Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

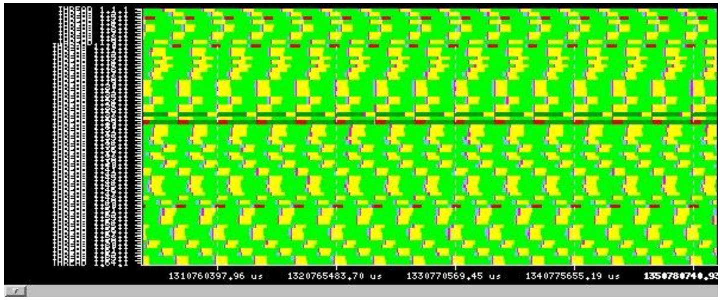
Understanding applications with Paraver

tools@bsc.es

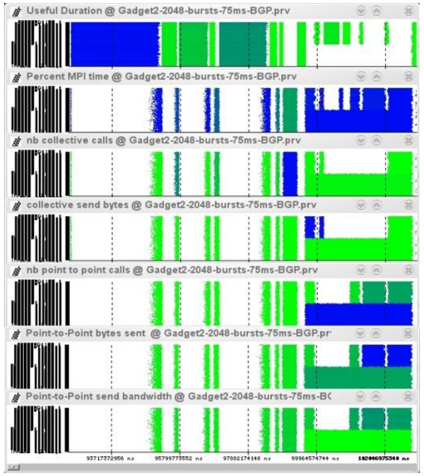
2018

Our Tools

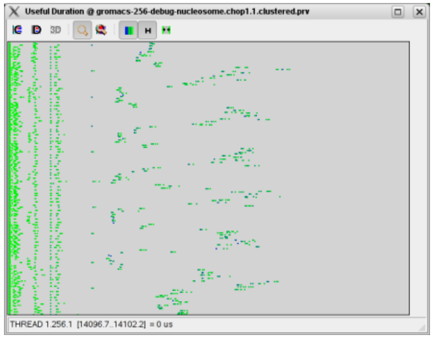
- Since 1991
- Based on traces
- Open Source
- http://tools.bsc.es



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



- Focus
  - Detail, variability, flexibility
  - Behavioral structure vs. syntactic structure
  - Intelligence: Performance Analytics

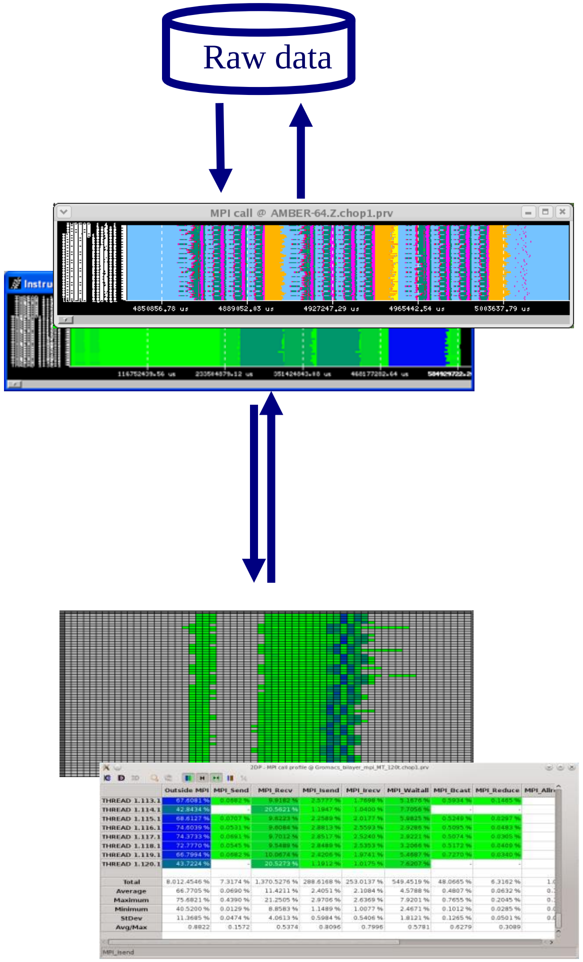


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Paraver

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Paraver – Performance data browser



Timelines

2/3D tables (Statistics)

Trace visualization/analysis

+ trace manipulation

Goal = Flexibility

No semantics

Programmable

Comparative analyses

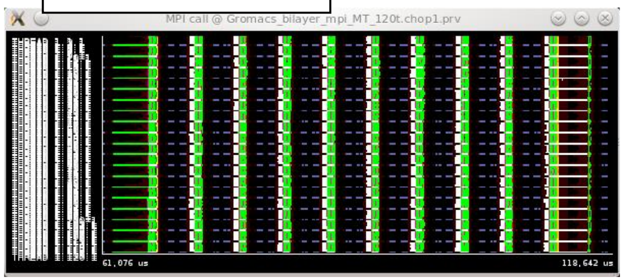
Multiple traces

Synchronize scales

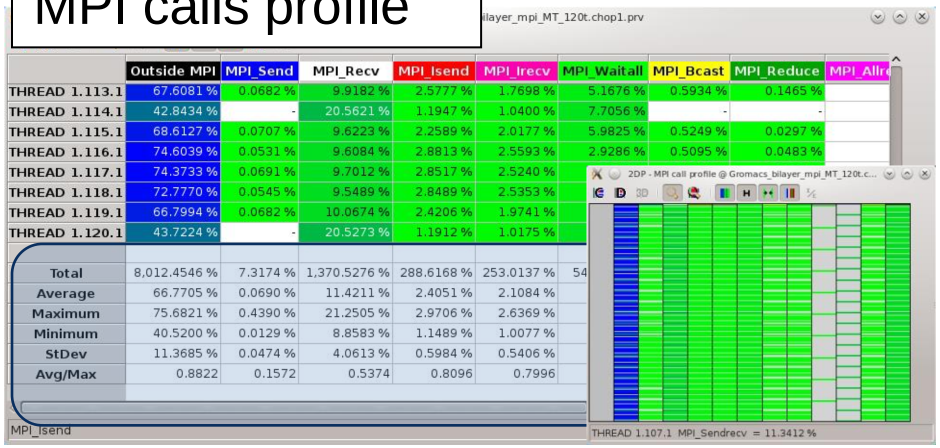
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

From timelines to tables

MPI calls

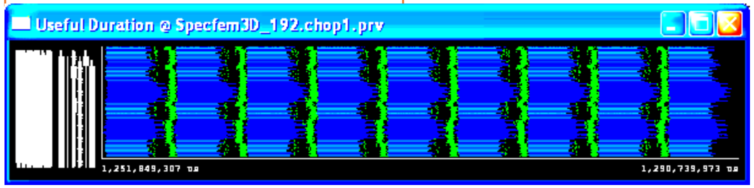


MPI calls profile

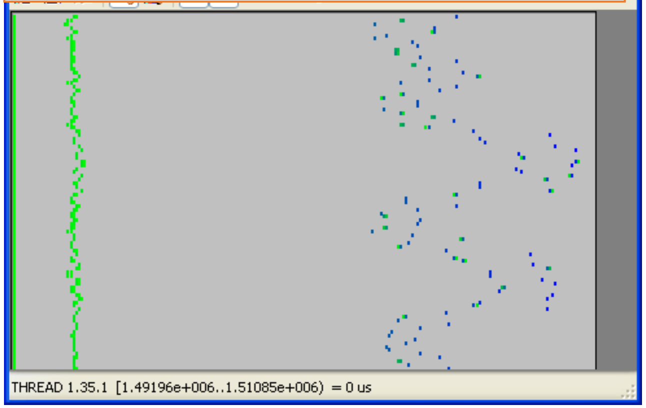


	Outside MPI	MPI_Send	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Bcast	MPI_Reduce	MPI_Allre
THREAD 1.113.1	67.6081 %	0.0682 %	9.9182 %	2.5777 %	1.7698 %	5.1676 %	0.5934 %	0.1465 %	
THREAD 1.114.1	42.8434 %	-	20.5621 %	1.1947 %	1.0400 %	7.7056 %	-	-	
THREAD 1.115.1	68.6127 %	0.0707 %	9.6223 %	2.2589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

Useful Duration

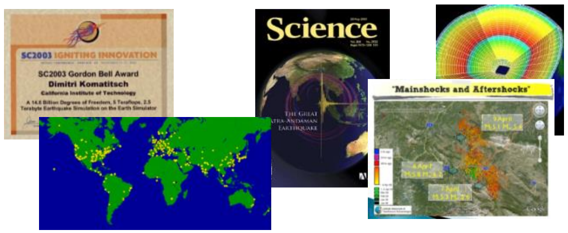


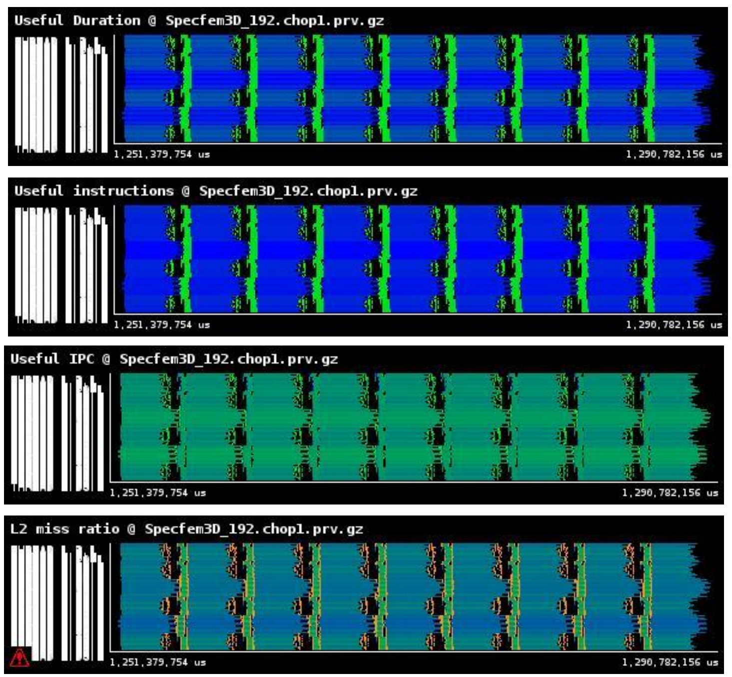
Histogram Useful Duration

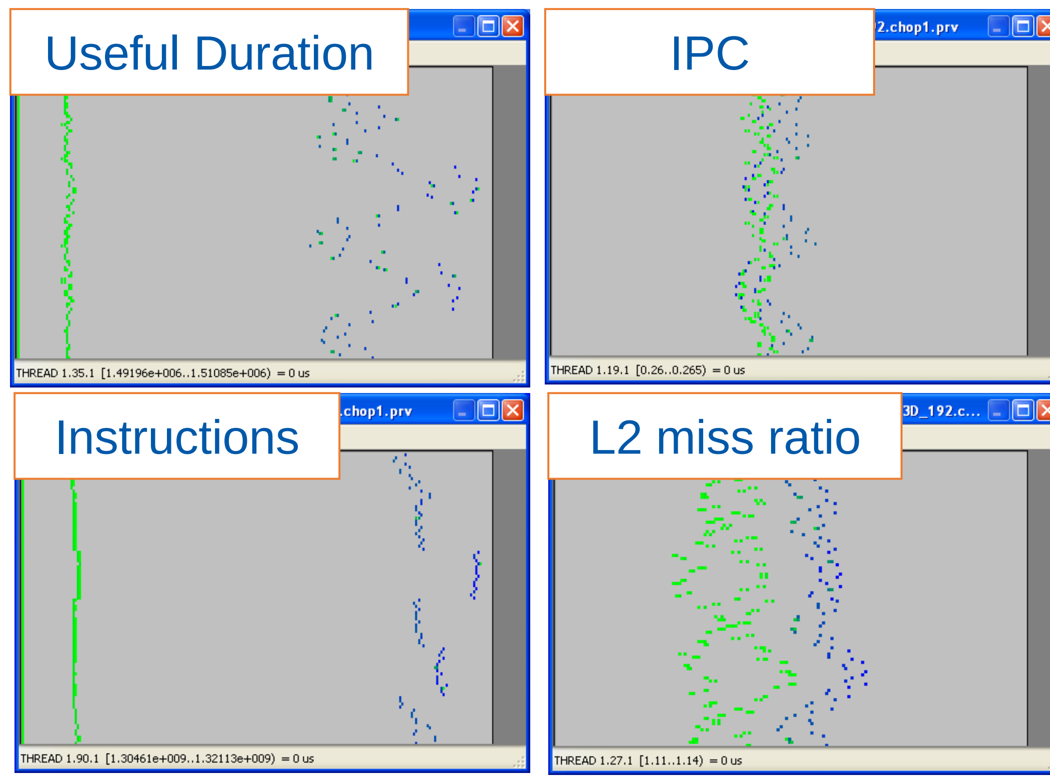


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Analyzing variability



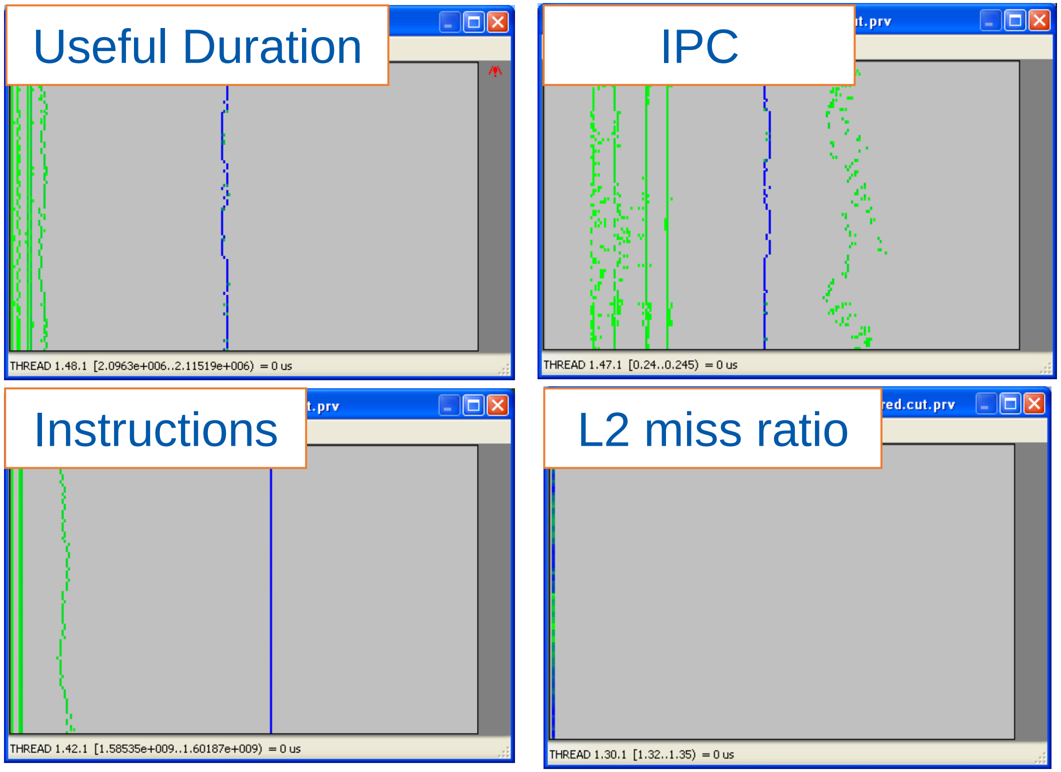




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Analyzing variability

- By the way: six months later ….

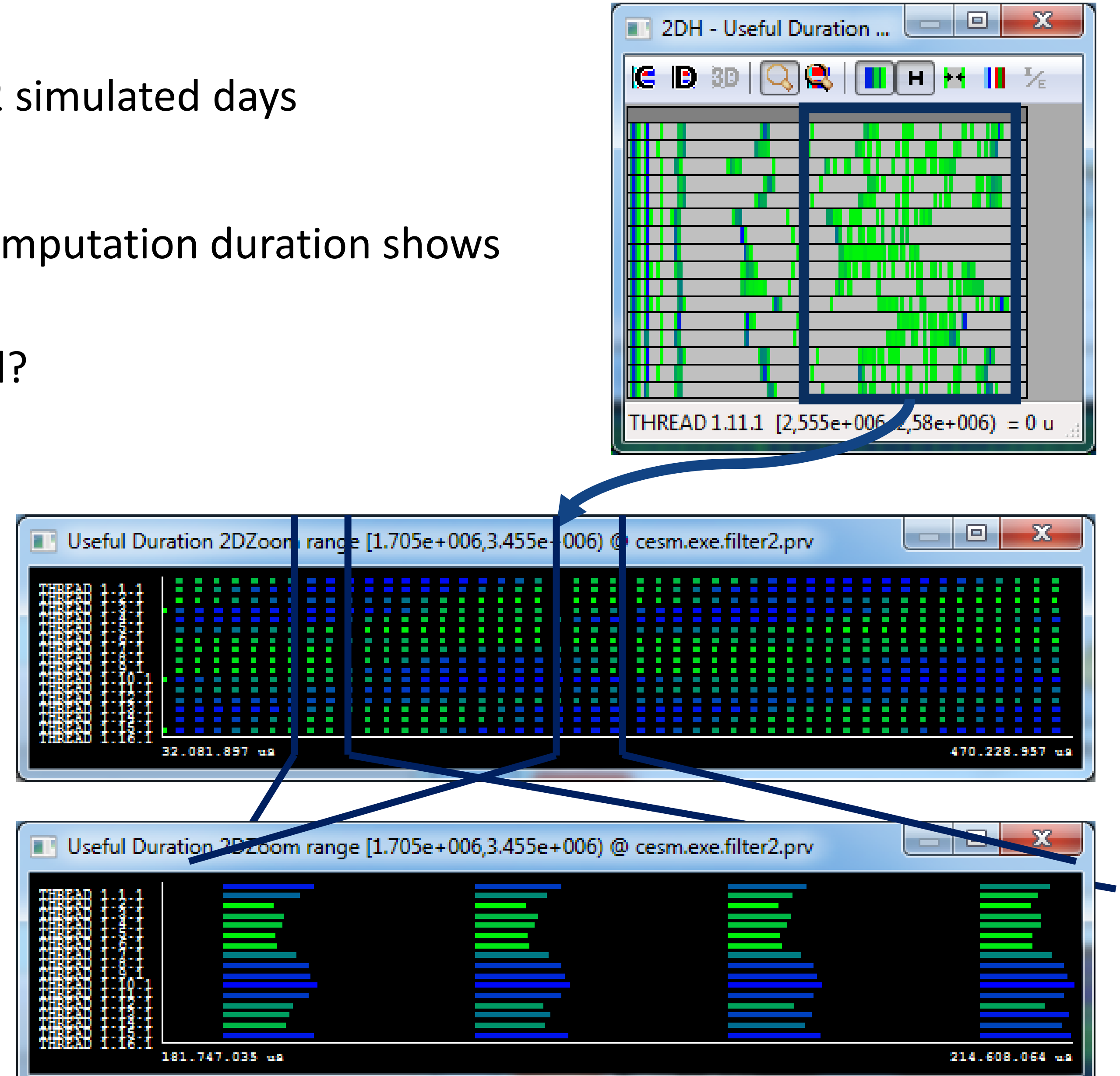


From tables to timelines

CESM: 16 processes, 2 simulated days

- Histogram useful computation duration shows high variability
- How is it distributed?

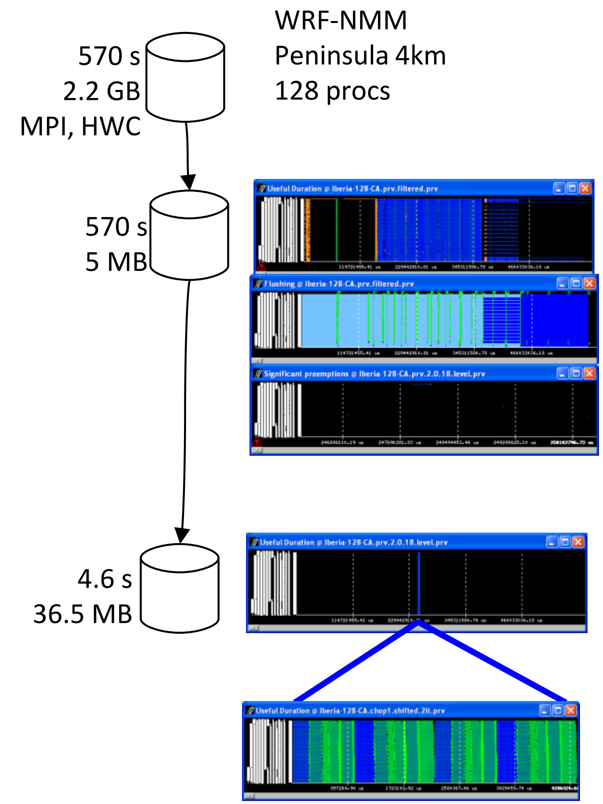
- Dynamic imbalance
  - In space and time
  - Day and night.
  - Season ? ☺



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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,…



Extrae

Extrae features

- Platforms
  - Intel, Cray, BlueGene, MIC, ARM, Android, Fujitsu Sparc…
- Parallel programming models
  - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Java, Python…
- Performance Counters
  - Using PAPI interface
- Link to source code
  - Callstack at MPI routines
  - OpenMP outlined routines
  - Selected user functions (Dyninst)
- Periodic sampling
- User events (Extrae API)

No need to recompile / relink!

Extrae overheads

	Average values
Event	150 – 200 ns
Event + PAPI	750 ns – 1.5 us
Event + callstack (1 level)	1 us
Event + callstack (6 levels)	2 us

How does Extrae work?

- Symbol substitution through LD_PRELOAD
  - Specific libraries for each combination of runtimes
    - MPI
    - OpenMP
    - OpenMP+MPI
    - …

Recommended

- Dynamic instrumentation
  - Based on Dyninst (developed by U.Wisconsin / U.Maryland)
    - Instrumentation in memory
    - Binary rewriting

- Alternatives
  - Static link (i.e., PMPI, Extrae API)

Extrae XML configuration

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

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

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

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

Trace the MPI calls
(What's the program doing?)

Trace the call-stack
(Where in my code?)

Extrae XML configuration (II)

```
<counters enabled="yes">
  <cpu enabled="yes" starting-set-distribution="1">
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_L1_DCM, PAPI_L2_DCM, PAPI_L3_TCM
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_BR_MSP, PAPI_BR_UCN,
      PAPI_BR_CN, RESOURCE_STALLS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_VEC_DP, PAPI_VEC_SP,
      PAPI_FP_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_LD_INS, PAPI_SR_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, RESOURCE_STALLS:LOAD,
      RESOURCE_STALLS:STORE, RESOURCE_STALLS:ROB_FULL, RESOURCE_STALLS:RS_FULL
    </set>
  </cpu>
  <network enabled="no" />
  <resource-usage enabled="no" />
  <memory-usage enabled="no" />
</counters>
```

Select which HW counters are measured

(How's the machine doing?)

Extrae XML configuration (III)

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

<sampling enabled="no" type="default" period="50m" variability="10m" />

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

Trace buffer size
(Flush/memory trade-off)

Enable sampling
(Want more details?)

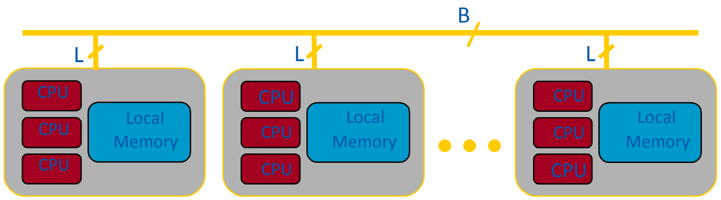
Automatic post-processing to generate the Paraver trace

Dimemas

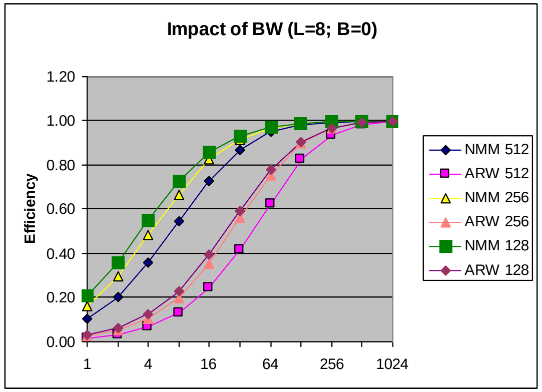
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Dimemas – Coarse grain, Trace driven simulation

- Simulation: Highly non linear model
  - MPI protocols, resource contention…



- Parametric sweeps
  - On abstract architectures
  - On application computational regions
- What if analysis
  - Ideal machine (instantaneous network)
  - Estimating impact of ports to MPI+OpenMP/CUDA/…
  - Should I use asynchronous communications?
  - Are all parts equally sensitive to network?
- MPI sanity check
  - Modeling nominal
- Paraver – Dimemas tandem
  - Analysis and prediction
  - What-if from selected time window

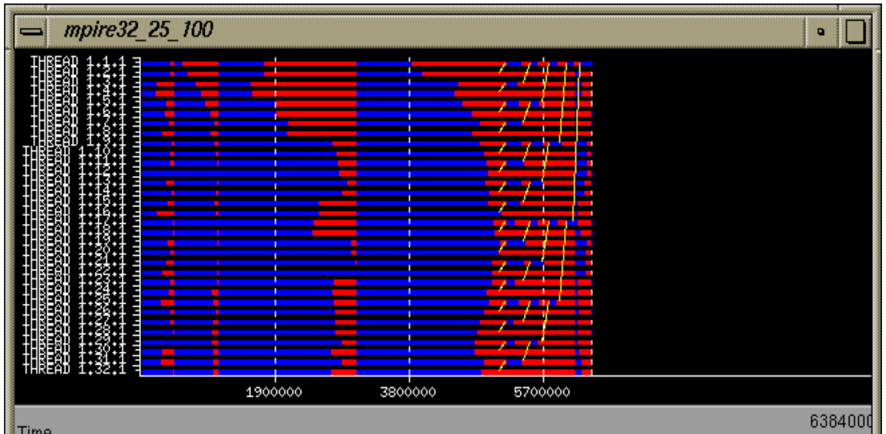


Detailed feedback on simulation (trace)

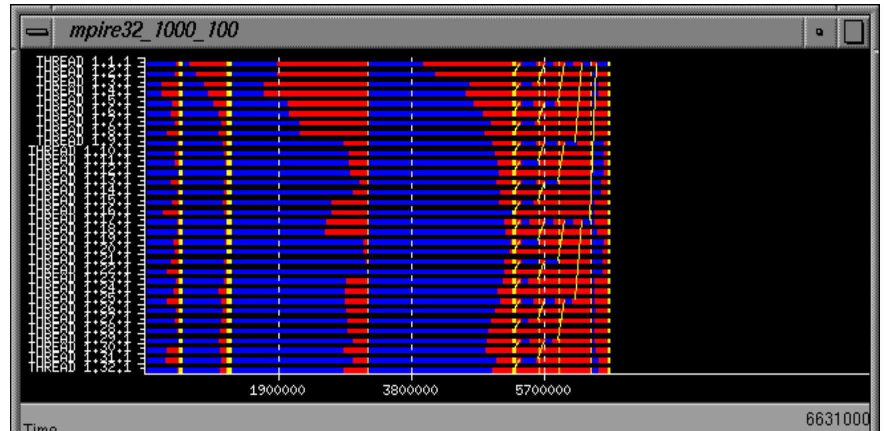
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Network sensitivity

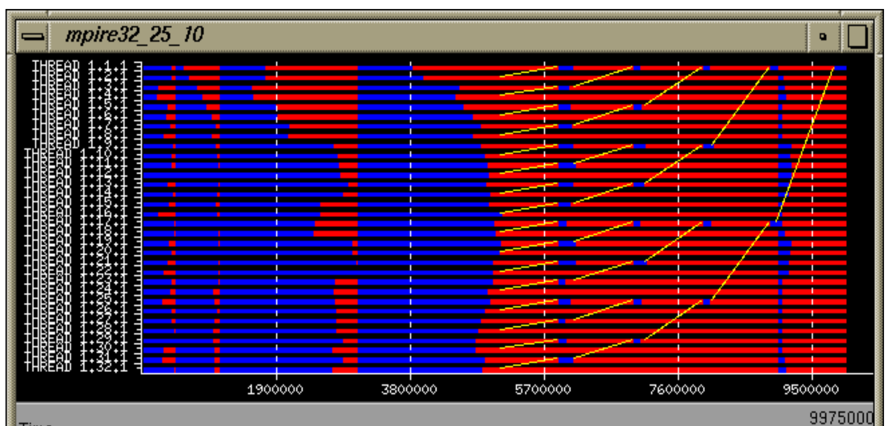
- MPIRE 32 tasks, no network contention



L = 5μs – BW = 1 GB/s



L = 1000μs – BW = 1 GB/s



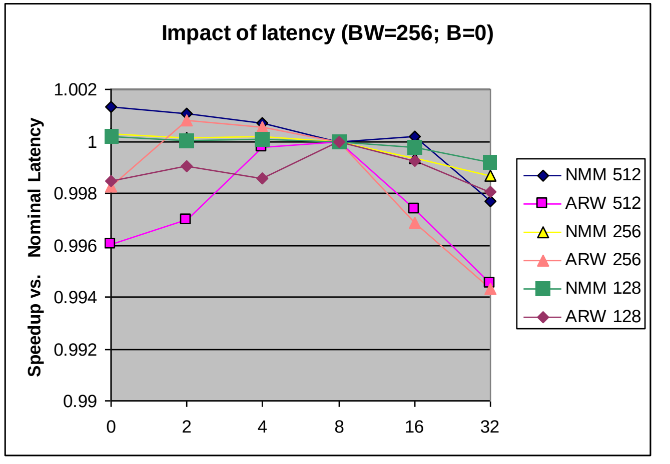
L = 5μs – BW = 100MB/s

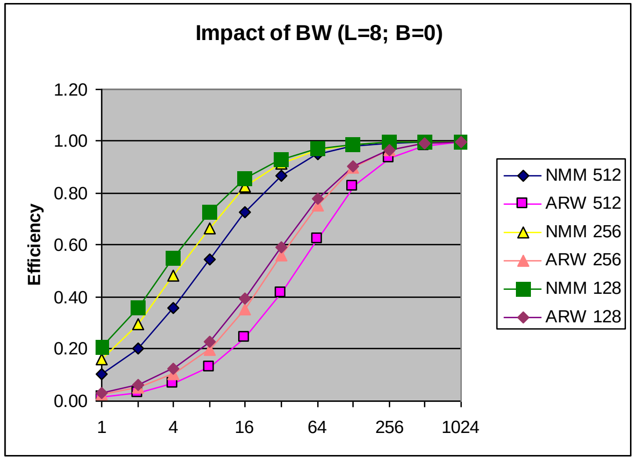
All windows same scale

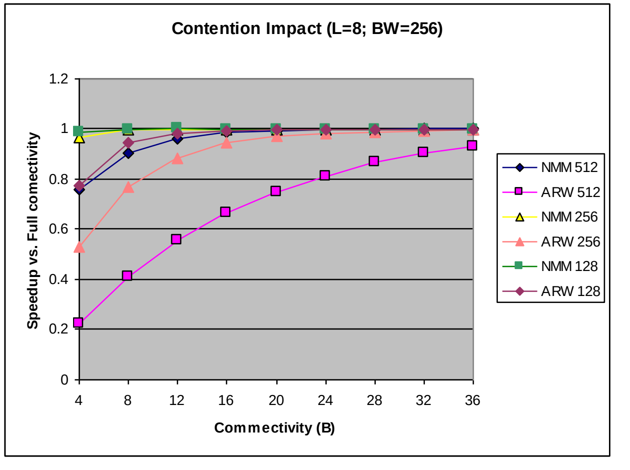
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Network sensitivity

- WRF, Iberia 4Km, 4 procs/node
  - Not sensitive to latency
  - NMM
    - BW – 256MB/s
    - 512 – sensitive to contention
  - ARW
    - BW - 1GB/s
    - Sensitive to contention



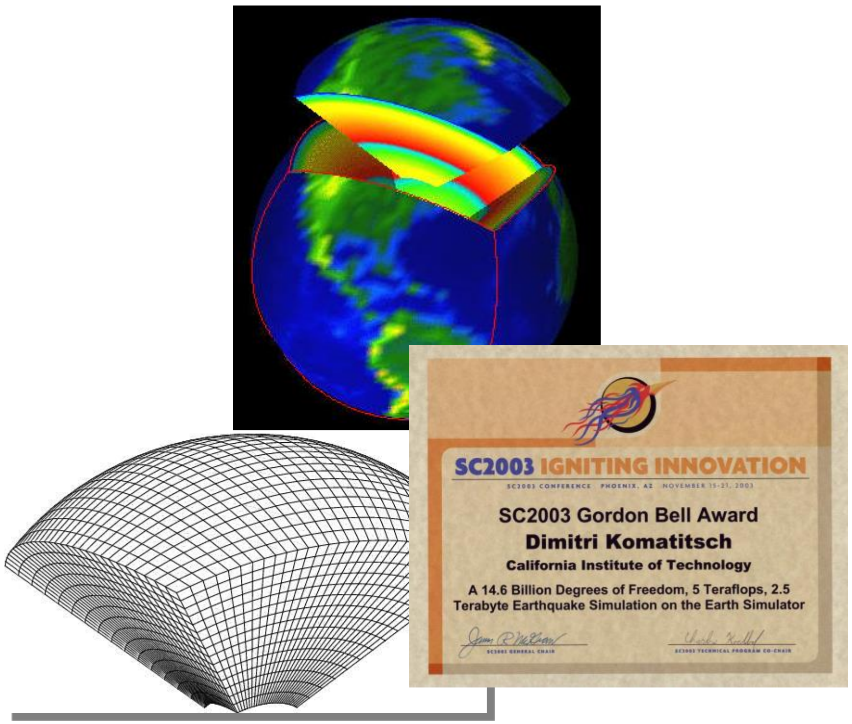




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Would I will benefit from asynchronous communications?

SPECFEM3D



Courtesy Dimitri Komatitsch

Real

Ideal

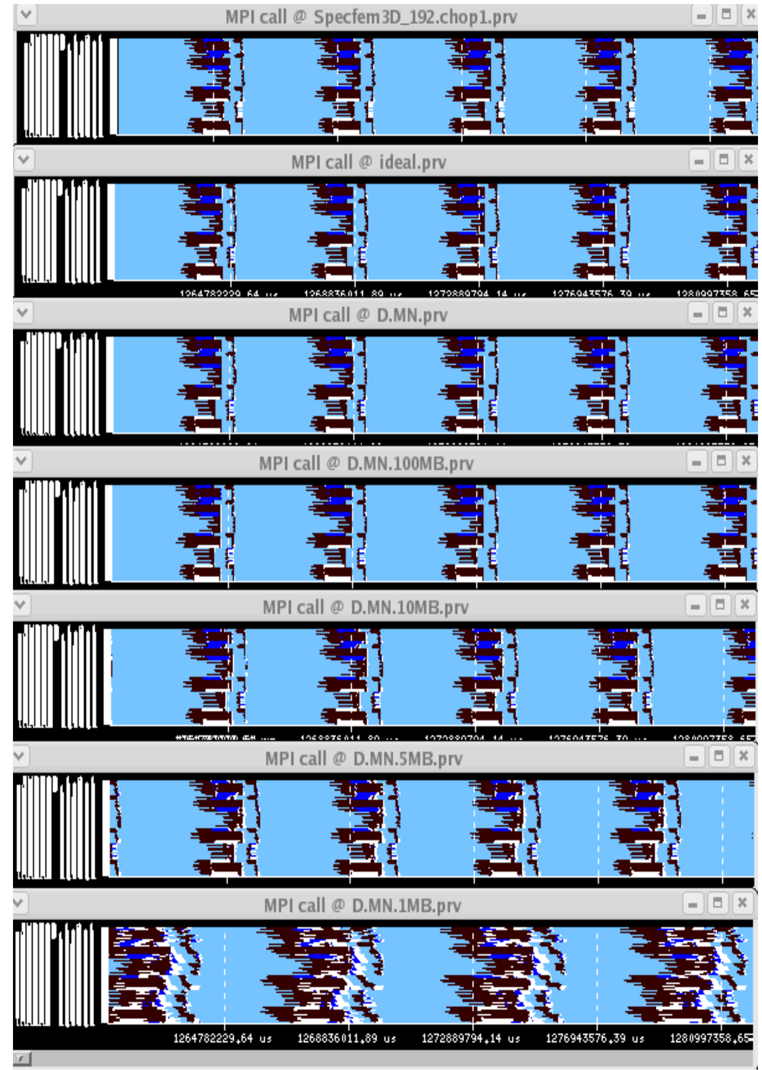
Prediction MN

Prediction 100MB/s

Prediction 10MB/s

Prediction 5MB/s

Prediction 1MB/s



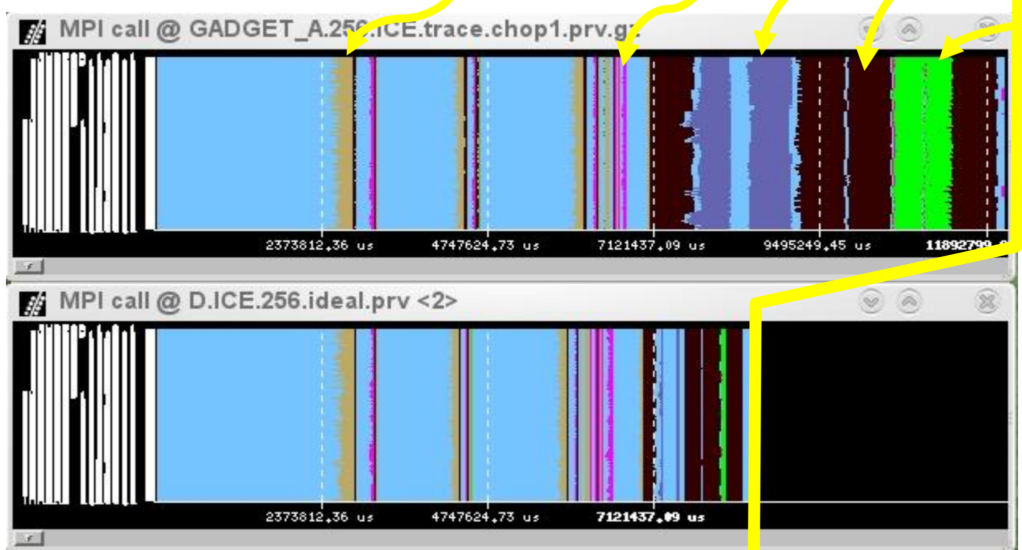
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Ideal machine

The impossible machine: BW = ∞, L = 0

- Actually describes/characterizes Intrinsic application behavior
  - Load balance problems?
  - Dependence problems?

GADGET @ Nehalem cluster
256 processes



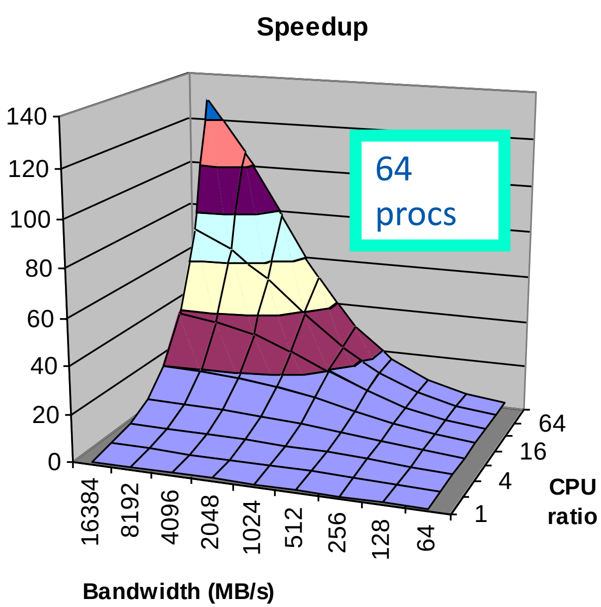
Impact on practical machines?

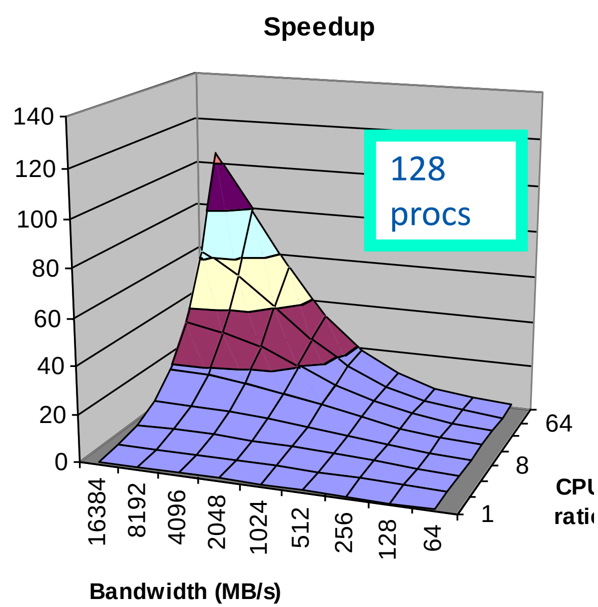
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

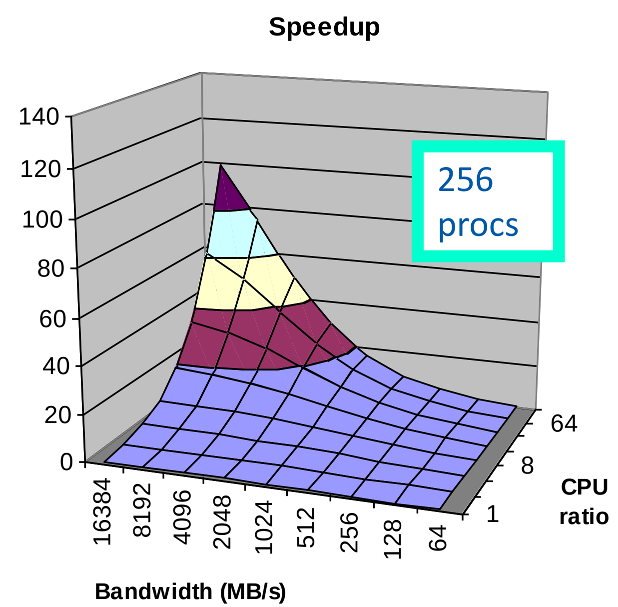
Impact of architectural parameters

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

GADGET

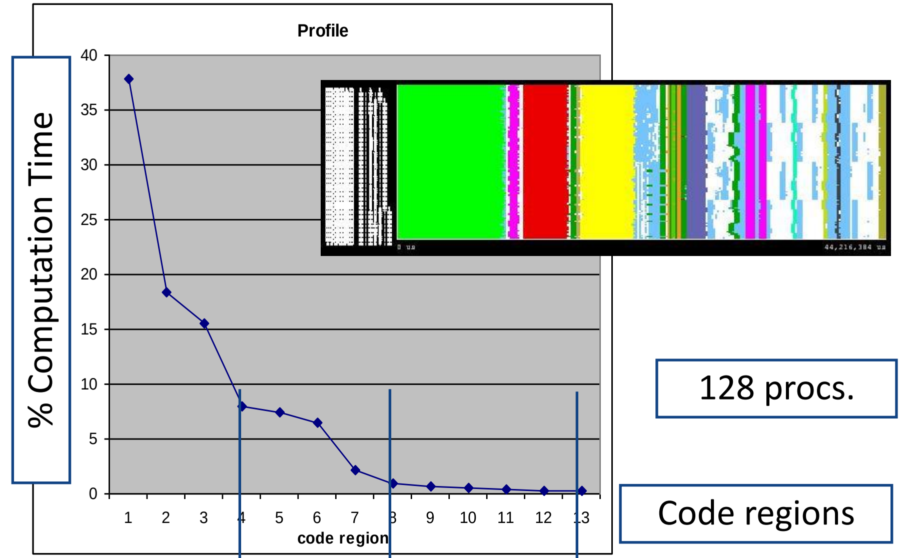


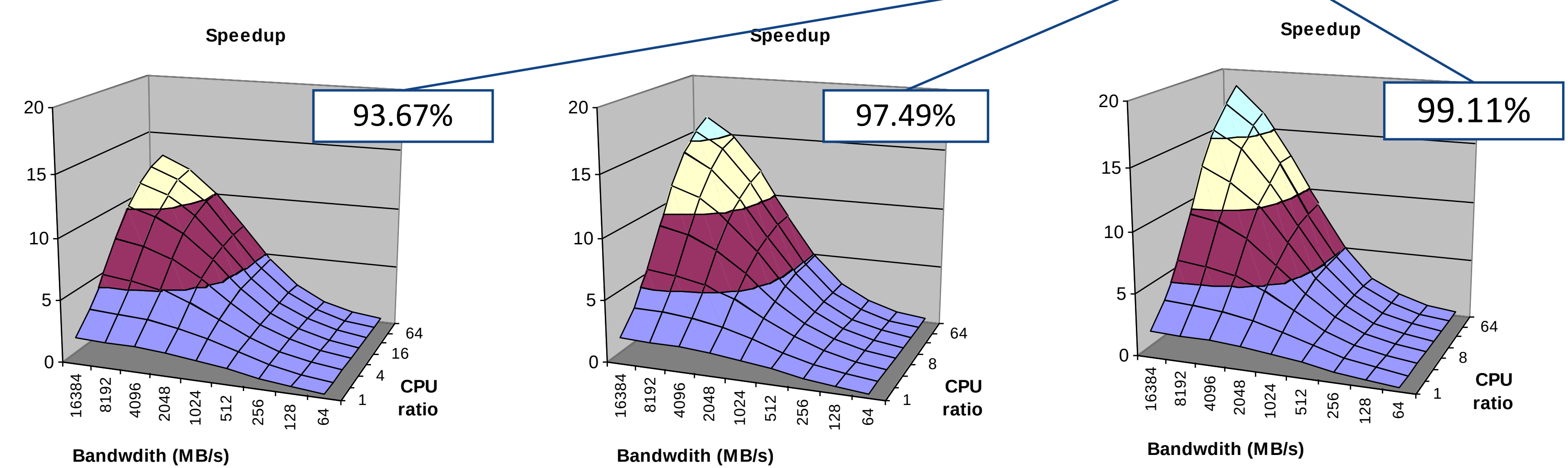




Hybrid parallelization

- Hybrid/accelerator parallelization
  - Speed-up SELECTED regions by the CPUratio factor





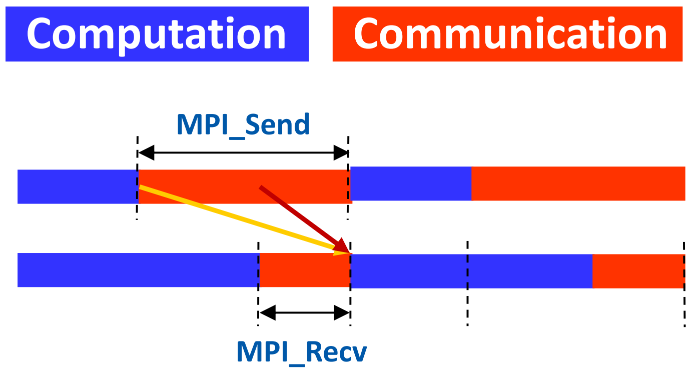
(Previous slide: speedups **up to 100x**)

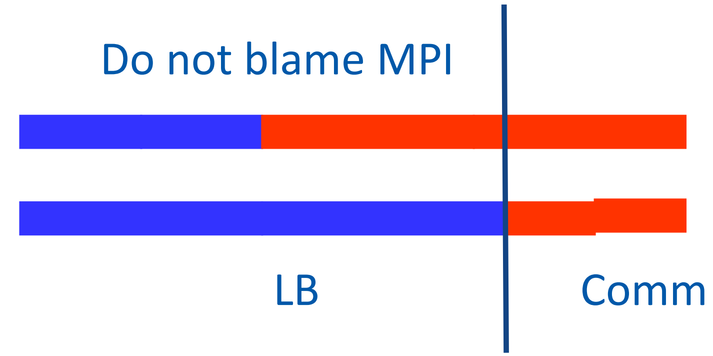
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Efficiency Models

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

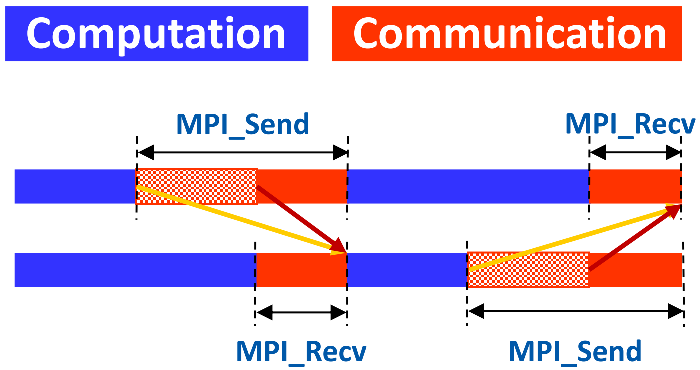
Parallel efficiency model

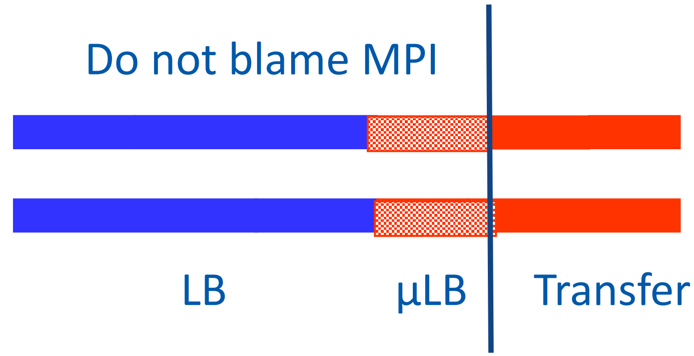




- Parallel efficiency = LB eff * Comm eff

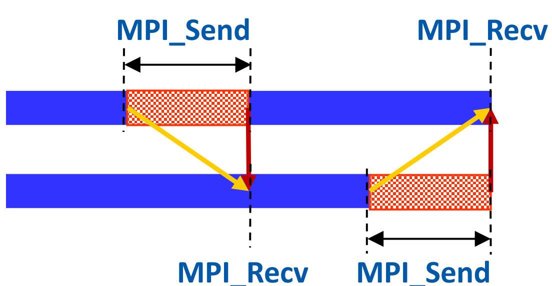
Parallel efficiency refinement: LB * µLB * Tr





$LB = 1$

- Serializations / dependences (µLB)
- Dimemas ideal network → Transfer (efficiency) = 1



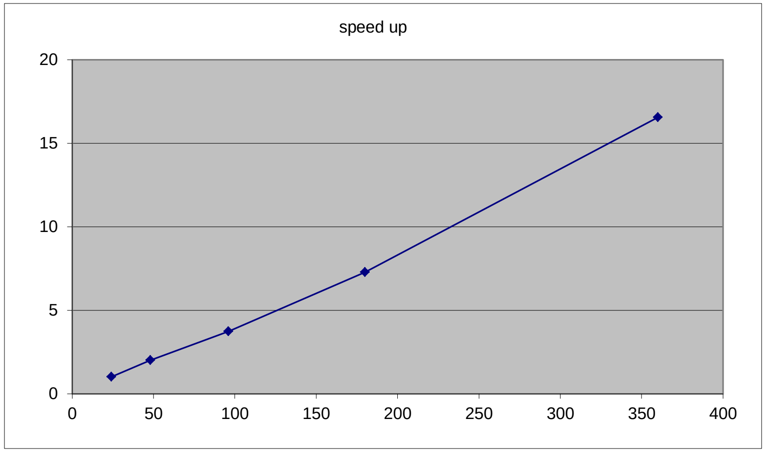
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

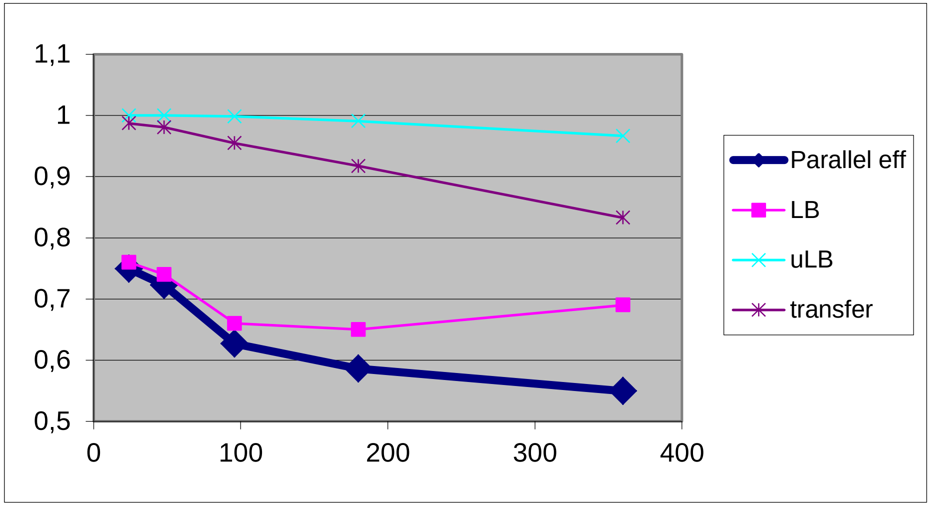
Why scaling?

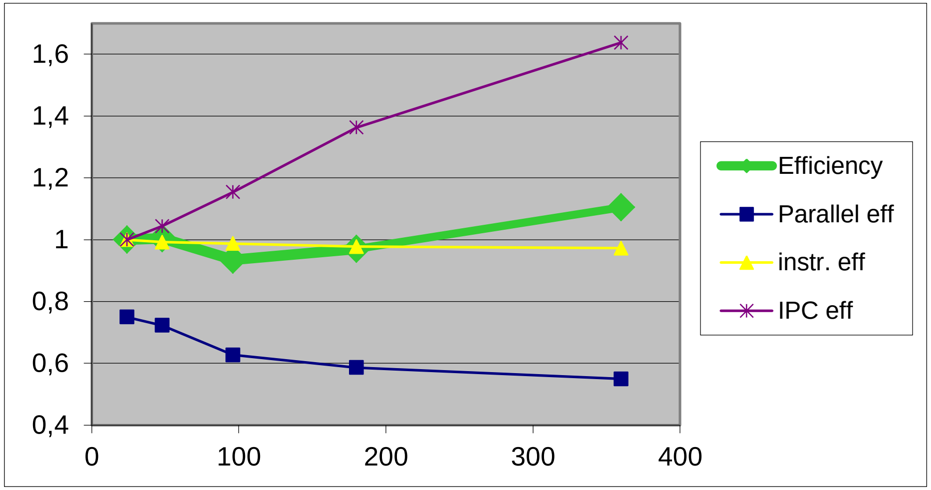
$$\eta_{\parallel} = LB * Ser * Trf$$

CG-POP mpi2s1D - 180x120

Good scalability !!
Should we be happy?







Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Some examples of efficiencies

Code	Parallel efficiency	Communication efficiency	Load Balance efficiency
Gromacs@mt	66.77	75.68	88.22
BigDFT@altamira	59.64	78.97	75.52
CG-POP@mt	80.98	98.92	81.86
ntchem_mini@pi	92.56	94.94	97.49
nicam@pi	87.10	75.97	89.22
cp2k@jureca	75.34	81.07	92.93
icon@mistral	79.86	84.02	95.05
k-Wave@salomon	89.08	92.84	95.96
fleur@claix	76.22	90.66	84.07

Same code, different behaviour

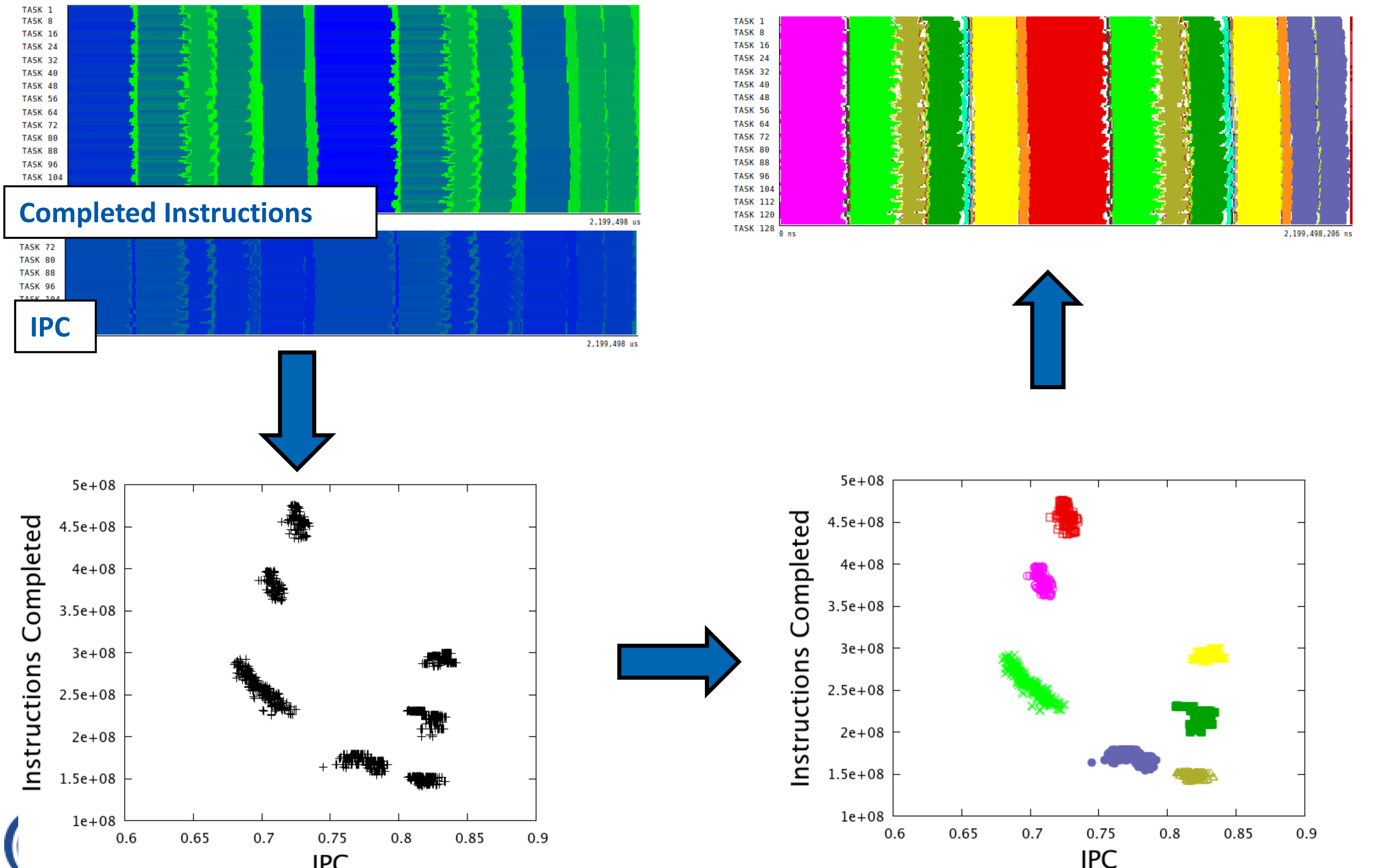
Code	Parallel efficiency	Communication efficiency	Load Balance efficiency
lulesh@mn3	90.55	**99.22**	91.26
lulesh@leftraru	**69.15**	99.12	**69.76**
lulesh@uv2 (mpt)	70.55	96.56	73.06
lulesh@uv2 (impi)	85.65	95.09	90.07
lulesh@mt	83.68	95.48	87.64
lulesh@cori	**90.92**	98.59	**92.20**
lulesh@thunderX	73.96	97.56	75.81
lulesh@jetson	75.48	**88.84**	84.06
lulesh@claix	77.28	92.33	83.70
lulesh@jureca	88.20	98.45	89.57
lulesh@mn4	86.59	98.77	87.67
lulesh@inti	88.16	98.65	89.36

Warning::: Higher parallel efficiency does not mean faster!

Analytics

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Using Clustering to identify structure



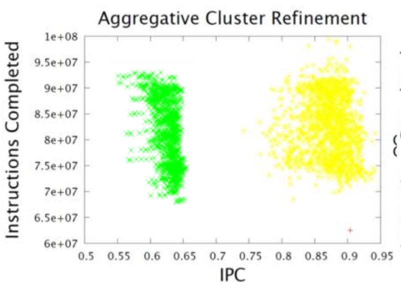
Automatic Detection of Parallel Applications Computation Phases (IPDPS 2009)

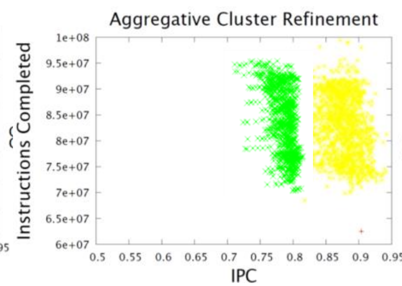
What should I improve?

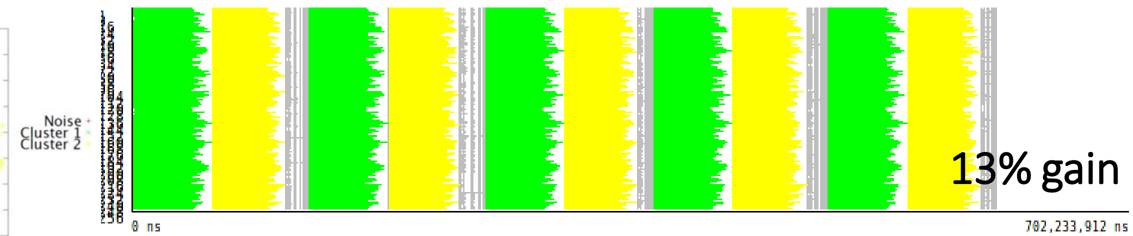
What if ….

PEPC

… we increase the IPC of Cluster1?

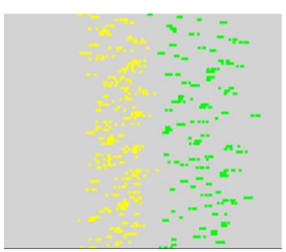


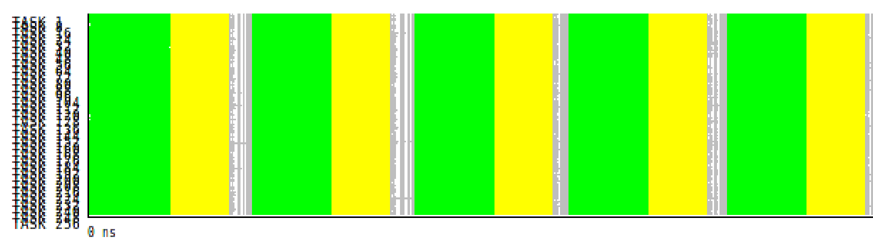




13% gain

… we balance Clusters 1 & 2?

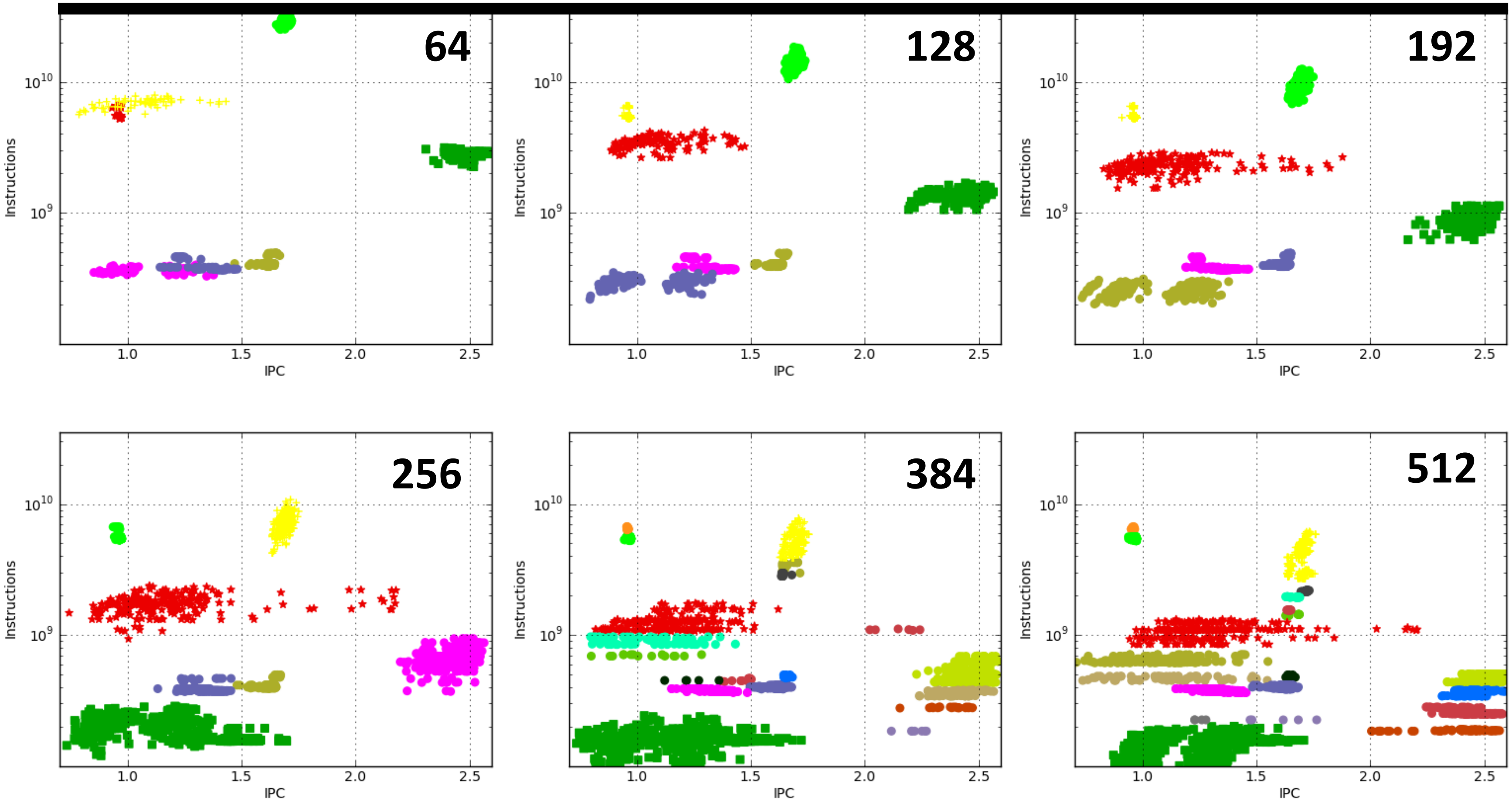




19% gain

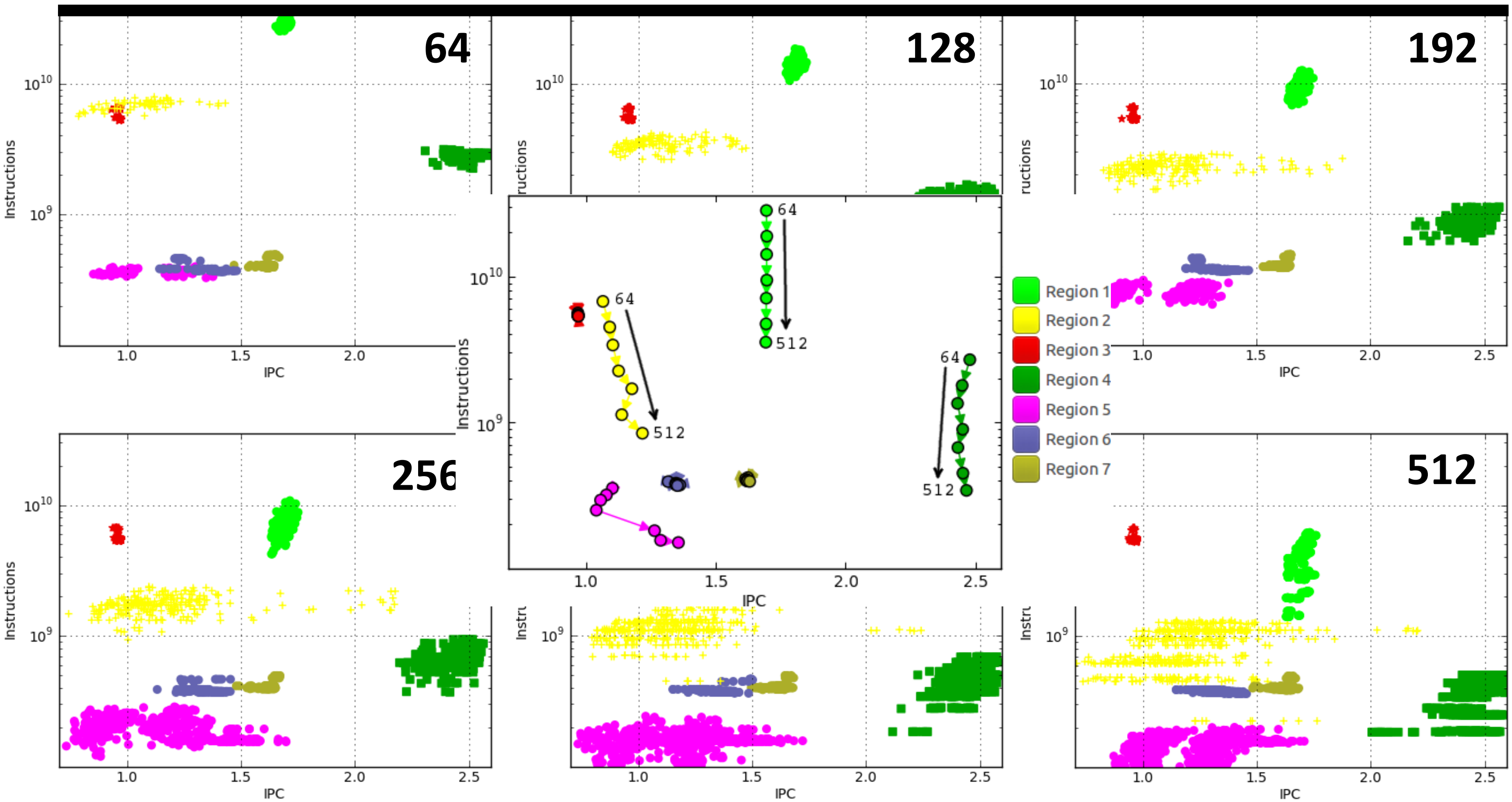
Tracking scability through clustering

- OpenMX (strong scale from 64 to 512 tasks)



Tracking scability through clustering

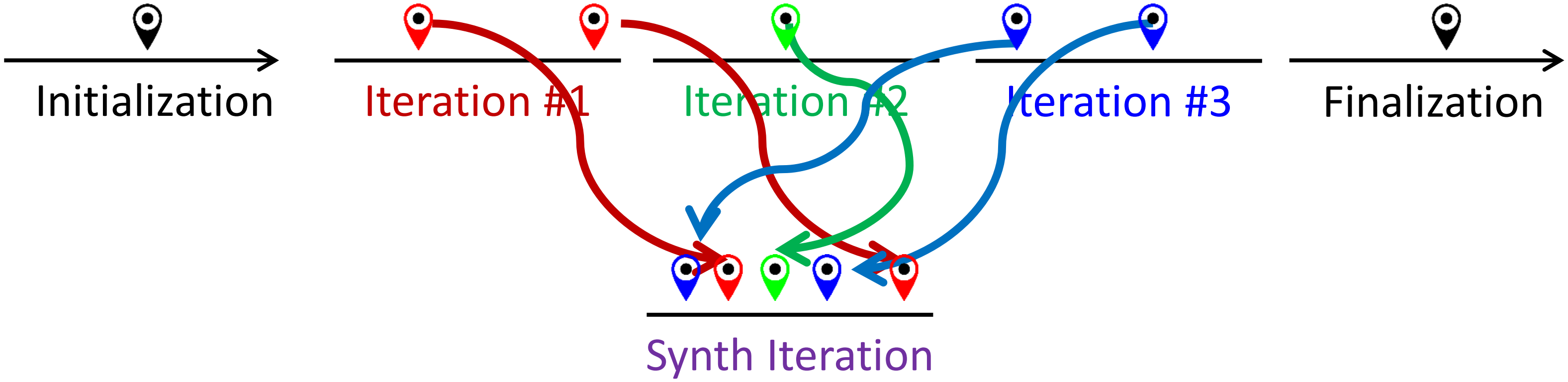
- OpenMX (strong scale from 64 to 512 tasks)



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Folding

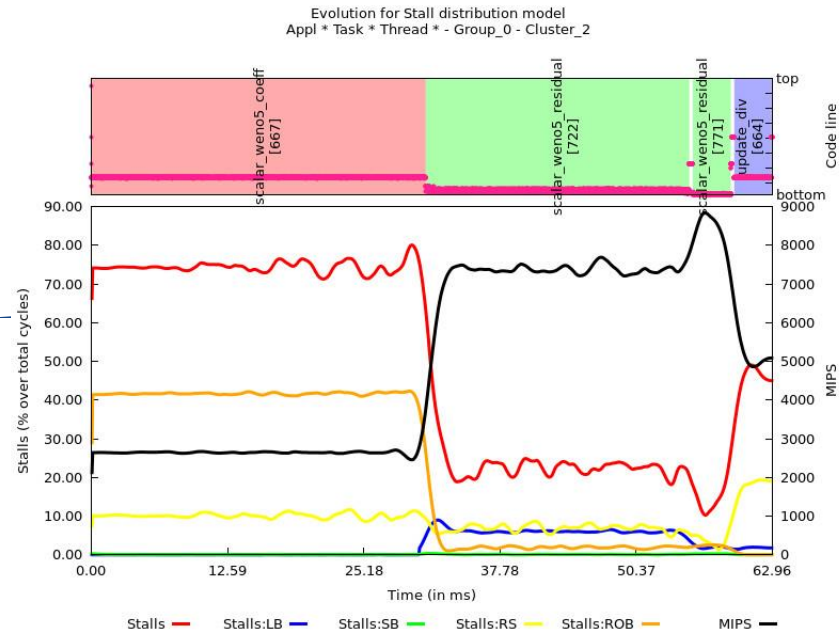
- Instantaneous metrics with minimum overhead
  - Combine instrumentation and sampling
    - Instrumentation delimits regions (routines, loops, …)
    - Sampling exposes progression within a region
  - Captures performance counters and call-stack references



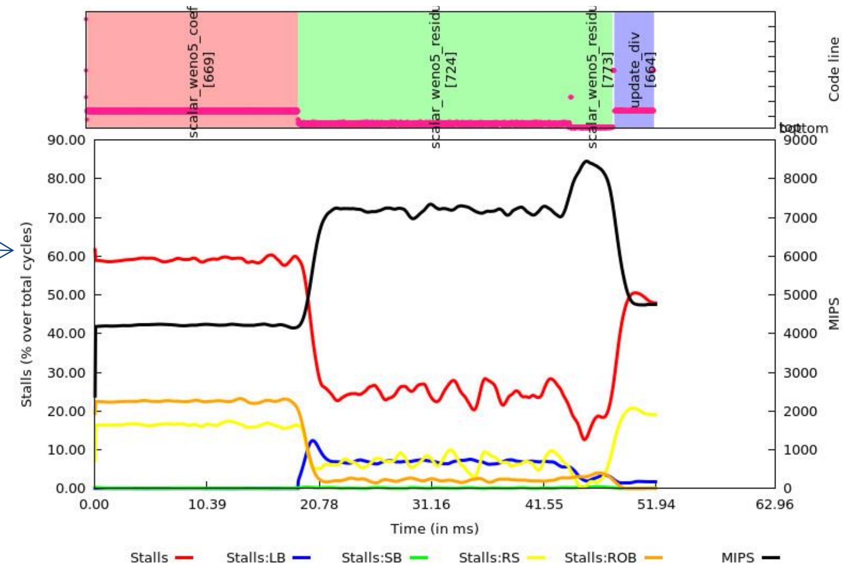
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

“Blind” optimization

- From folded samples of a few levels to timeline structure of “relevant” routines



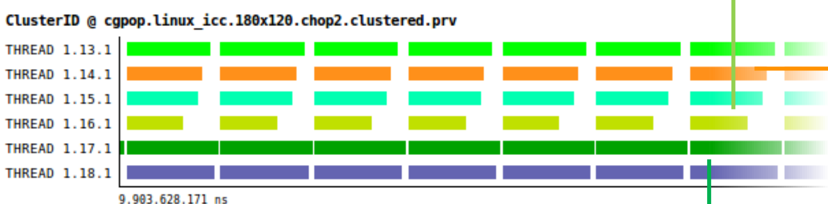
Recommendation without access to source code

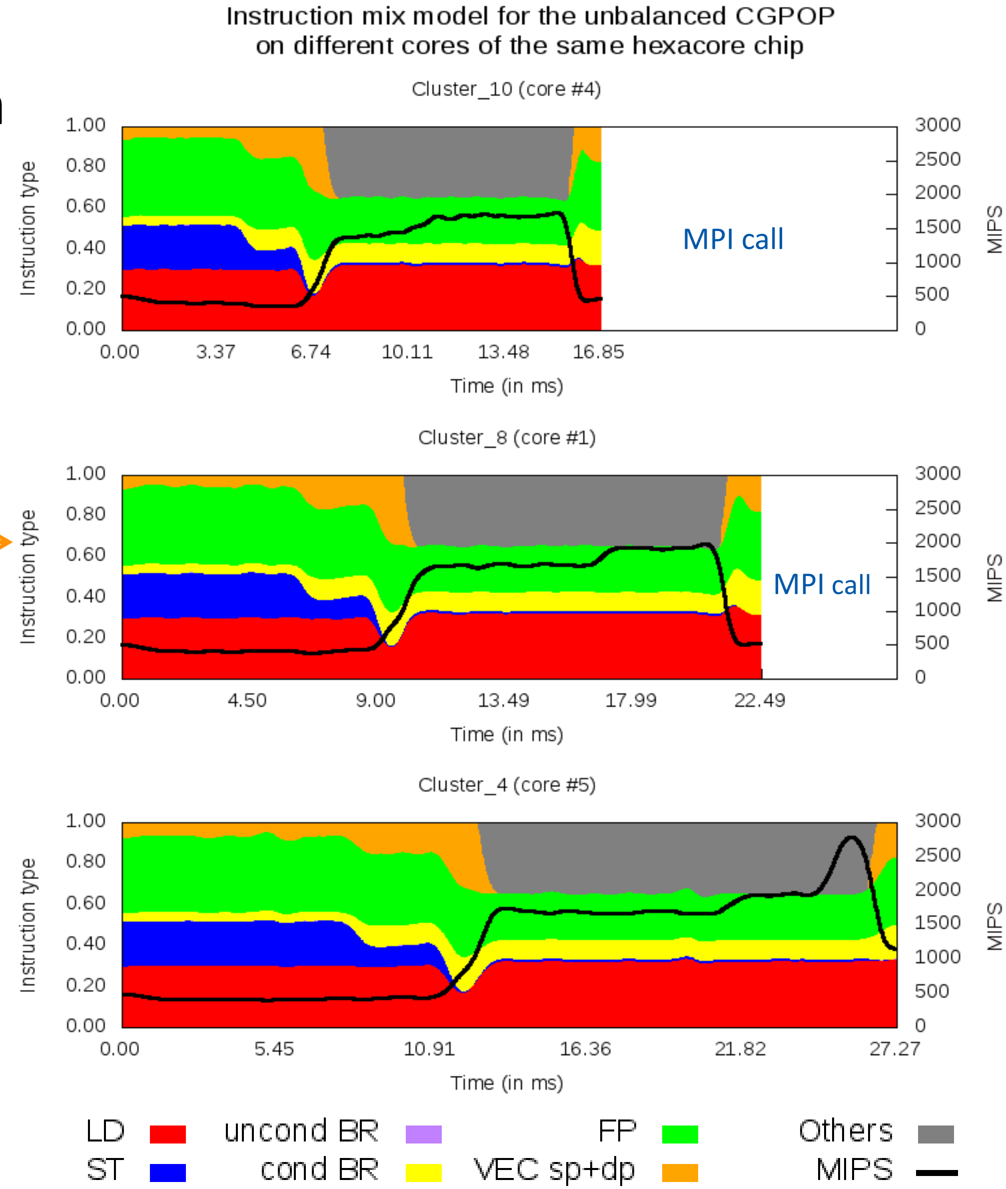


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

CG-POP multicore MN3 study

- Unbalanced MPI application
- Same code
- Different duration
- Different performance





Methodology

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Performance analysis tools objective

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?

BSC Tools web site

- tools.bsc.es
  - 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