www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Performance Analytics
From visualization to insight

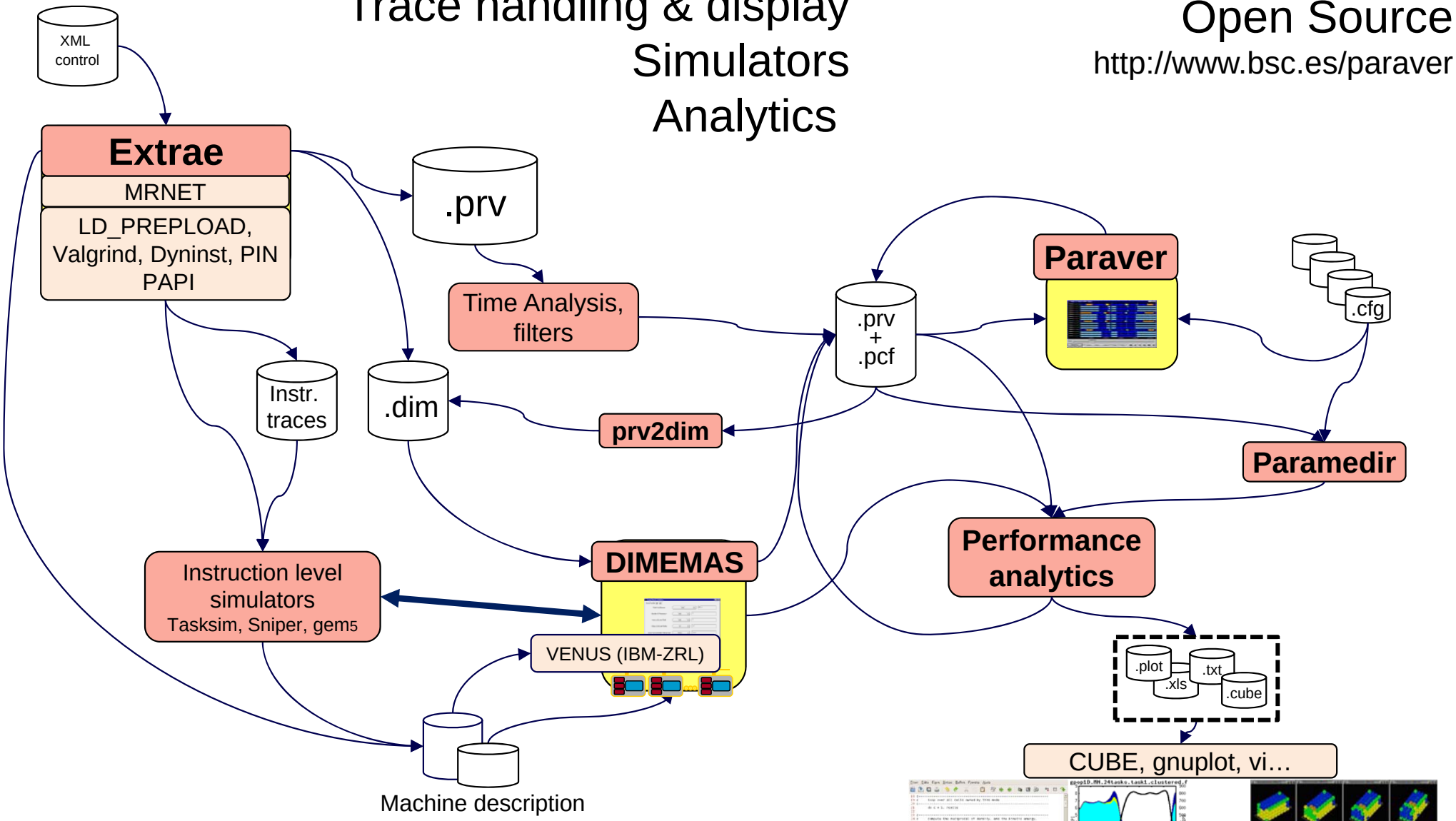
Jesús Labarta
BSC

VI-HPC 10th Anniversary Workshop
Frankfurt, June 23rd 2017

BSC Tools framework

Trace handling & display
Simulators
Analytics

Open Source
http://www.bsc.es/paraver



The importance of detail and flexibility
Since ~ 1991

Performance Analytics

- Performance analysis: THE big data app
  - 10000 cores x 1 event/100us x 100 bytes/event x 1000s = 10 GB
  - Easily underestimated terms by orders of magnitude

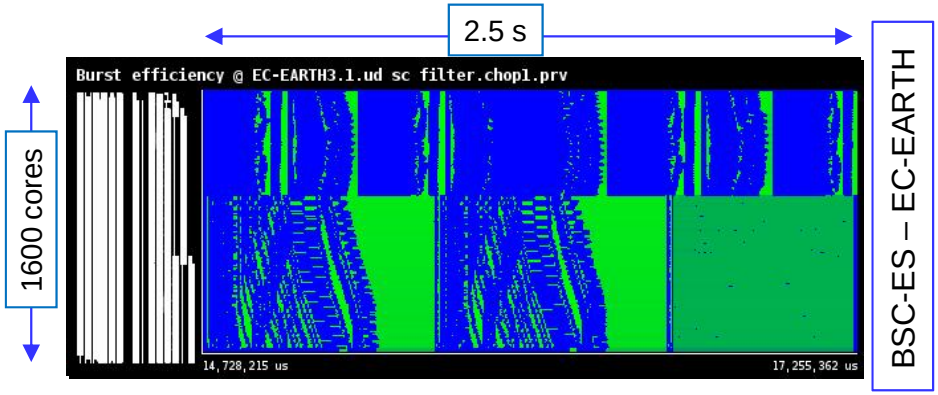
- Performance Analytics:
  - Squeezing the information in the captured data → Insight
    - About Machine Learning … and beyond, and before
  - Some BSC activities in the last 10 year (~)

- Vision
  - Have to leverage data processing methods from ALL areas
  - Have to balance between first principles, pure statistical, black box

BSC Performance Analytics

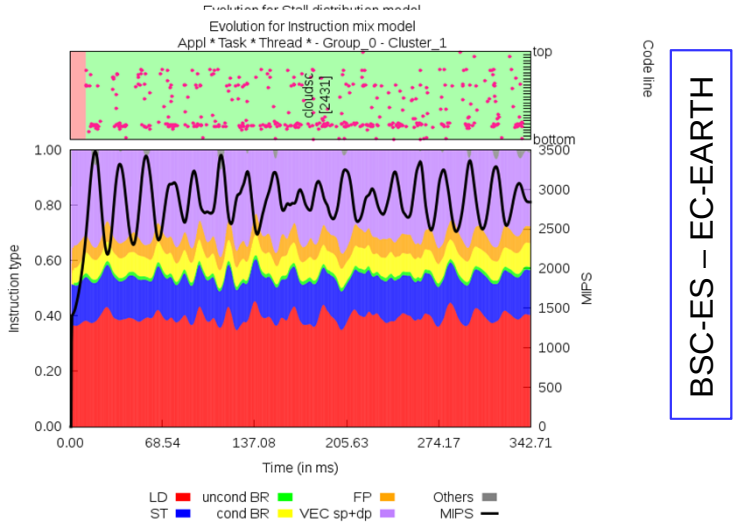
Flexible trace visualization and analysis

Adaptive burst mode tracing

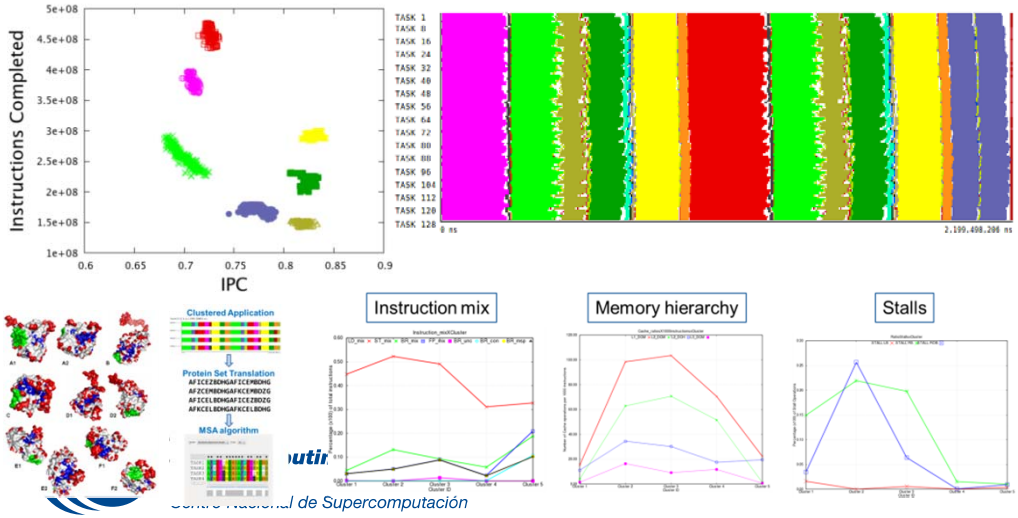


26.7MB trace
Eff: 0.43; LB: 0.52; Comm:0.81

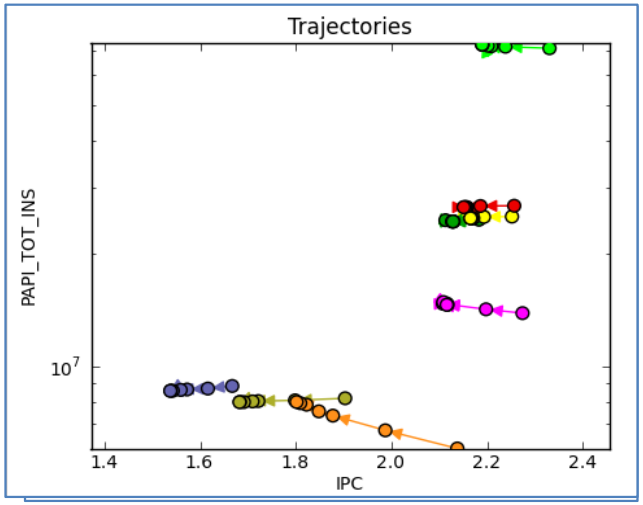
Instantaneous metrics for ALL hardware counters at “no” cost



Advanced clustering algorithms

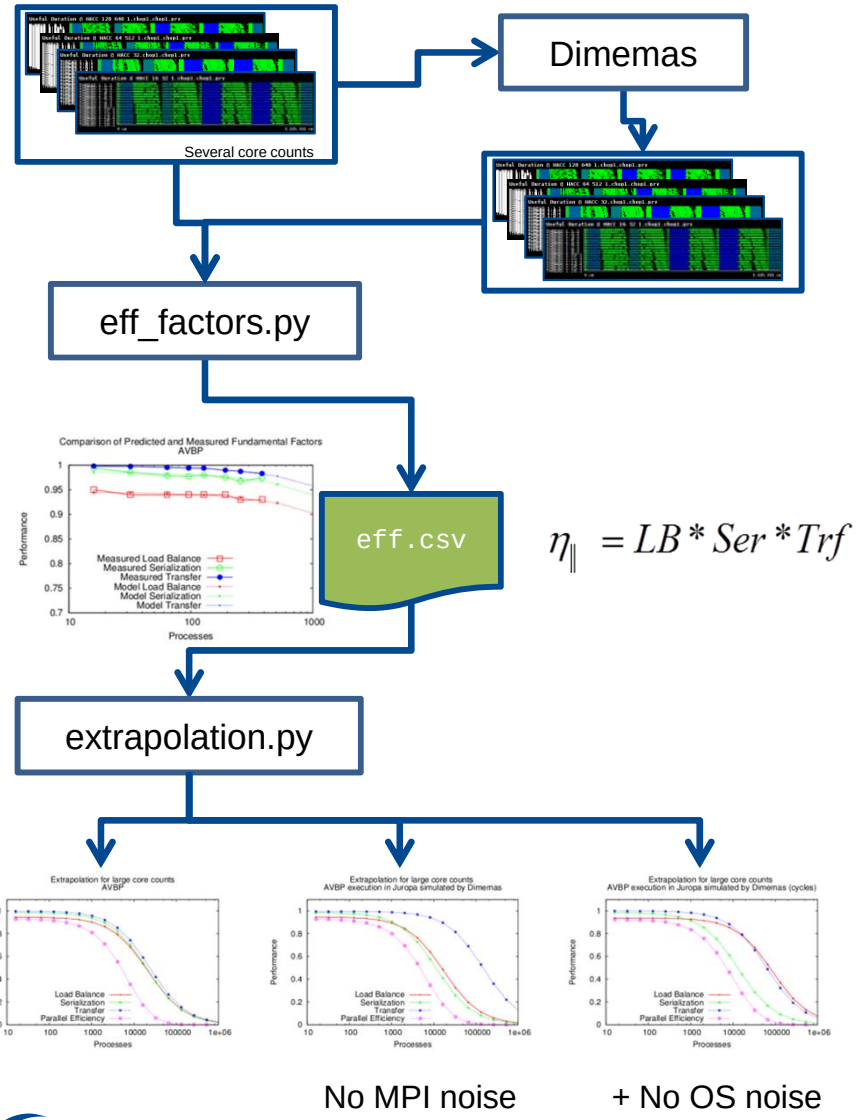


Tracking performance evolution



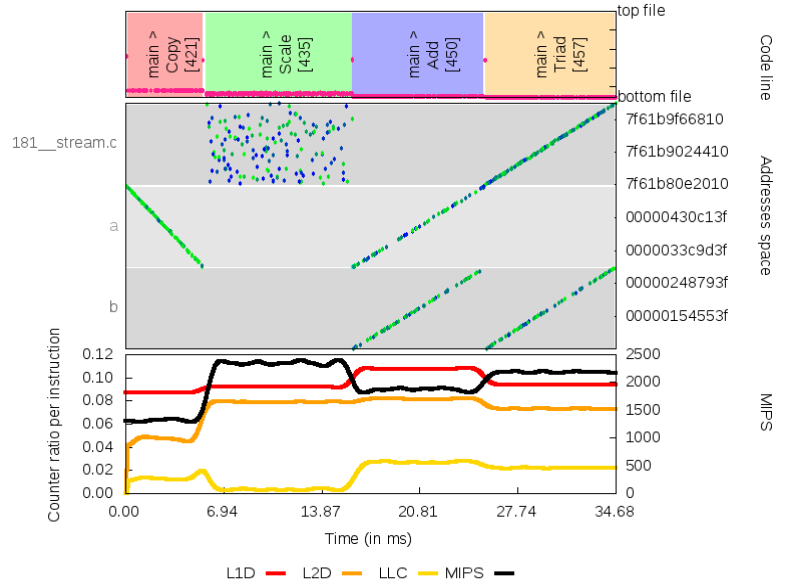
BSC Performance Analytics

Models and Projection



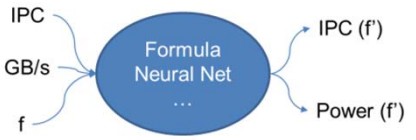
Intel –BSC Exascale Lab

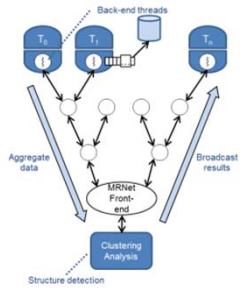
Data access patterns

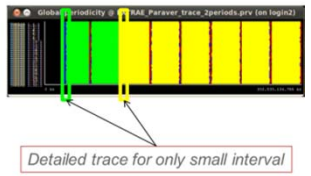


Structure detection

Runtime Energy and performance model (EAR)







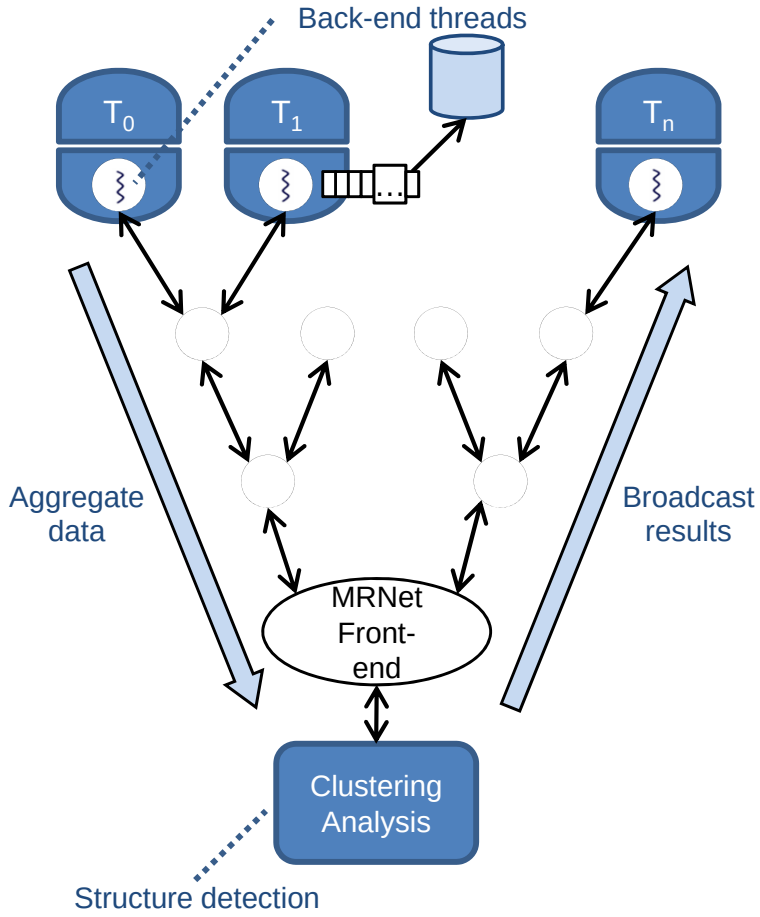
Lenovo - BSC

"Scalability prediction for fundamental performance factors " J. Labarta et al. SuperFRI 2014

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

About Analytics AND infrastructure

MRNET



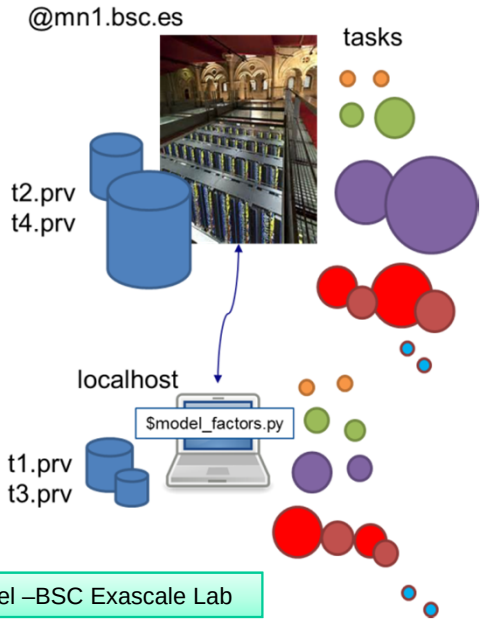
PyCOMPSs

Main

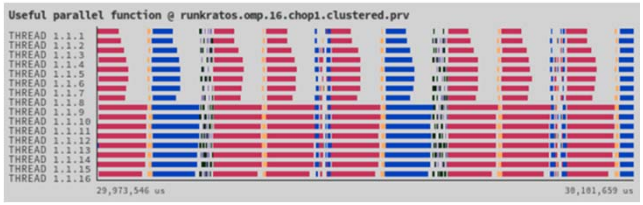
```
for name in list_traces:
    prv = name
    row = prv[:-4] + '.row'
    pcf = prv[:-4] + '.pcf'

    dim = os.path.basename(prv)[:-4] + '.dim'
    prv_ideal = 'D.' + os.path.basename(prv)
    pcf_ideal = 'D.' + os.path.basename(pcf)
    row_ideal = 'D.' + os.path.basename(row)

    np = get_num_prcs_trace(prv)

    prv2dim(prv, pcf, row, dim)

    dimemas(pcf, row, dim, prv_ideal, pcf_ideal, row_ideal,
            head_file, tail_file, coll4dim, dim_ideal_sim)

    ulb = paramedir(prv_ideal, pcf_ideal, row_ideal,
            '2dp_mpi_stats.cfg')
    lb, trf = paramedir(prv, pcf, row, '2dp_mpi_stats.cfg')

    collect_per_trace.append([np, ulb, lb, trf])

from pycompss.api.api import compss_wait_on
    collect_per_trace = compss_wait_on(collect_per_trace)
```

Tasks

```
@task(trace_prv=FILE_IN, trace_pcf=FILE_IN,
trace_row=FILE_IN, cfg=FILE_IN, returns=list)
def paramedir (trace_prv, trace_pcf, trace_row, cfg,
               trace_id_name):
    //Body of function
```



What we use: POP

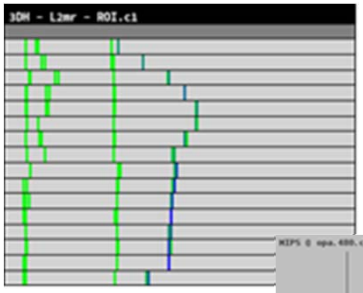


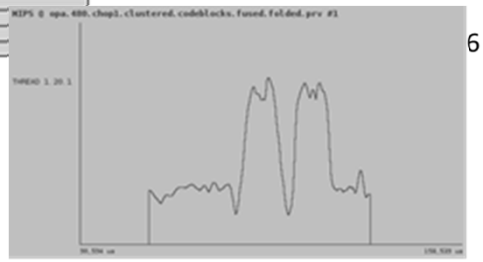
16

32

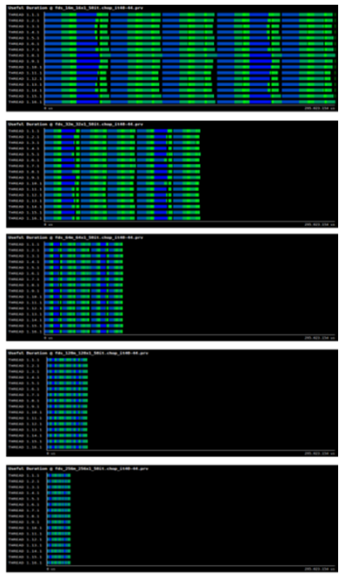
64

128



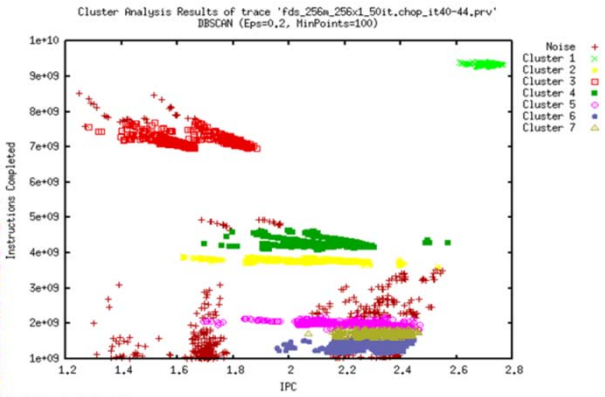


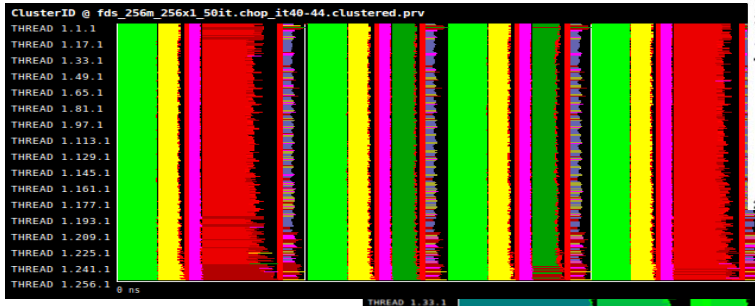
6

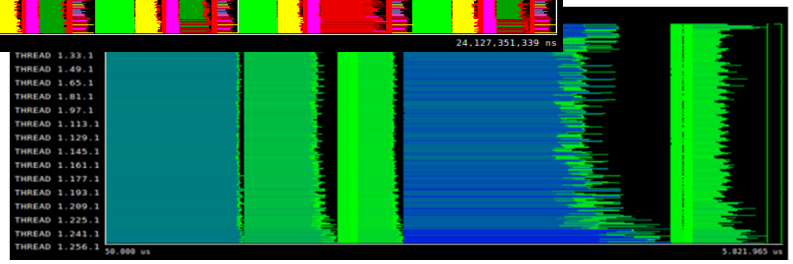


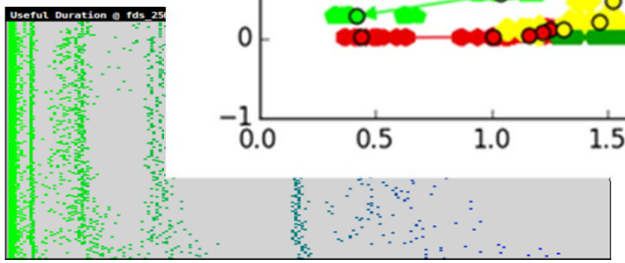
	2	4	8	16
Parallel Efficiency	0.9834	0.9436	0.8980	0.8478
Load Balance	0.9871	0.9687	0.9099	0.9177
Serialization efficiency	0.9975	0.9770	0.9938	0.9395
Transfer Efficiency	0.9988	0.9970	0.9931	0.9833
Computation Efficiency	1.000	0.9590	0.8680	0.6953
Global efficiency	0.9834	0.9049	0.7795	0.5894

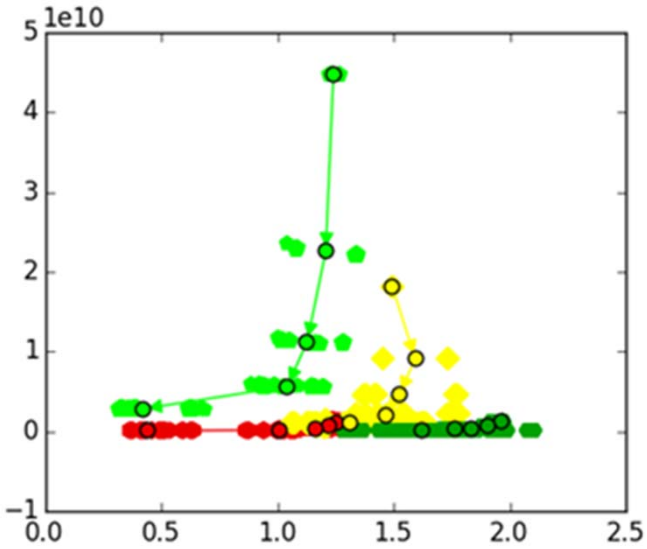
	2	4	8	16
IPC Scaling Efficiency	1.000	0.9932	0.9591	0.8421
Instruction Scaling Efficiency	1.000	0.9721	0.9393	0.9075
Core frequency efficiency	1.000	0.9932	0.9635	0.9098











Performance Analytics

- Leverage methods from ALL areas
- Balance between first principles, pure statistical, black box

  - "Proper" choice of feature vector
- Leverage big data infrastructure
- Lots of opportunities for the next 10 years !!!


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación