www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

BSC TOOLS: INSTRUMENTATION & ANALYSIS

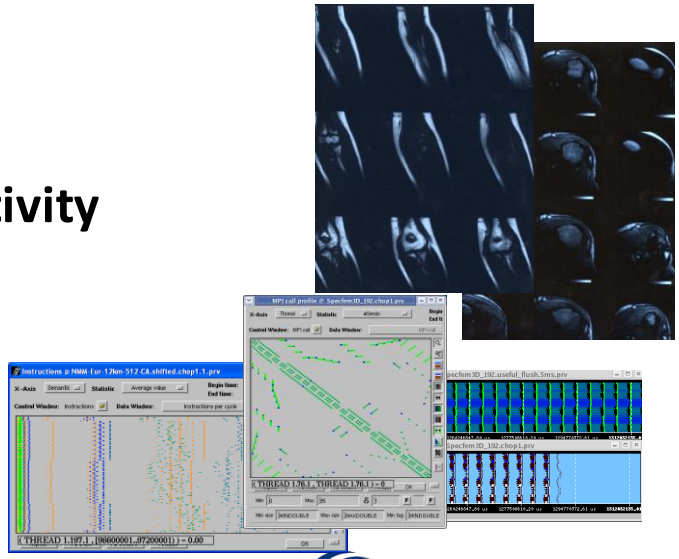
Juan Gonzalez

JLPC Summer School, Sophia Antipolis, June 2014

Why tools?

- Measurements as science enablers

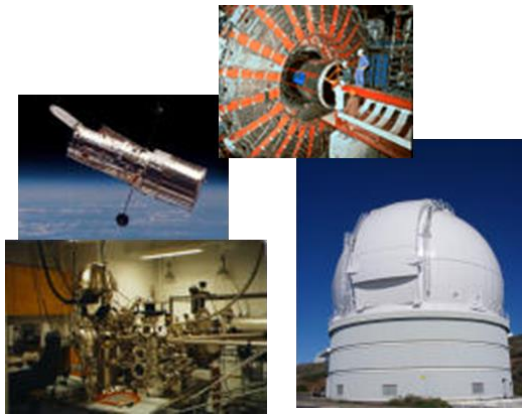
- Vital for app. development at Exascale
  - Flight instrumentation
  - Work in the right direction
- Important for "lawyers"
  - Who to blame?
- Vital for system architects
  - Understand our system → Increase productivity



- Performance analyst
  - Expert understanding displays

Tools in…

Science



Computer science

- `printf()`
- Timers
- Lots of speculation
  - We see $\int_a^b f(t)dt$
  - We talk about $f(t)$

Physics

- One “expensive” experiment
- Lots of data to analyse
- To understand a nature system

Parallel computing

- Just another system
- We should use similar practices and techniques

Outline

Performance Analysis Tools

- Ecosystem
- Data acquisition
- Data presentation

BSC Tools

- Extrae
- Paraver
- Performance Analytics

Performance analysis universe…

Acquisition

Presentation

Processors/Processes

Time

Event types

Sampling Land

PAPI

HPCTOOOLKIT

gprof

Instrumentation Land

PMPI
Dyninst
Score-P
Compiler Flags

$\int_a^b f(t)dt$

Profile

Trace

Profile Land

CUBE
Xprofiler
Paraprof
Perfexplorer
mpiP

Scalasca
Expert
TAU

Time-line Land

Vampir
Projections
Paraver

+ SIMULATION!
To be presented on
Friday Session

Performance analysis universe…

Acquisition

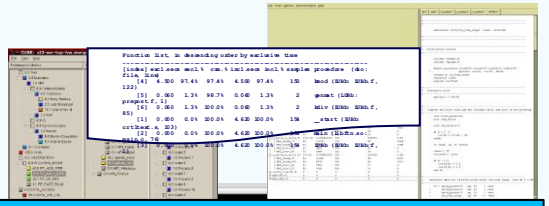
Presentation

Profile Land

Sampling Land

$\int_a^b f(t)dt$

Profile



Processors/Processes

The information available determines the possible analyses

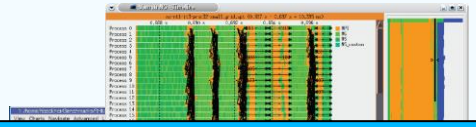
Analytics capabilities can maximize the insight delivered

Event types

Time-line Land

Instrumentation Land

$f(\text{Trace})$



May put different amounts of computation in the different phases, resulting in differences in emitted/manipulated data size and analysis flexibility

Performance tools

OBJECTIVE:

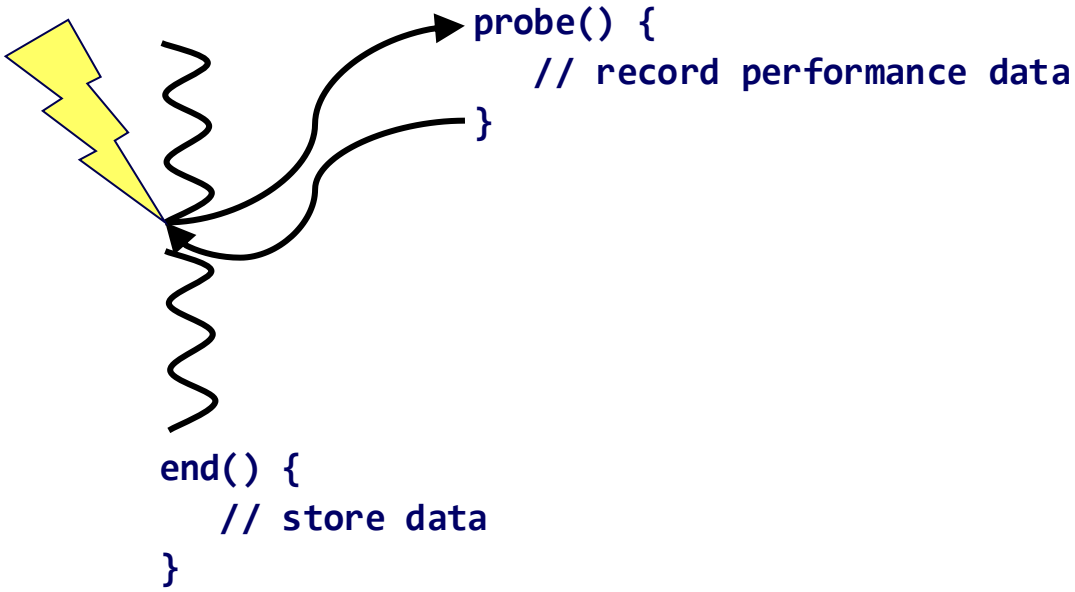
- Identify performance problems to help optimizing apps.

PHASES:

- Data acquisition
- (Data emission)
  - Compaction
  - Summarization
- Data presentation

Data acquisition

Insert probes into the running program



Issues

- **Control flow: *When a probe is called?***
- ***Which is the information available?***

Data acquisition: control flow

Sampling

- Punctual records of app. activity
- Interrupt driven: correlated (or not) with app. activity
  - POSIX clocks, PAPI overflows
- Need to project samples to total application behaviour

Instrumentation

- Probes driven by relevant application events
- Statically: link with an instrumentation library
  - PMPI / OMPTI / OMPSs / Charm++
- Dynamic: rewrite the binary
  - DynInst / DPCL

Data acquisition: Information available

- Control flow
  - Program counter (PC) register / Call stack
- Control flow arguments
  - e.g. MPI call parameters through PMPI
- Communications / Synchronizations
  - Messages transmitted / Tasks Dependencies
- Hardware Counters
  - PAPI / PMAPI
- Operating System
  - `rusage`
- Runtime internals
  - PERUSE (MPI statistics)

… and timing information too!! ☺

Data acquisition: perturbation

Probe effect in the application execution

Granularity: number of probes

- Application
- Sampling frequency / what to instrument

Overhead: cost of each probe

- Control flow
- Time measurement
- Inline processing
- Storage to buffer (or disk!)

Data presentation

What metrics?

- Time / Counts
- Absolute / Relative

How?

- Textual
- Graphical

Which type?

- Profile: accumulated statistics per app. abstraction
- Timeline: instantaneous value of metrics vs. time vs. process

Data presentation

Keep in mind this objective

- **Maximize the flow of information you require…**
  - Qualitatively: colours, shapes
  - Quantitatively: numbers
- **… by a proper balance of the approaches**

Analyse your requirements!

Which of my subroutines consumes the most time?

- Do I need time measurements?
- Do I need MPI measurements?
- Do I need a timeline?

Where is the computation performing inefficiently?

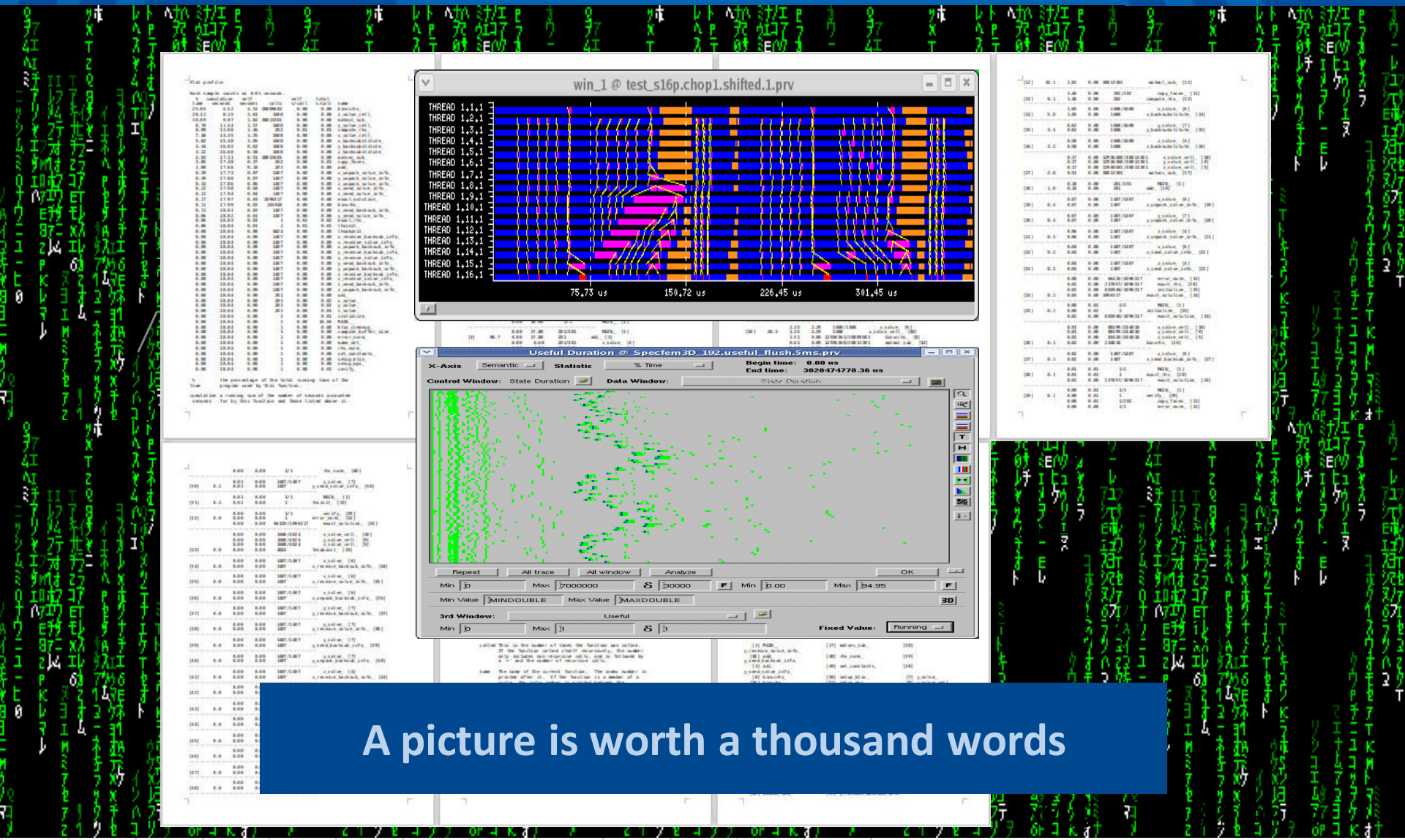
- Do I need MPI measurements?
- Do I need hardware counters?
- Do I need a timeline?

I want to simulate the MPI message exchange

- Do I need time measurements?
- Do I need MPI measurements?
- Do I need a timeline?


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
BSC TOOLS

The ways of debugging & performance analysis



A picture is worth a thousand words

Performance tools @ BSC

Since 1991

Based on traces

Flexibility and detail

Core Tools

Trace generation - Extrae

Trace analyzer - Paraver

Message passing simulator - Dimemas

Open-source

Do not speculate about your code performance

LOOK AT IT

A “different” view…

Behavioural structure vs. syntactic structure

- Algorithmic and performance
- In space and time

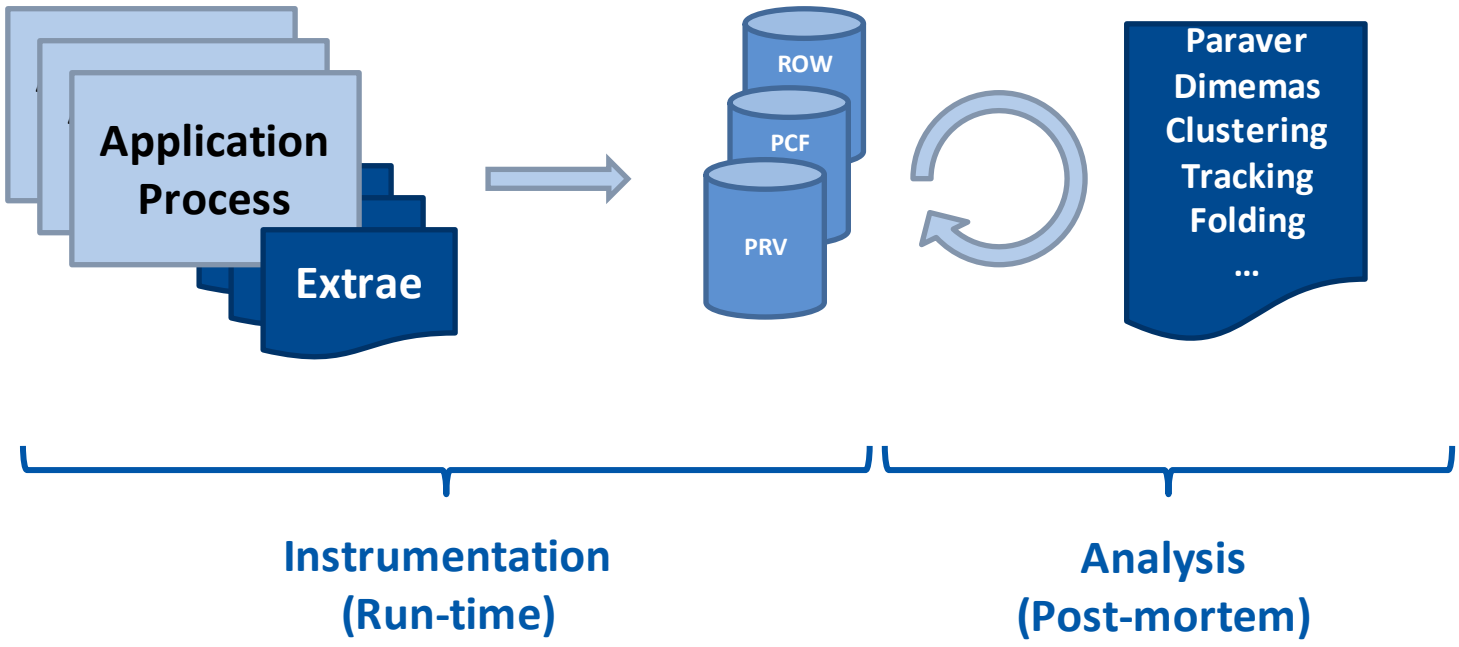
Variability

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

Flexibility

- Let the analyst navigate through capture data
- Gain insight minimizing a single app. execution

Basic workflow



The (Paraver) trace

Sequence of time stamped records (.prv)

- **Punctual events**
  - Something happened: when (time) & where (object/entity e.g. thread)
  - One record (type:value) per specific information
    - About the event
    - About the interval from previous event
- **Relations between objects (… communications)**
  - Source and destination
  - Attributes (… tag, value)

Symbolic information (.pcf)

- **Strings to name types and events**

System information (.row)

- **Strings to name the trace entities**


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
EXTRAE

Extrae features

- Parallel programming models
  - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Intel MIC…
- Performance Counters
  - Using PAPI and PMAPI interfaces
- Link to source code
  - Callstack at MPI routines
  - OpenMP outlined routines and their containers
  - Selected user functions
- Periodic samples
- User events (Extrae API)

How does Extrae work?

Symbol substitution through LD_PRELOAD

- For production binaries
- Specific library for runtime/s
- No knowledge about the app. required

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

- For production binaries
- Specify the functions to be instrumented

Based on Dyninst
U.Wisconsin/U.Maryland

Other possibilities

- Static linking (e.g. PMPI @ BG/Q)
- OmpSs instrumentation calls
- Extrae API

How to use Extrae?

Adapt your submission script

- Point the Extrae library require in the LD_PRELOAD
- Point to the XML instrumentation control file

Specify the data to be captured

- Editing 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

Extrae API

application.job

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

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

trace.sh

```
#!/bin/sh

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

# Run the desired program
$*
```

Extrae API

- If the source code is available you can use it to…
- … scalability
  - `Extrae_shutdown()`
  - `Extrae_restart()`
- … inject your own events
  - `Extrae_[n]event(type[s], value[s])`
  - `Extrae_[n]eventandcounters(…)`

Approaches towards scalability

Manual selection

- Tracing On-Off
  - Internal / External
- Trace/Buffer maximum sizes
- Information gathered
  - Call path depth
- *Bursts Mode* + Software Counters

Automatic reduction

- Spectral analysis
- Computation structure detection

Emitting "relevant" data (i.e. *Burst Mode)*

- Important information → Details
- Not that important → Software counters

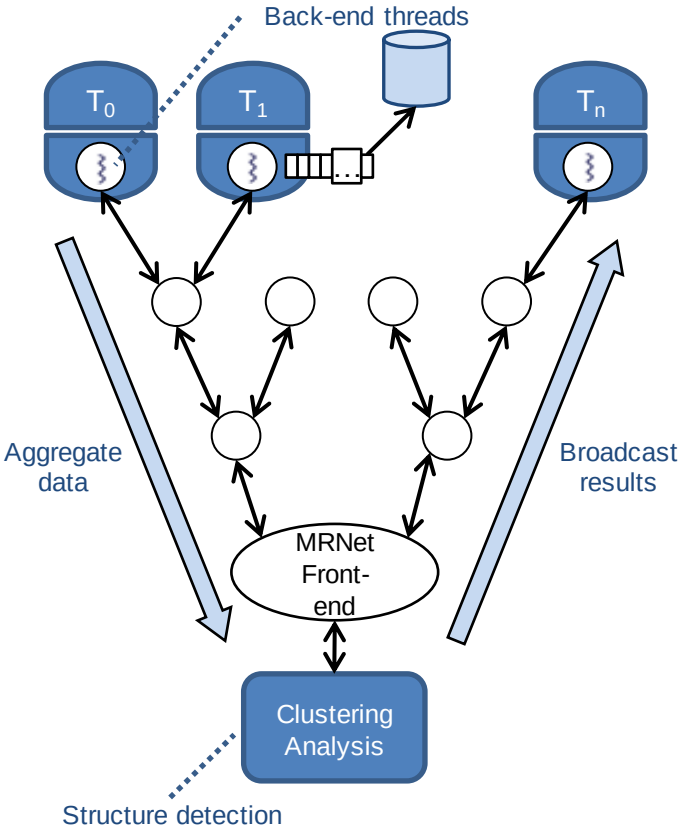
- What is important?
  - First order approach: Computation !!!
  - MPI: a gas. Fills whatever space you give it. Very often not the major cause of problems

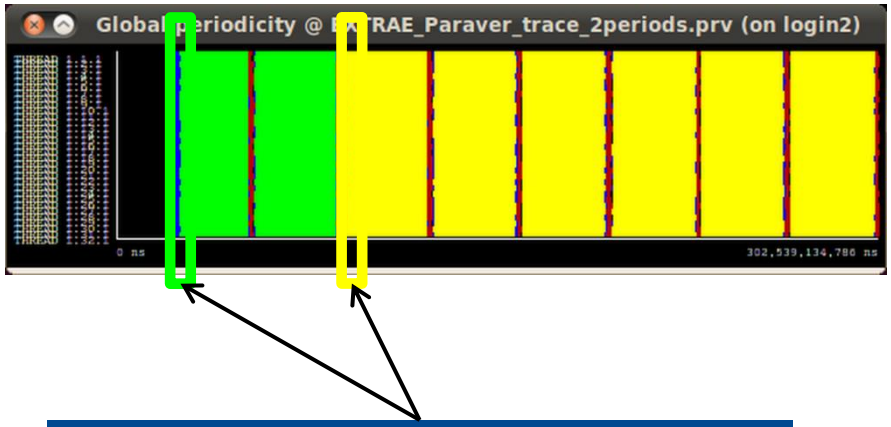
- Major computation bursts (i.e. > X ms)
  - Entry and exit timestamps and hardware counters

- Communication phases.
  - Software counters:
    - # MPI calls, aggregated bytes, %time in MPI, …

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

Scalability: online automatic interval selection





Detail trace only for small interval

G. Llort et al., *Scalable tracing with dynamic levels of detail,* ICPADS 2011


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
PARAVER

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

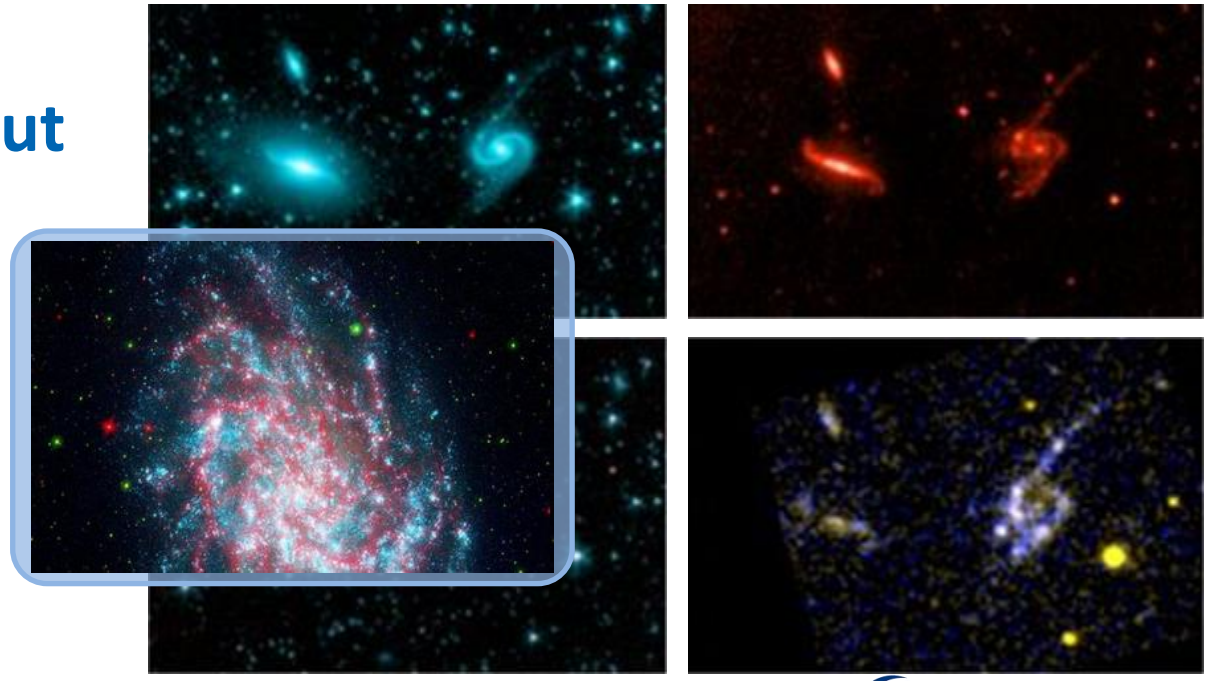
Multispectral imaging

Different looks at one reality

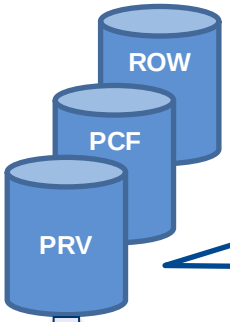
- Different light sources using filters

Highlight different aspects

Zooming In & Out

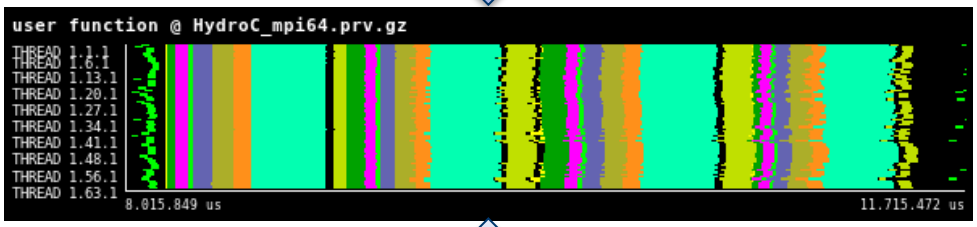


Paraver displays



The trace

MPI calls, OpenMP regions, user functions, peer-to-peer & collective communications, performance counters, samples…



Timelines

2D / 3D Tables (statistics)

Timelines: description

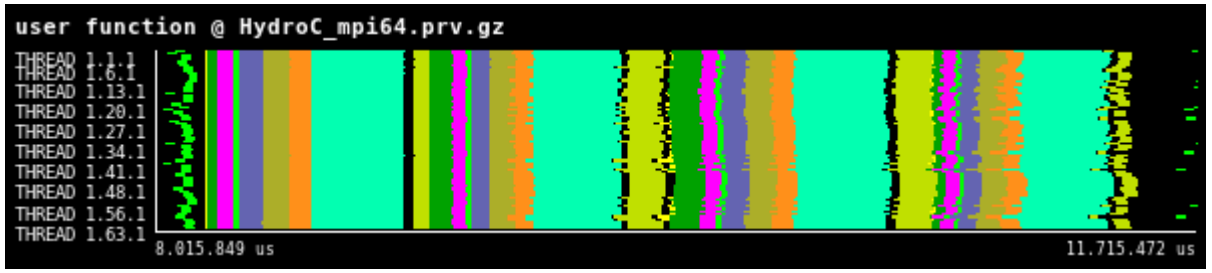
Objects

Process dimension

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

Resource dimension

- **CPU**
- **Node**
- **System**



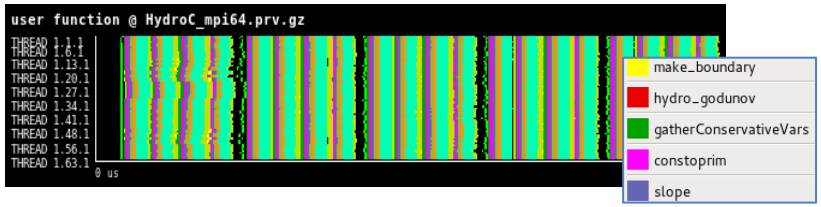
Time

Timelines: Semantics

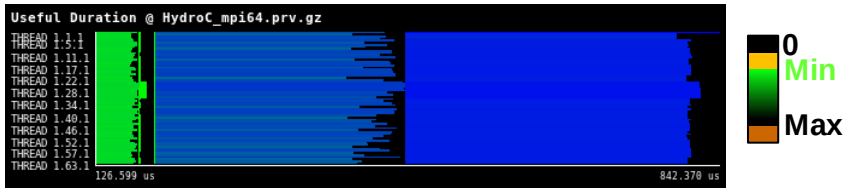
Each window computes a function of time per object

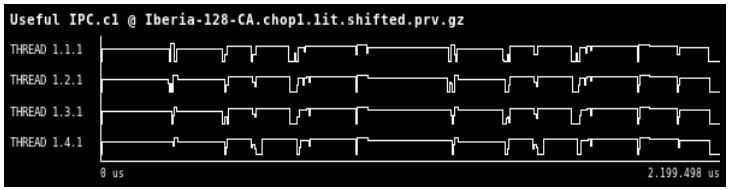
Two types of functions

- **Categorical**
  - State, User function, MPI call…
  - Color encoding
    - 1 color per value



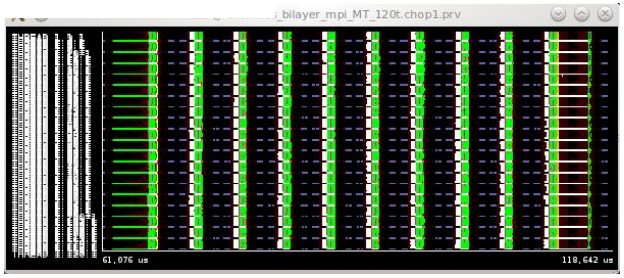
- **Numerical**
  - IPC, instructions, cache misses, computation duration…
  - Gradient encoding
    - **Black** (or background) for zero
    - From **light green** to **dark blue**
    - Limits in yellow and orange
  - Function line encoding



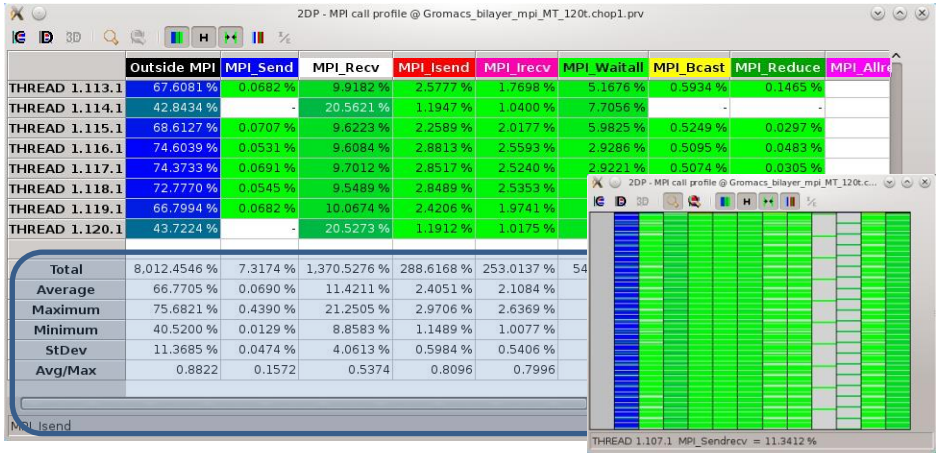


From timelines to tables

MPI calls



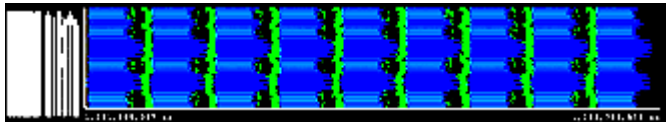
MPI calls profile



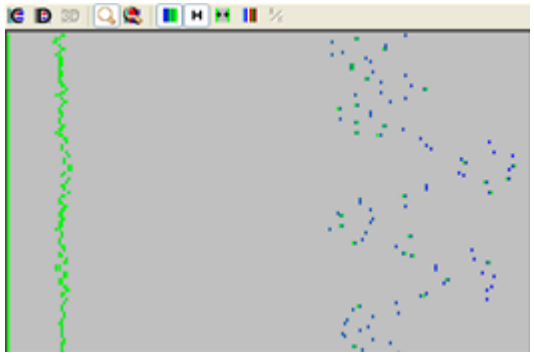
	Outside MPI	MPI_Send	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Bcast	MPI_Reduce
THREAD 1.113.1	67.6081 %	0.0682 %	9.9182 %	2.5777 %	1.7698 %	5.1676 %	0.5934 %	0.1465 %
THREAD 1.114.1	42.8434 %	-	20.5621 %	1.1947 %	1.0400 %	7.7056 %	-	-
THREAD 1.115.1	68.6127 %	0.0707 %	9.6223 %	2.2589 %	2.0177 %	5.9825 %	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 %	2.9221 %	0.5074 %	0.0305 %
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			

MPI_Isend

THREAD 1.107.1 MPI_Sendrecv = 11.3412 %

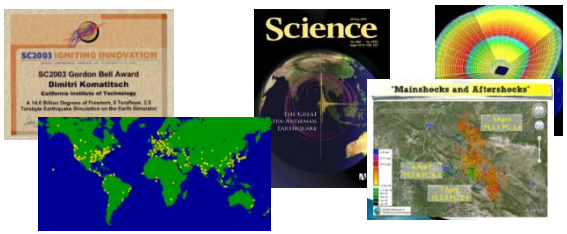


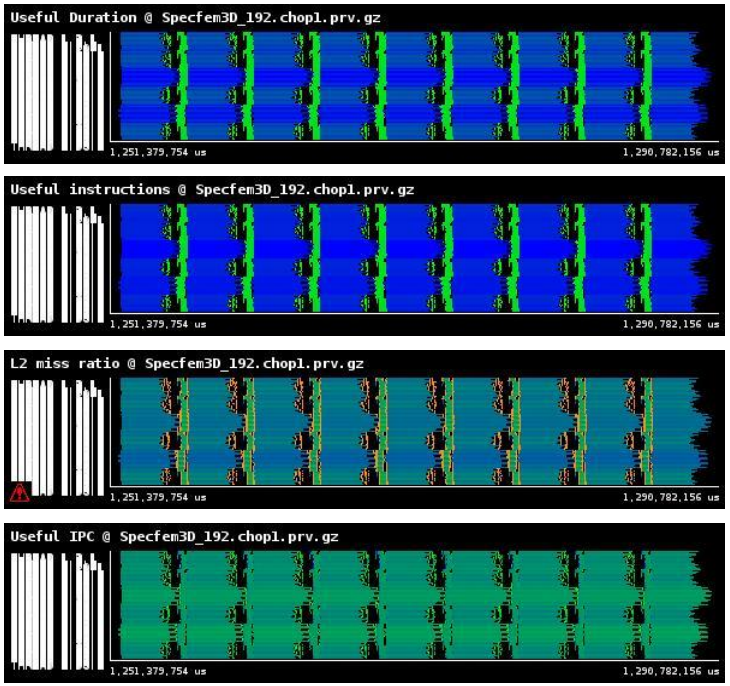
Computation duration

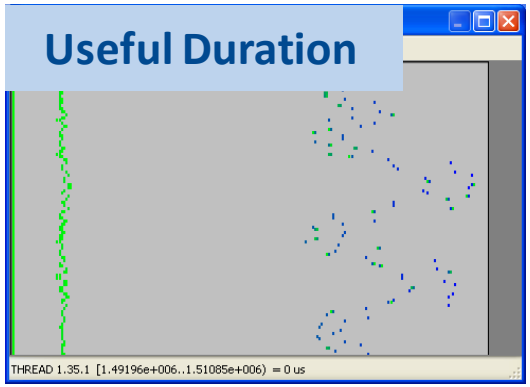


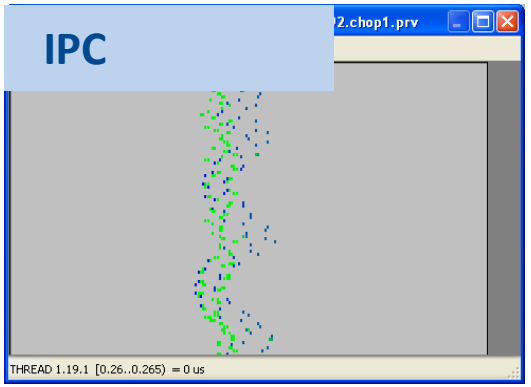
Computation duration histogram

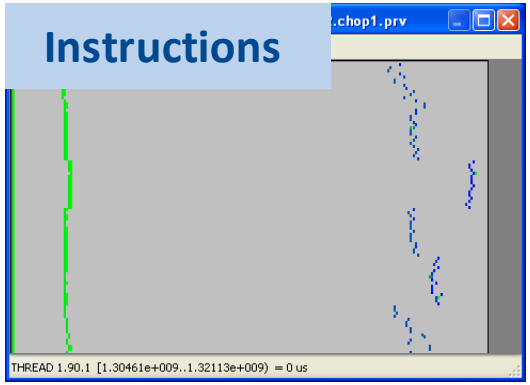
Analyzing variability through histograms and timelines

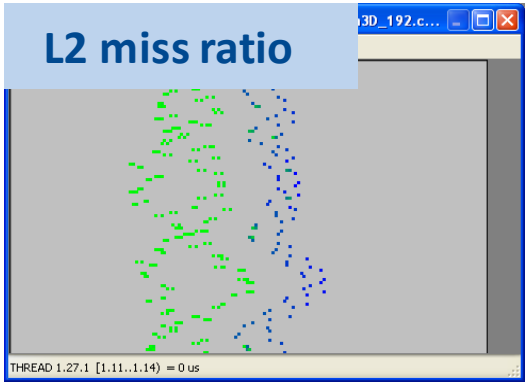






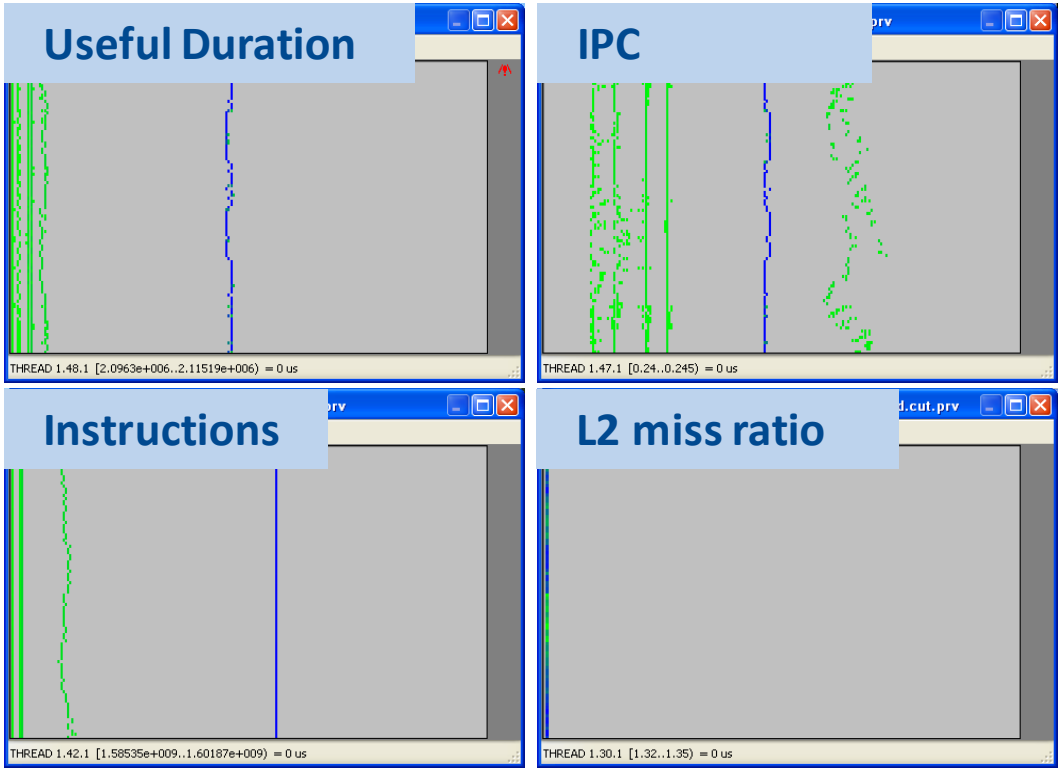






Analyzing variability through histograms and timelines

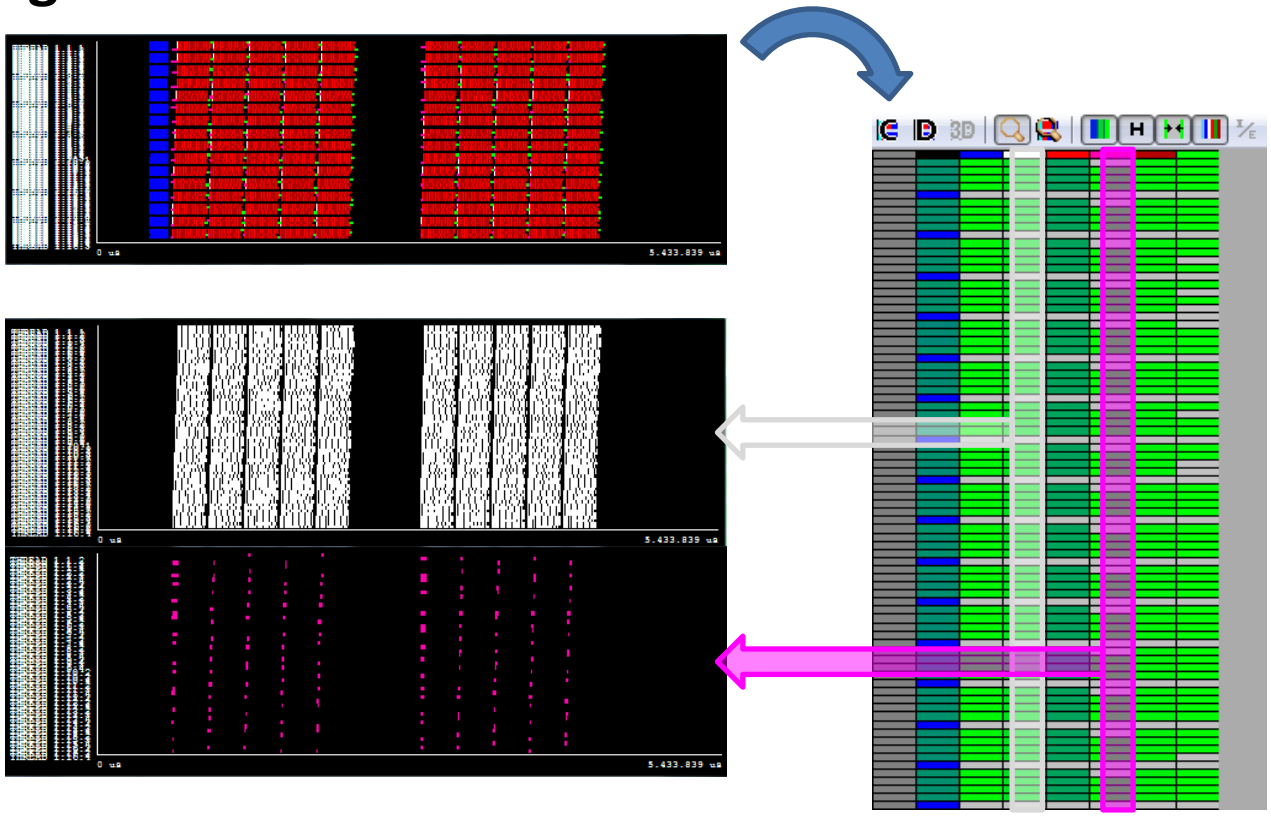
By the way: six months later…



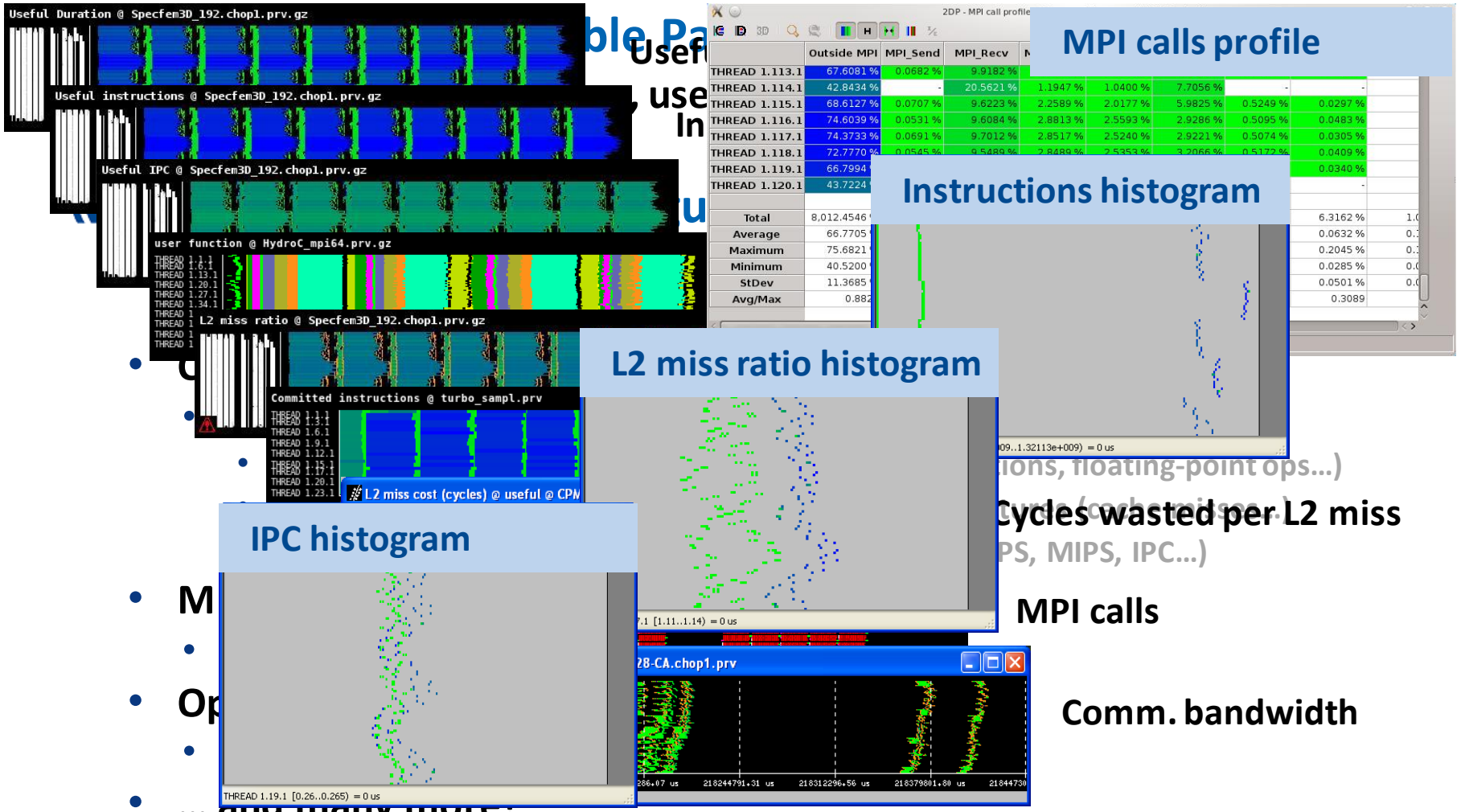
Tables: back to timelines

Where in the timeline do certain values appear?

- e.g. which is the time distribution of a given routine?
- e.g. when does a routine occur in the timeline?



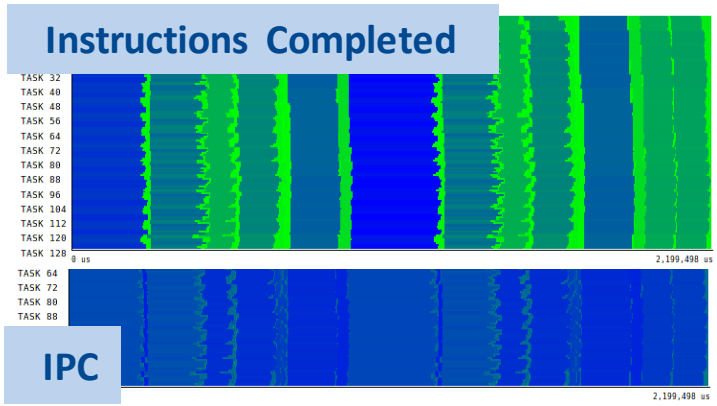
Configuration files

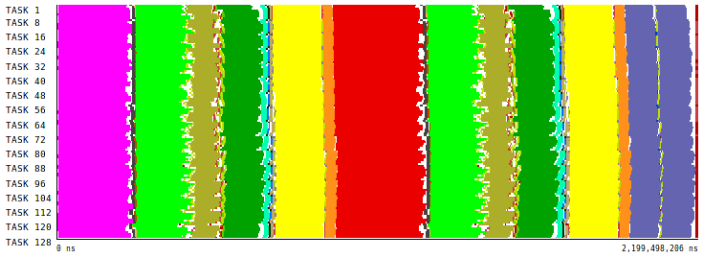


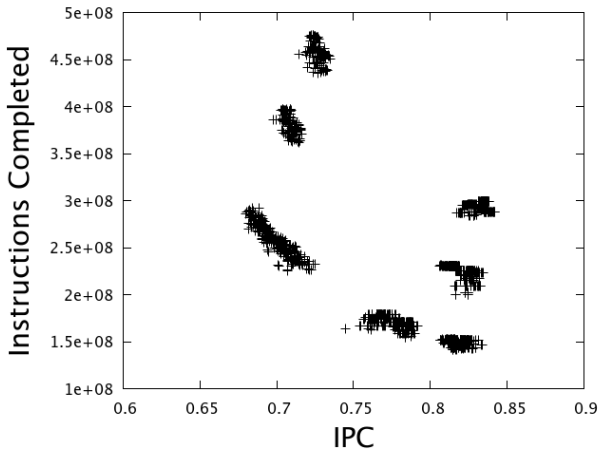
- Useful Pa
- use
- In
- u
- ions, floating-point ops…)
- Cycles wasted per L2 miss
- PS, MIPS, IPC…)
- M
- MPI calls
- Op
- Comm. bandwidth
- … and many more!

PERFORMANCE ANALYTICS

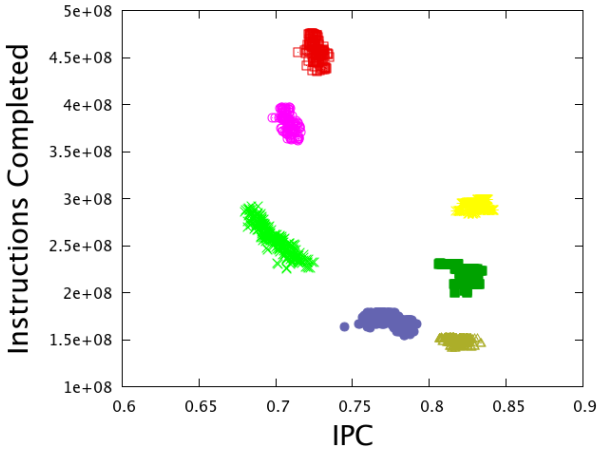
Clustering to identify structure







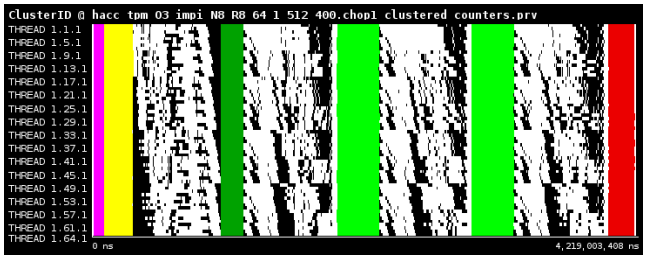
DBSCAN

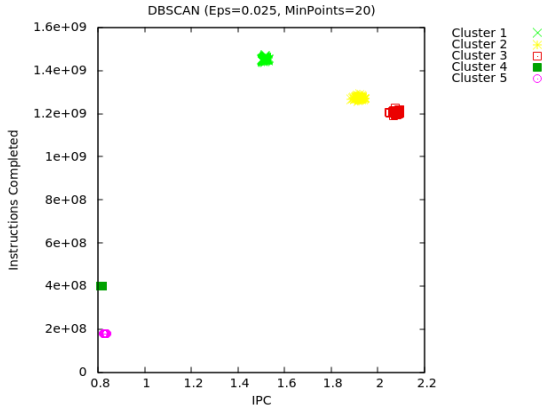


J. Gonzalez et. al., *Automatic detection of parallel applications computation phases*, IPDPS'09

Projecting hardware counters based on clustering

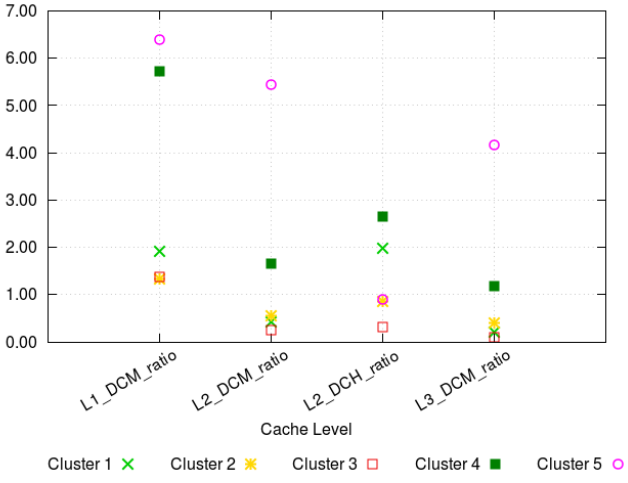
Full per region HWC characterization from a single run



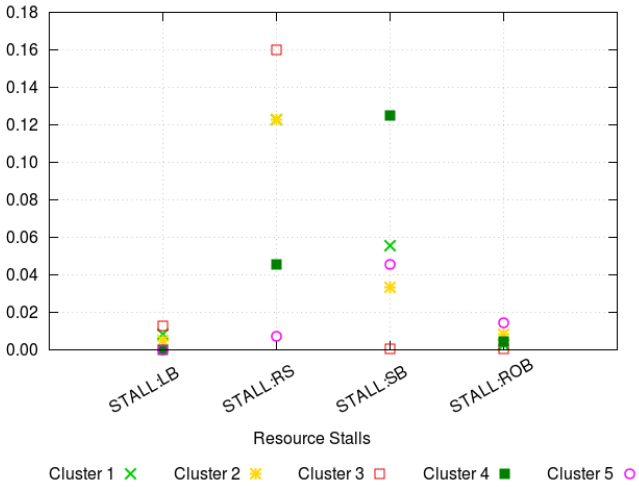


Instruction mix

Miss ratios



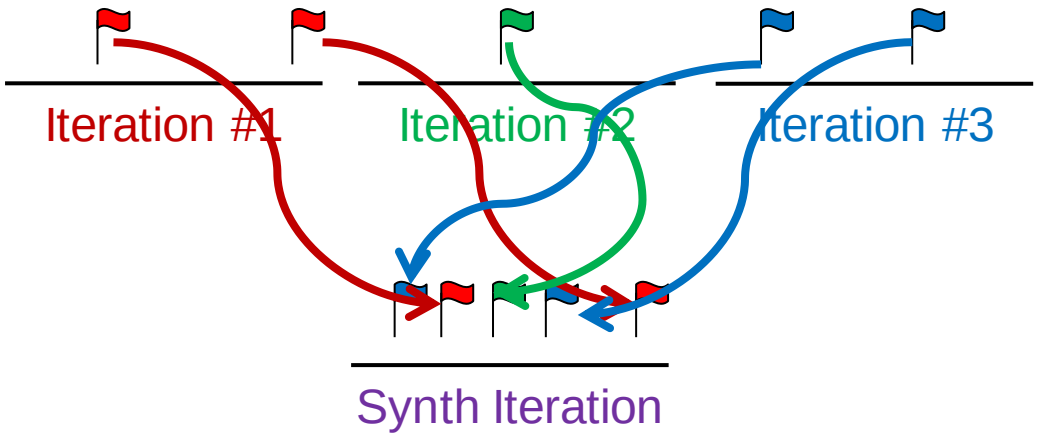
Stalls

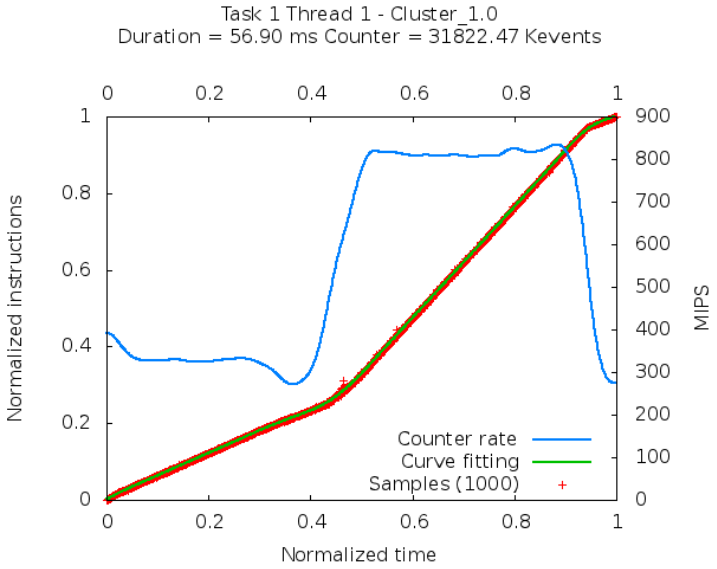


Folding

Folding mechanism

- Detailed time evolution of metrics with low-frequency sampling
- Combining sampling and instrumentation
- Taking advantage of repetitiveness



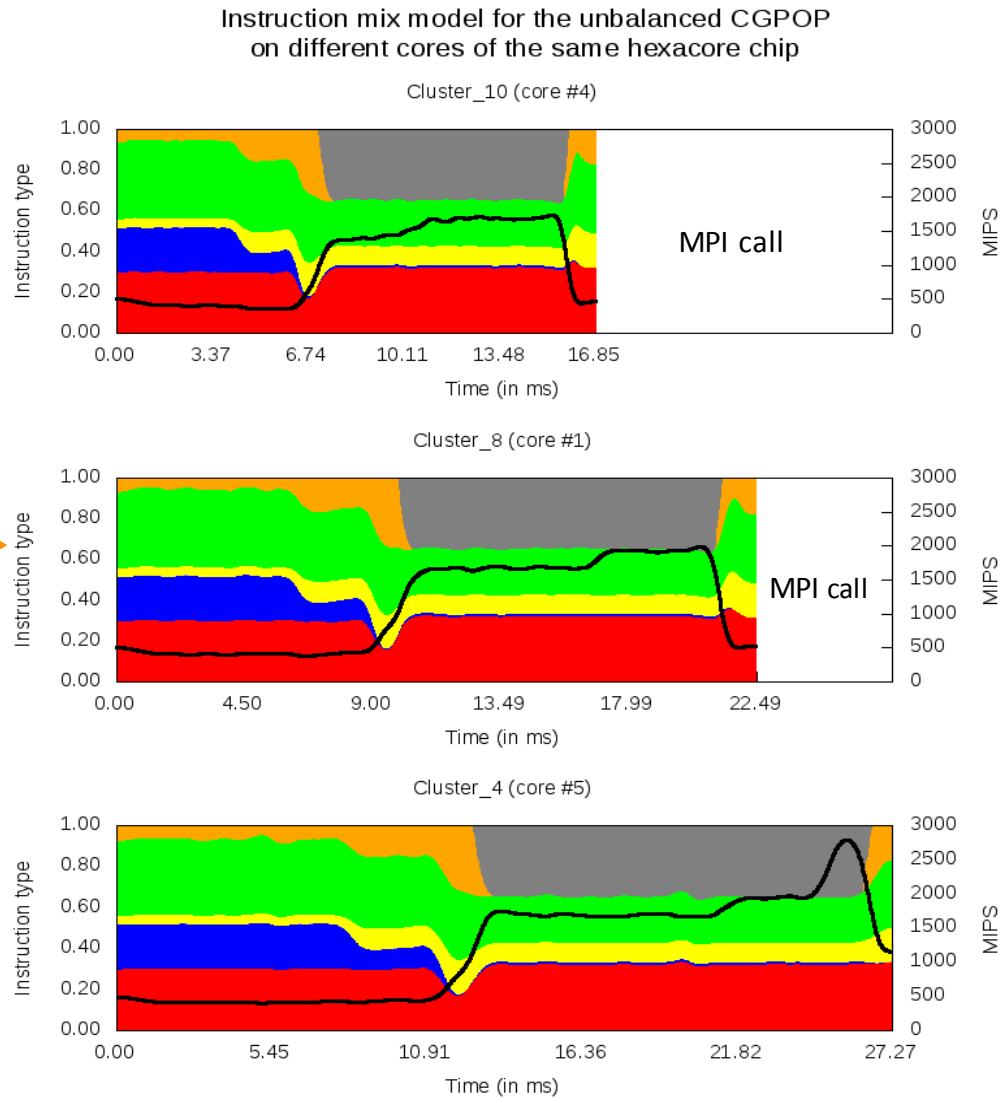


Unveiling Internal Evolution of Parallel Application Computation Phases (ICPP 2011)

Combined clustering + folding

- Instantaneous values
- All metrics
- From a single run
- "No" overhead

CGPOP -1D



CONCLUSIONS

Measure!

- Measuring your application performance is crucial
- Different acquisition and presentation techniques
  - Sampling / Instrumentation
  - Profiles / Time-lines
  - Provide different granularity
- Make the appropriate questions and select the appropriate tools

BSC Tools

Extrae: tracing mechanism

- Supports the most commonly used parallel runtimes
- Is able to capture multiple types of information

Paraver: trace visualizer & analyser

- Time-lines & Tables
- Flexible & Highly configurable (CFGs!)

Performance Analytics

- Maximum insight from a single execution
- Detailed and understendable

TOOLS DEMO + HANDS-ON


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
EXTRA!

TUNING THE EXTRAE XML FILE

Trace control xml

extrae.xml

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

<trace enabled="yes" home="/gpfs/apps/CEPBATOOLS/64.hwc">

  <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>
  </callers>

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

…
```

Activate/not MPI/OpenMP tracing and features

Add call stack info (number of levels) at tracing points

Add instrumentation at specified user functions
Requires Dyninst based mpitrace

Trace control xml

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_L1_DCM
      </set>
      <set enabled="yes" domain="user" changeat-globalops="5">
        PAPI_TOT_INS,PAPI_FP_INS,PAPI_TOT_CYC
      </set>
    </cpu>

    <network enabled="yes" />

    <resource-usage enabled="yes" />
  </counters>

…
```

Specification of counters emitted to trace

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

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/bsc41/bsc41273</final-
directory>
    <gather-mpits enabled="no" />
  </storage>

  <buffer enabled="yes">
    <size enabled="yes">150000</size>
    <circular enabled="no" />
  </buffer>

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

…
```

Control of emitted trace …

… name, tmp and final dir …

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

Size of in core buffer (#events)

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

Trace control xml

mpitrace.xml (cont)

```
…

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

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

  <cell enabled="no">
    <spu-file-size enabled="yes">5</spu-file-size>
    <spu-buffer-size enabled="yes">64</spu-buffer-size>
    <spu-dma-channel enabled="yes">2</spu-dma-channel>
  </cell>

</trace>
```

Stop tracing after elapsed time …

… or when signal received

… emit only computation bursts of a minimal duration …

… plus summarized MPI events

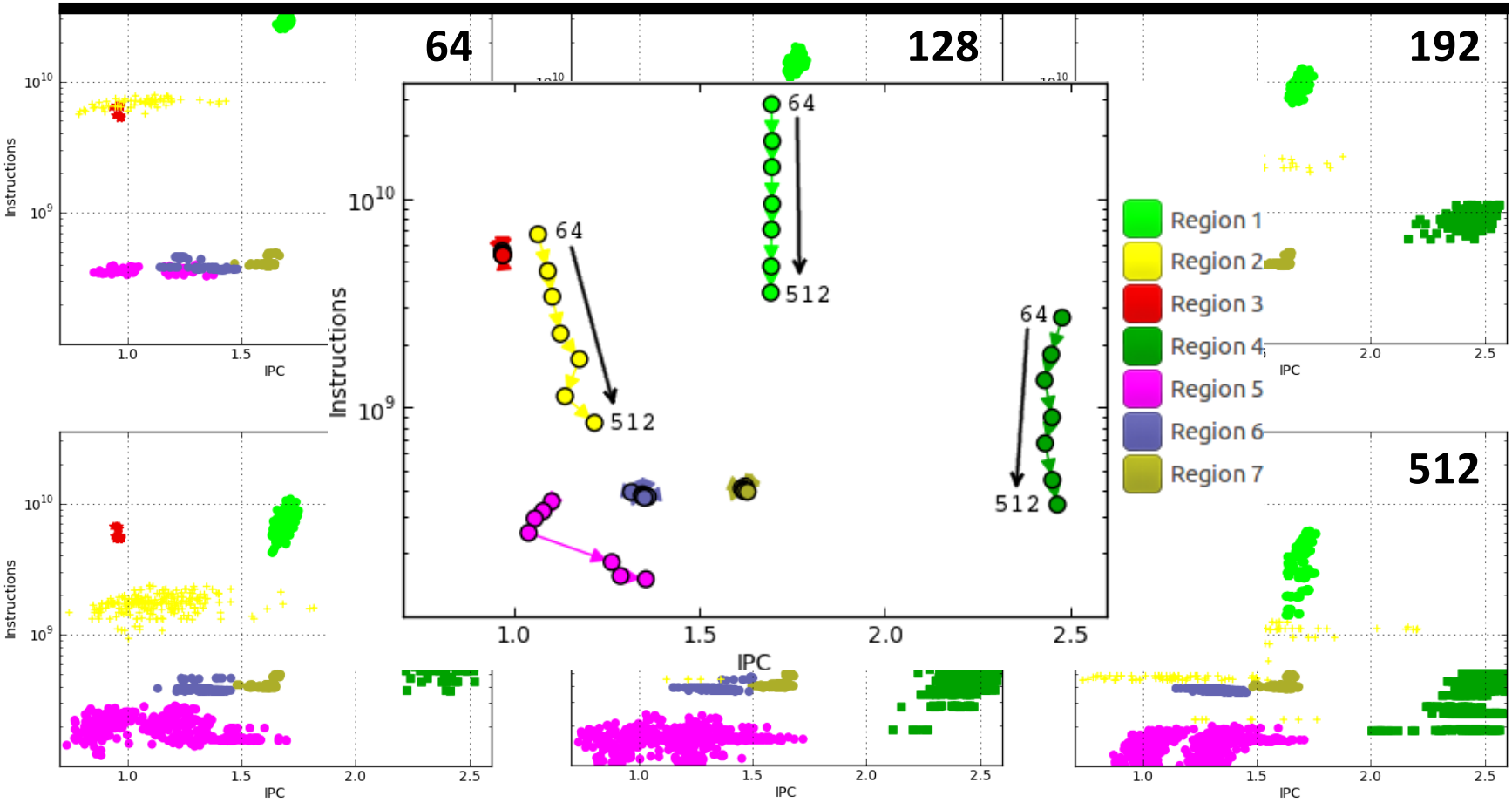
Cell specifics

More on Analytics

Tracking structural evolution

- Frame sequence: clustered scatterplot as core counts increases

OpenMX Strong scaling



G.Llort et. al., *On the Usefulness of Object Tracking Techniques in Performance Analysis*, SC 2013

Sampling frequency

Trade-off:

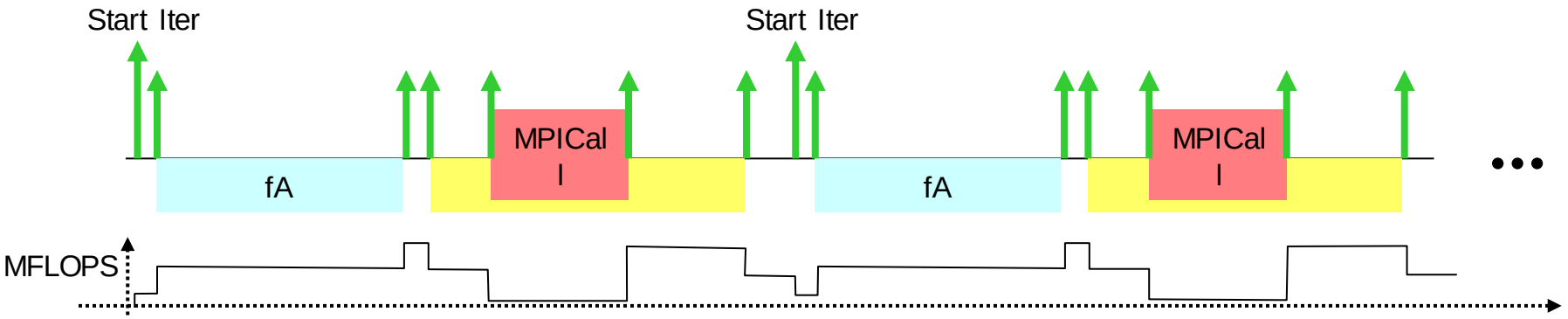
- Too low → no detail
- Too high → too much overhead

Challenge: Can we get

- lot of detail, very fine grain information :
  - i.e. “instantaneous” performance metric rates
- With very little overhead:
  - ie. sampling a few times per second

Instrumentation

- Events correlated to specific program activity
  - Start/exit iterations, functions, loops,…

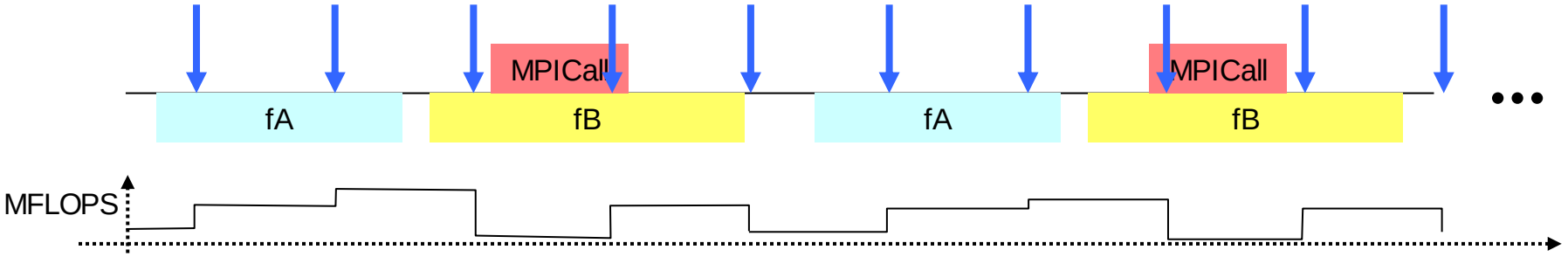


- Different intervals:
  - May be very large, may be very short
  - Variable precision
- Captured data:: Hardware counters, call arguments, call path,….
- Accurate statistics: profiles, …

Sampling

Events uncorrelated to program activity (at least not specific)

- Time (or counter) overflow



Controlled granularity:

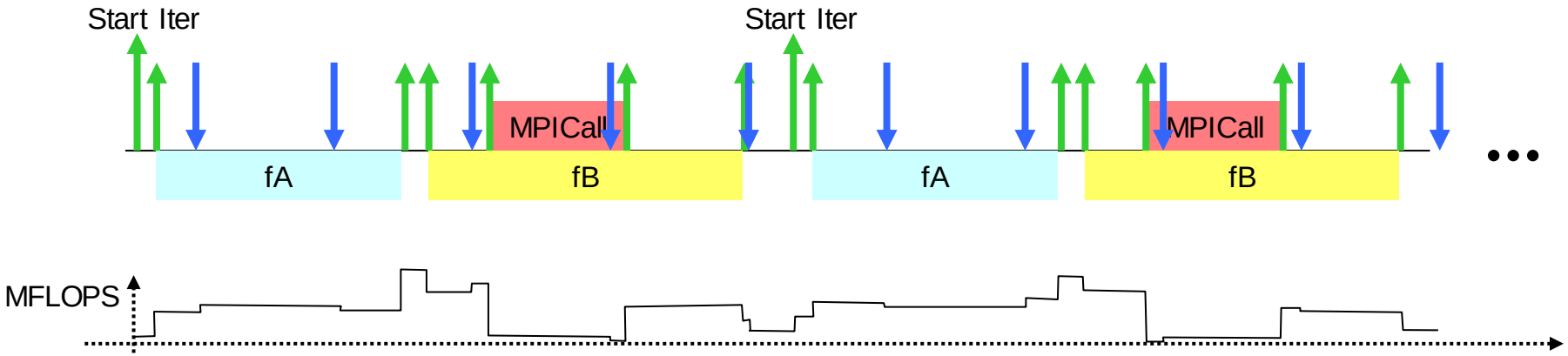
- Sufficiently large to minimize overhead
- Guaranteed acquisition interval/precision

Statistical projection

- %time (or metric) = f(%counts)
- Assuming no correlation, sufficiently large #samples

Instrumentation + sampling

Both



Guaranteed interval

Captured data:

- Hardware counters (since previous probe)
- call path
- Call arguments in some probes