRODUCTIVITY
FAST SOLUTIONS
PAPI_L1_DCM
PAPI_L1_ICM
PAPI_L2_DCM
PAPI_L2_ICM
PAPI_L1_TCM
PAPI_L2_TCM

Paraver

- A browser …

…to manipulate (visualize, filter, cut, combine, …) ….

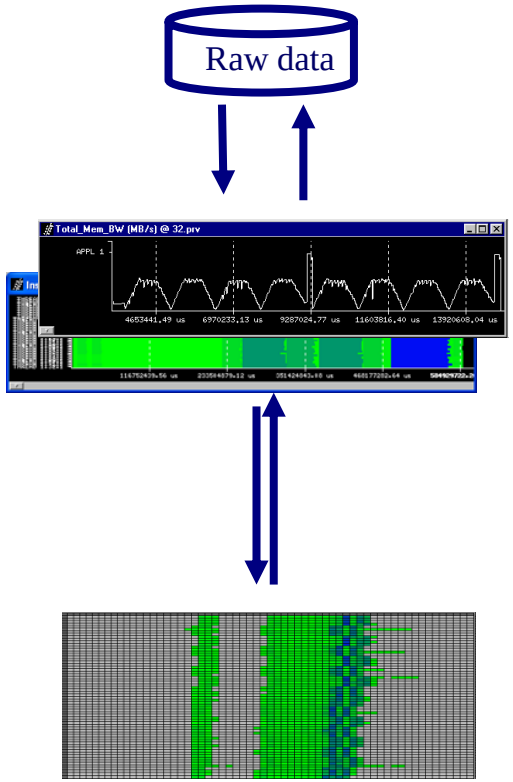
… sequences of time-stamped events …

… 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

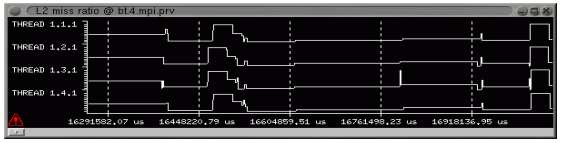
Programmable

Comparative analyses

Multiple traces

Synchronize scales

Timelines



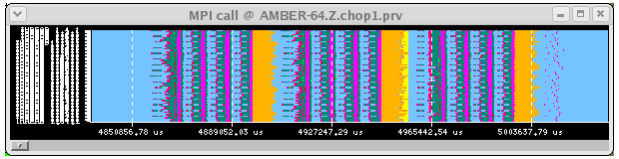
- Each window displays one view
  - **Piecewise constant** function of time

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

- Types of functions
  - Categorical

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

    - State, user function, outlined routine
  - Logical

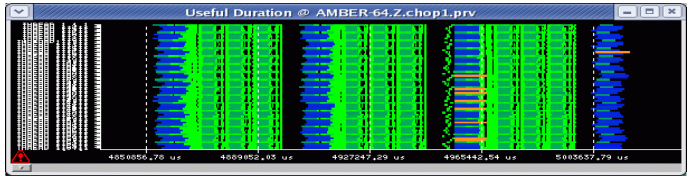


$$S_i \in \{0, 1\}$$

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

$$S_i \in R$$

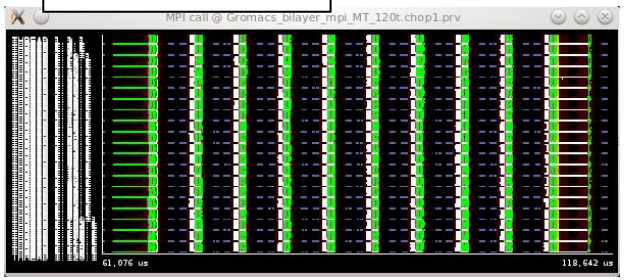
    - IPC, L2 miss ratio, Duration of MPI call, duration of computation burst



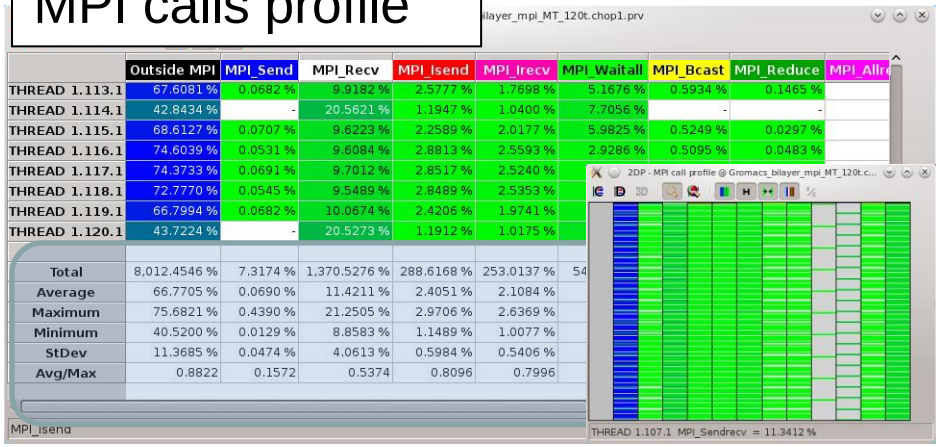
Tables: Profiles, histograms, correlations

- From timelines to tables

MPI calls

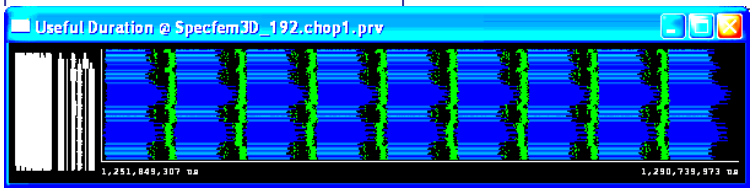


MPI calls profile

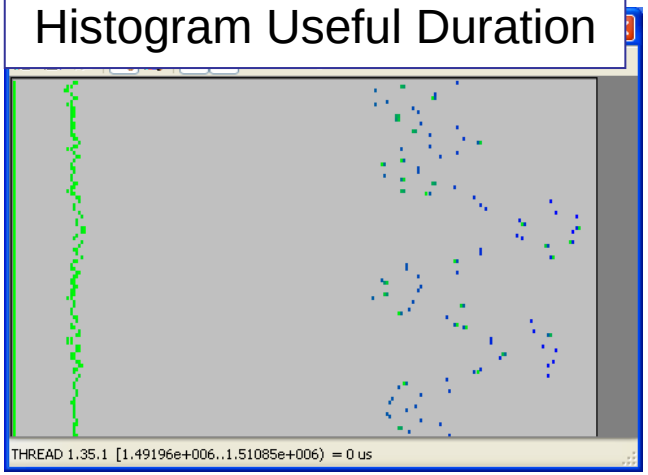


	Outside MPI	MPI_Send	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Bcast	MPI_Reduce	MPI_Allr
THREAD 1.113.1	67.6081 %	0.0682 %	9.9182 %	2.5777 %	1.7698 %	5.1676 %	0.5934 %	0.1465 %	
THREAD 1.114.1	42.8434 %	-	20.5621 %	1.1947 %	1.0400 %	7.7056 %	-	-	
THREAD 1.115.1	68.6127 %	0.0707 %	9.6223 %	2.2589 %	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				

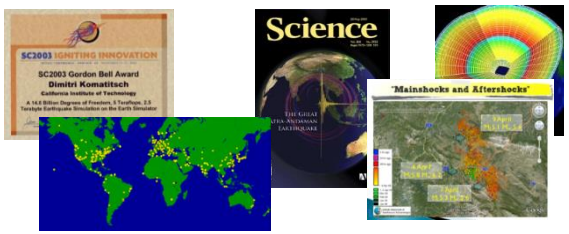
Useful Duration

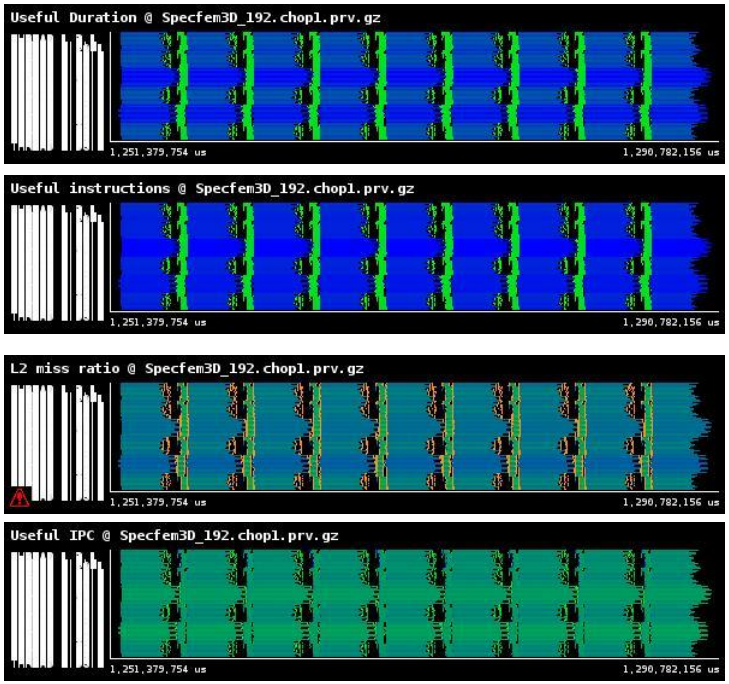


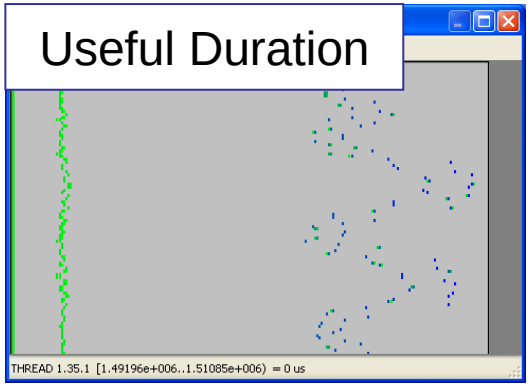
Histogram Useful Duration

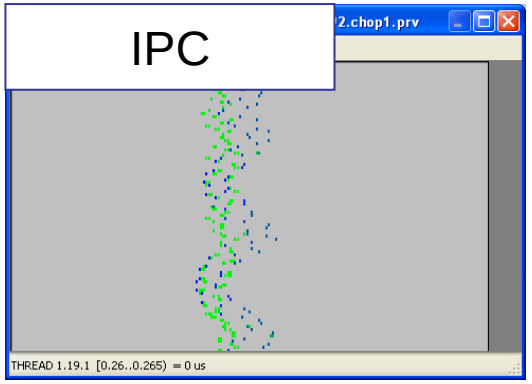


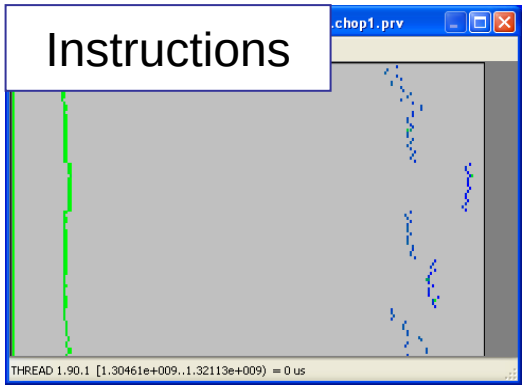
Analyzing variability through histograms and timelines

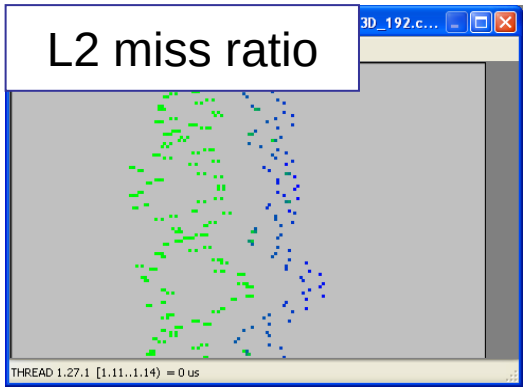






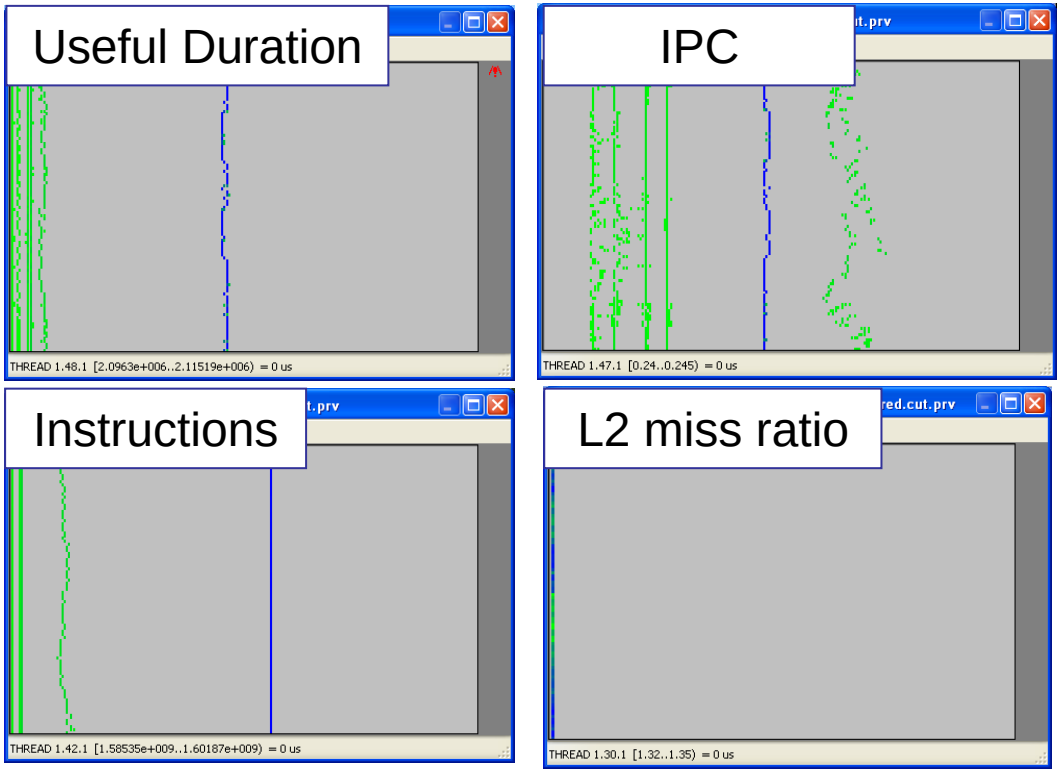






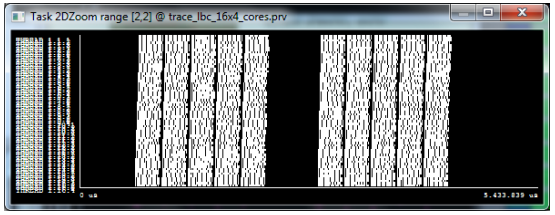
Analyzing variability through histograms and timelines

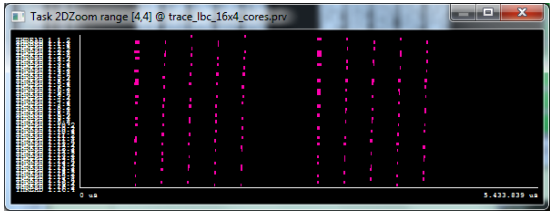
- By the way: six months later ….



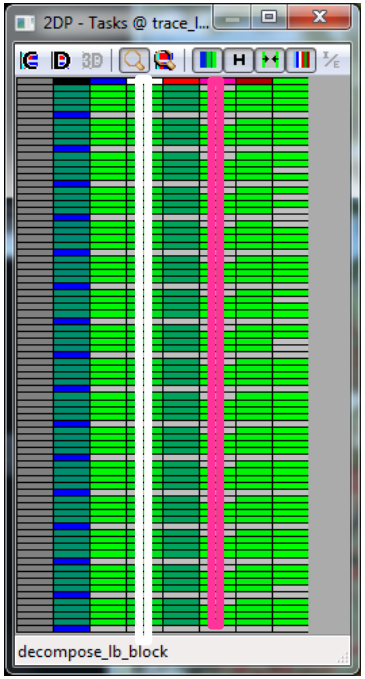
From tables to timelines

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



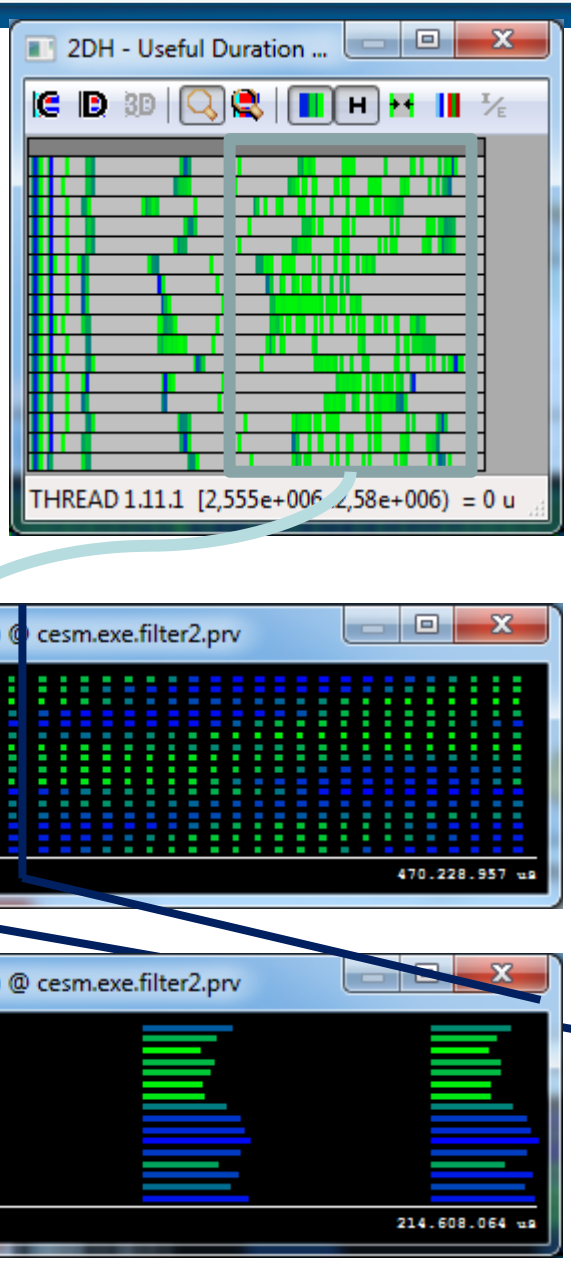


Only showing when a given value happens



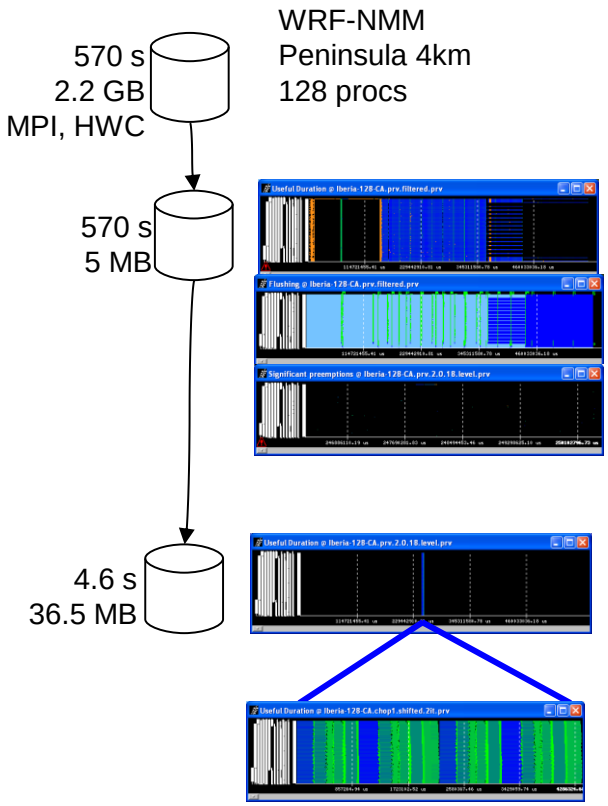
Variability … is everywhere

- 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 ? ☺



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,…

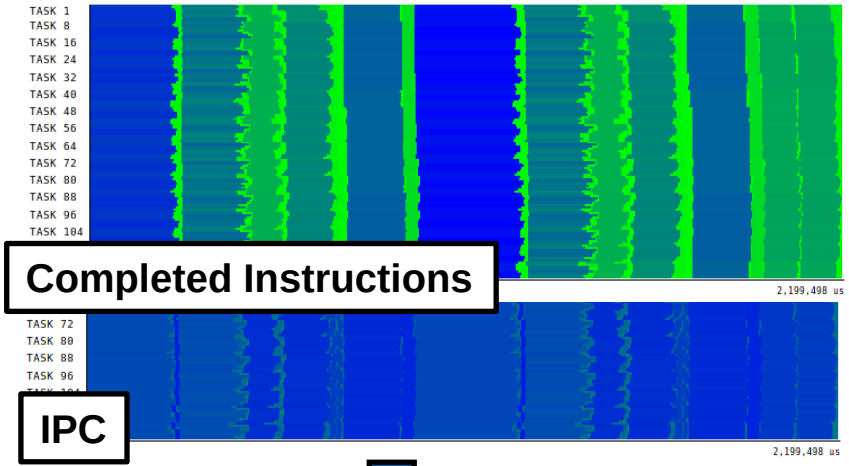


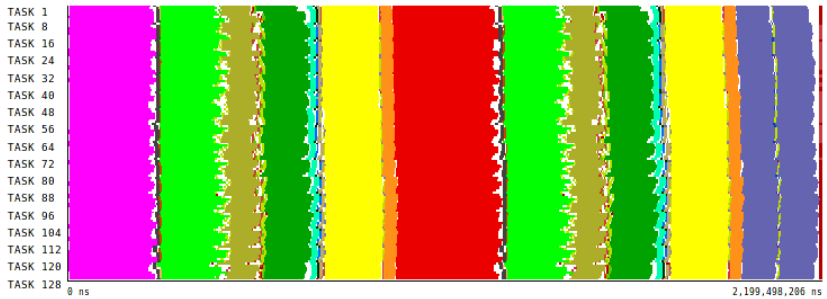
See slides at end of presentation for details

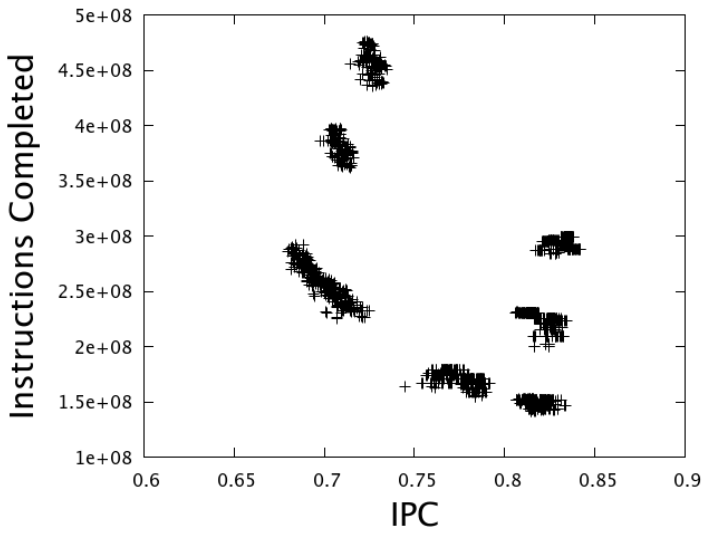
Performance Analytics

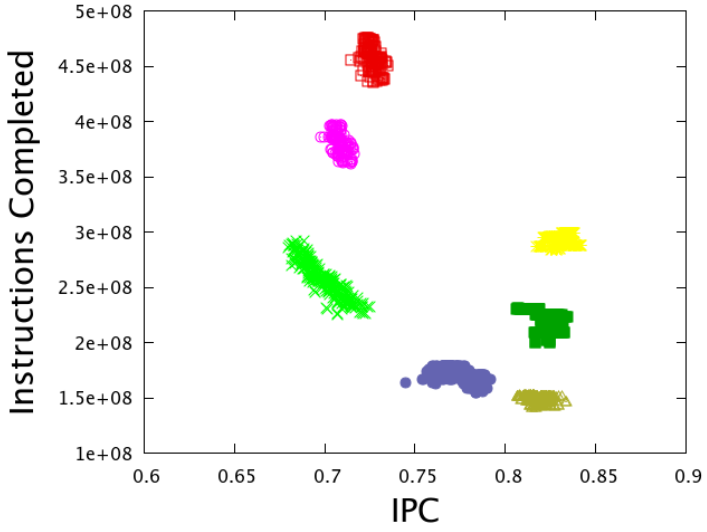
Clustering

Using Clustering to identify structure

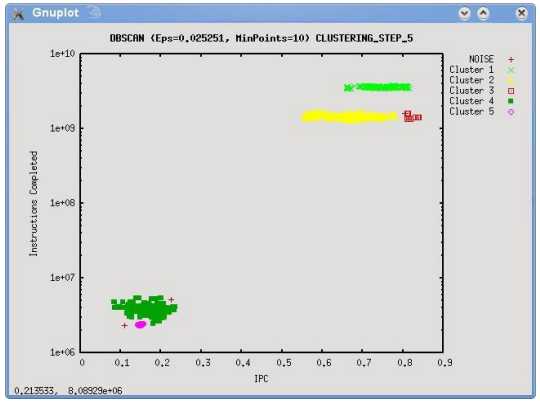




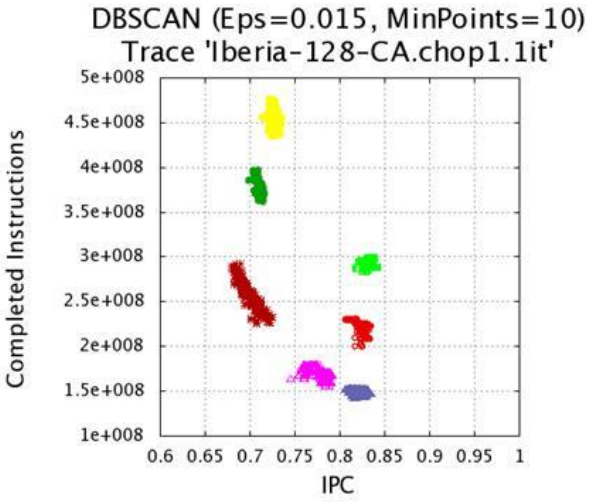




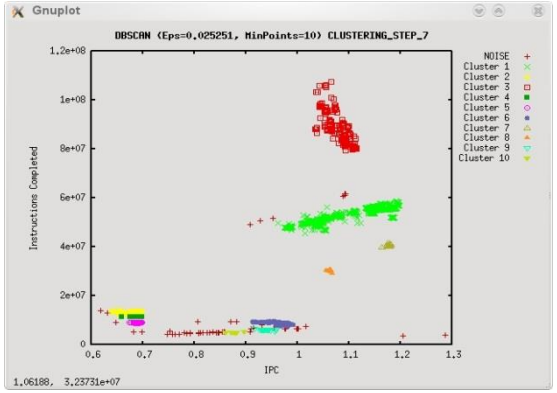
Performance @ serial computation bursts



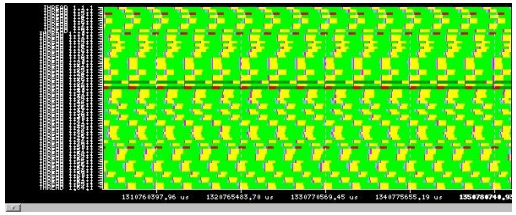
SPECFEM3D



WRF 128 cores

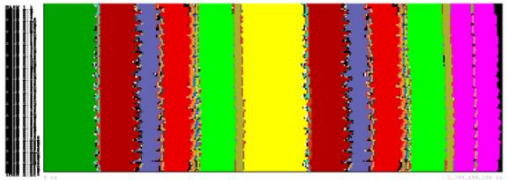


GROMACS



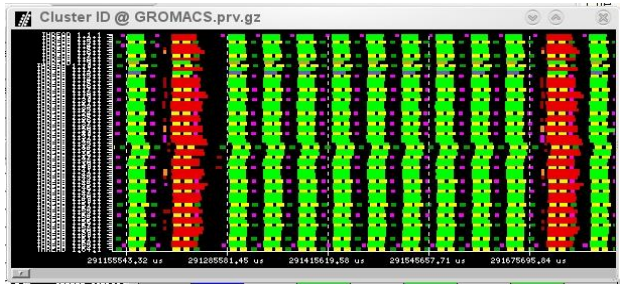
Asynchronous SPMD

Balanced #instr
variability in IPC



SPMD
Repeated substructure

Coupled imbalance

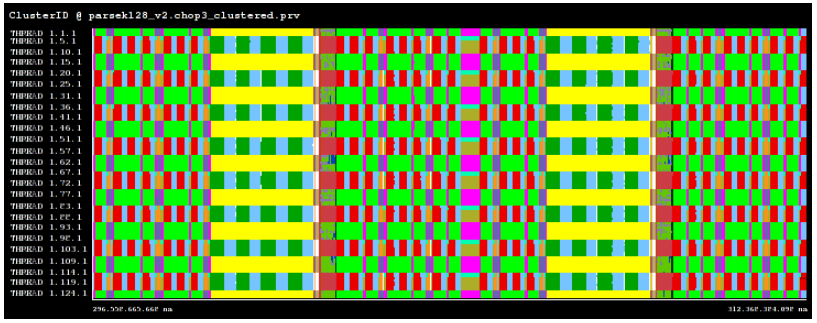


MPMD structure

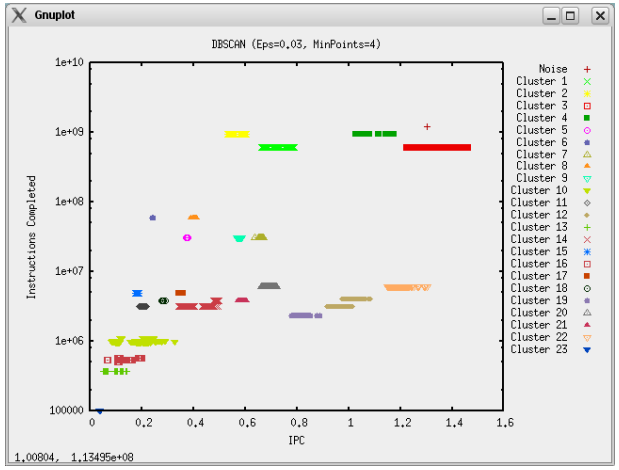
Different coupled
imbalance trends

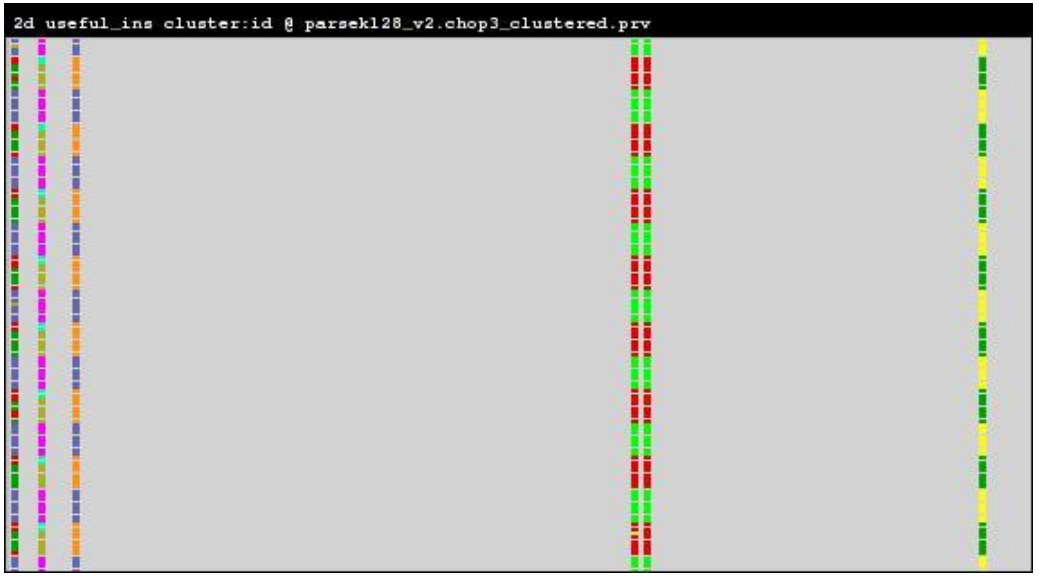
Example PARSEK (DEEP)

duration vs. cluster



instr. vs. cluster

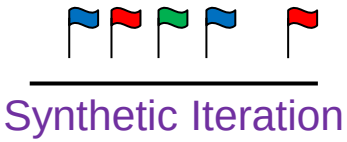


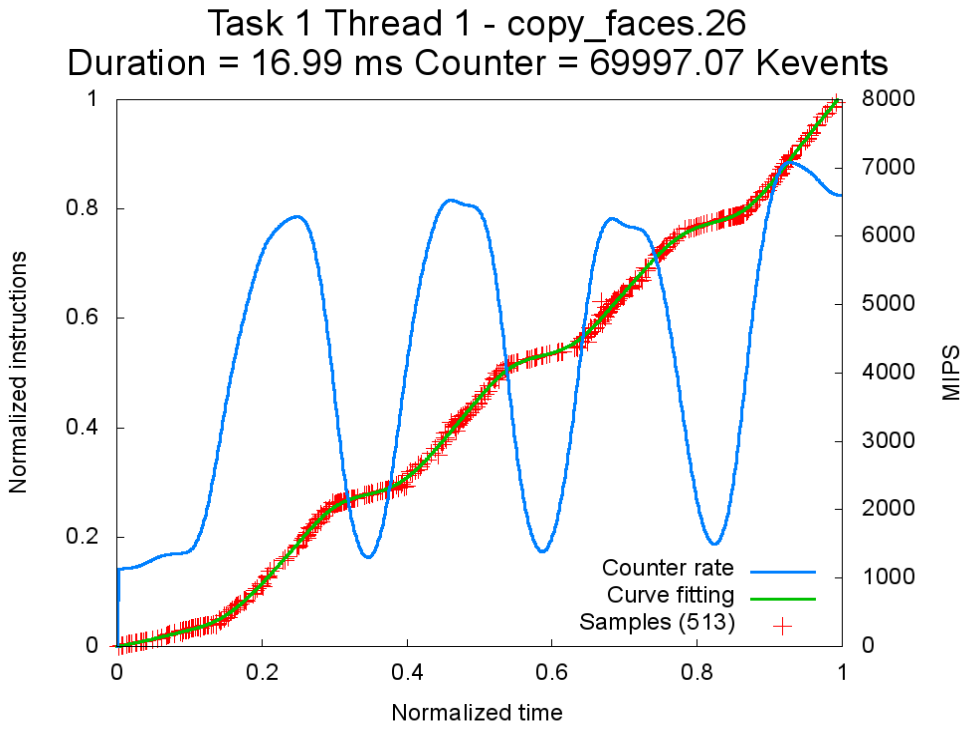


Folding

Folding: Detailed time evolution

- Benefit from applications' repetitiveness

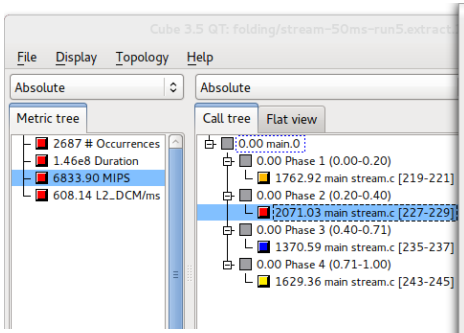


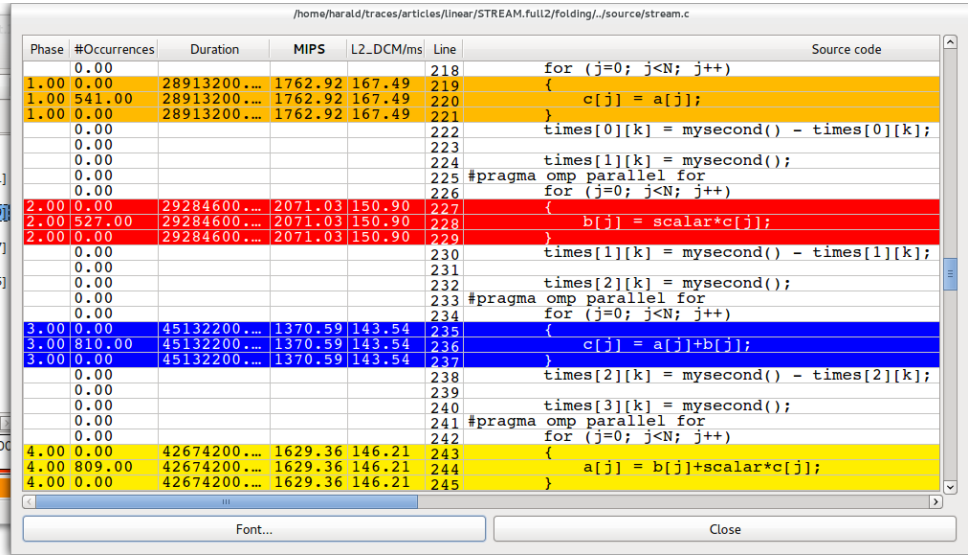


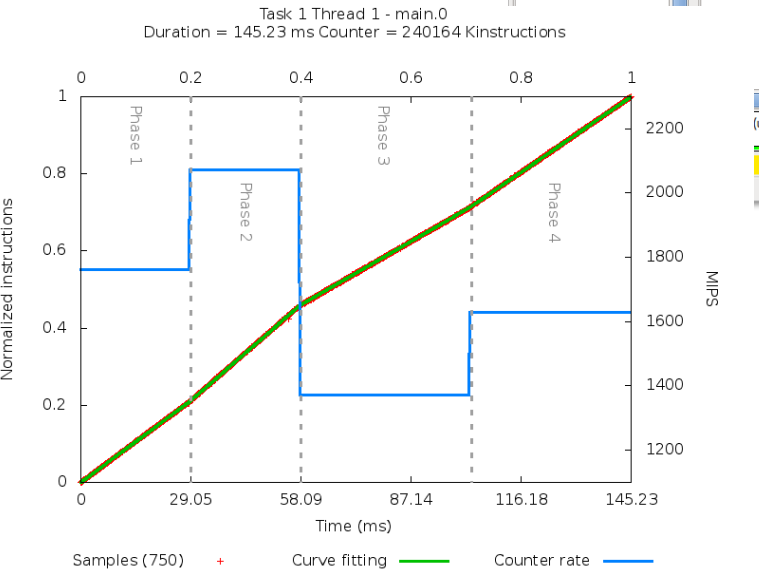
Unveiling Internal Evolution of Parallel Application Computation Phases (ICPP 2011)

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





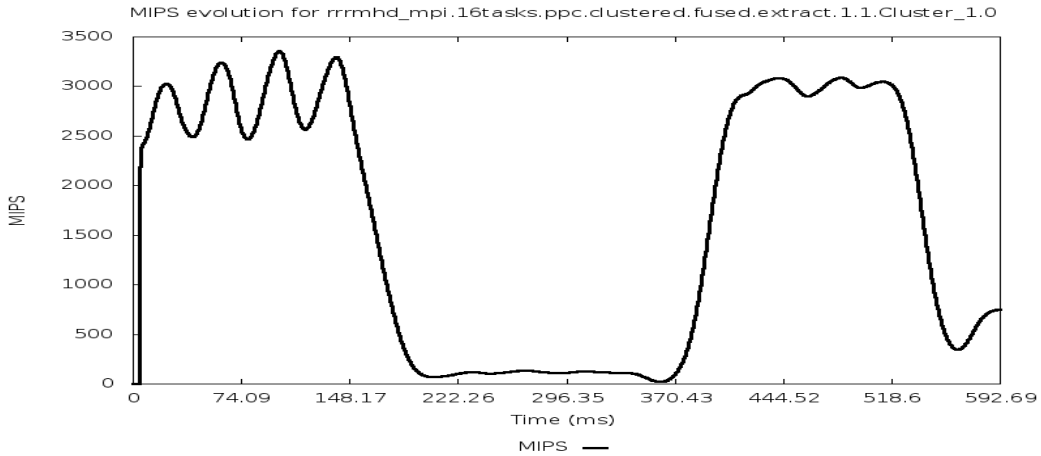


What is a good performance?

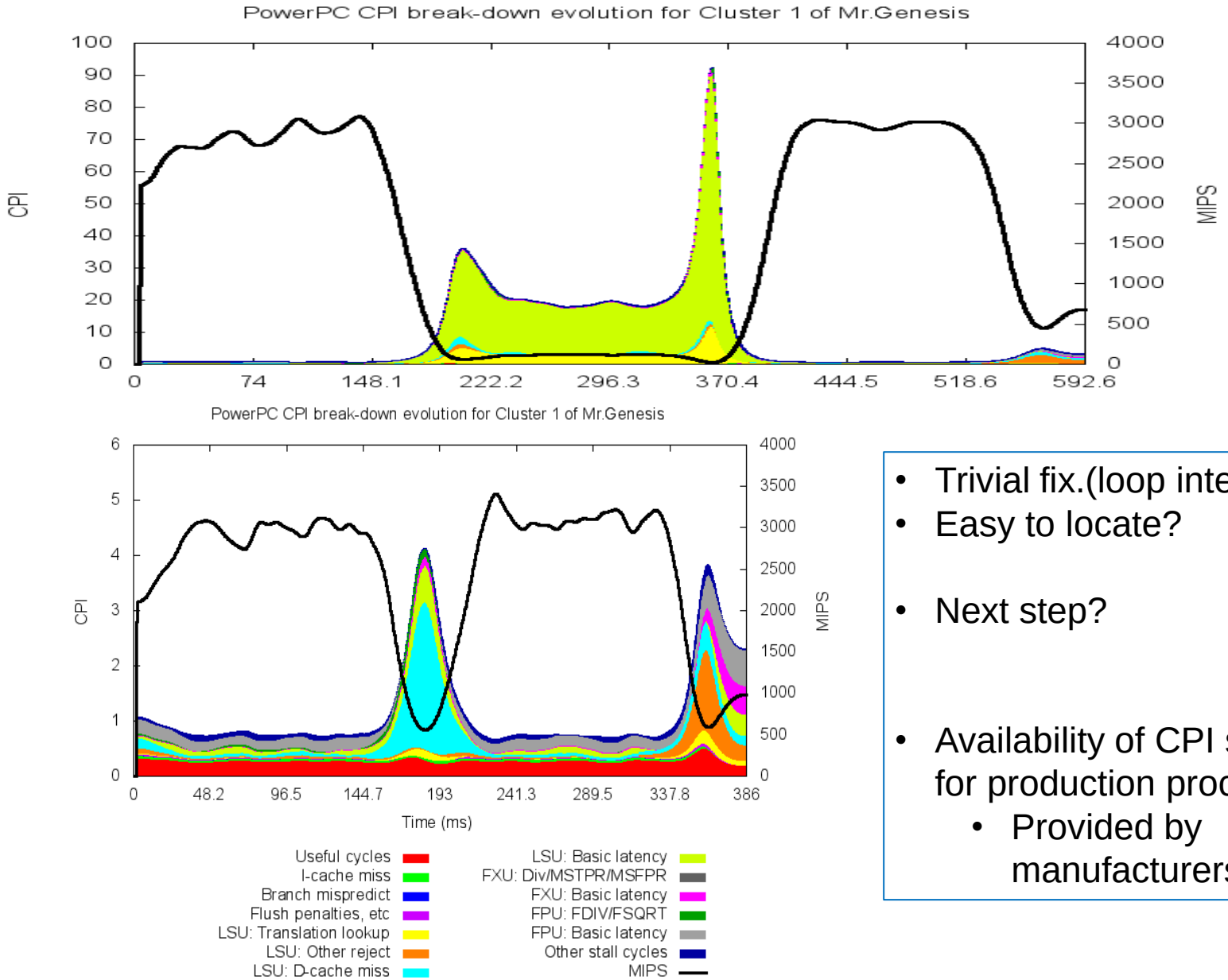
- Performance of a sequential region = 2000 MIPS

  Is it good enough?

  Is it easy to improve?



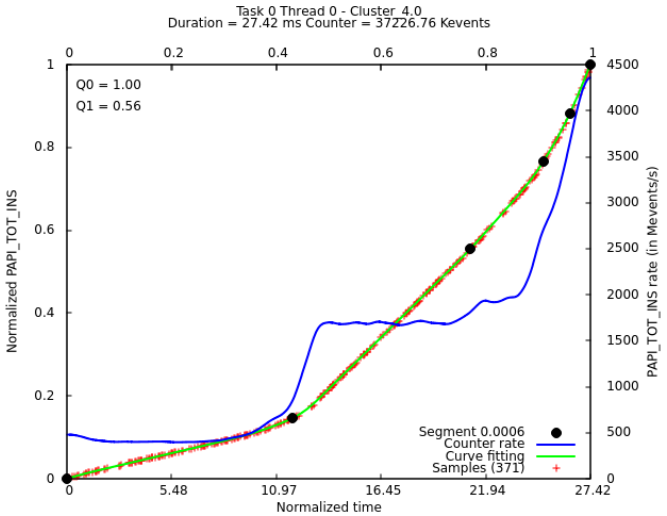
Instantaneous CPI stack



MRGENESIS

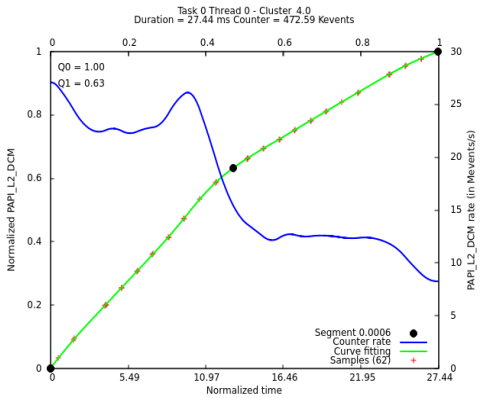
- Trivial fix.(loop interchange)
- Easy to locate?
- Next step?
- Availability of CPI stack models for production processors?
  - Provided by manufacturers?

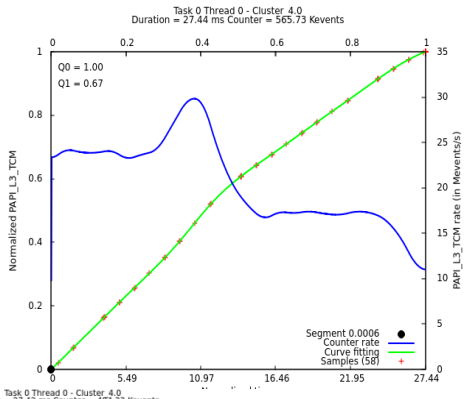
Correlating counters

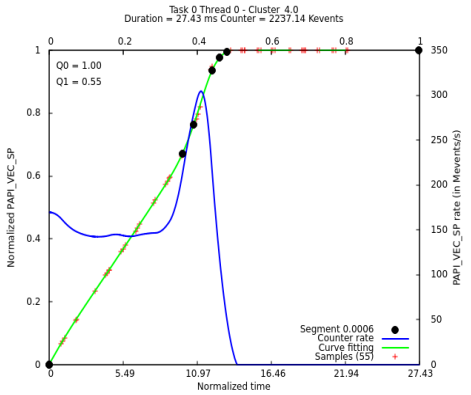


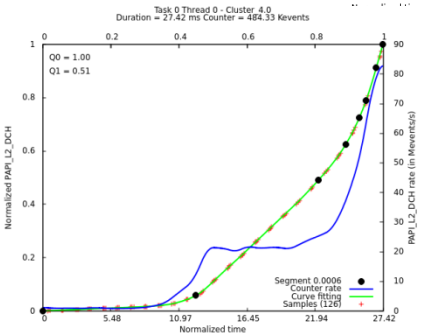
CG-POP

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

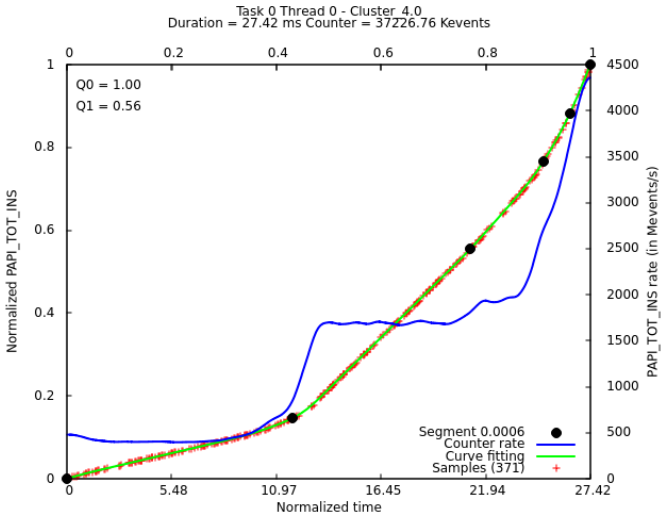






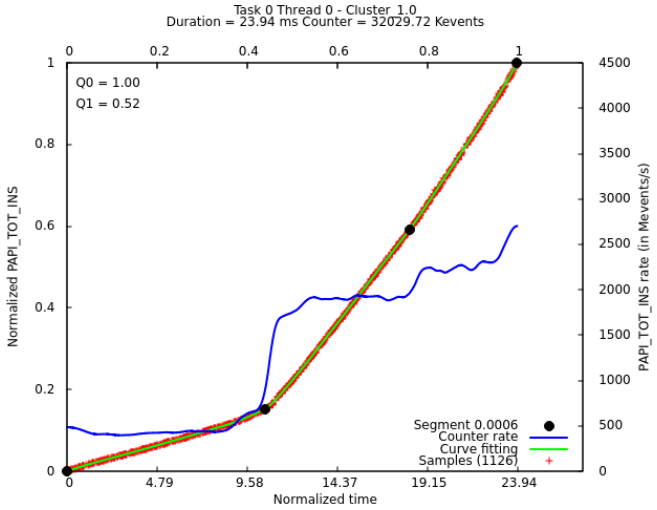


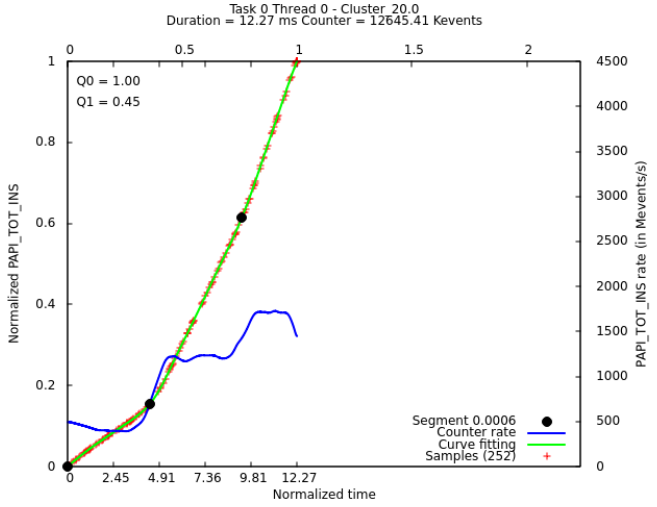
Correlating counters



CG-POP

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



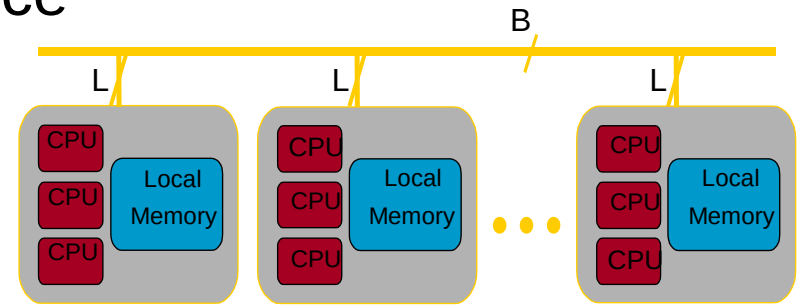



VI-HPS
SOFTWARE
FAST SOLUTIONS
PRODUCTIVITY
PAPI_L1_DCM
PAPI_L1_ICM
PAPI_L2_DCM
PAPI_L2_ICM
PAPI_L1_TCM
PAPI_L2_TCM

Dimemas

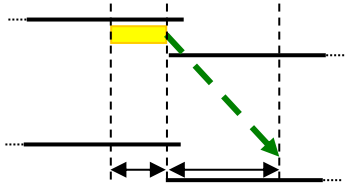
Dimemas: Coarse grain trace driven simulator

- Key factors influencing performance
  - Abstract architecture
  - Basic MPI protocols
  - No attempt to model details

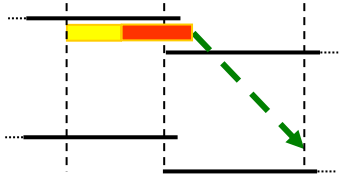


- Objectives
  - Simple / general, Fast simulations
- Linear components
  - Point to point communication
  - Sequential processor performance (global CPU speed, per block/subroutine

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



- Non-linear components
  - Synchronization semantics (blocking receives, rendezvous)
  - Resources contention (CPU, links half/full duplex, busses)

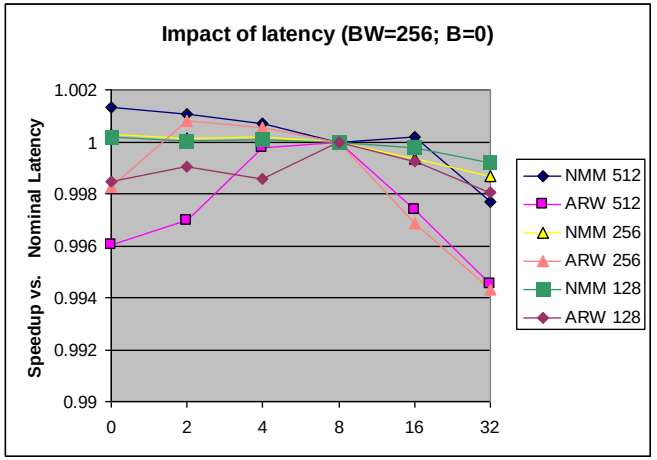


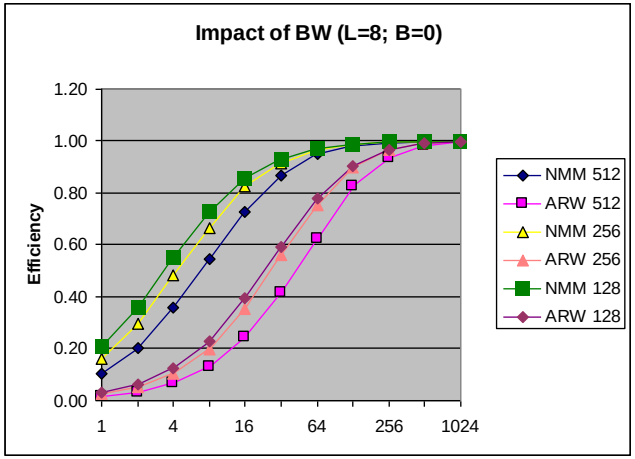
Dimemas vs. Paraver traces

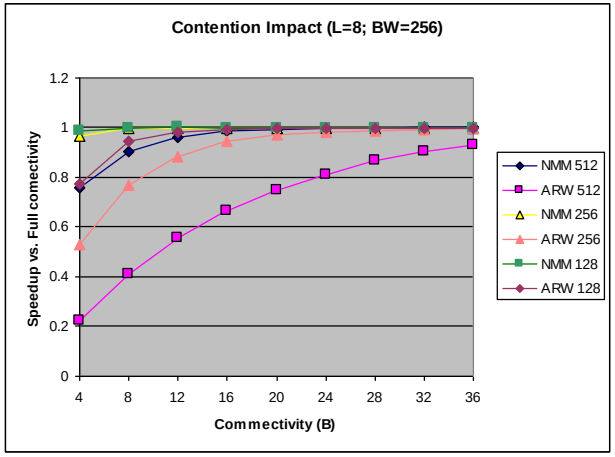
- Paraver trace: what happens when
  - Actual wall clock time of events
- Dimemas trace: sequence of resource demands
  - Duration of computation bursts
  - Type of communication, partners and bytes
- Can be generated from Paraver trace
  - prv2dim input.prv output.dim
- Dimemas generates as output a Paraver file of the simulated run

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

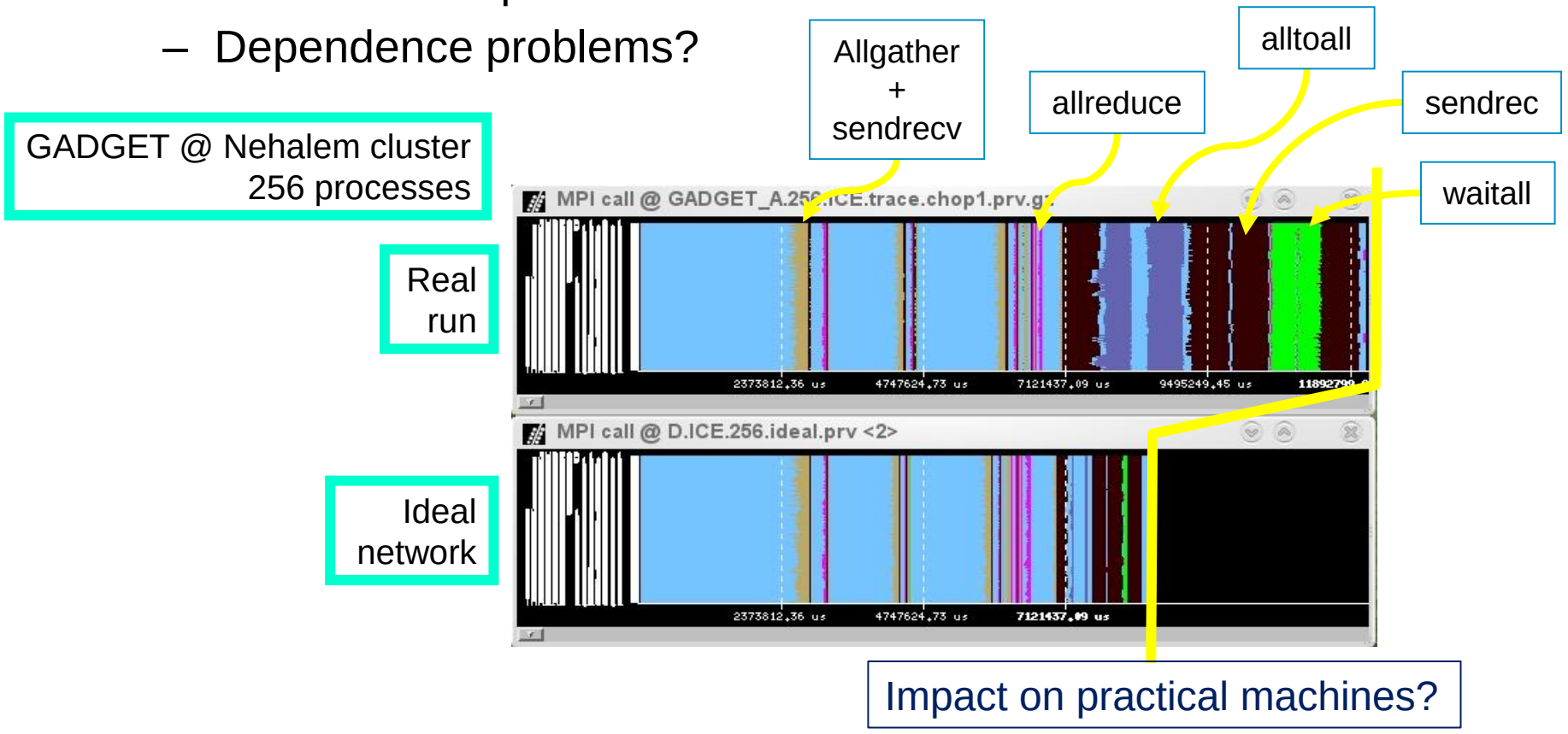






Ideal network machine (instantaneous communication)

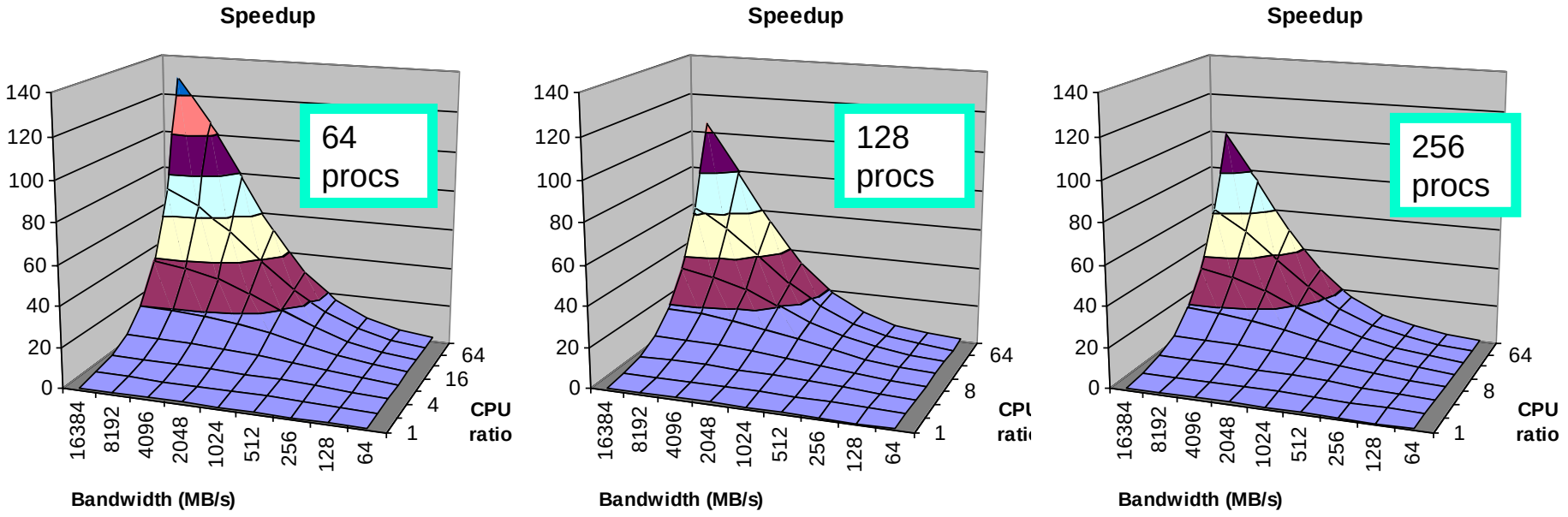
- The impossible machine: BW = ∞, L = 0
- Actually describes/characterizes Intrinsic application behavior
  - Load balance problems?
  - Dependence problems?



Impact of architectural parameters

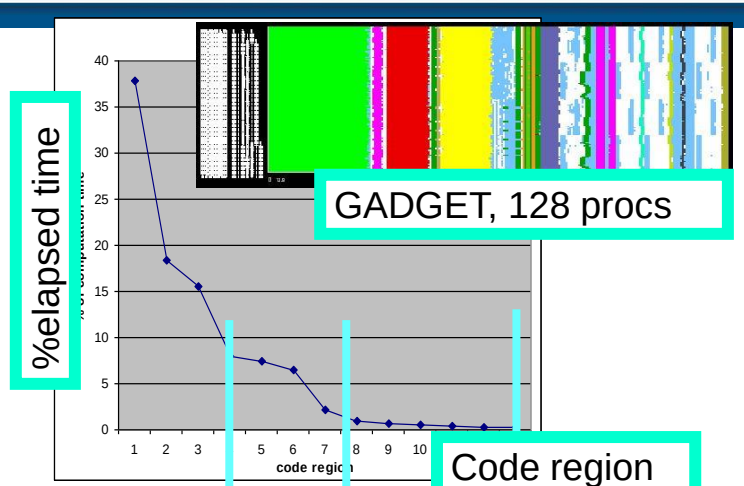
- **Ideal speeding up ALL** the computation bursts by the CPUratio factor
  - The more processes the less speedup (higher impact of bandwidth limitations) !!

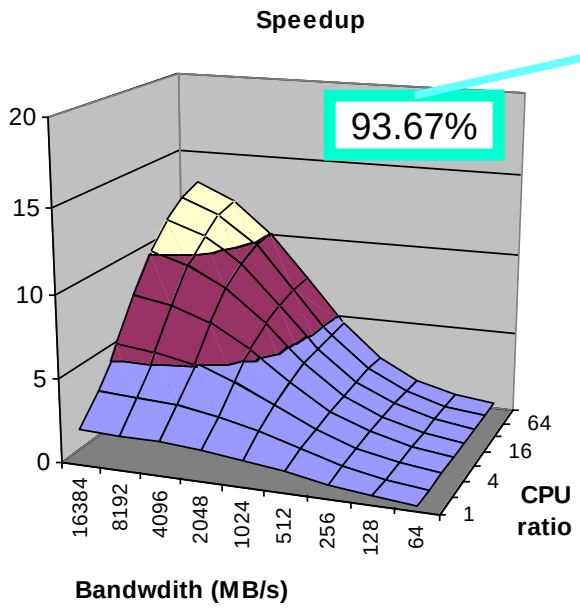
GADGET

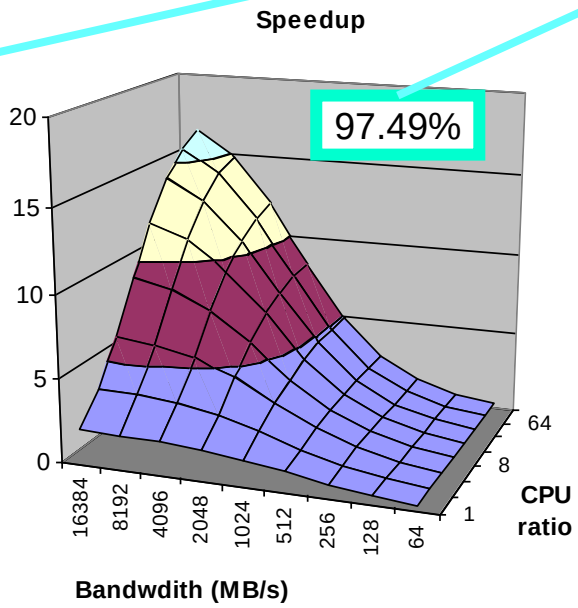


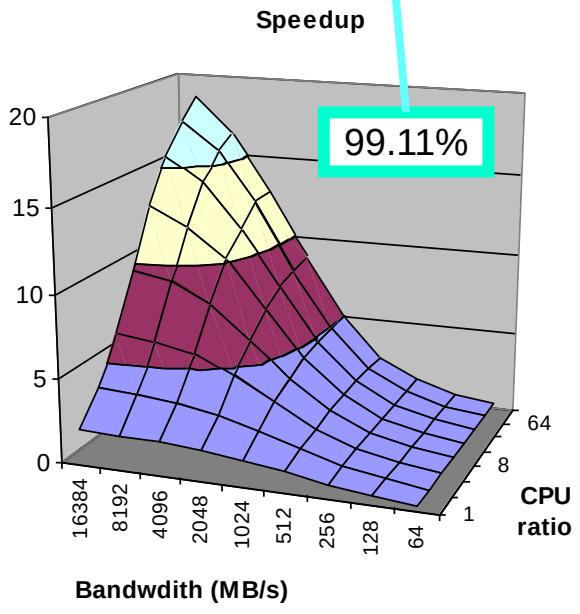
The potential of hybrid/accelerator parallelization

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

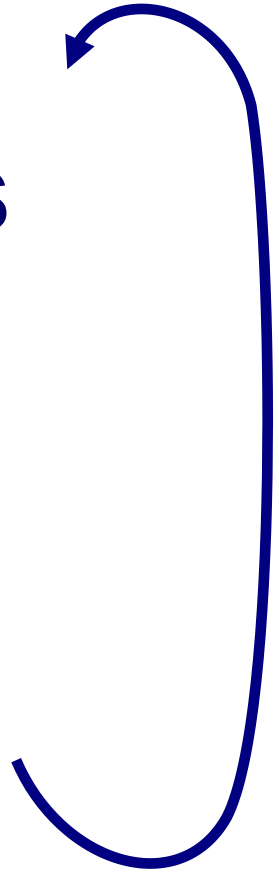








Methodology



Help generate hypotheses

Help validate hypotheses

Qualitatively

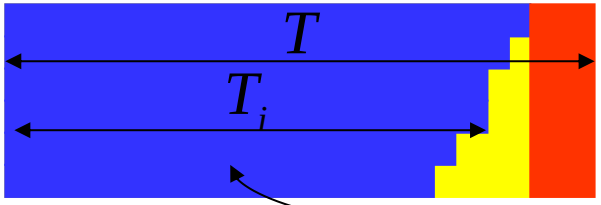
Quantitatively

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, correlate with other counters
- Scalability – code replication?
  - Total #instructions
- Behavioral structure? Variability?

Scaling model

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



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

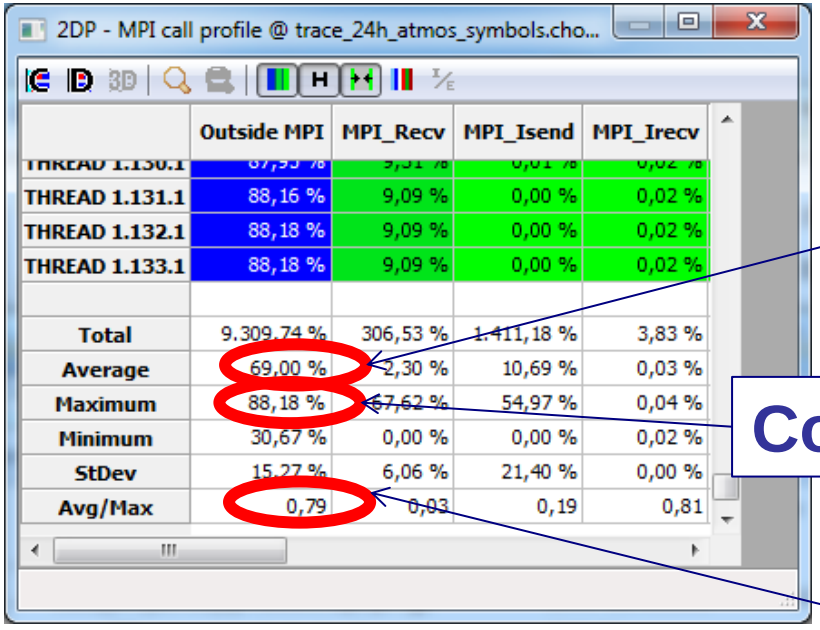
$$CommEff = \max(eff_i)$$

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

IPC

instr

Directly from real execution metrics



η

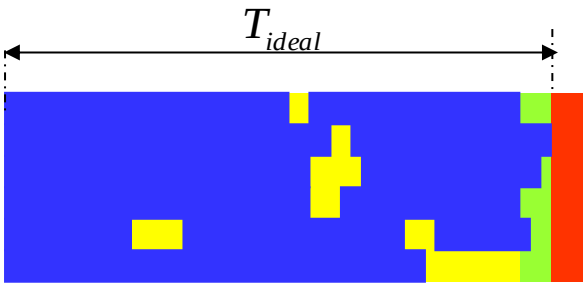
CommEff

LB

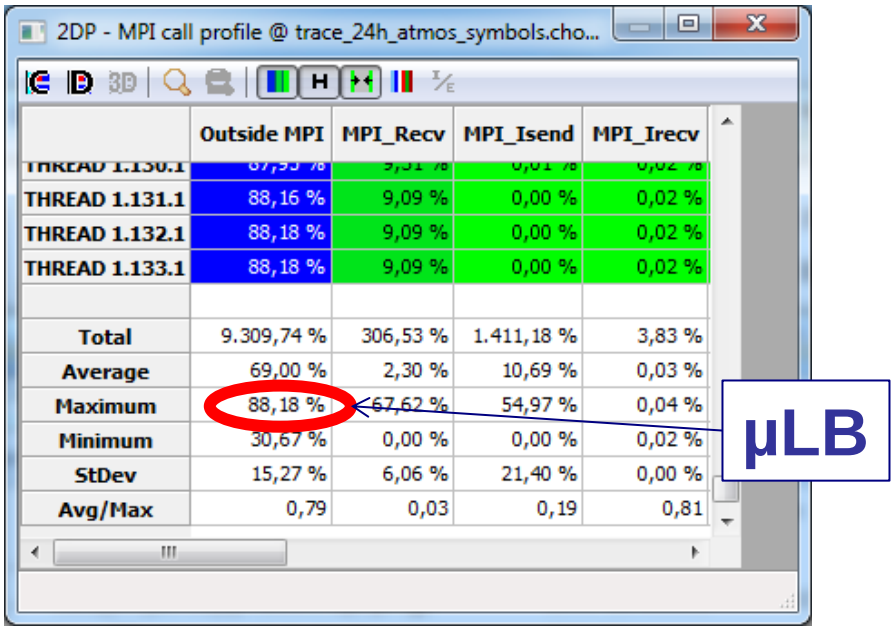
Scaling model

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

$$CommEff = \mu LB * Transfer$$



Migrating/local load imbalance
Serialization



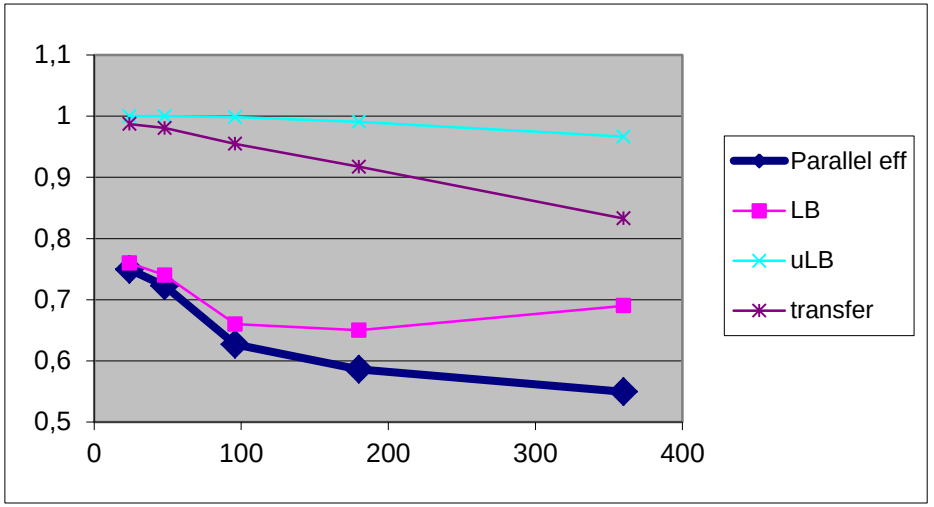
$$\mu LB = \frac{\max(T_i)}{T_{ideal}} \qquad Transfer = \frac{T_{ideal}}{T}$$

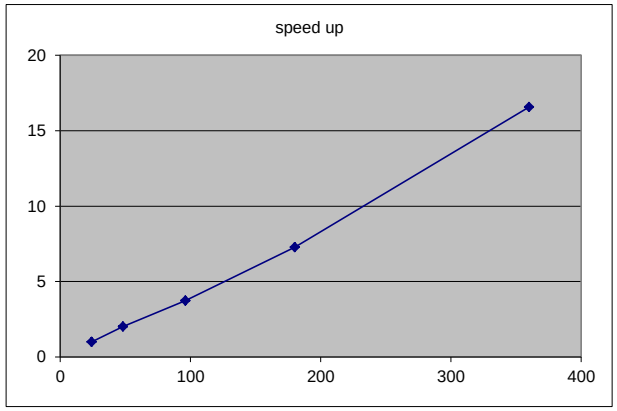
Modelling efficiency

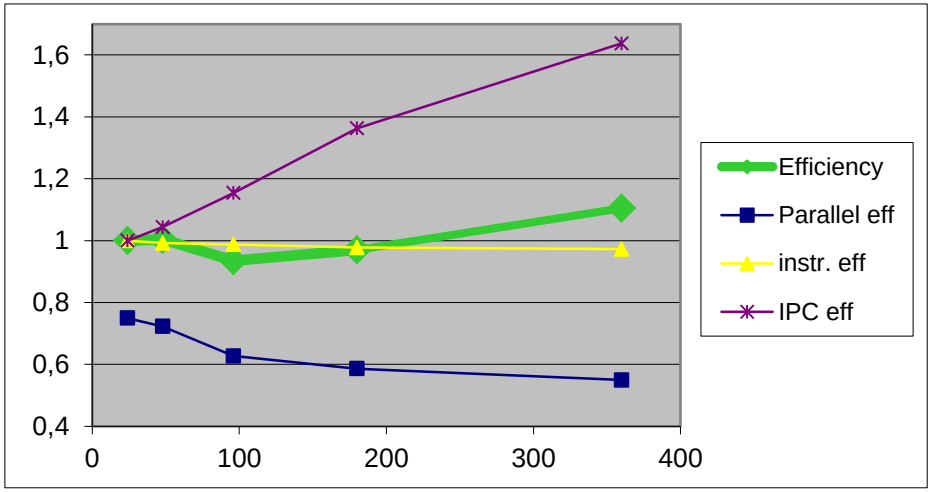
$$\eta_{\parallel} = LB * \mu LB * Transfer$$

CG-POP mpi2s1D - 180x120

Good scalability !!
Should we be happy?







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

BSC Tools web site

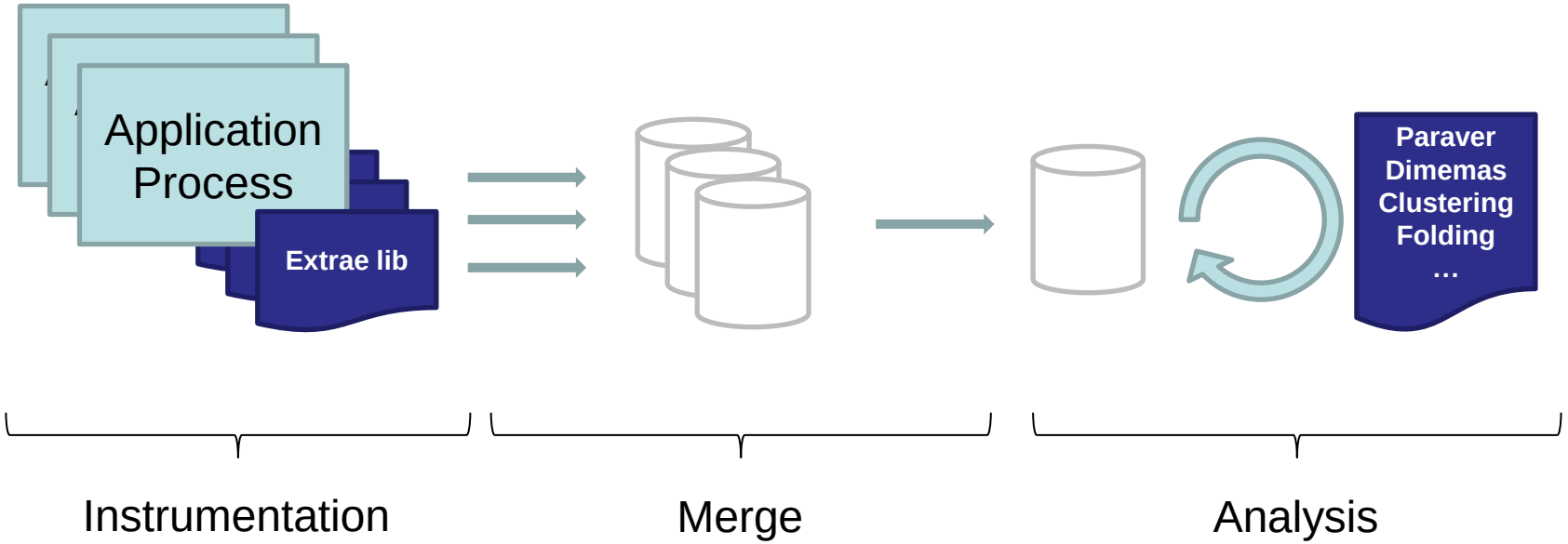
- www.bsc.es/paraver
  - downloads
    - Sources / Binaries
    - Linux / windows / MAC
  - documentation
    - Training guides
    - Tutorial slides

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

Short Paraver Demo

BSC tools info for Juqueen & Juropa

Trace Generation Workflow



BSC tools material

- USB or BSC Tools account ~train036
  - in Juqueen and Juropa

- Content
  - Binaries for your laptops
    - Paraver (Linux, Windows, and Mac)
    - Dimemas (Linux only)
    - ClusteringSuite (Linux only)
  - Tutorials
  - Extrae Documentation
  - Slides
  - Traces
  - 2 application examples: CG-POP, GTC

BSC tools installation

- BSC Tools account is ~train036
  - in Juqueen and Juropa

- To use the tools
  - source ~train036/env-bsc-tools

Instrumentation in Juqueen

- Currently, only MPI statically linked programs can be traced

- To facilitate compilation
  - Add to your link stage
    - $(EXTRAE_LIBS) for C/C++ apps
    - $(EXTRAE_LIBS_F) for Fortran apps

- We provide 2 full-examples
  - GTC2, CGPOP
    - In ~train036/analysis-apps

Examples for Juqueen (II)

- Within ~train036/sample-extrae-job directory
  - job.ll – Execution of the instrumented application
    - Load-Leveler Submission script.
      - Executes the application, and,
      - Submits the merging process
  - merge.ll – Merging process to generate a Paraver trace-file
    - Automatically launched through job.ll
  - extrae/extrae.xml – Extrae configuration

Examples for Juqueen (III)

- job.ll

```
#@ job_name  = TRACE
#@ …
#@ queue

export BINARY= # Write the binary here

export EXTRAE_CONFIG_FILE=extrae/extrae.xml
runjob –n N –exe ${BINARY} –env-all

llsubmit merge.ll
```

Examples for Juqueen (IV)

- merge.ll

```
#@ job_name  = MERGE
#@ …
#@ queue

export TRACEFILE= # Write here the name of the resulting tracefile
export BINARY= #Write here the name of the binary

source ~train036/env-bsc-tools
runjob –n 16 : $EXTRAE_HOME/bin/mpimpi2prv -f TRACE.mpits
   -e ${BINARY} –o ${TRACEFILE}
```

Extrae configuration (I)

- extrae.xml

```
<counters enabled="yes">
   <cpu enabled="yes" starting-set-distribution="cyclic">
      <set enabled="yes" domain="all" changeat-globalops="125">
PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L1_DCM,PAPI_TLB_DM,PAPI_LD_INS,PAPI_BR_INS,PAPI_BR_MSP,PAPI_
FP_OPS
      </set>
      <set enabled="yes" domain="all" changeat-globalops="125">
PAPI_TOT_INS,PAPI_TOT_CYC,PEVT_AXU_FP_DS_ACTIVE_CYC,PEVT_IU_IL1_MISS_CYC,PEVT_IU_RELOAD_
COLLISIONS_CYC,PEVT_IU_IU0_REDIR_CYC,PEVT_IU_IERAT_MISS_CYC
      </set>
      <set enabled="yes" domain="all" changeat-globalops="125">
PAPI_TOT_INS,PAPI_TOT_CYC,PEVT_IU_ICACHE_FETCH_CYC,PEVT_IU_INSTR_FETCHED_CYC,PEVT_IU_FX
U_ISSUE_PRI_LOSS_CYC,PEVT_IU_RAW_DEP_HIT_CYC
      </set>
      …
      <set enabled="yes" domain="all" changeat-globalops="125">
PAPI_TOT_INS,PAPI_TOT_CYC,PEVT_L1P_BAS_LU_STALL_SRT_CYC,PEVT_L1P_BAS_LU_STALL_MMIO_DCR
_CYC,PEVT_L1P_BAS_LU_STALL_STRM_DET_CYC,PEVT_L1P_BAS_LU_STALL_LIST_RD_CYC,PEVT_L1P_BAS
_LU_STALL_LIST_WRT_CYC
      </set>
   </cpu>
   <network enabled="no" />
   <resource-usage enabled="no" />
   <memory-usage enabled="no" />
</counters>
```

Hardware counters

Extrae configuration (II)

- extrae.xml

```
<storage enabled="no">
        <trace-prefix enabled="yes">TRACE</trace-prefix>
        <size enabled="no">5</size>
        <temporal-directory enabled="yes">/scratch</temporal-directory>
        <final-directory enabled="yes">/gpfs/scratch/bsc41/bsc41273</final-directory>
        <gather-mpits enabled="no" />
</storage>
```

Storage options

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

Buffering options

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

MPI section

```
<sampling enabled="no" type="default" period="50m" variability="20m" />
```

Sampling

```
<callers enabled="yes">
        <mpi enabled="yes">1-3</mpi>
        <sampling enabled="yes">1-5</sampling>
</callers>
```

Callstack information

- Within ~train036/sample-extrae-job directory
  - job.moab – Execution of the instrumented application
    - Submission script.
      - Executes the application,
  - merge.moab – Merging process to generate a Paraver trace-file
    - It is NOT automatically launched through job.moab
  - extrae/extrae.xml – Extrae configuration
  - extrae/trace.sh – intercept execution with tracing library

```
#!/bin/bash
source ~/train036/env-bsc-tools
export EXTRAE_CONFIG_FILE=extrae/extrae.xml
export LD_PRELOAD=$EXTRAE_HOME/lib/libmpitracef.so
$@
```

- Instrumentation through shared library preloading
  - Specific libraries for each combination of runtimes
    - MPI
    - OpenMP
    - OpenMPI+MPI
    - …

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

Examples for Juropa

- job.moab

```
#!/bin/bash –x
#MSUB –e trace.err
#MSUB –o trace.out
..

export BINARY= # Write here the binary you want to trace
export NP= # Write here the number of processes

mpiexec –np ${NP} extrae/trace.sh ${BINARY}
```

Examples for Juropa

- merge.moab

```
#!/bin/bash –x
#MSUB –e merge.err
#MSUB –o merge.out
..

export TRACEFILE= # Write here the tracefile name

source ~train036/env-bsc-tools
mpiexec –np ${NP} ${EXTRAE_HOME}/bin/mpimpi2prv –f
TRACE.mpits –o ${TRACEFILE}
```

Recommended steps

1. Follow guidelines for paraver basic navigation (Tut #1)
   Tracefile provided
2. (if Juqueen) Build your binary with extrae library
3. Adapt scripts to instrument your application
4. Obtain a tracefile
5. Follow first steps of methodology guidelines (Tut #5)
   - Parallel efficiency
   - Distribution of computations
   - Instruction balance
   - …
6. Depending on the diagnosis use clustering, folding, dimemas…


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
THANKS

Detailed material

Semantic Module

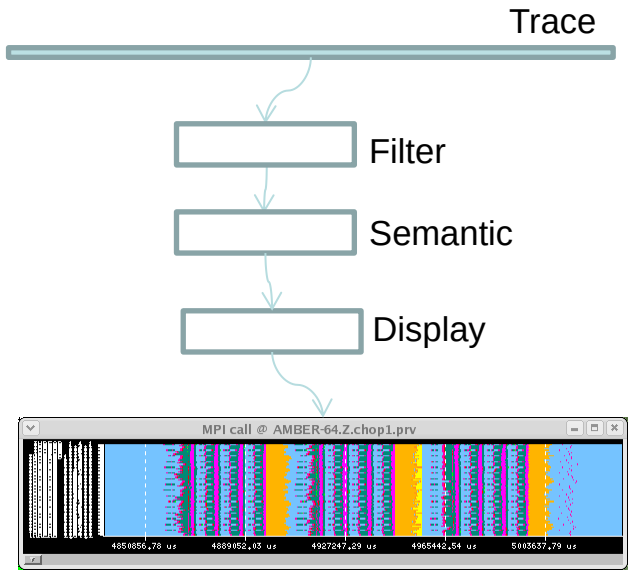
Basic functions of time

- 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

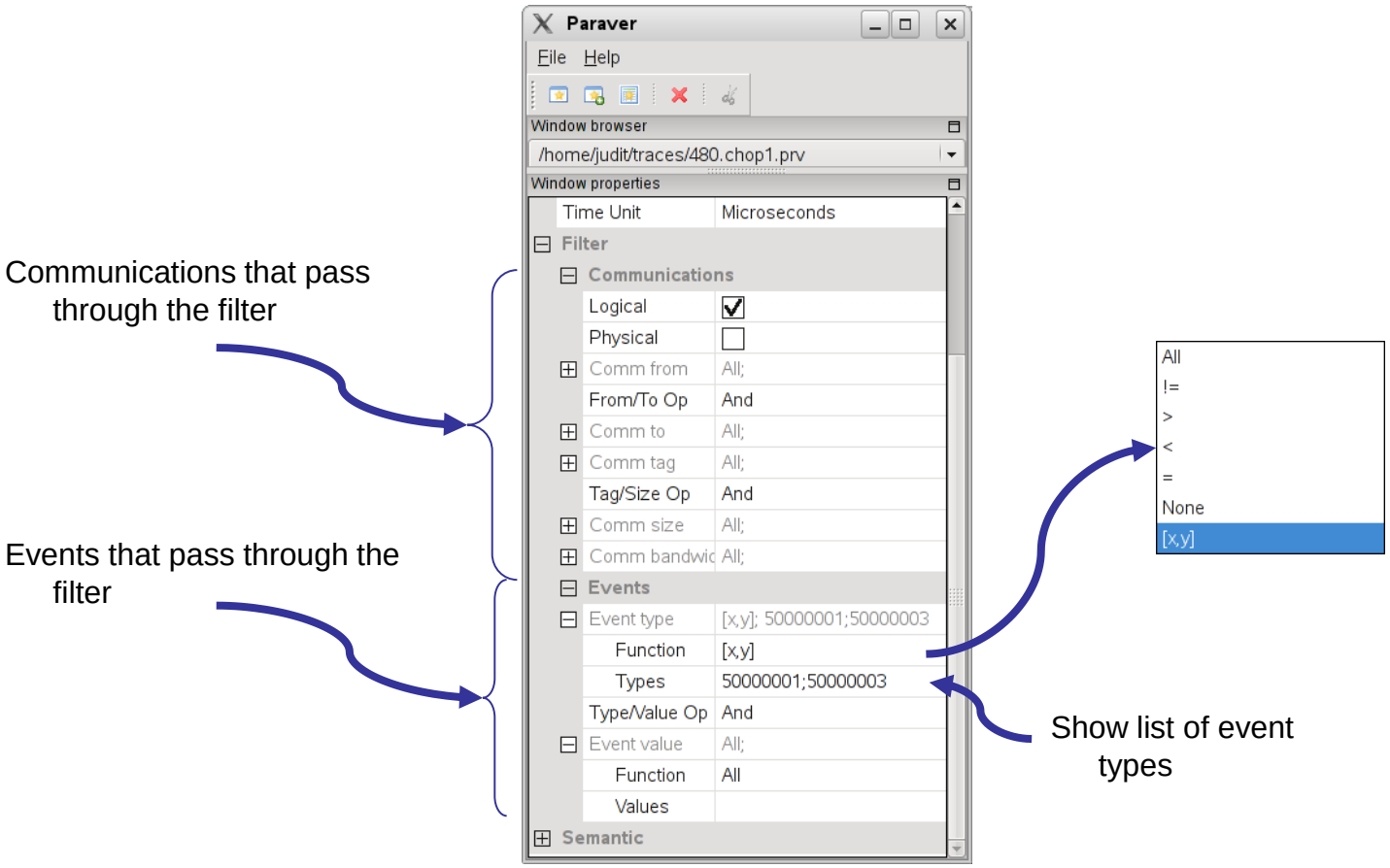
- **Semantic module** builds

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

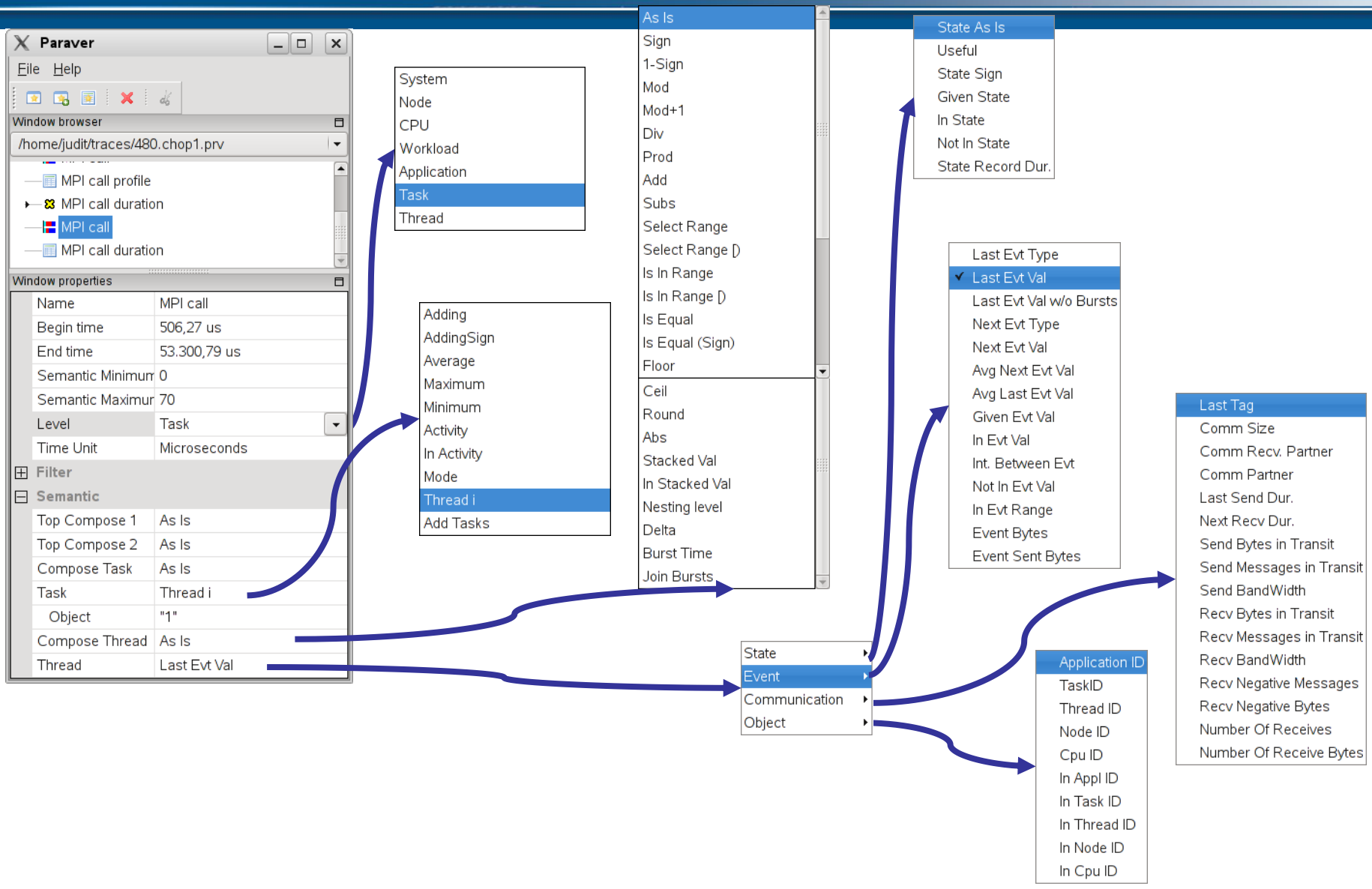
Function of time Series of values



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

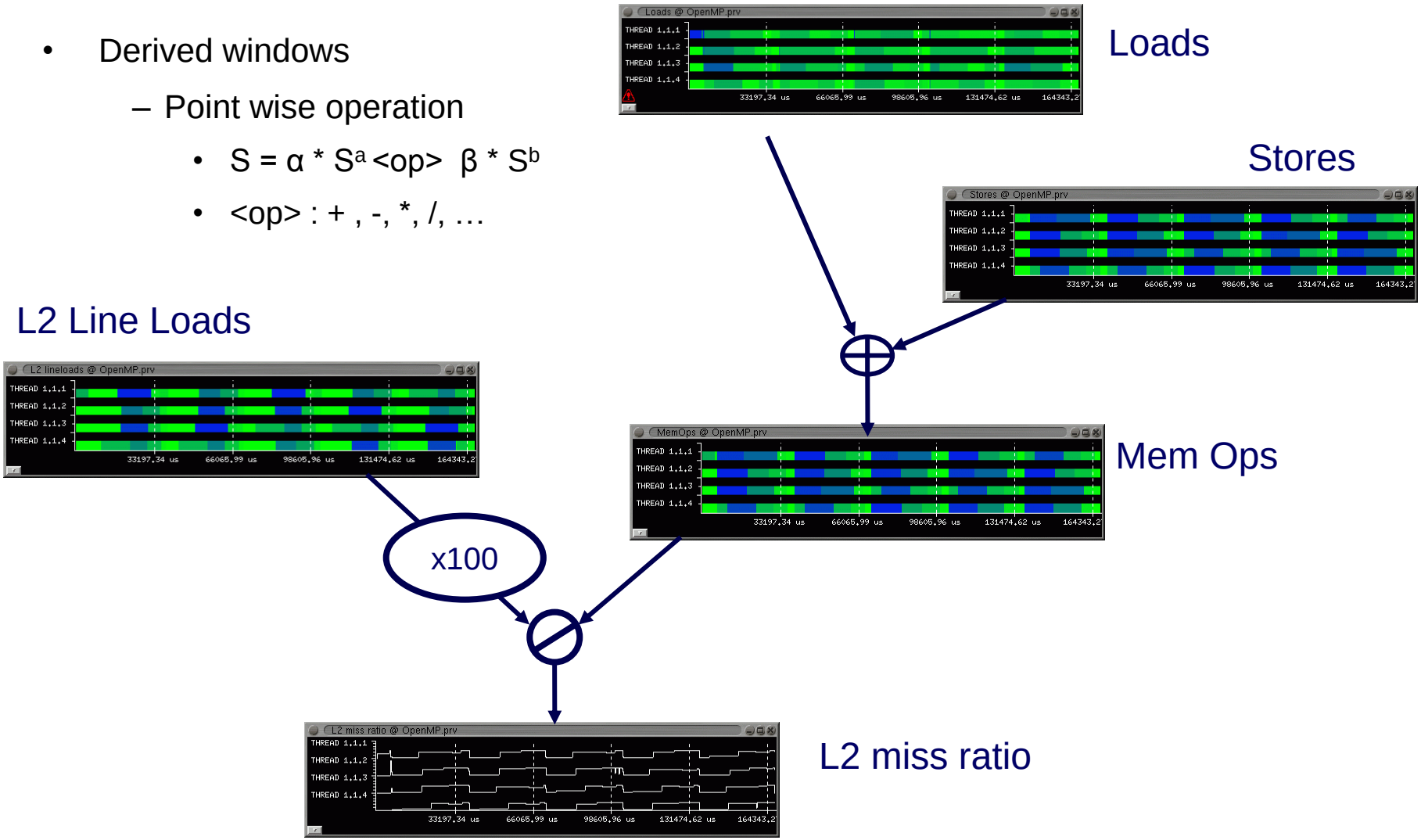
- From communication records to functions of time
  - Send Bytes $$s(t) = \sum_j Sz(C_j),\ j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$$
  - Send Bandwidth $$s(t) = \sum_j \frac{Sz(C_j)}{T_R(C_j) - T_S(C_j)},\ j \mid (T_S(C_j) < t) \wedge (T_R(Cj) > t) \wedge (Dir(Cj) == send)$$
  - 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 $$S(i) = \sum_j Sz(C_j),\ j \mid T_S(C_j) \in [T(Ev_i), T(Ev_{i+1})) \vee T_R(C_j) \in [T(Ev_i), T(Ev_{i+1}))$$

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

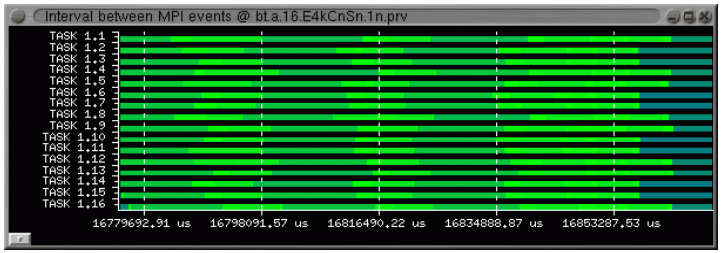
Semantic module

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

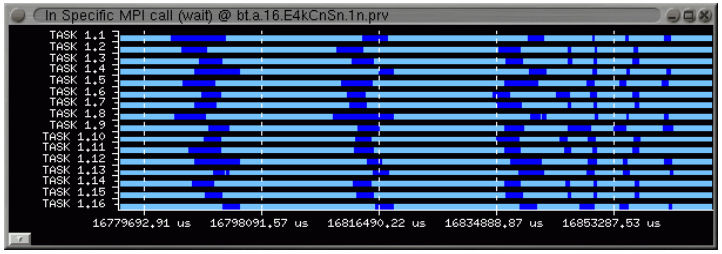


Semantic module

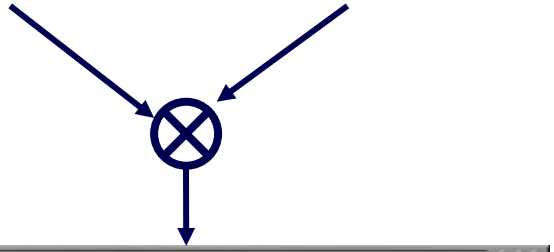
- Derived windows
  - Point wise operation
    - $S = \alpha * S^a \ <op> \ \beta * S^b$
    - <op> : + , -, *, /, …

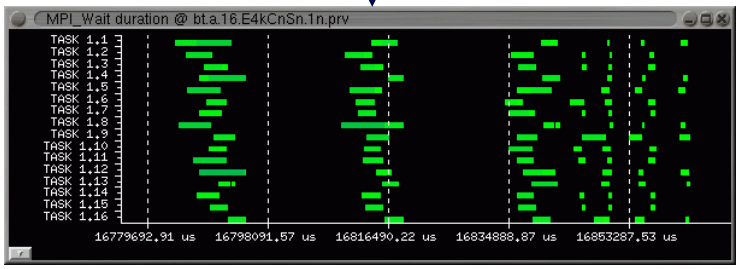


Interval between MPI events



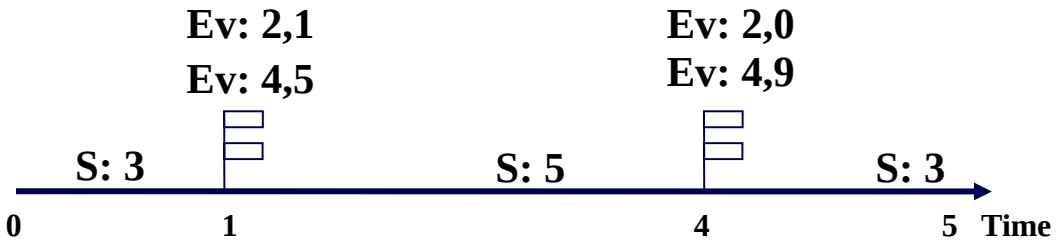
In MPI call



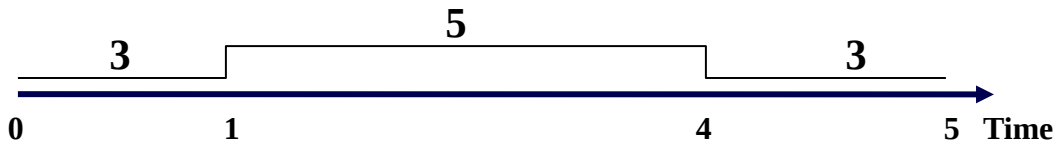


MPI call duration

Semantic module: Examples

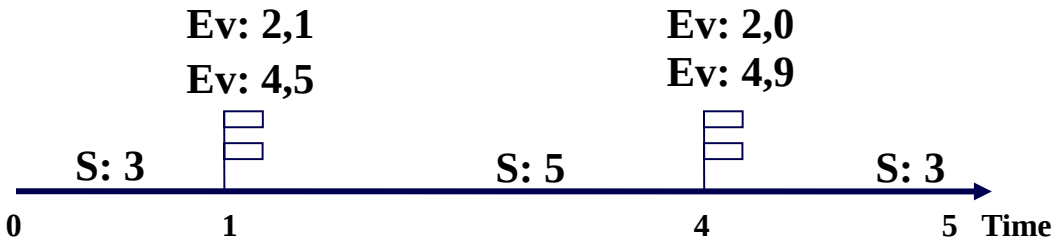


- Thread function: State as is



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

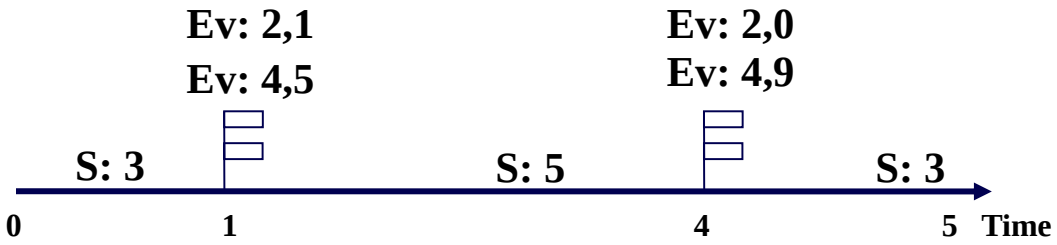
Semantic module: Examples



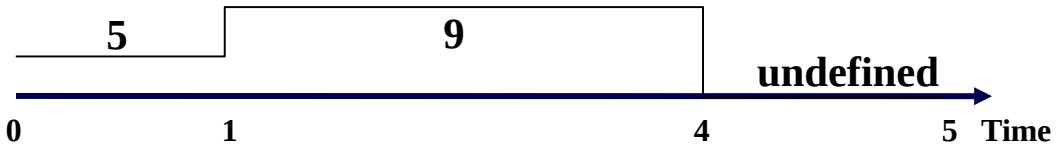
- Filter: type == 2
  - Thread function: Last event value

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

Semantic module: Examples

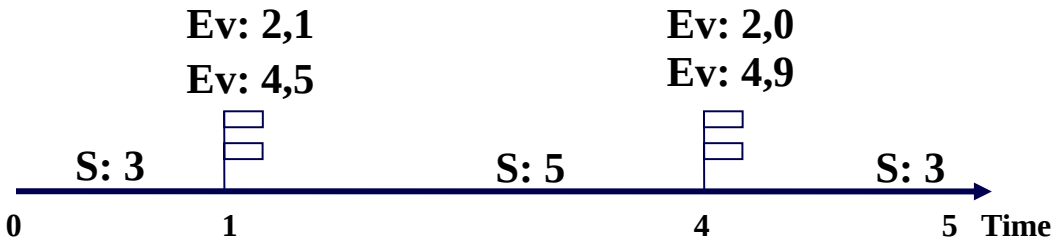


- Filter: type == 4
  - Thread function: Next event value

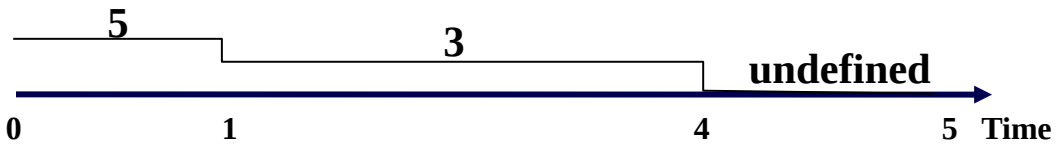


- Useful for
  - Hwc events (TLB, L1 misses,...) within interval

Semantic module: Examples



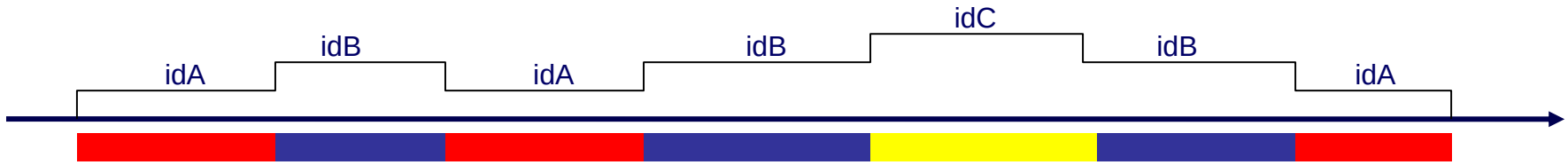
- Filter: type == 4
  - Thread function: Average next event value



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

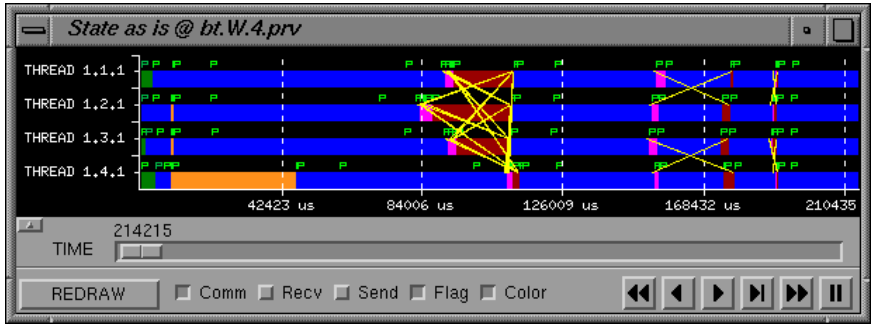
Semantic module: Examples

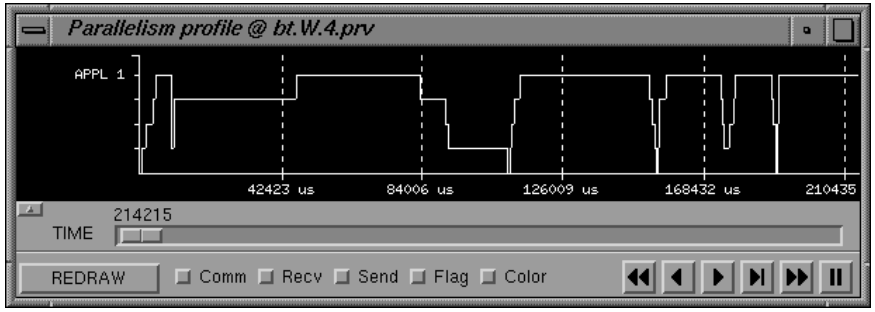
- Filter: type == USR_FCT
  Thread function: Last event value
  Compose: Stacked value



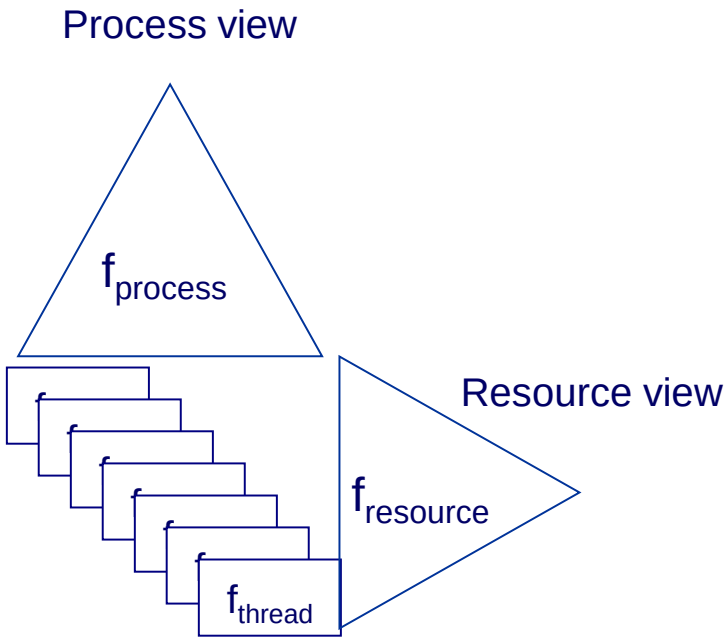
- Useful for
  - Routine

Semantic module perspective



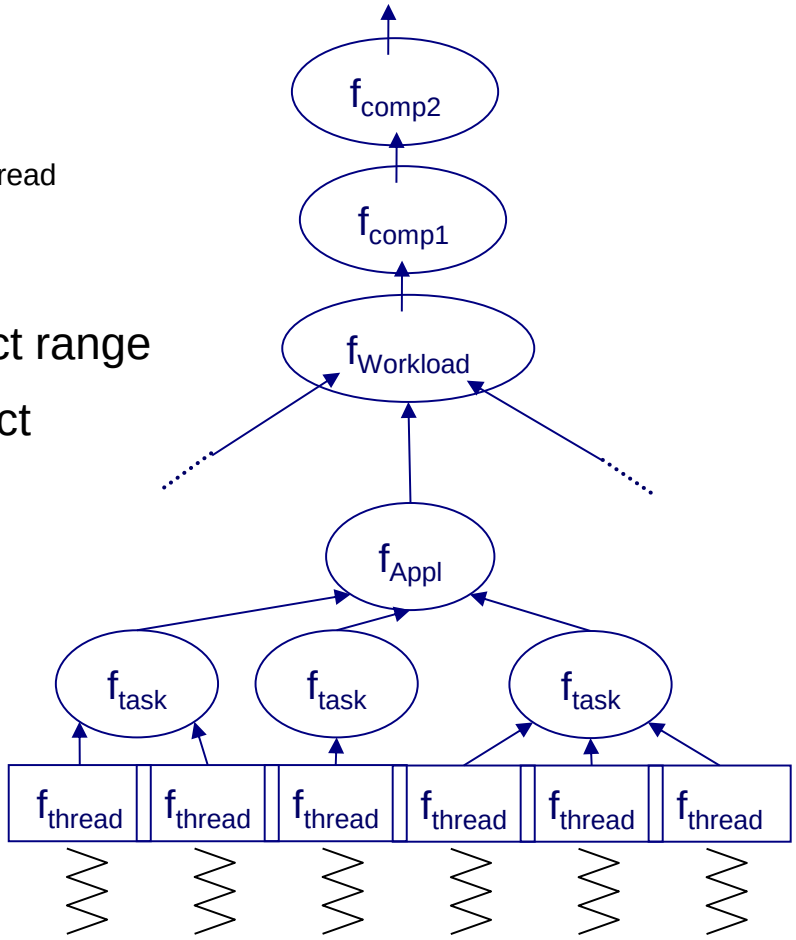


- Process model
  - Thread, task, application, workload
- Resource model
  - CPU, node, system



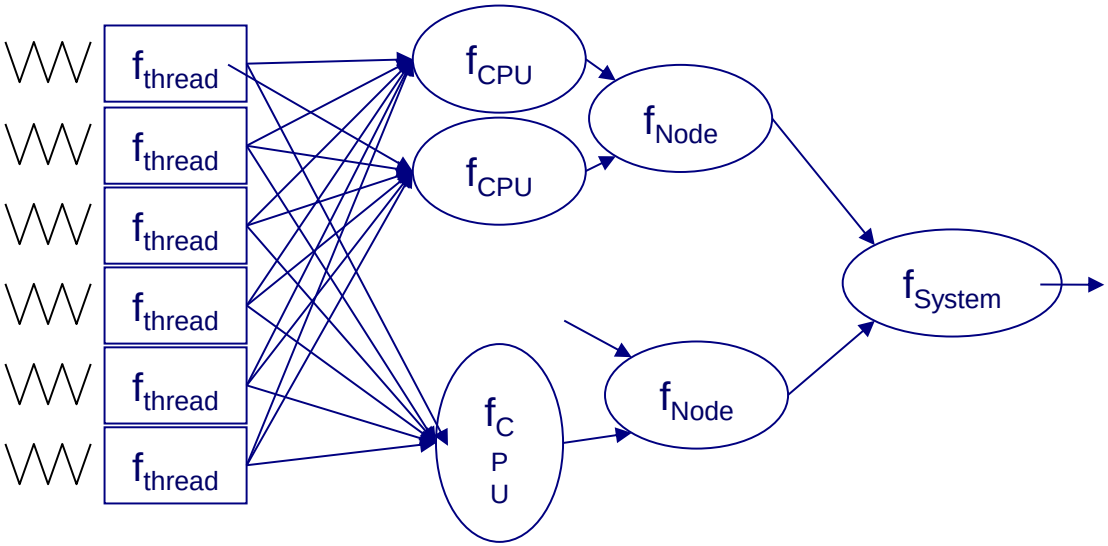
Process model perspective

- 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

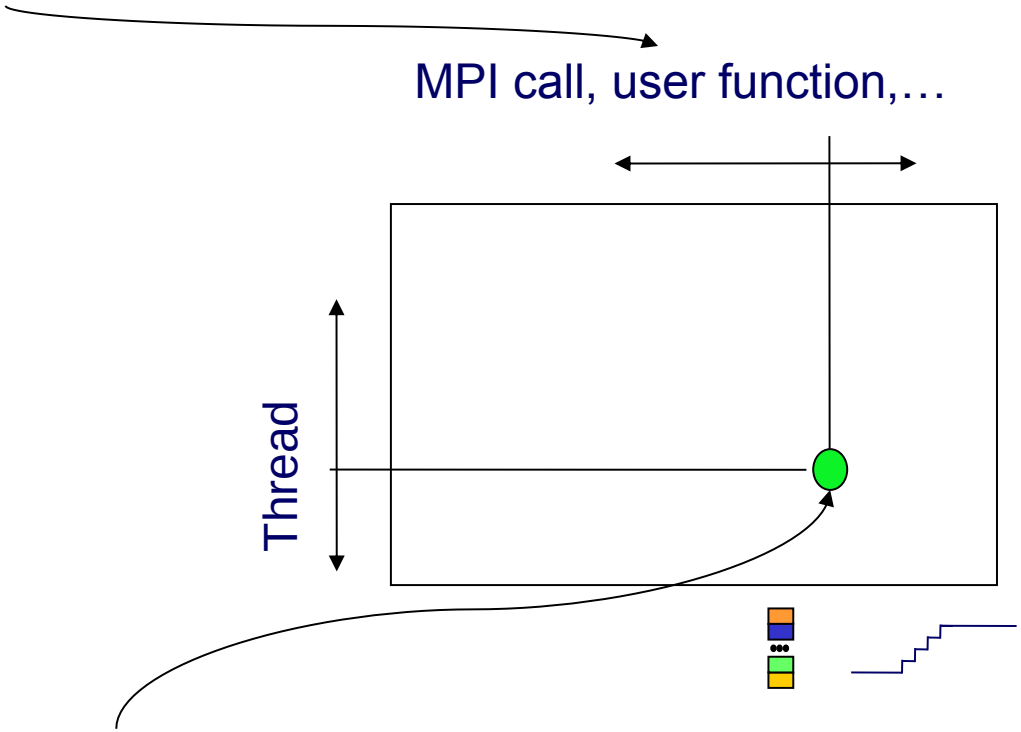
How to read profiles

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

MPI 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

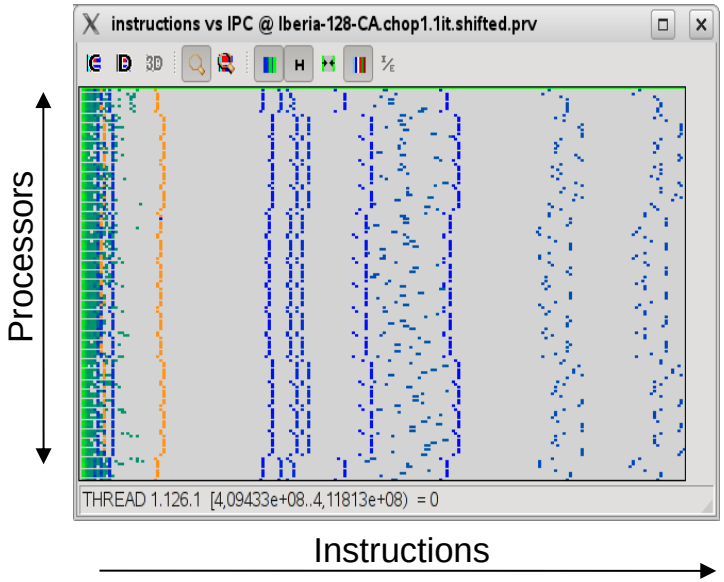


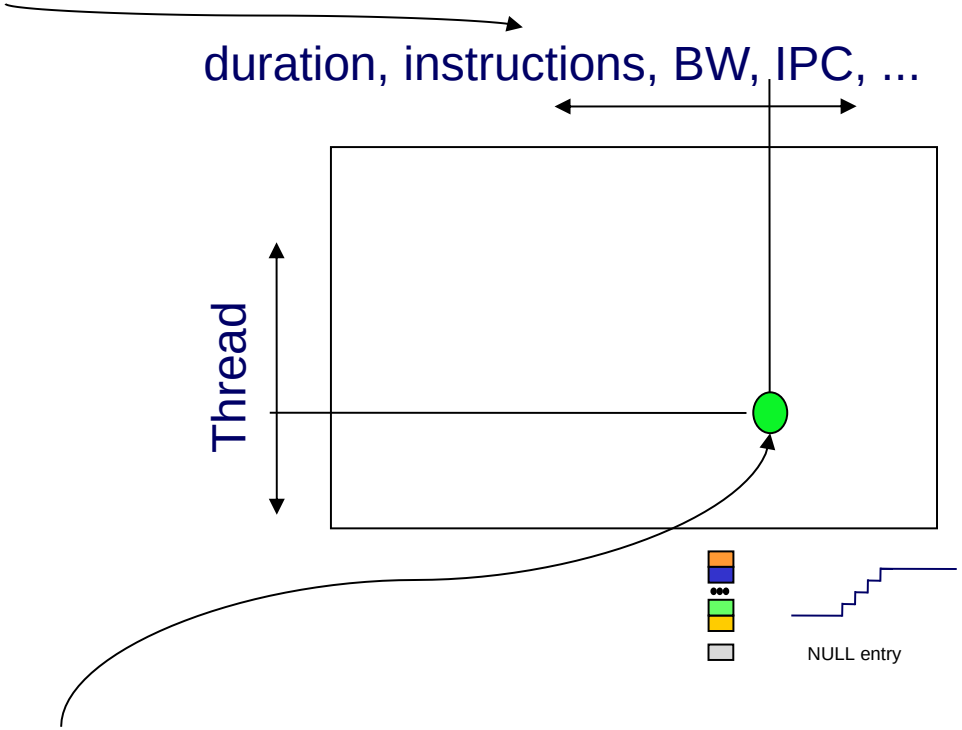
Value/color is a statistic computed for the specific thread
when control window had the value corresponding to the column

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**

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

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

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

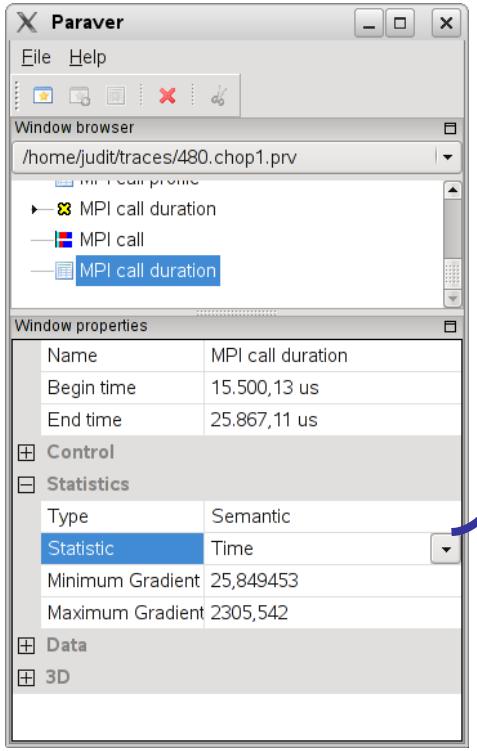
Identify regions with cw's within the (j,k) bin

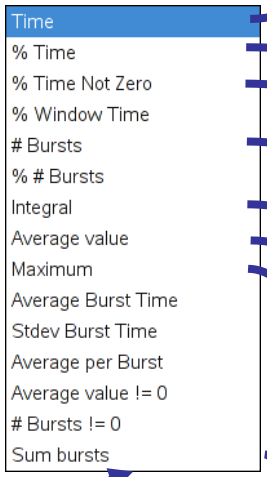
- The 3D analysis module computes a cube (or plane in the case of 2D) of statistics

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

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

2D analysis module





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

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

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

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

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

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

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

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

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

Distributed Configurations

Distribution of cfg directories

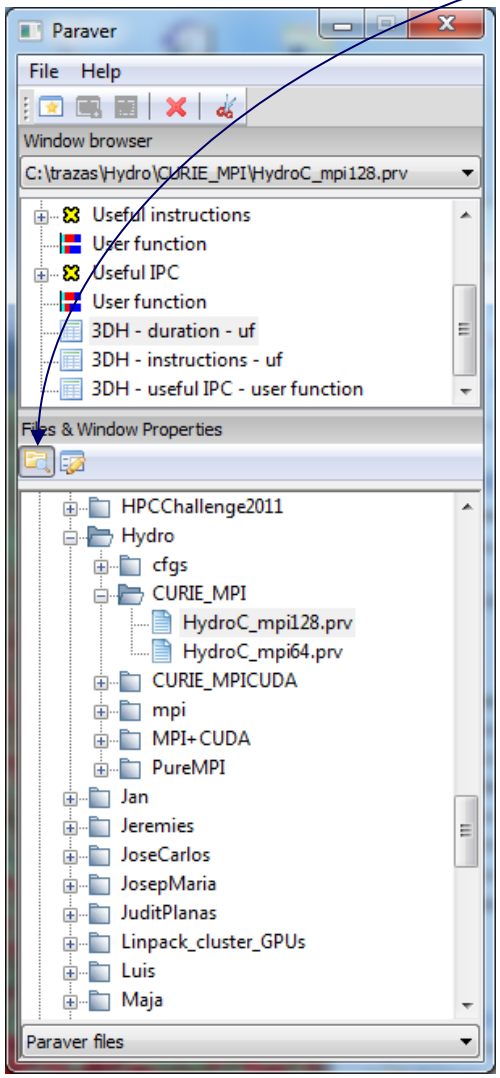
- CFG

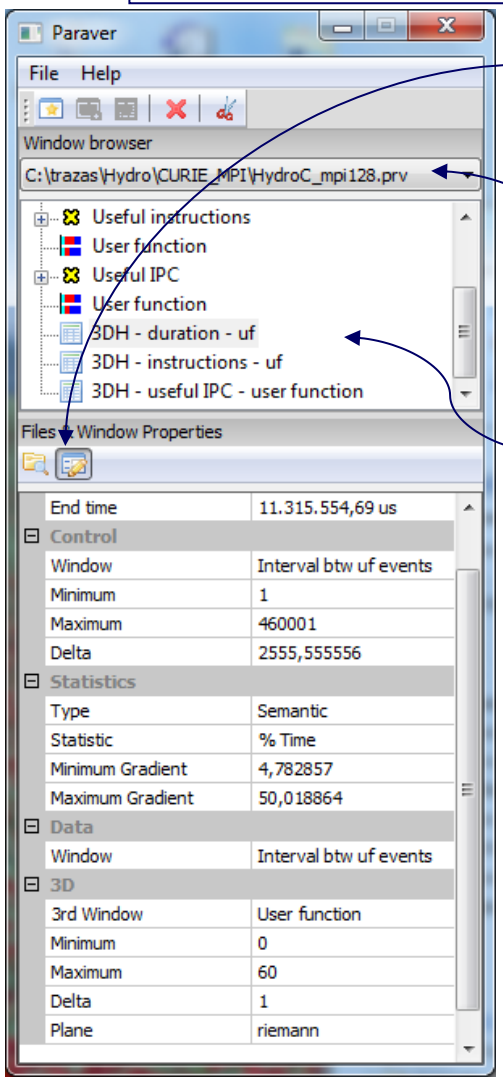
$PARAVER_HOME/cfgs

  - 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

How to …

Main Paraver window





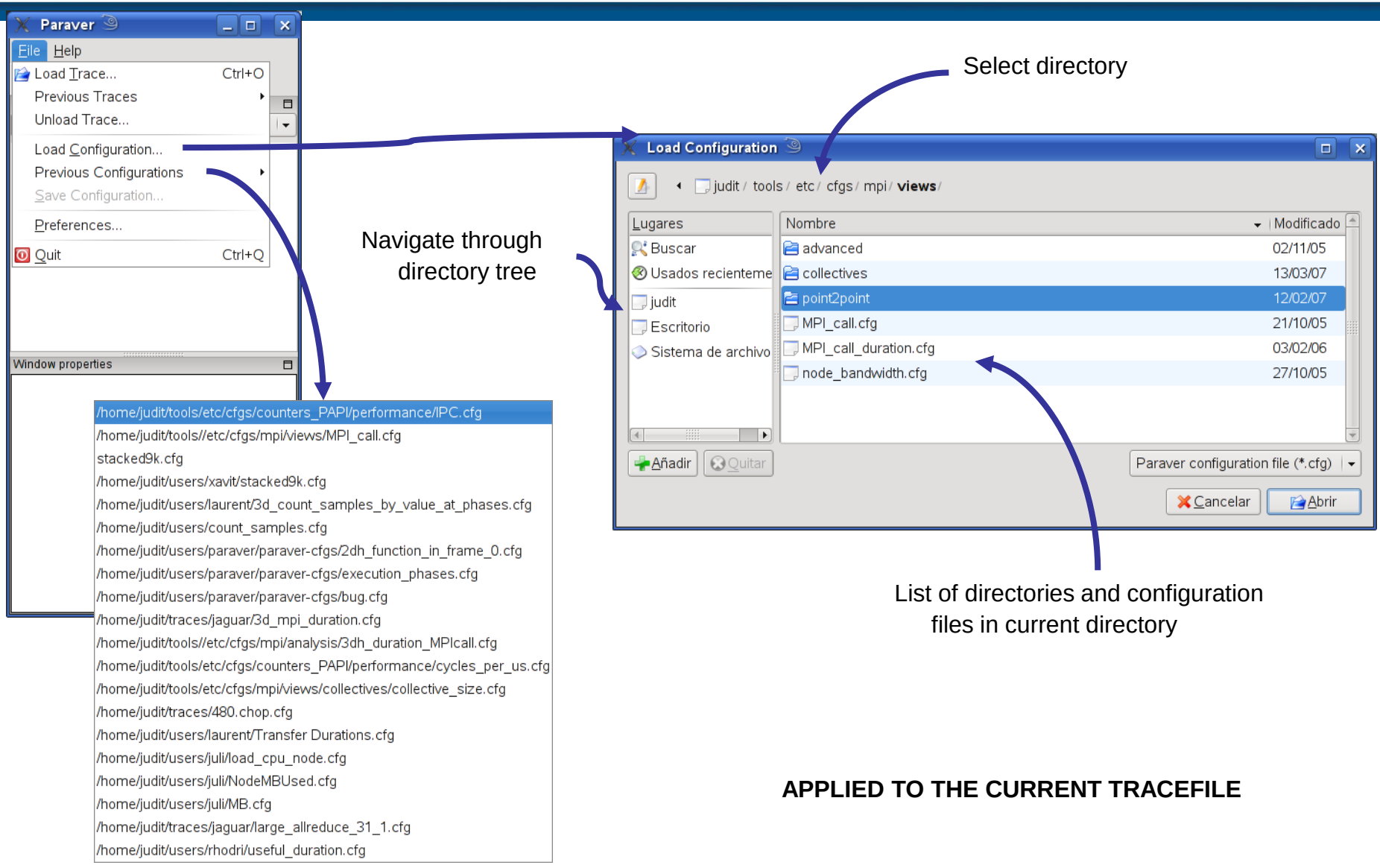
Select to browse in lower panel for traces or cfgs

Select to browse characteristics of active view or table

Active trace

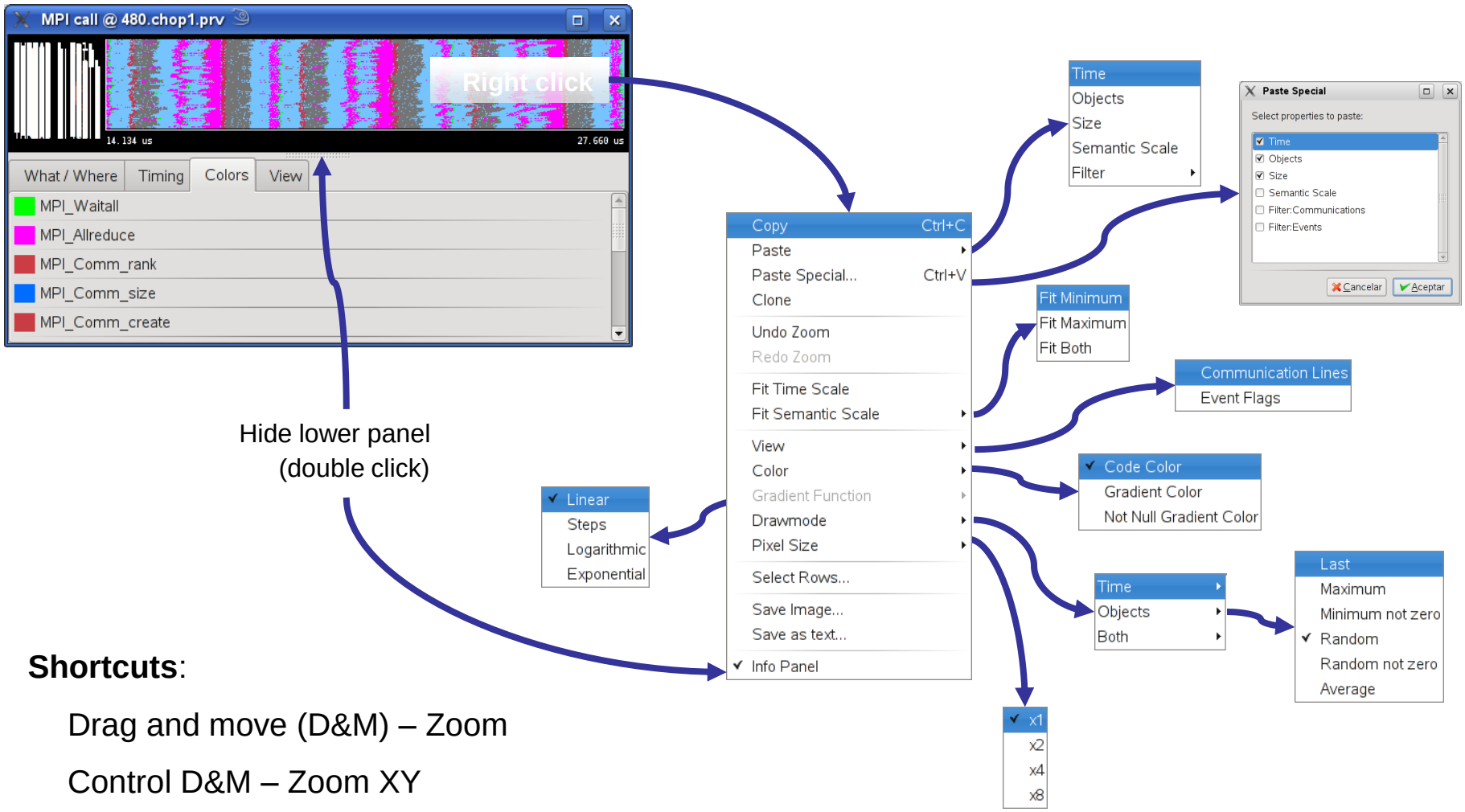
Available views and tables
Active view or table highlighted

Load configuration files



APPLIED TO THE CURRENT TRACEFILE

Navigation



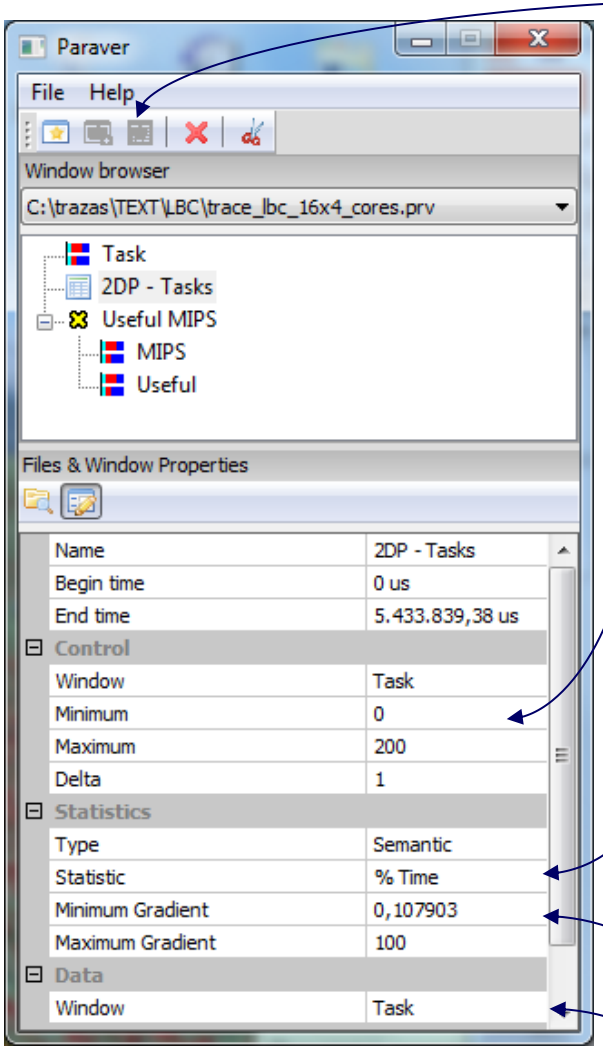
Shortcuts:

Drag and move (D&M) – Zoom

Control D&M – Zoom XY

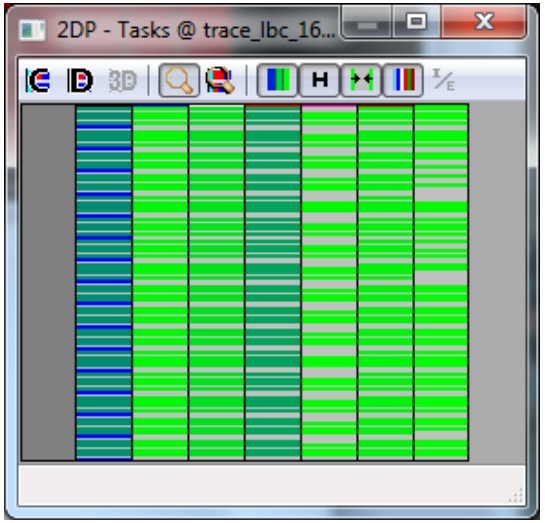
Shift D&M – Timing

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

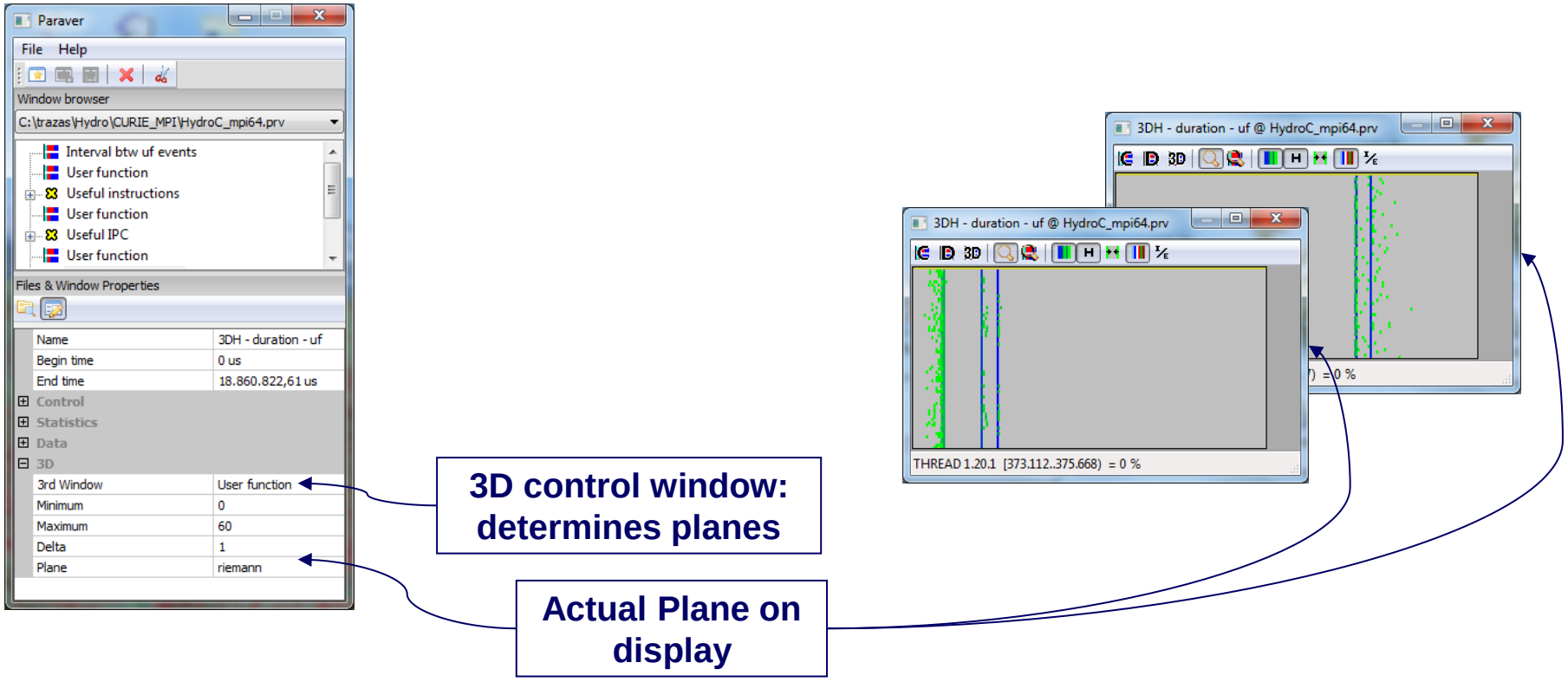
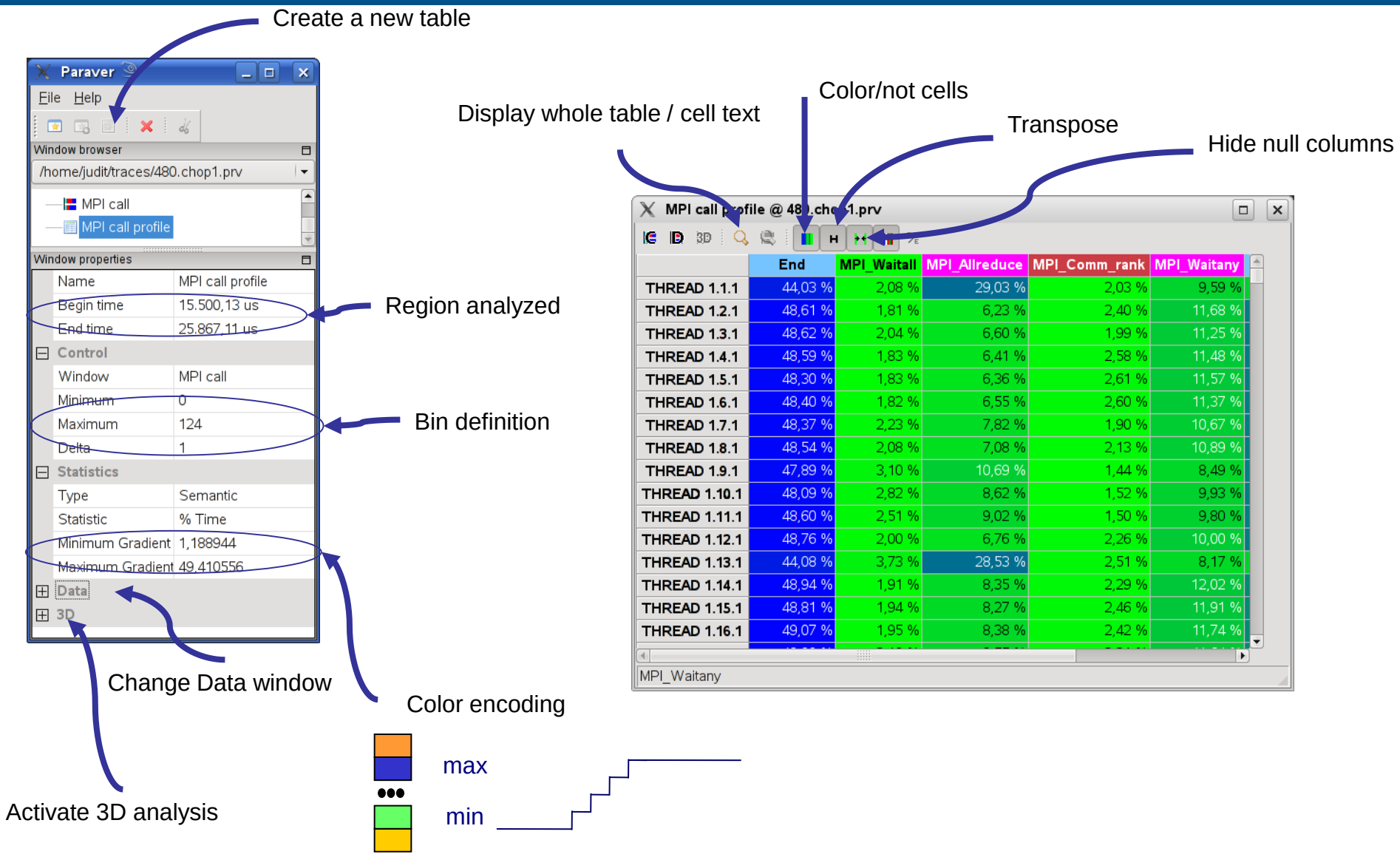
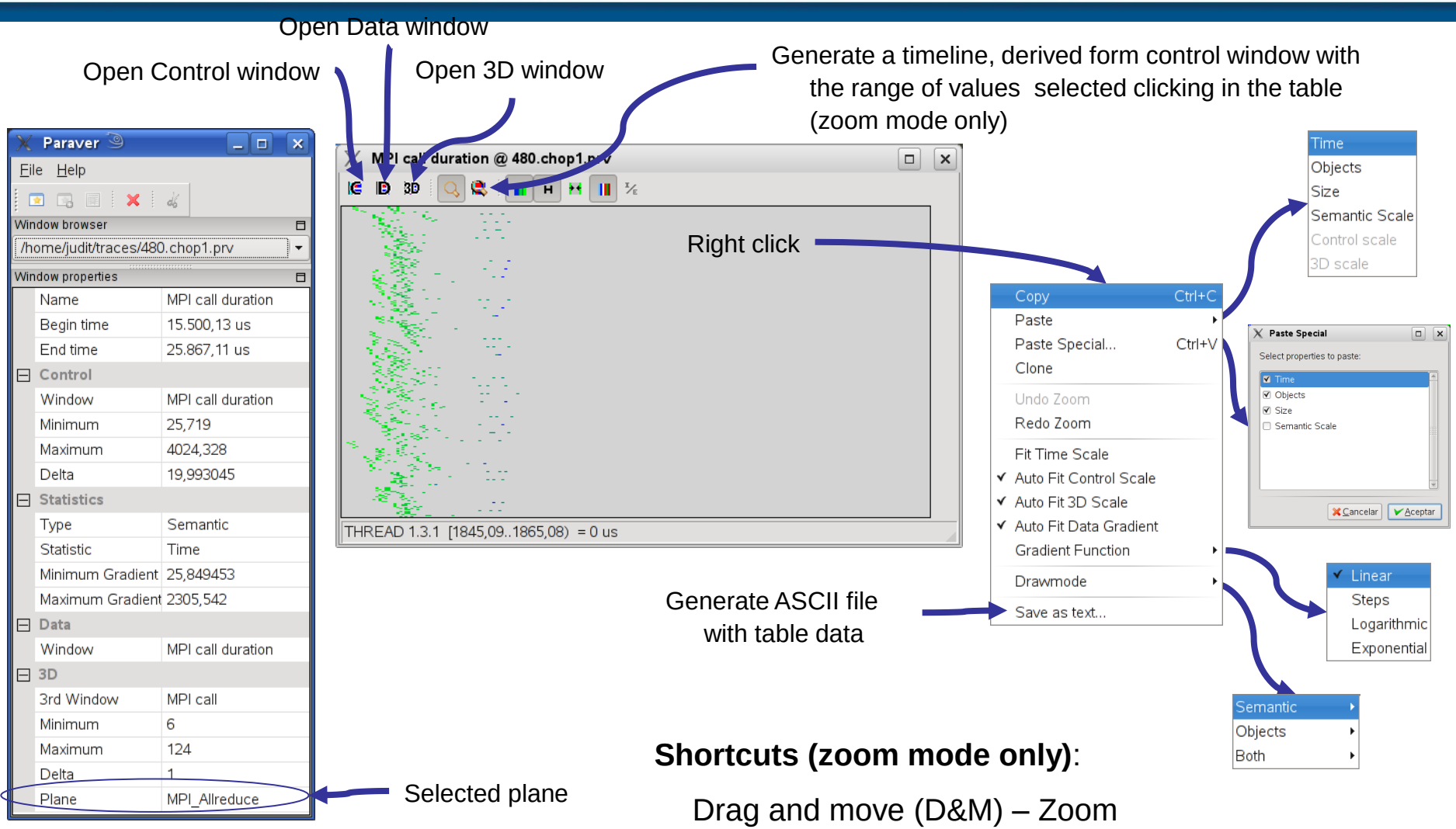


Table information and control



	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



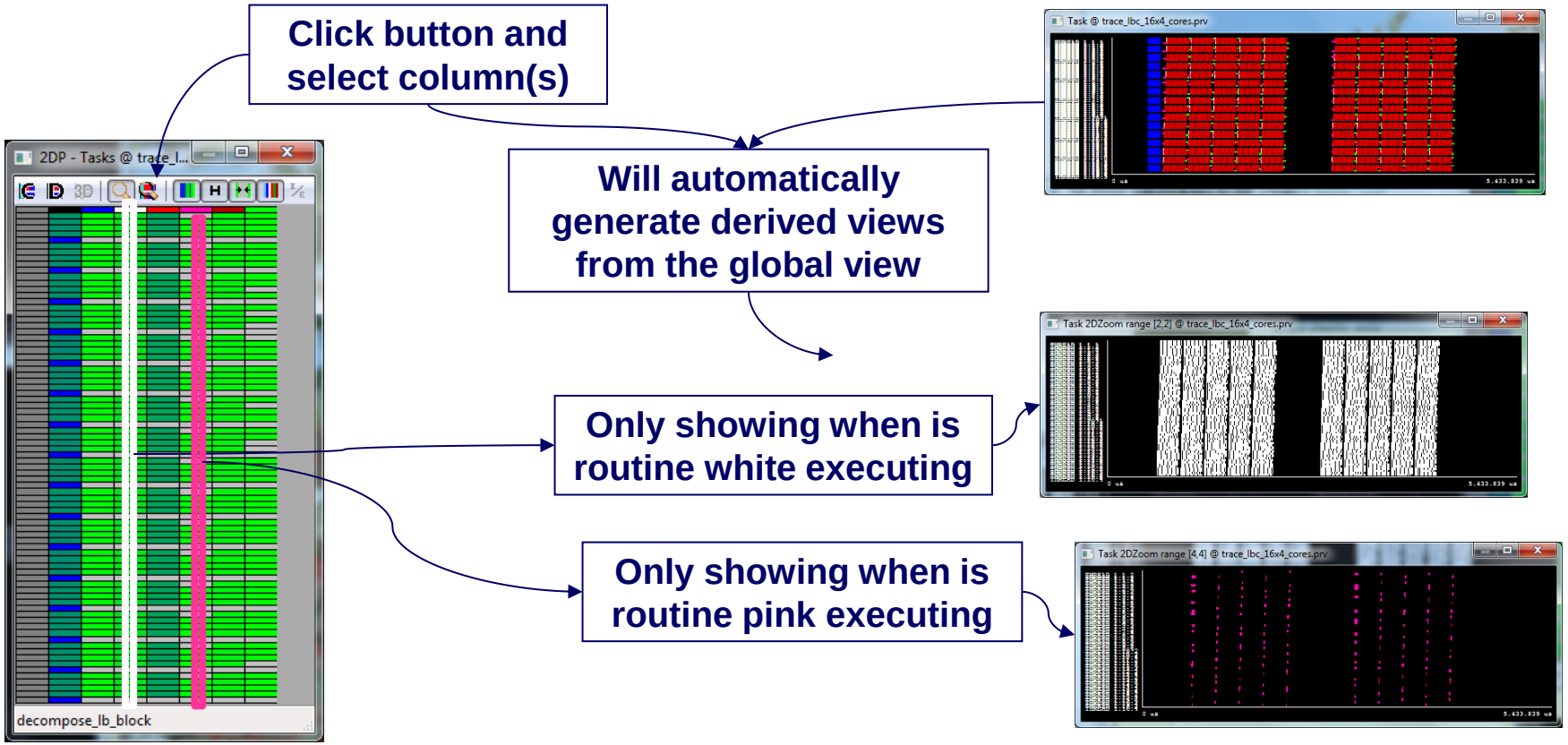
Shortcuts (zoom mode only):

Drag and move (D&M) – Zoom

Control D&M – Zoom XY

From tables to timelines

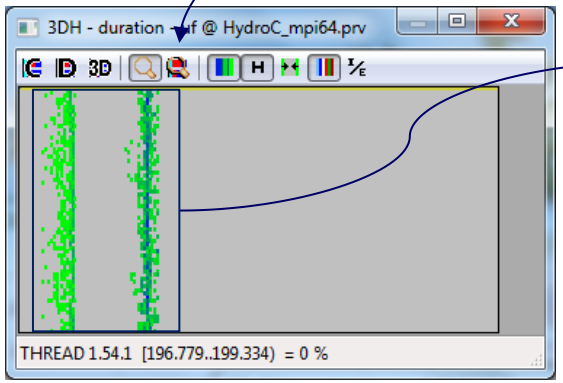
- Where in the timeline do the values in certain table columns appear?
  - ie. want to see the time distribution of a given routine?



From tables to timelines

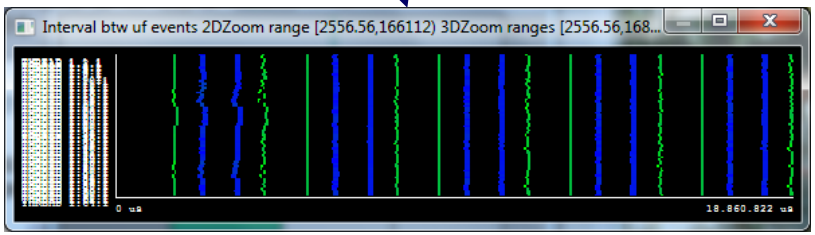
- Where in the timeline do the values in certain table columns appear?
  - ie. want to see where the timeline happen computation bursts of a given length?

Click button and select column(s)



3D histogram of duration of routine foo

Will automatically generate



Only showing duration of routine foo

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

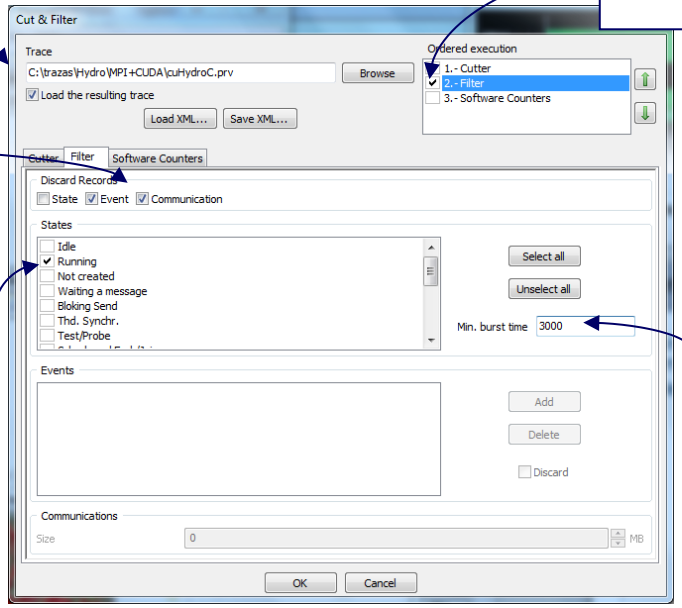
A trace with basename.filter1.prv will be generated

Select filtering option

Discard events and communications

Keep only Running bursts ….

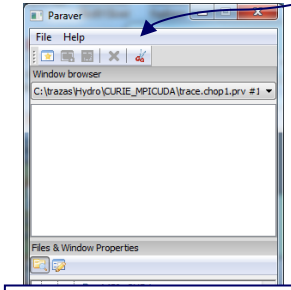
--- longer than 3000 ns



Cutting very large traces

- Load a filtered trace and use the scissors tool

Scissors tool



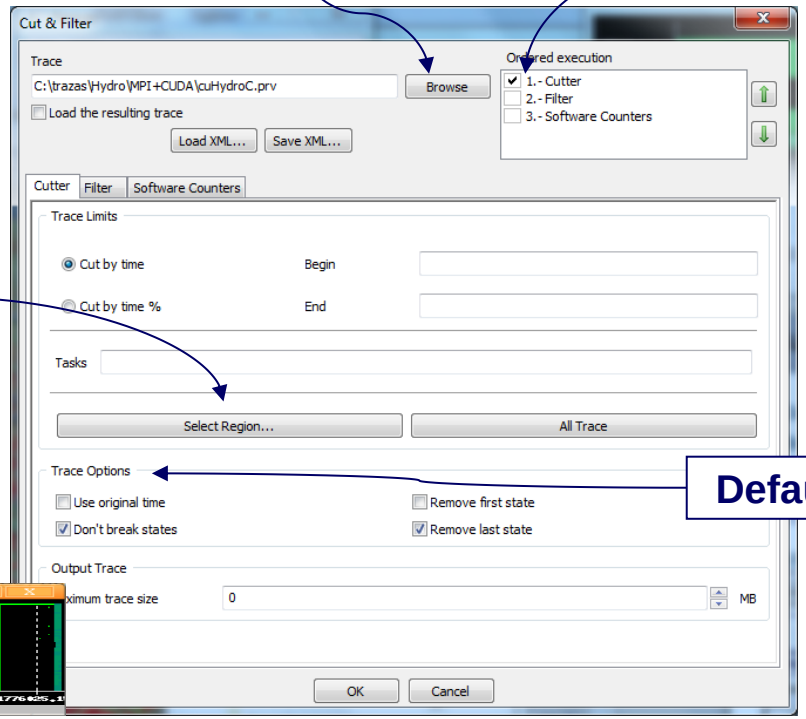
Browse to select file from which the cut will be obtained

Select cutter

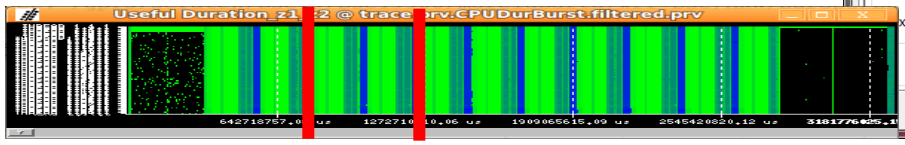
Click to select region

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

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)

extrae.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)

extrae.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)

extrae.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)

extrae.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

- 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

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

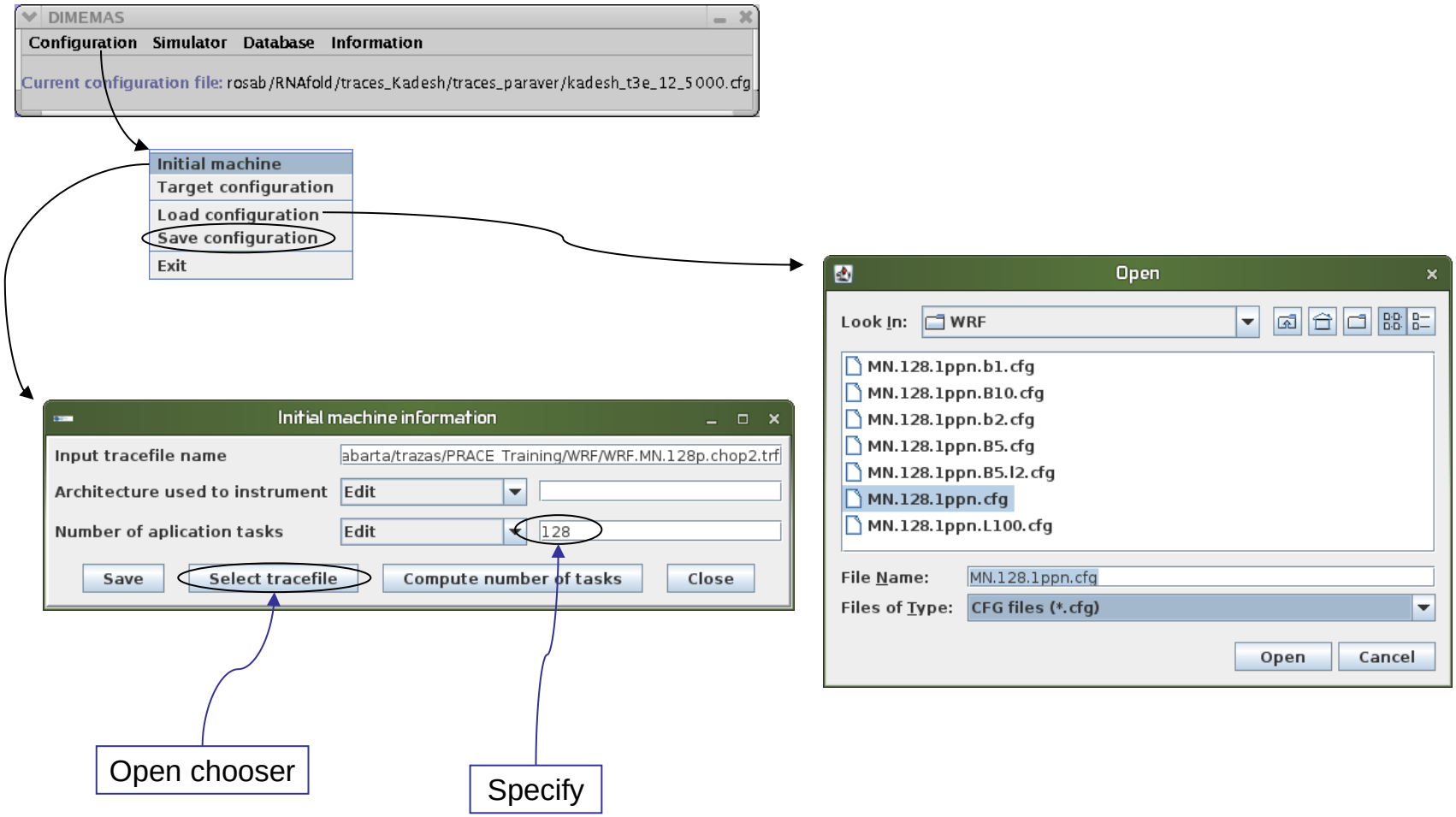
```
Usage:
  prv2dim -i <iprobe_miss_threshold> -b <hw_counter_type>,<factor>
  <paraver_trace> <dimemas_trace>

  -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
…
```

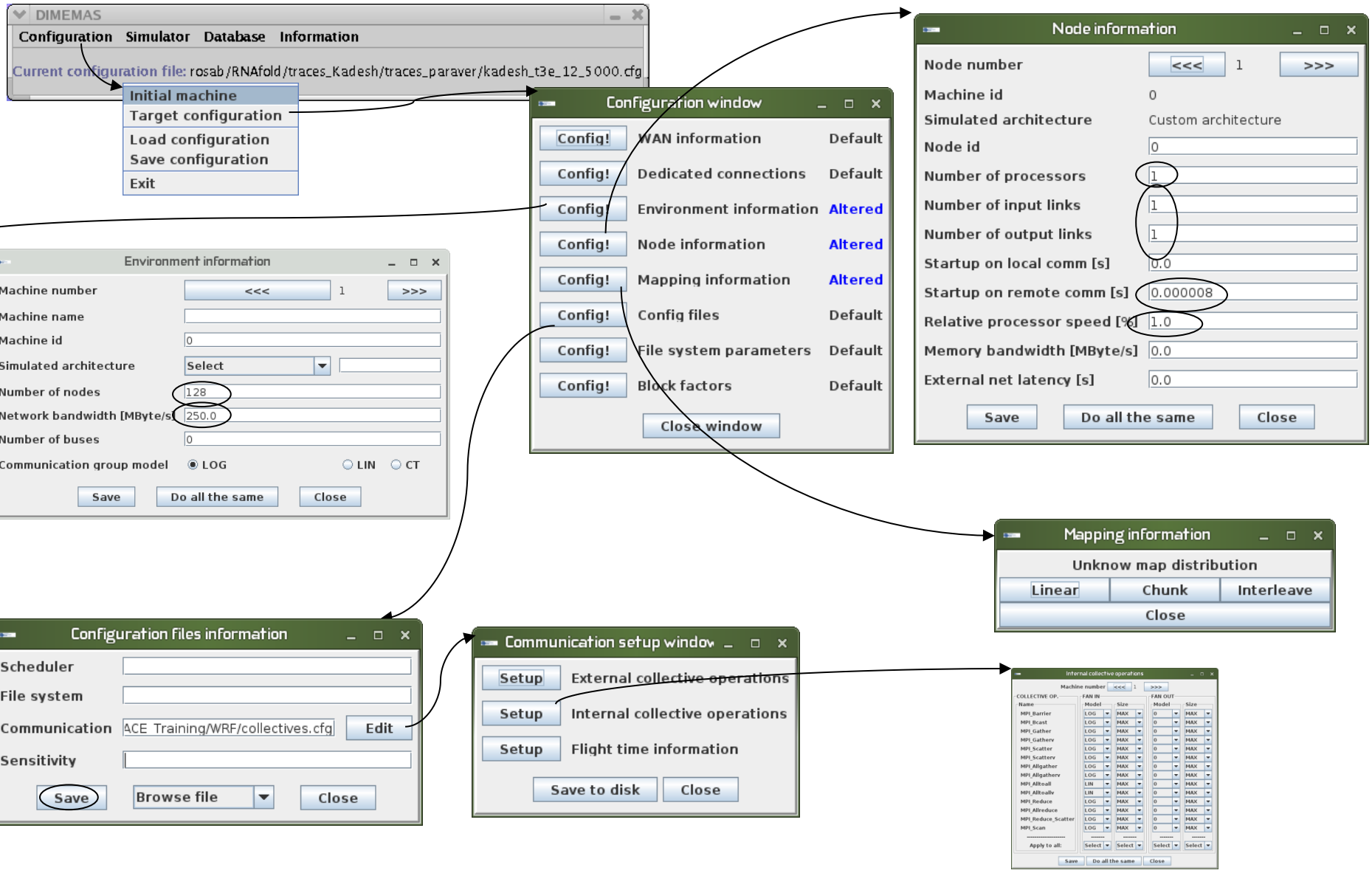
Force synchronized start of all threads

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

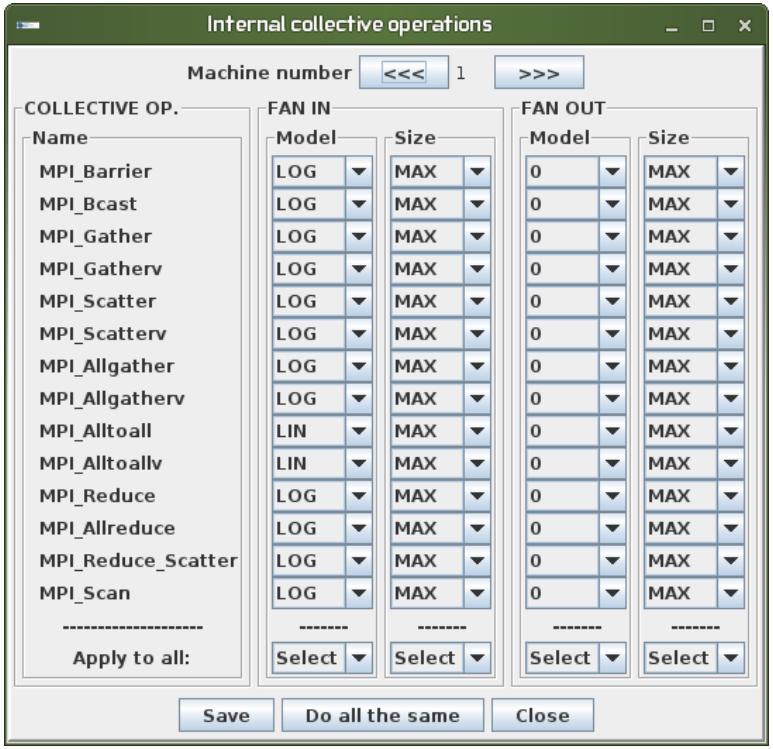


Dimemas GUI – Specify target machine



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



Scalability

Scalability of Presentation

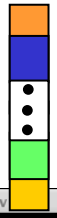
- Linpack @ Marenostrum: 10k cores x 1700 s

Dgemm duration

11.8 s

10 s

Dgemm IPC



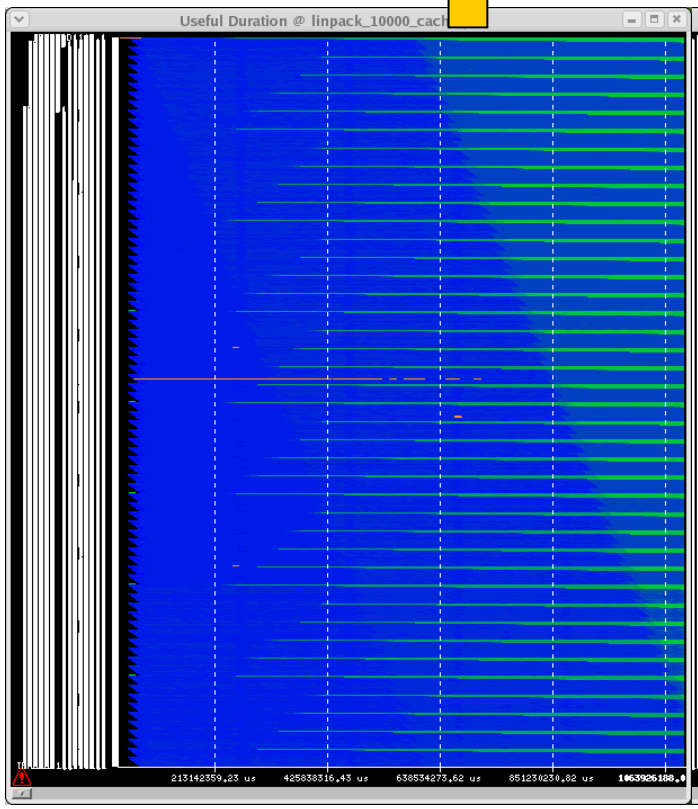
2.95

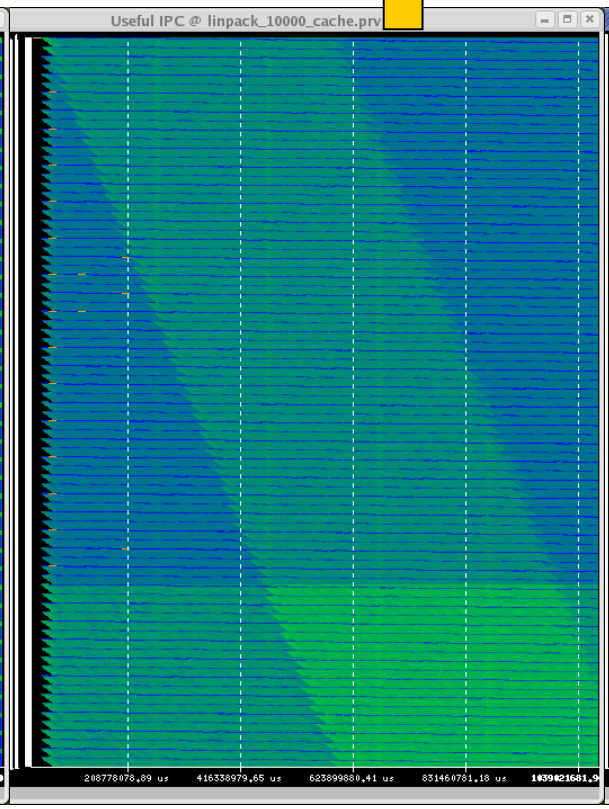
2.85

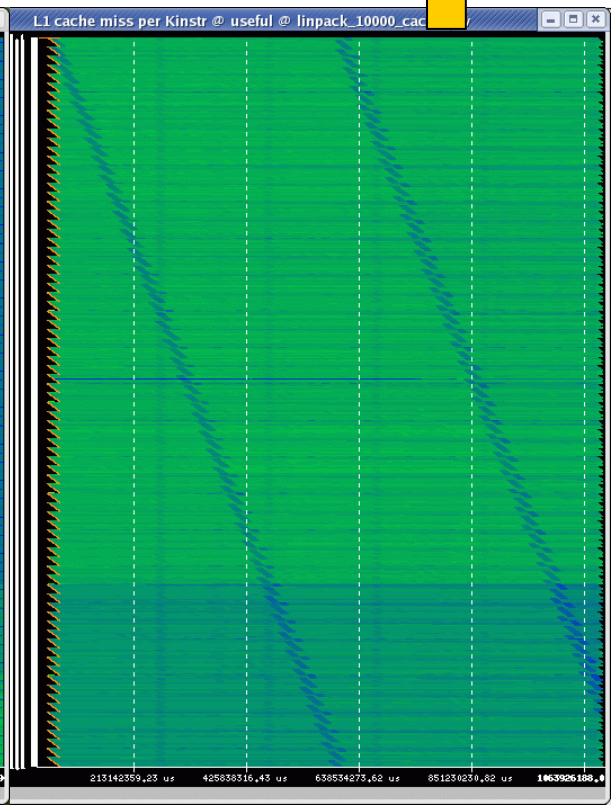
Dgemm L1 miss ratio

0.8

0.7







Scalability of analysis

Jugene

Jaguar

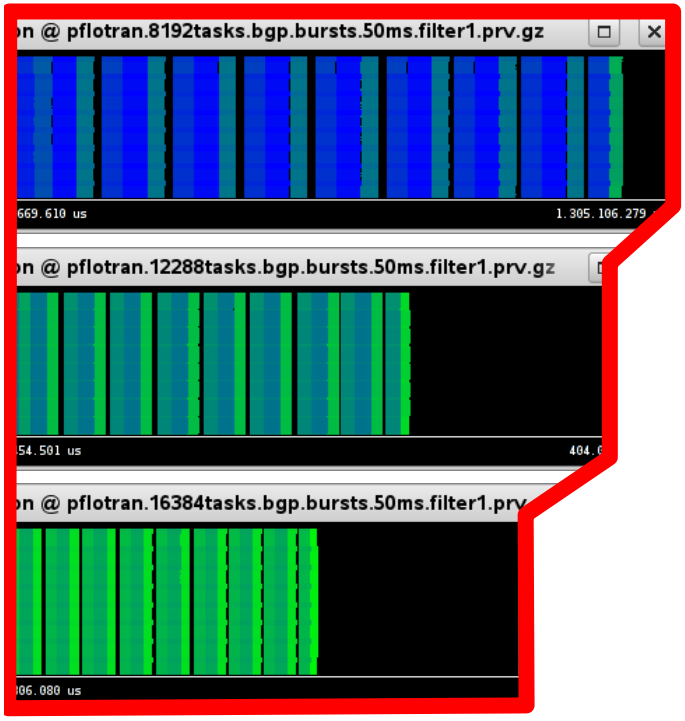
~ 105 seconds

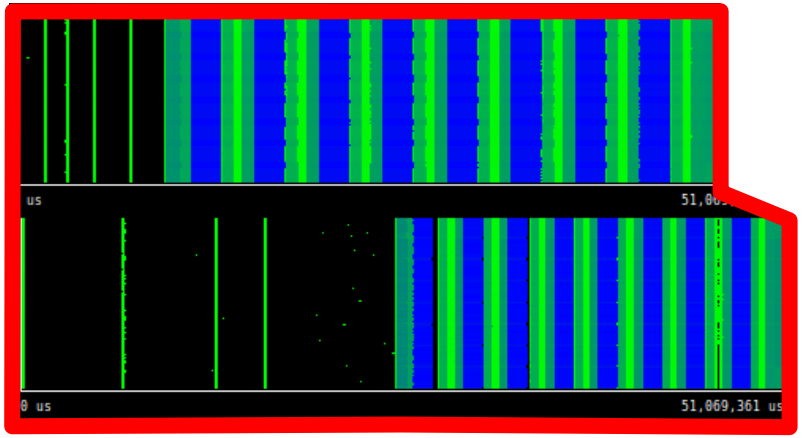
~ 47 seconds

8K cores

12K cores

16K cores





Tran

Flow

Flow

Tran

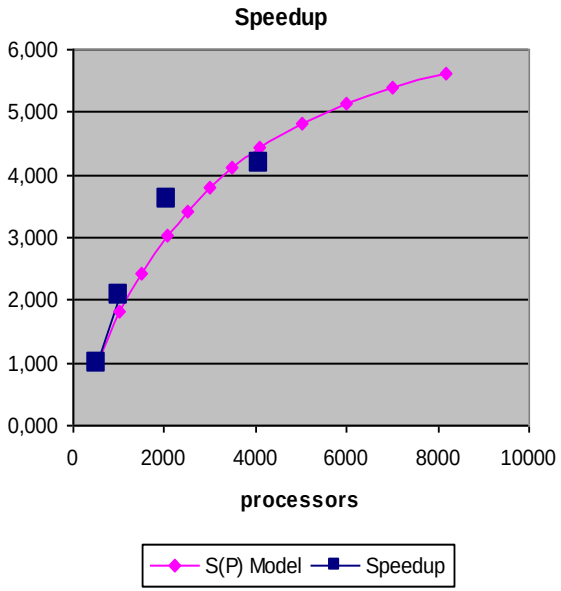
PFLOTRAN

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

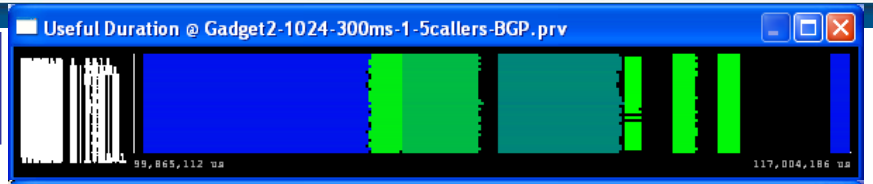
J. Labarta, et al.: “Scalability of tracing and visualization tools”, PARCO 2005

Software counters

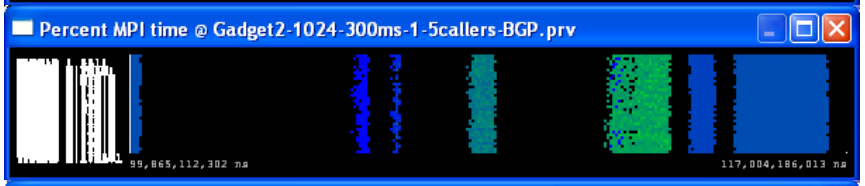


GADGET, PRACE Case A, 1024 procs

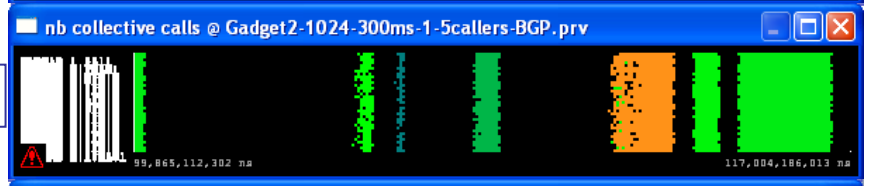
Useful duration



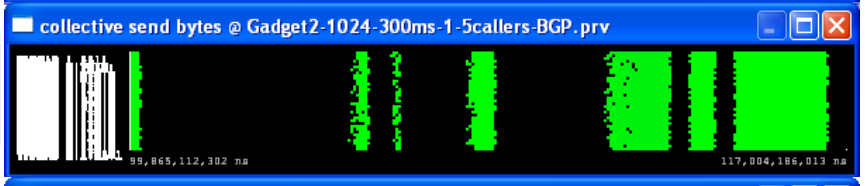
% MPI time



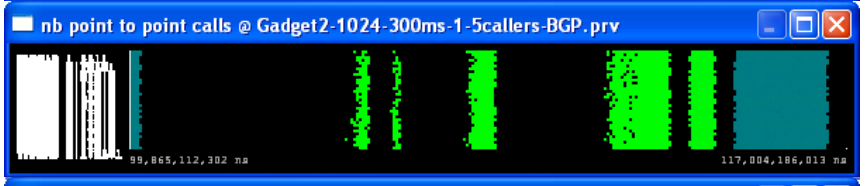
collectives



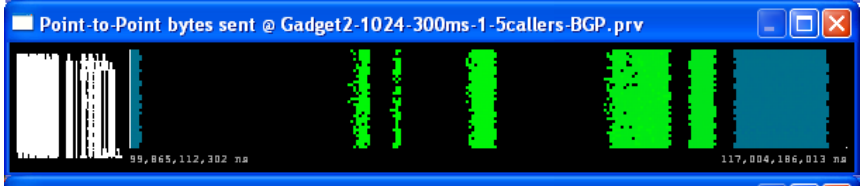
Collective bytes



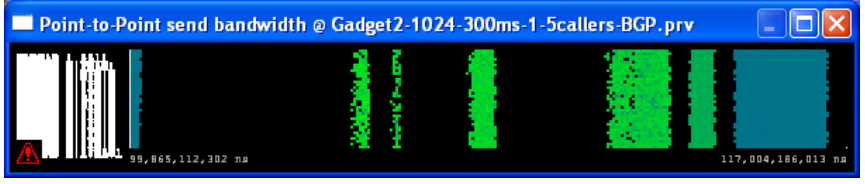
p2p



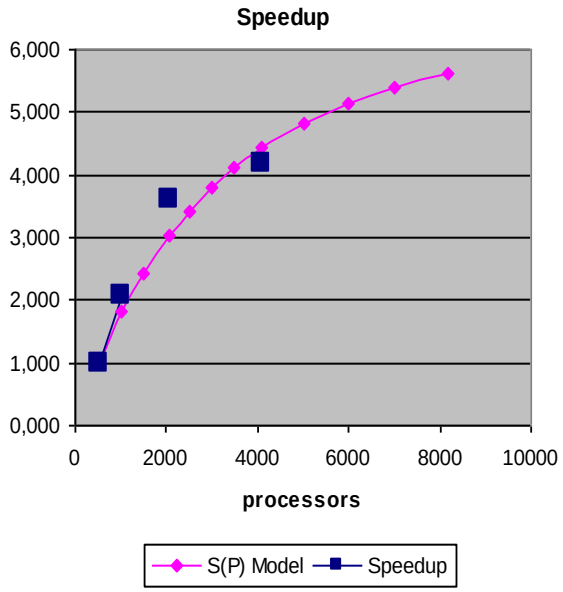
p2p bytes



p2p BW

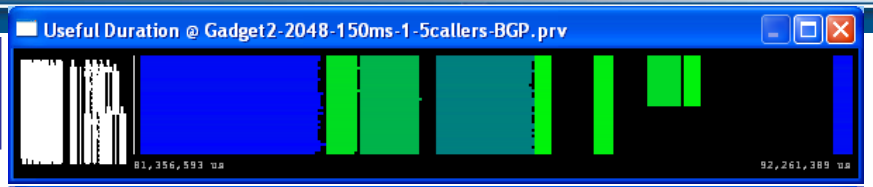


Software counters



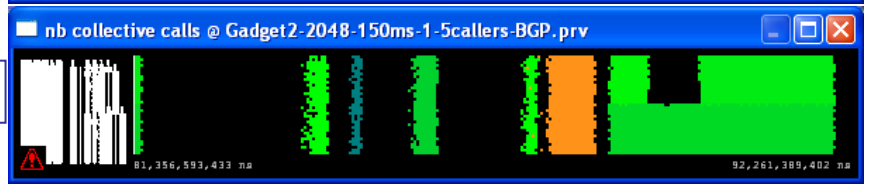
GADGET, PRACE Case A, 2048 procs

Useful duration

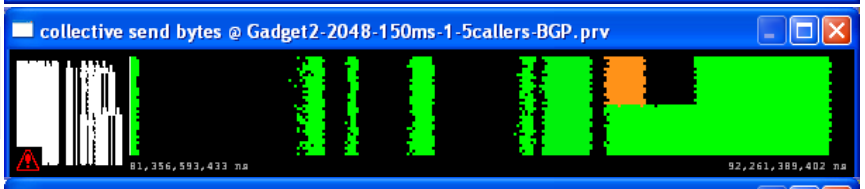


% MPI time

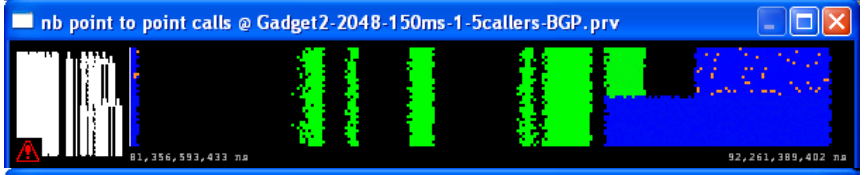
collectives



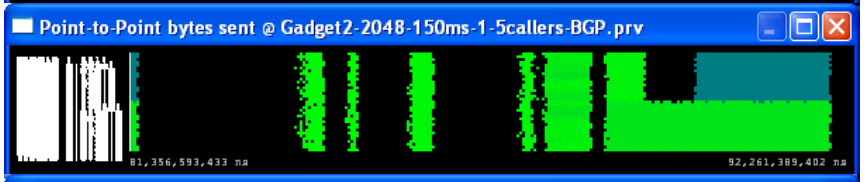
Collective bytes



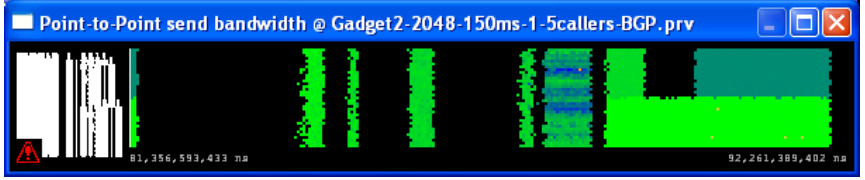
p2p



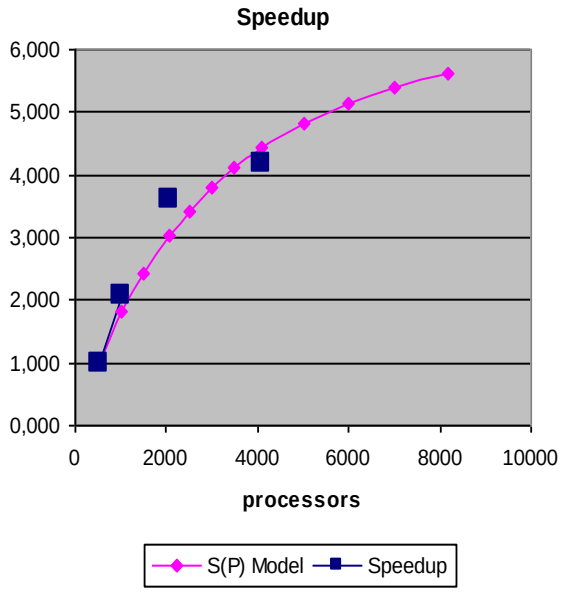
p2p bytes



p2p BW

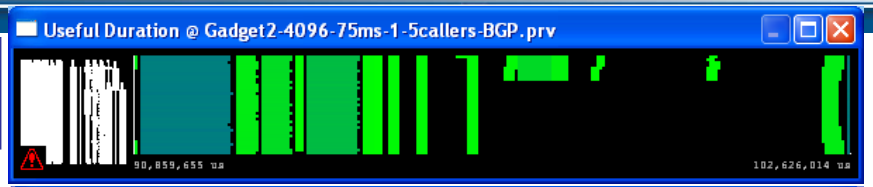


Software counters

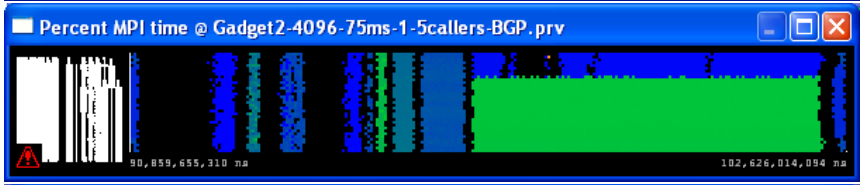


GADGET, PRACE Case A, 4096 procs

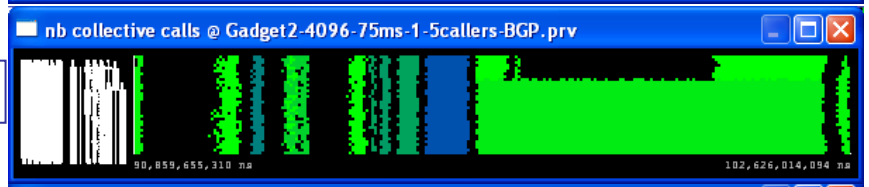
Useful duration



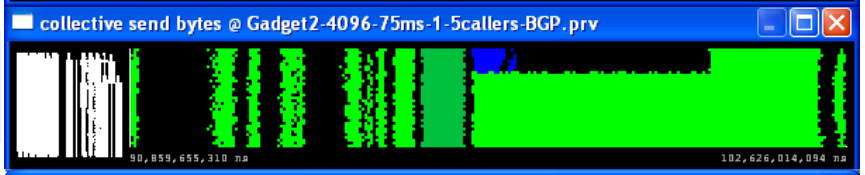
% MPI time



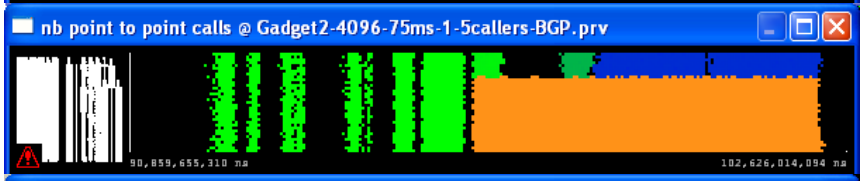
collectives



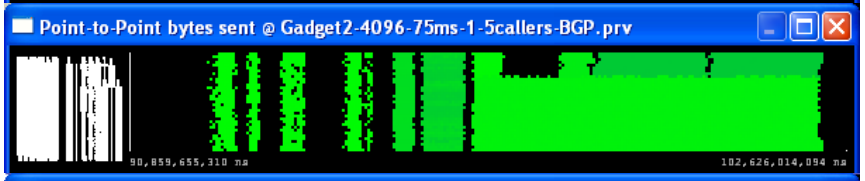
Collective bytes



p2p



p2p bytes



p2p BW