www.bsc.es

BSC

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

BSC Tools update

Using clustering and Folding

Judit Gimenez (judit@bsc.es)

CScADS – Snowbird, June 2012

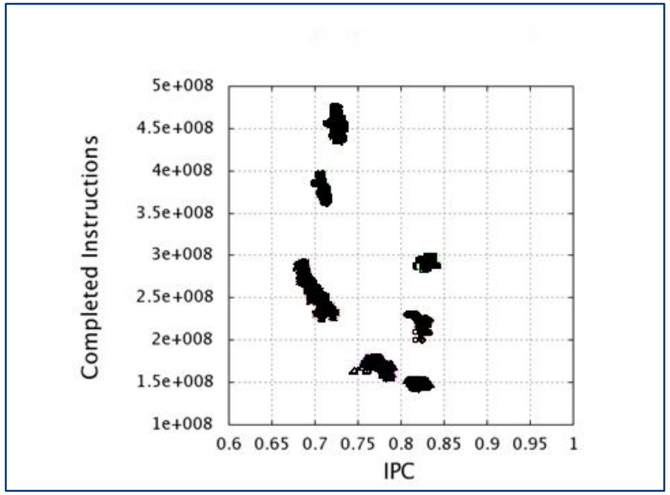
Outline

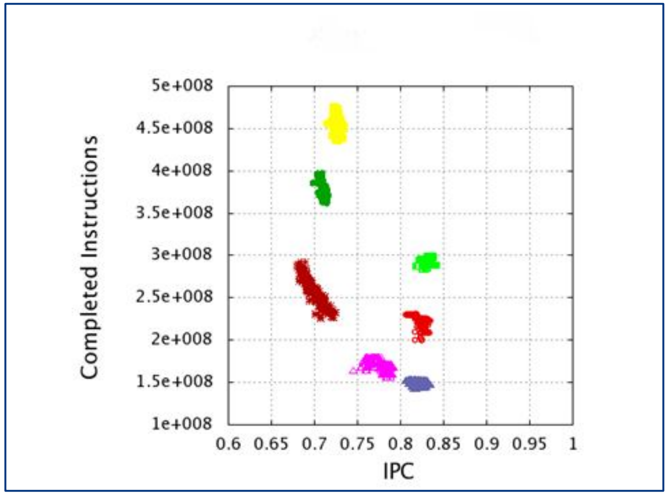
- Computation Structure detection
  - Short intro
  - Aggregative Refinement
  - Tracking program evolution
  - Scaling clustering algorithm

- Instantaneous performance metric
  - Clustering + Folding

Clustering

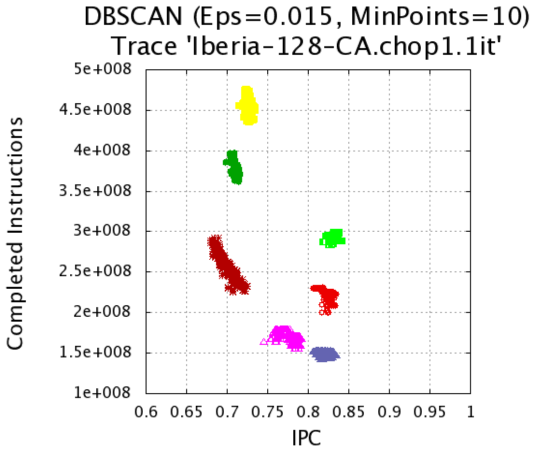
- Identification of computation structure
  - CPU burst = region between consecutive runtime calls
    - Described with performance hardware counters
    - Associated with call stack data
- Using DBSCAN density-cluster algorithm
  - Data not necessarily Gaussian



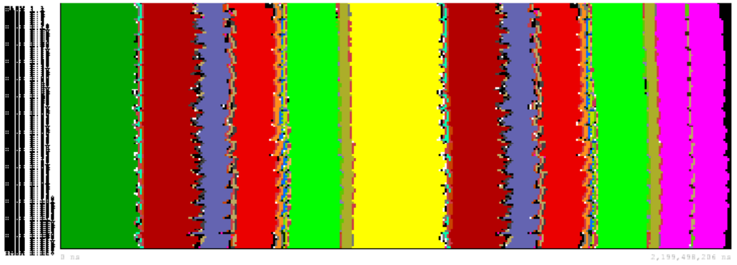


Outputs

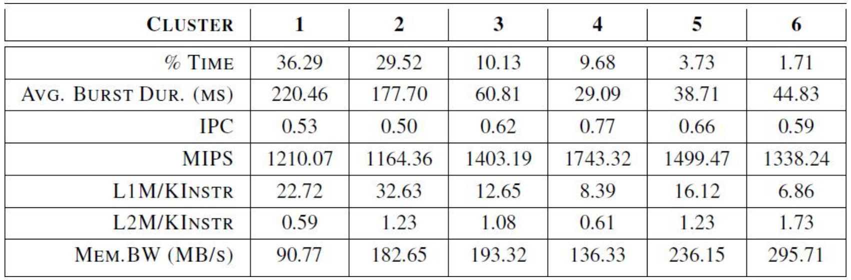
Scatter Plot of Clustering Metrics



Clusters Distribution Along Time

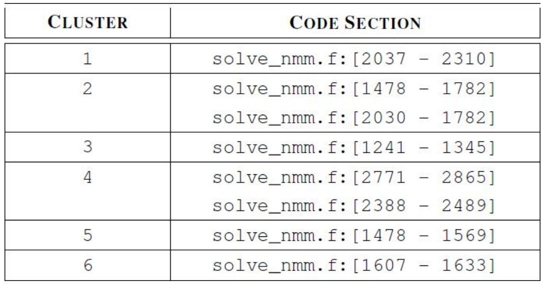


Cluster Statistics



Cluster	1	2	3	4	5	6
% Time	36.29	29.52	10.13	9.68	3.73	1.71
Avg. Burst Dur. (ms)	220.46	177.70	60.81	29.09	38.71	44.83
IPC	0.53	0.50	0.62	0.77	0.66	0.59
MIPS	1210.07	1164.36	1403.19	1743.32	1499.47	1338.24
L1M/KInstr	22.72	32.63	12.65	8.39	16.12	6.86
L2M/KInstr	0.59	1.23	1.08	0.61	1.23	1.73
Mem.BW (MB/s)	90.77	182.65	193.32	136.33	236.15	295.71

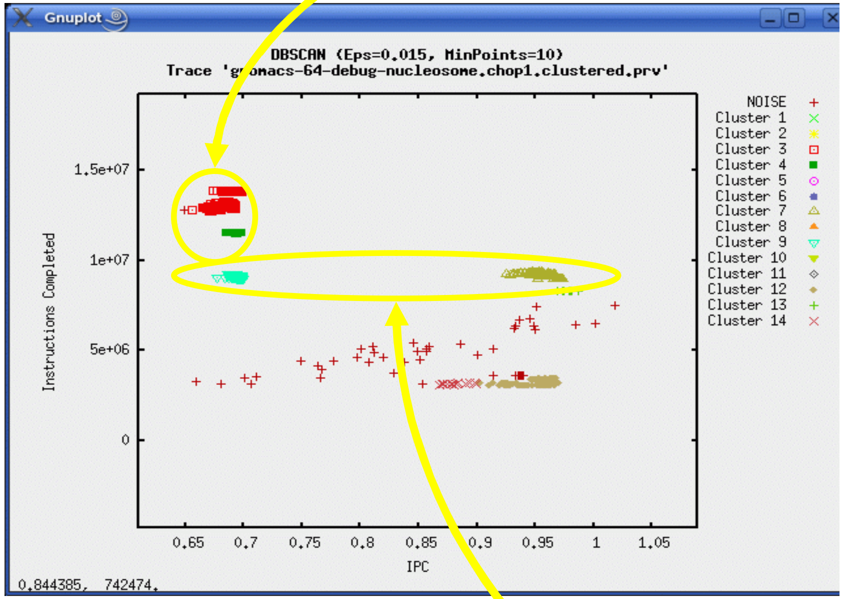
Code Linking



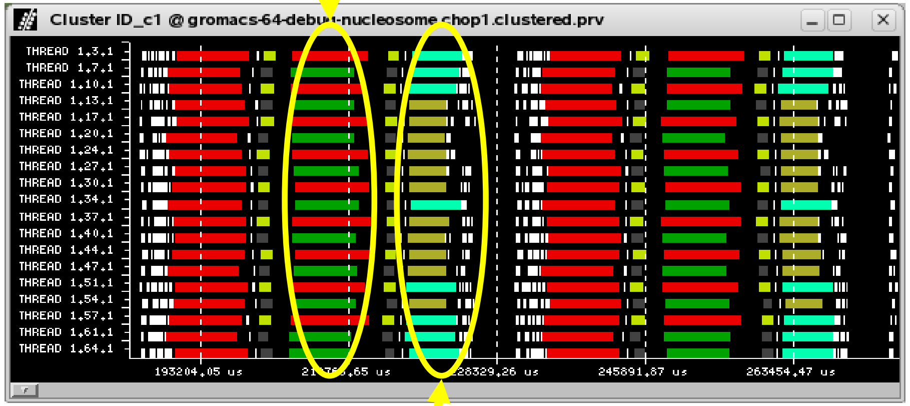
Cluster	Code Section
1	solve_nmm.f:[2037 – 2310]
2	solve_nmm.f:[1478 – 1782] solve_nmm.f:[2030 – 1782]
3	solve_nmm.f:[1241 – 1345]
4	solve_nmm.f:[2771 – 2865] solve_nmm.f:[2388 – 2489]
5	solve_nmm.f:[1478 – 1569]
6	solve_nmm.f:[1607 – 1633]

Using clusters to understand apps behavior (GROMACS)

Instructions imbalance



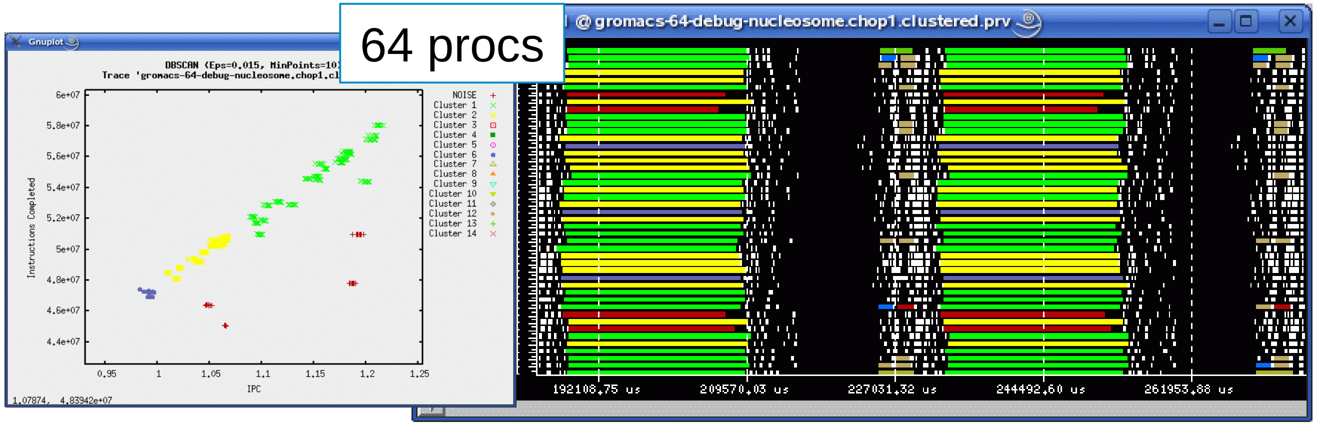
DBSCAN (Eps=0.015, MinPoints=10)
Trace 'gromacs-64-debug-nucleosome.chop1.clustered.prv'



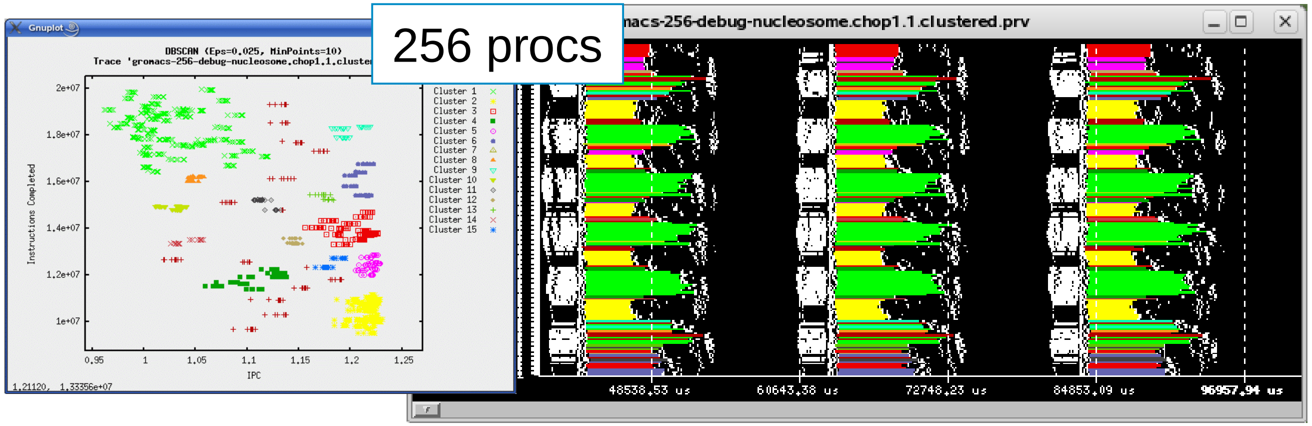
IPC Imbalance

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Using clusters to understand apps behavior (GROMACS)



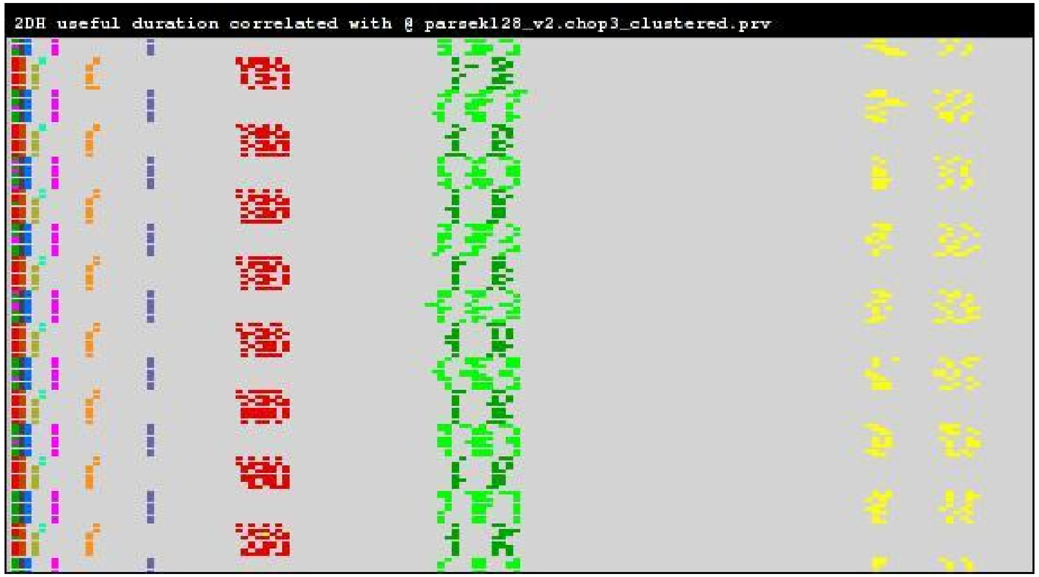
64 procs



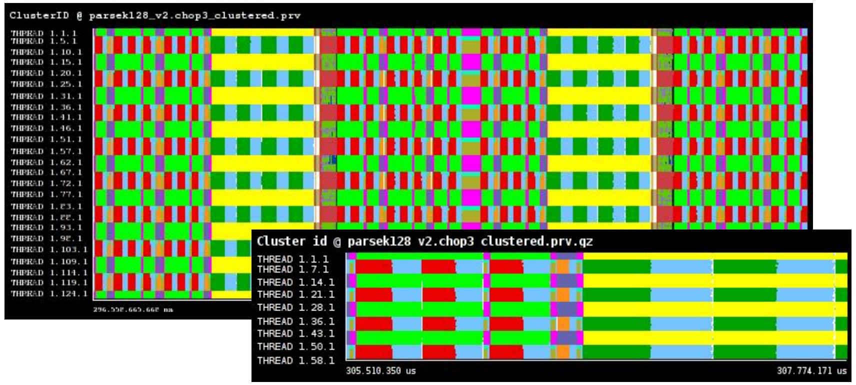
256 procs

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

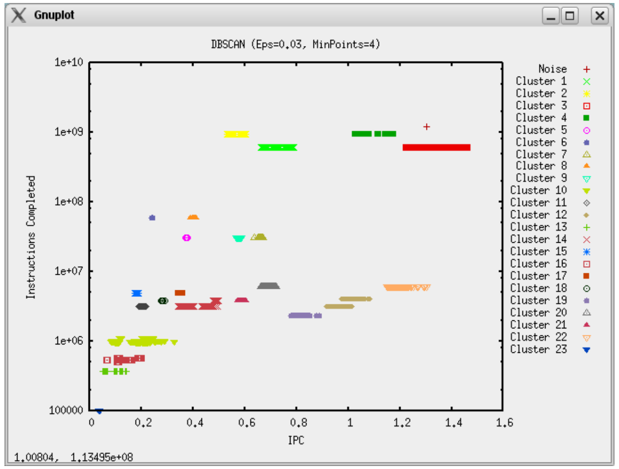
Identifying main code regions (PARSEK)

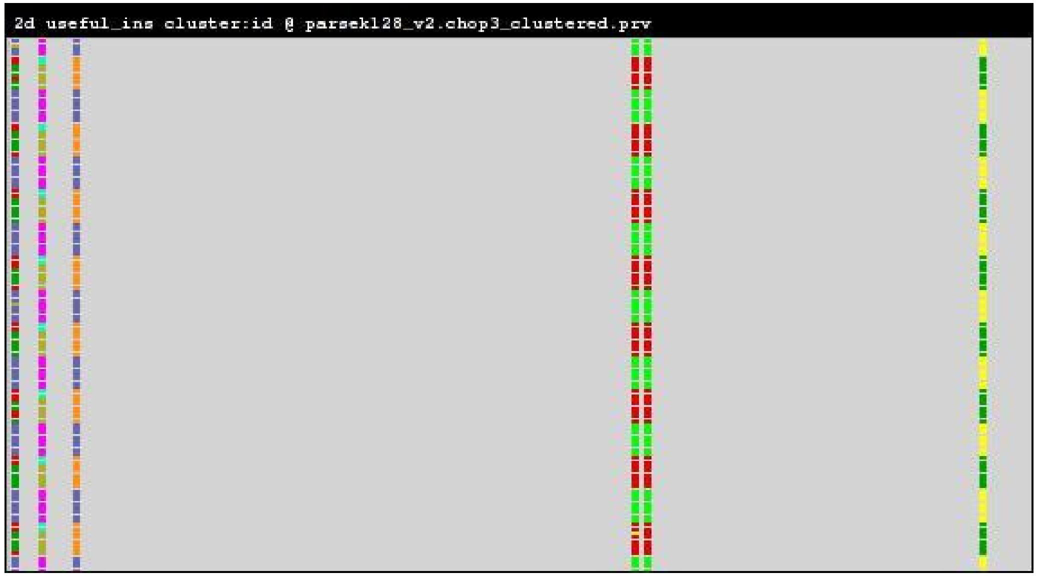

2DH useful duration correlated with @ parsek128_v2.chop3_clustered.prv

duration vs. cluster



instr. vs. cluster





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

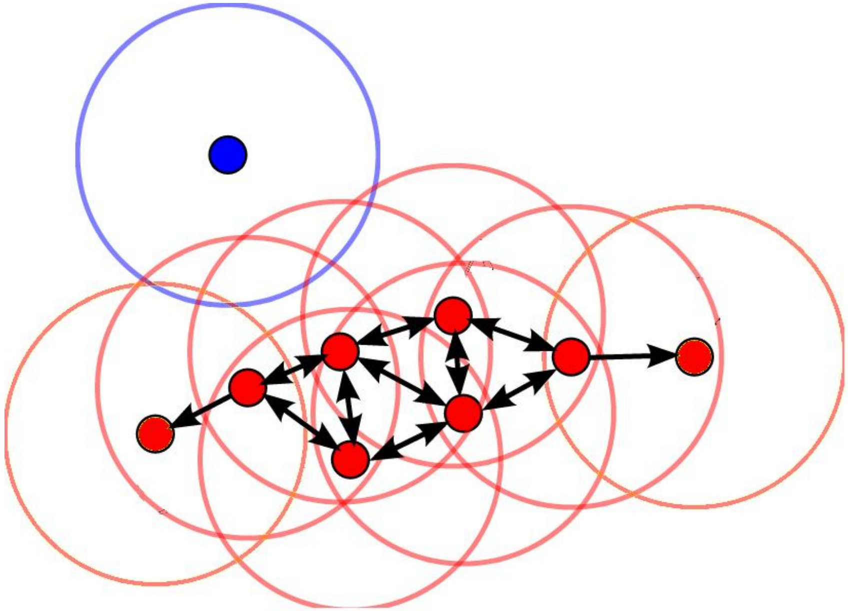
Outline

- Computation Structure detection
  - Short intro
  - **Aggregative Refinement**
  - Tracking program evolution
  - Scaling clustering algorithm

- Instantaneous performance metric
  - Clustering + Folding

DBSCAN characteristics

- Two parameters
  - Epsilon: search radius
  - MinPoints: minimum cluster density



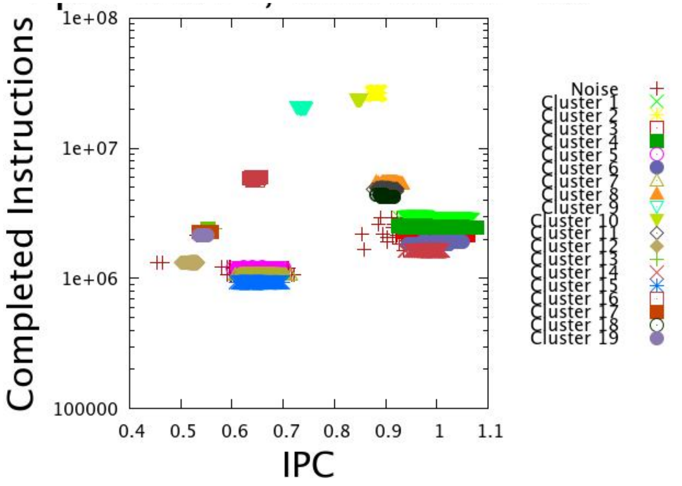
● Noise point

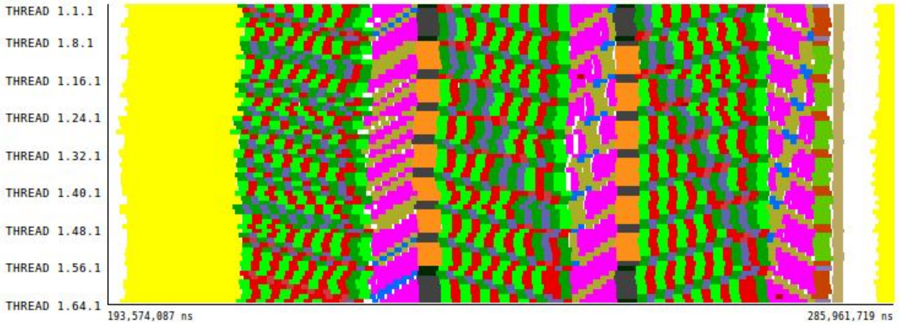
● Cluster points

DBSCAN *Eps* selection

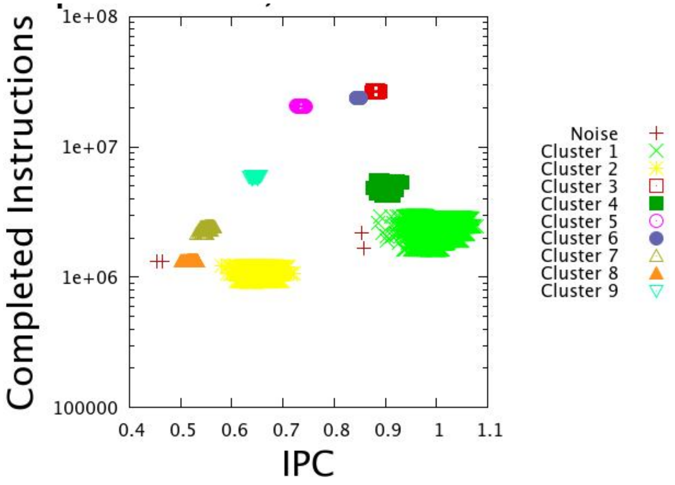
- Which results are better?

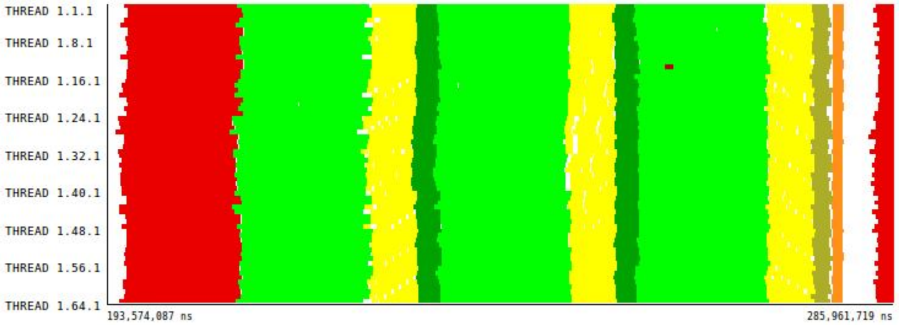
Eps=0.0140 (Low Value)





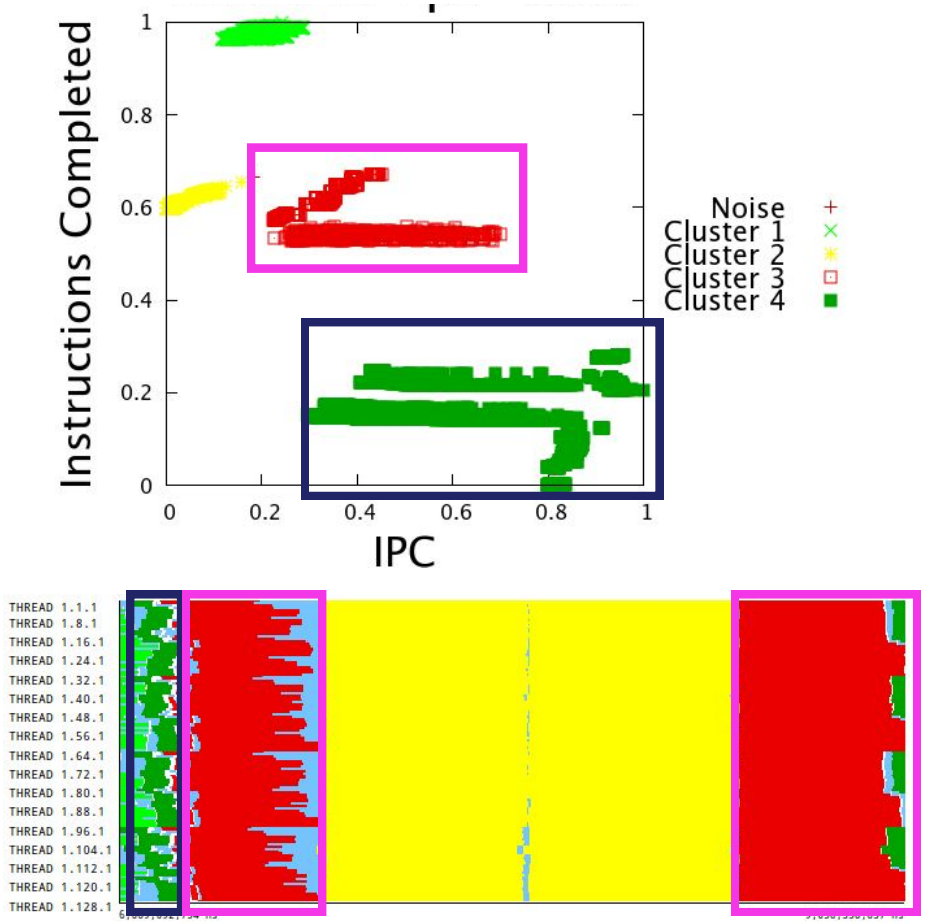
Eps=0.0400 (High Value)



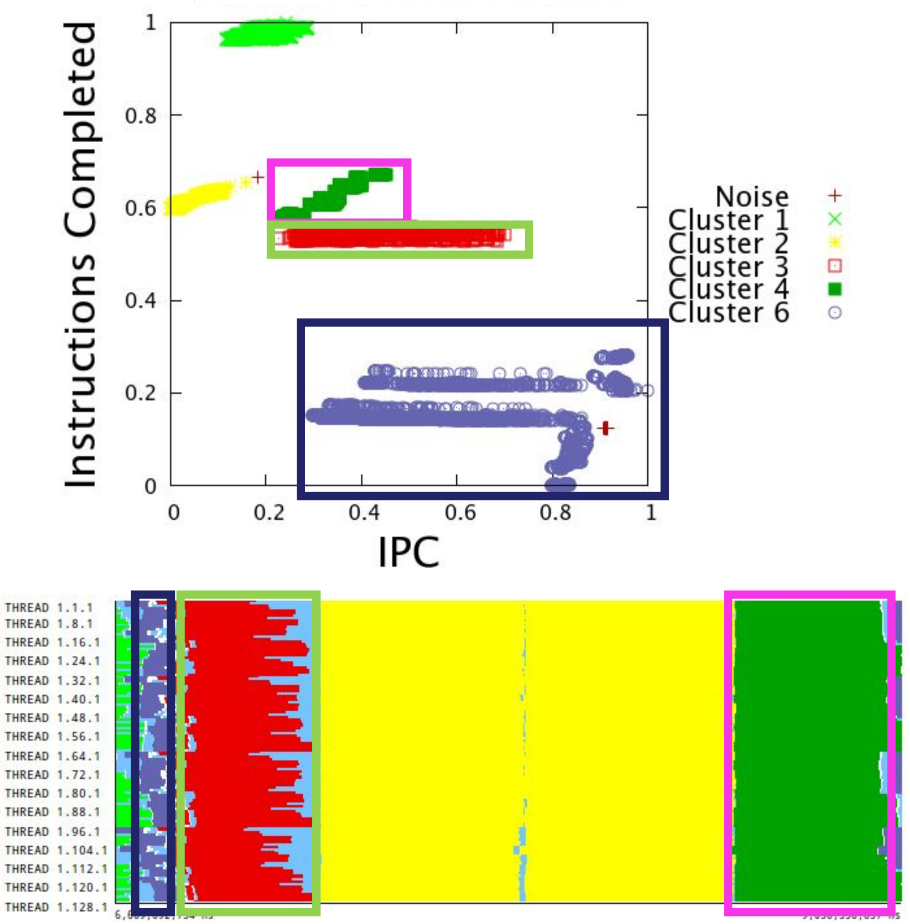


DBSCAN single *Eps* limitation

DBSCAN *Eps*=0.05

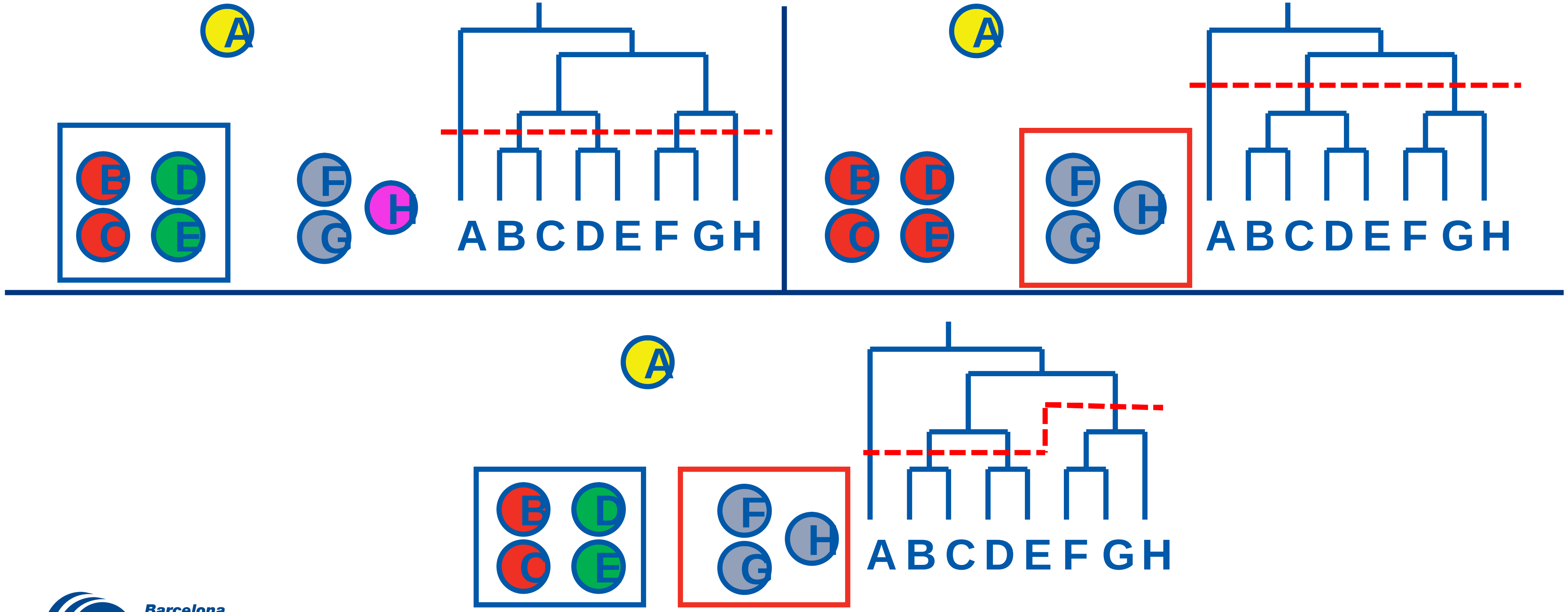


Desired results



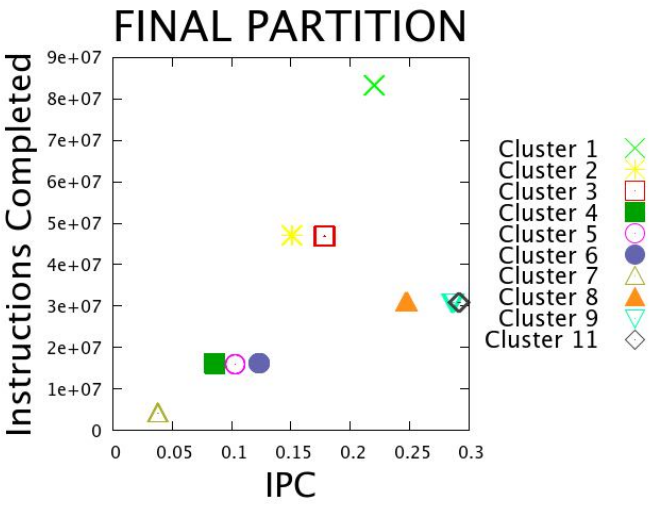
Refinement Algorithm Approach

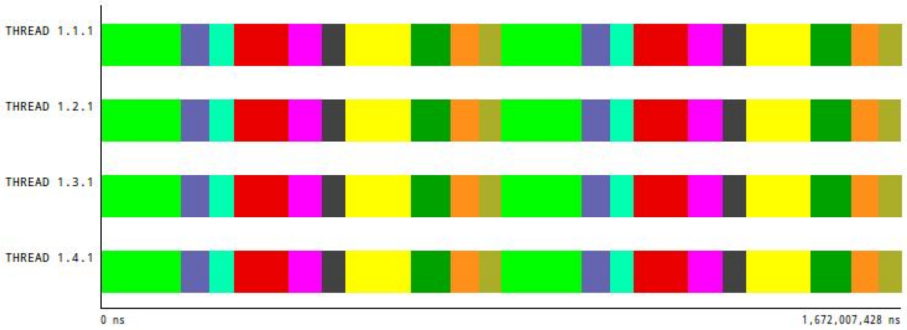
- Analogy between DBSCAN and hierarchical clustering
  - Iterative bottom up construction of a *pseudo-dendogram*
- Cluster Sequence Score as target
  - Similar to *X*-means approach to decide *K*-means *k* parameter

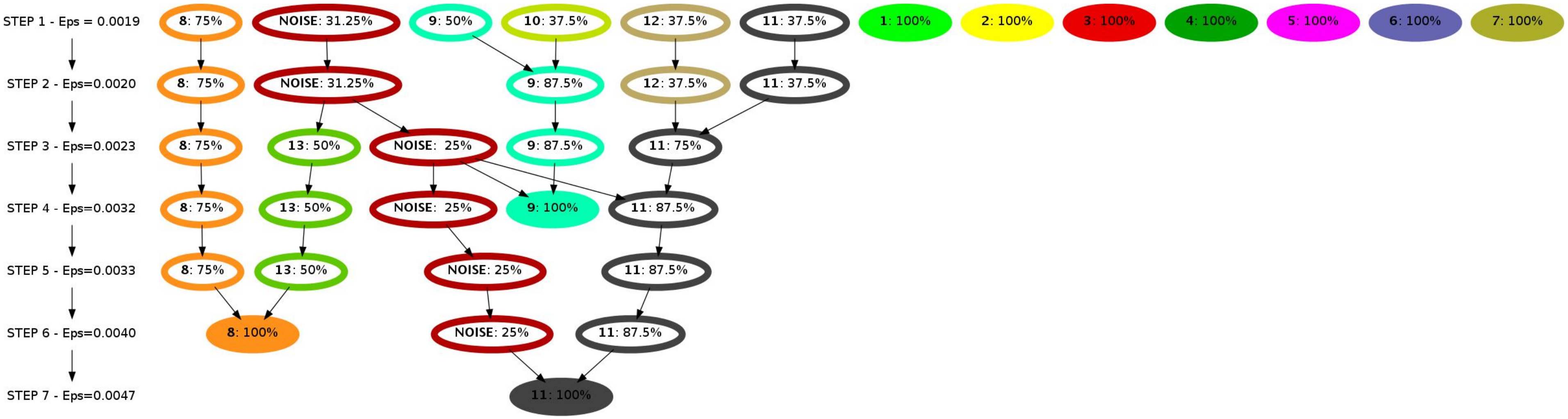


Automatic Refinement of Parallel Application Structure Detection (ICPADS 2011)

BT A 4 tasks







STEP 1 - Eps = 0.0019: 8: 75% | NOISE: 31.25% | 9: 50% | 10: 37.5% | 12: 37.5% | 11: 37.5% | 1: 100% | 2: 100% | 3: 100% | 4: 100% | 5: 100% | 6: 100% | 7: 100%

STEP 2 - Eps=0.0020: 8: 75% | NOISE: 31.25% | 9: 87.5% | 12: 37.5% | 11: 37.5%

STEP 3 - Eps=0.0023: 8: 75% | 13: 50% | NOISE: 25% | 9: 87.5% | 11: 75%

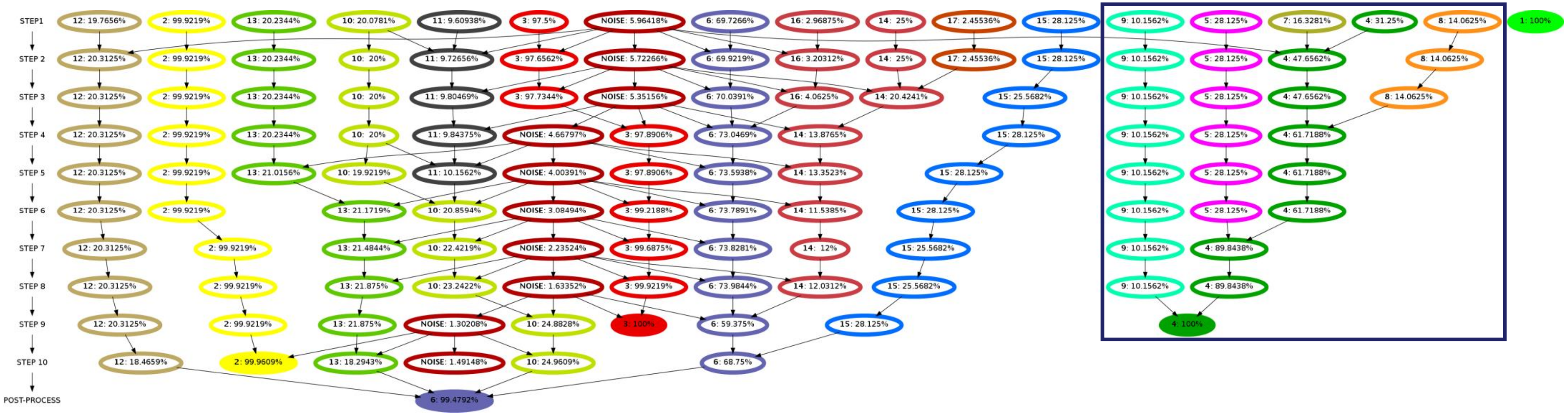
STEP 4 - Eps=0.0032: 8: 75% | 13: 50% | NOISE: 25% | 9: 100% | 11: 87.5%

STEP 5 - Eps=0.0033: 8: 75% | 13: 50% | NOISE: 25% | 11: 87.5%

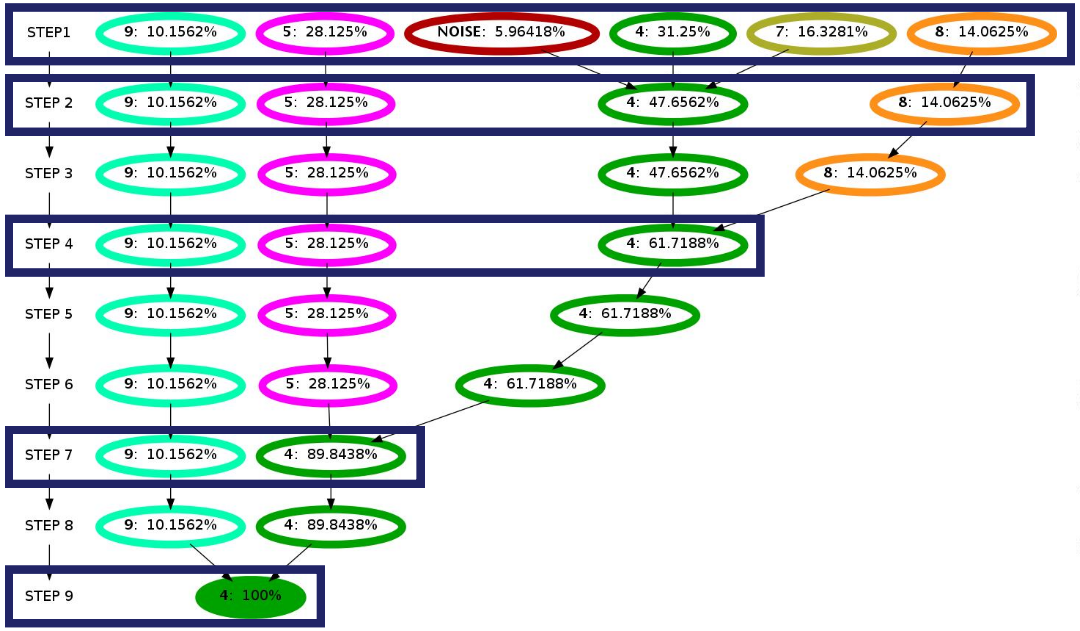
STEP 6 - Eps=0.0040: 8: 100% | NOISE: 25% | 11: 87.5%

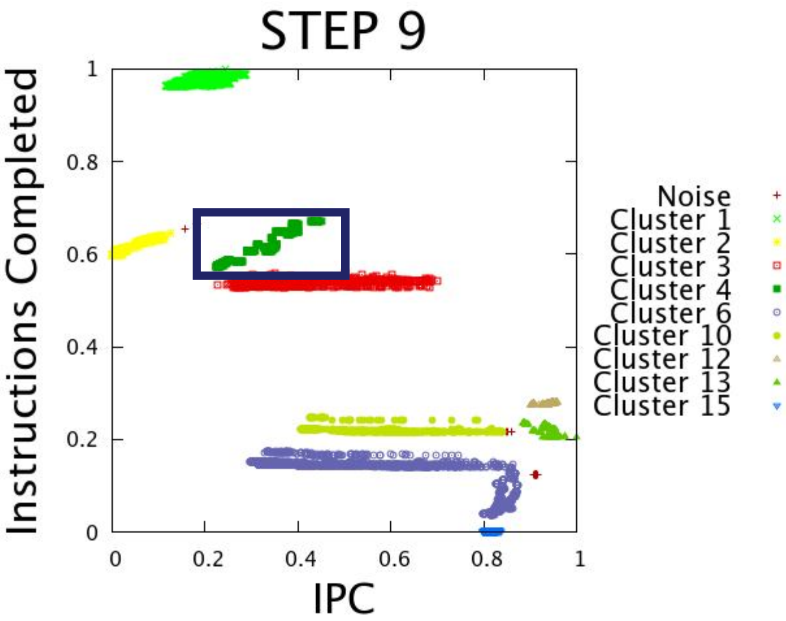
STEP 7 - Eps=0.0047: 11: 100%

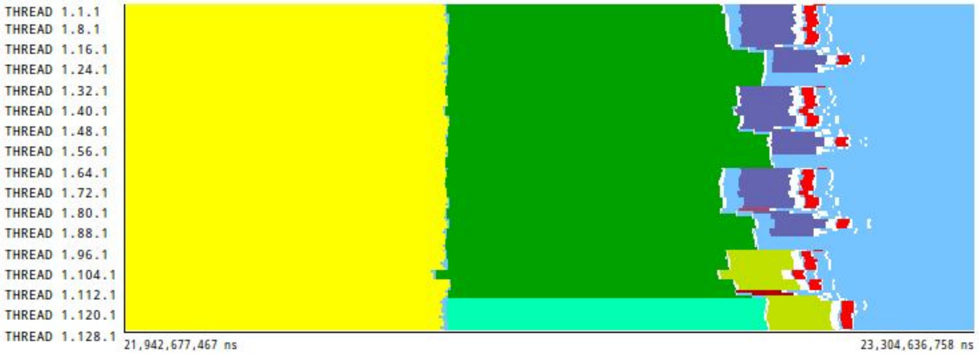
VAC4 128 tasks



VAC4 128 tasks

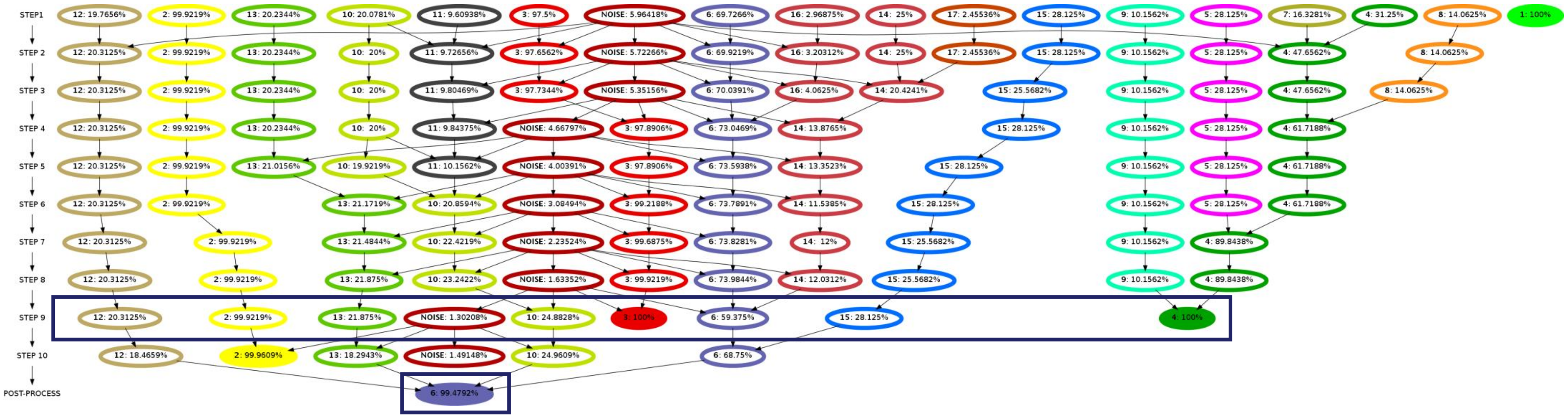


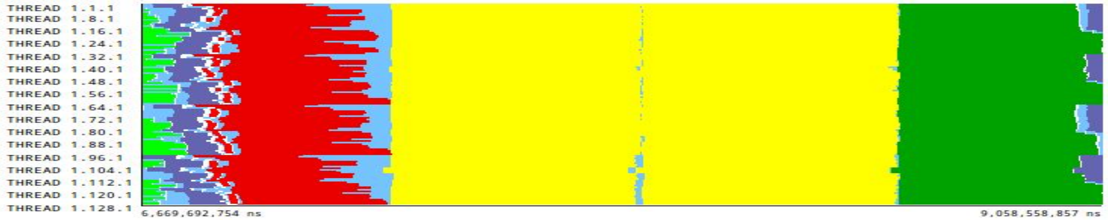


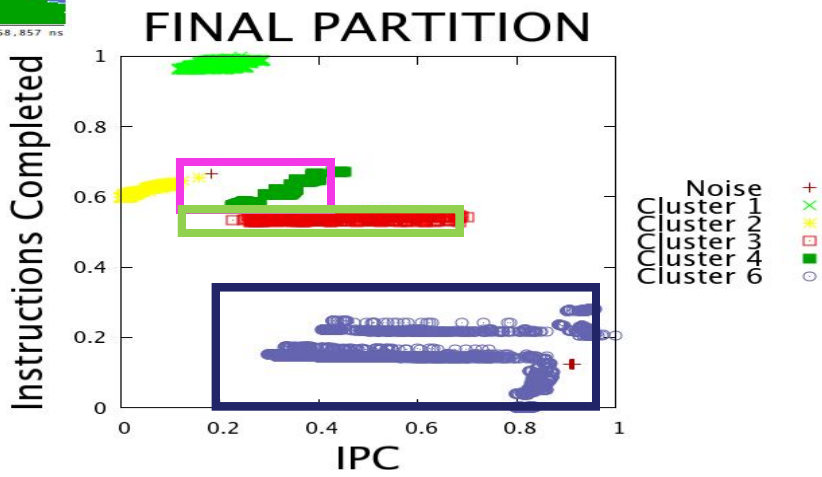


Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

VAC4 128 tasks







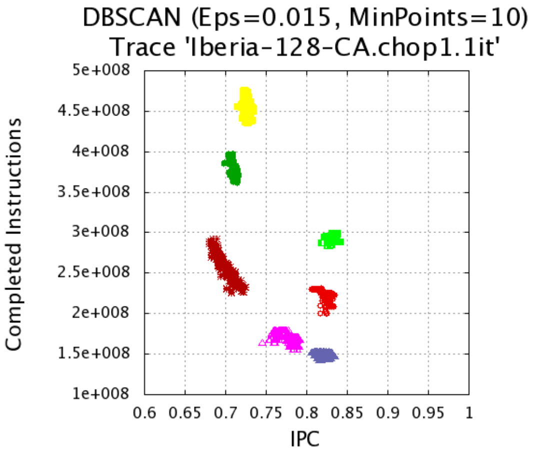
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Outline

- Computation Structure detection
  - Short intro
  - Aggregative Refinement
  - **Tracking program evolution**
  - Scaling clustering algorithm
- Instantaneous performance metric
  - Clustering + Folding

Correlating multiple runs

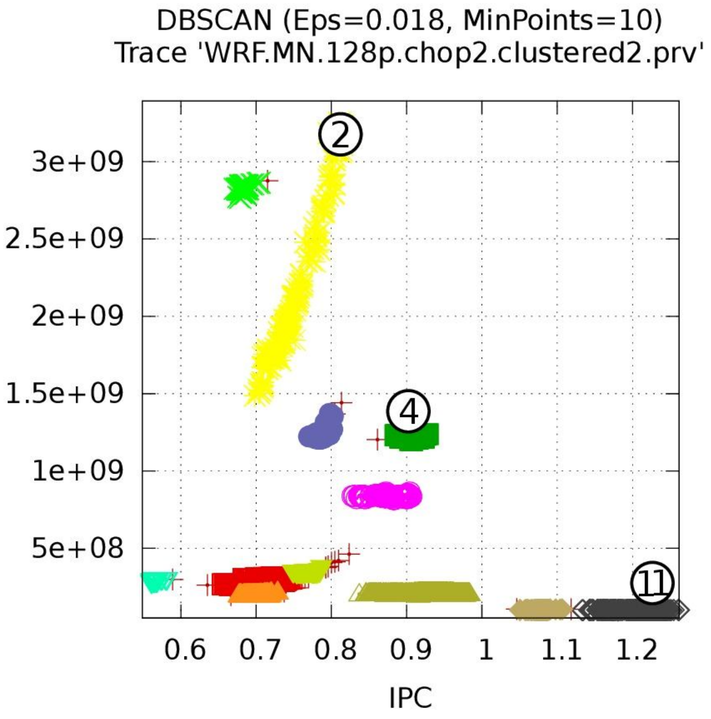
- Use and correlate information from different runs
  - Analysis of input parameters
  - Code improvements
  - Using different machines, compilers, flags, libraries
  - Scalability studies
  - Even for the same run: time evolution
- Scatter plot = performance picture
  - Identifies objects and their weight
  - Correlation → image tracking
- Based on heuristics
  - Code regions evolve smoothly (things keep closer)
  - No common callstack means not the same region
  - Time sequence identify regions within and between runs

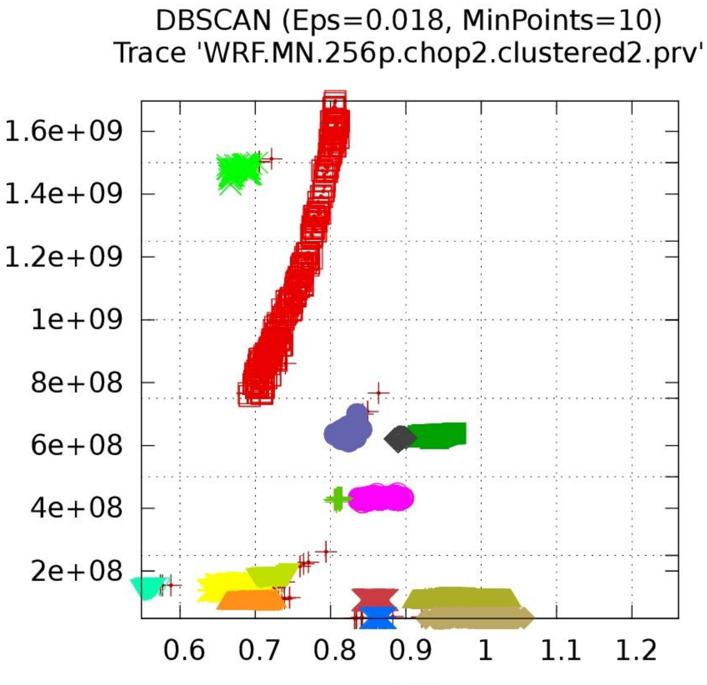


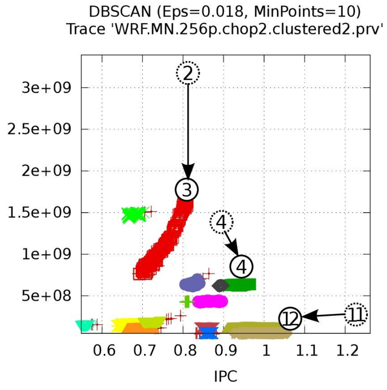
On the usefulness of object tracking techniques in performance analysis (UPC-DAC-RR-2012-18)

Scenario 1: Analysing scalability (WRF)

- 128 vs. 256

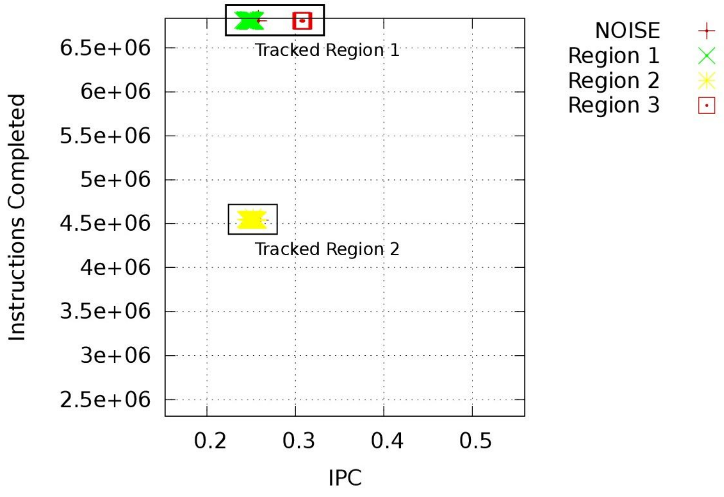




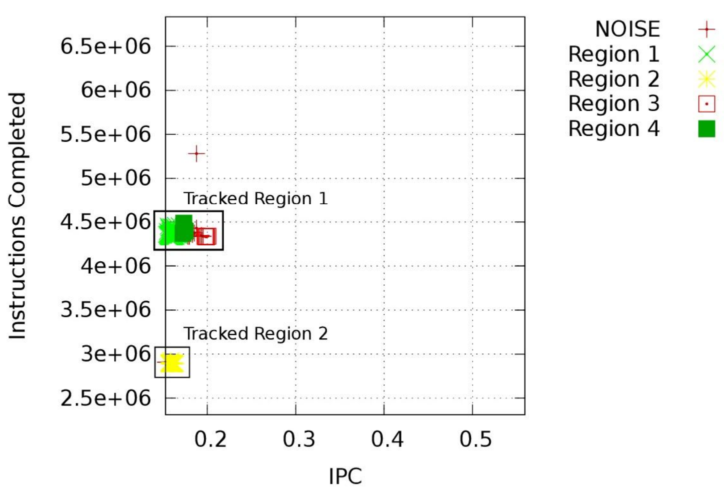


Scenario 2: Comparing machines & compilers (CG-POP)

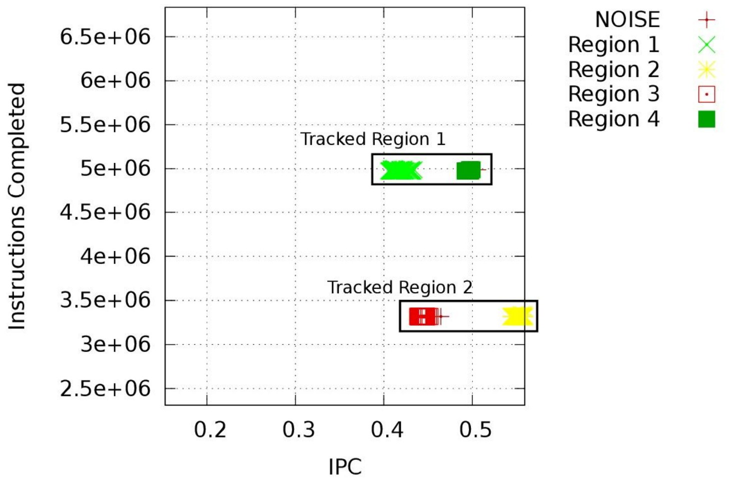
PowerPC, gfortran



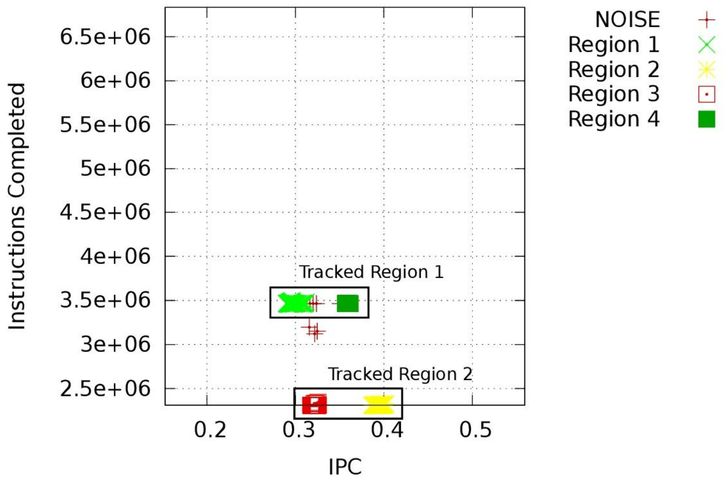
PowerPC, xlc



Intel, gfortran

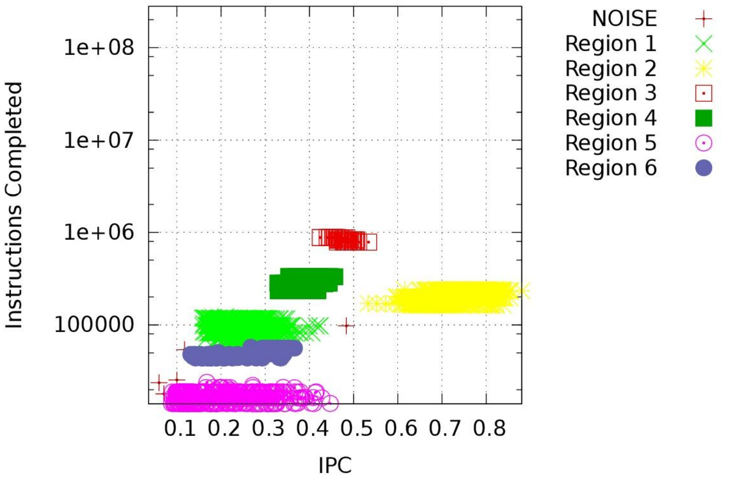


Intel, ifort

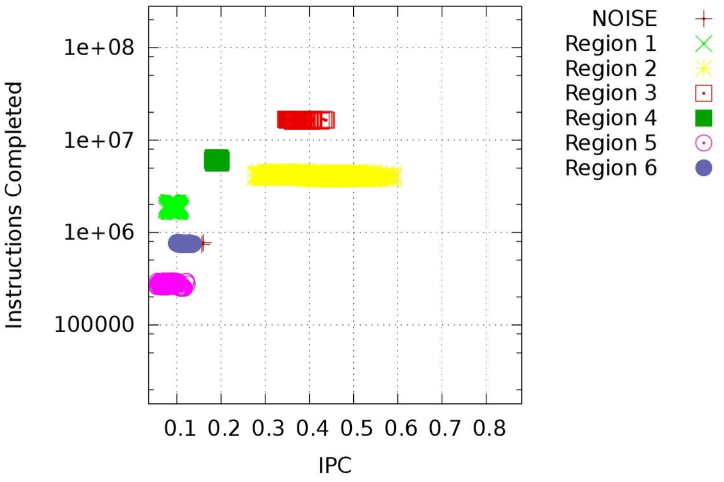


Scenario 3: Problem size impact (NAS-BT)

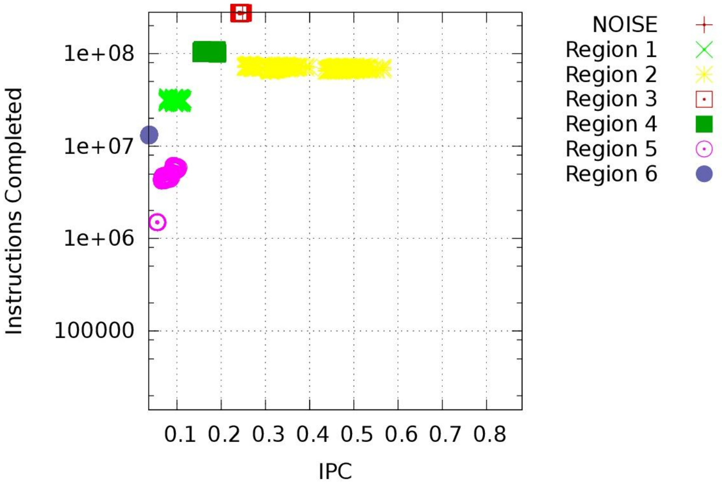
Class W



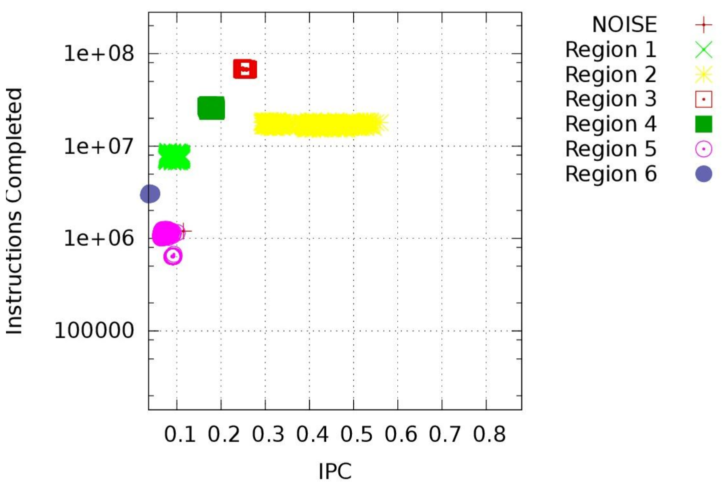
Class A



Class C



Class B



Outline

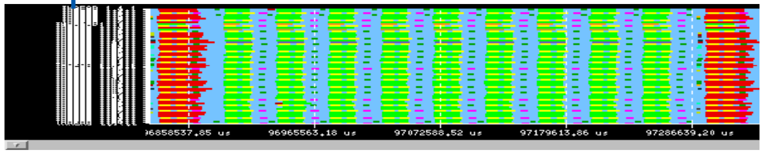
- Computation Structure detection
  - Short intro
  - Aggregative Refinement
  - Tracking program evolution
  - **Scaling clustering algorithm**

- Instantaneous performance metric
  - Clustering + Folding

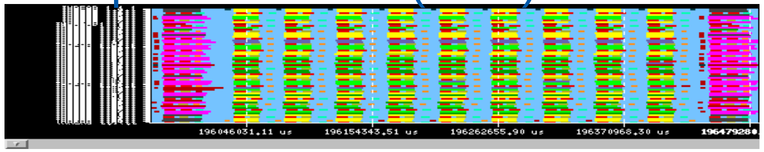
Sampling input data

- First target: online clustering
  - Centralised approach (global clustering at the MRNet frontend)
  - Data reduction trough sampling
  - Data classification based on the samples clustering

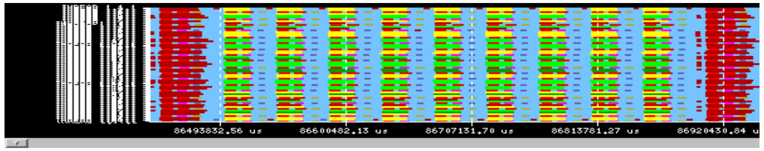
All processes



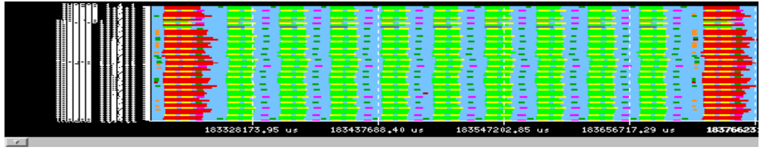
32 representatives (50%)



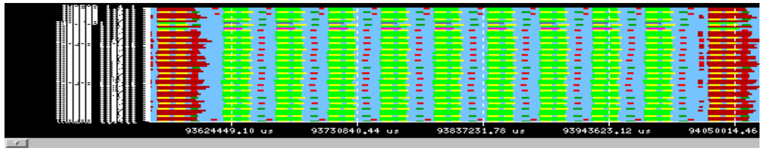
15% random records



8 representatives + 15% random



25% random records

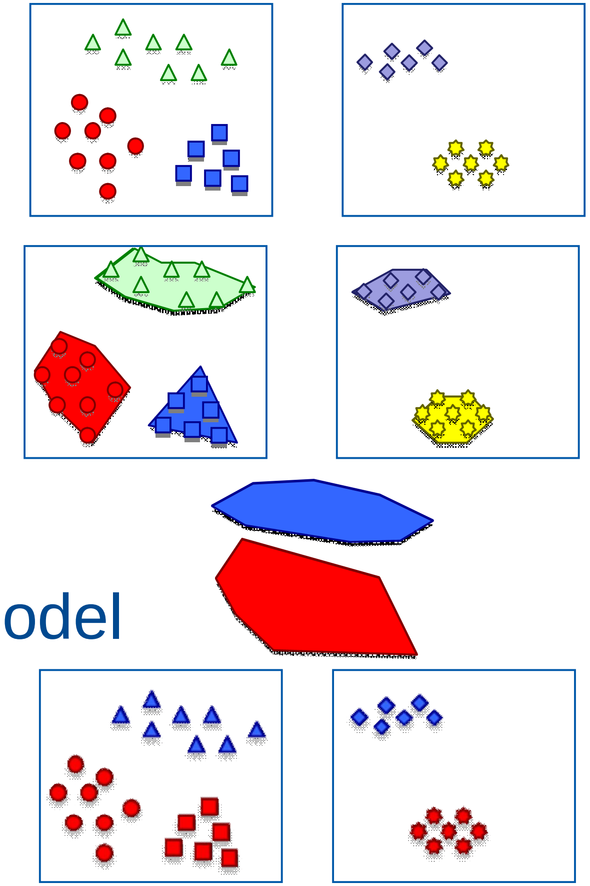


75% less data
6s down from 2m

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Hierarchical DBSCAN

1. Local clustering
   - Up to 20-30k points per local process
2. Generate models
   - Convex hull, medial axis…
3. Merge the hulls over the MRNet
   - Intersect?
4. Broadcast the global model
5. Classify data locally using the global model
   - Point inside the hull?

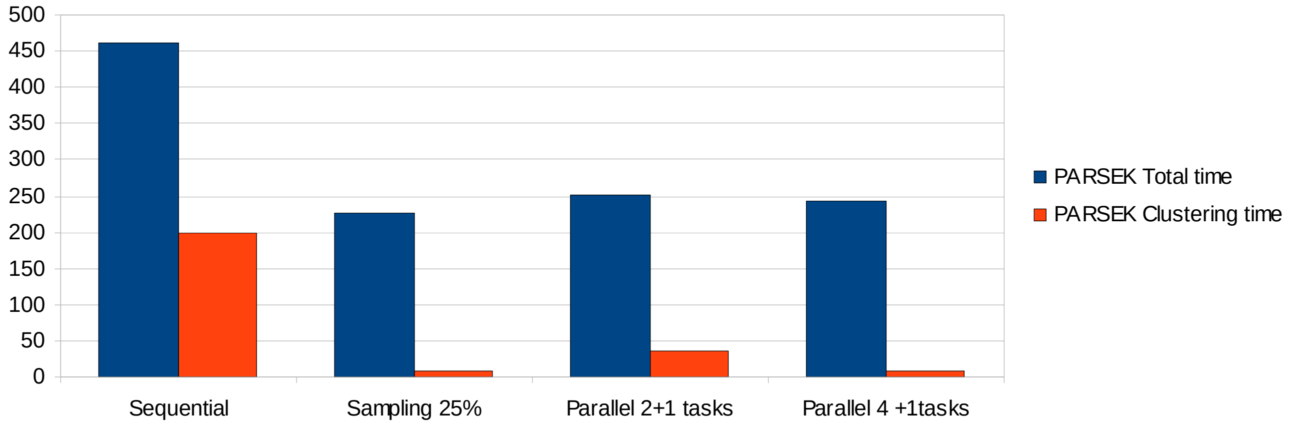


PEPC 4K tasks, 3095134 points, 273 tasks (16 way tree, 256 leaves, 12k points per local clustering) → clustering time 28.6 sec

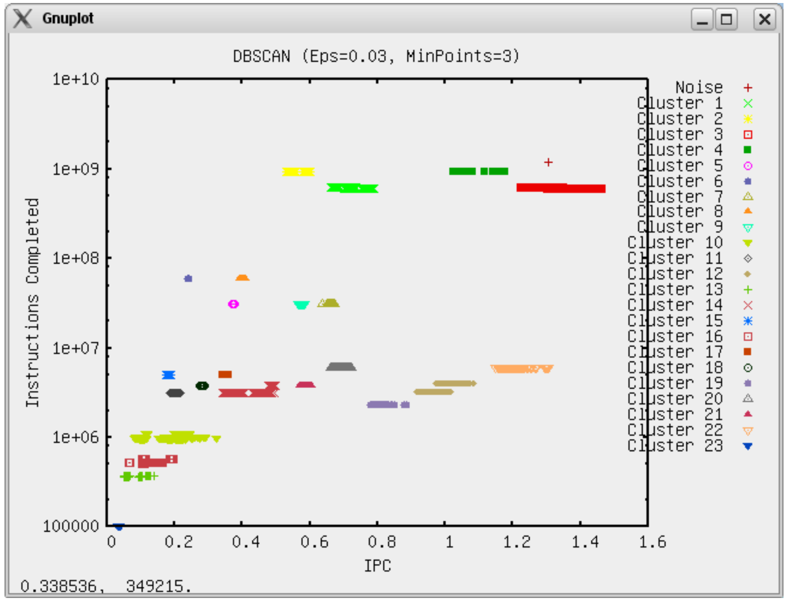
Distributed tree-based implementation of DBSCAN cluster algorithm for parallel applications analysis (UPC-DAC-RR-2011-38)

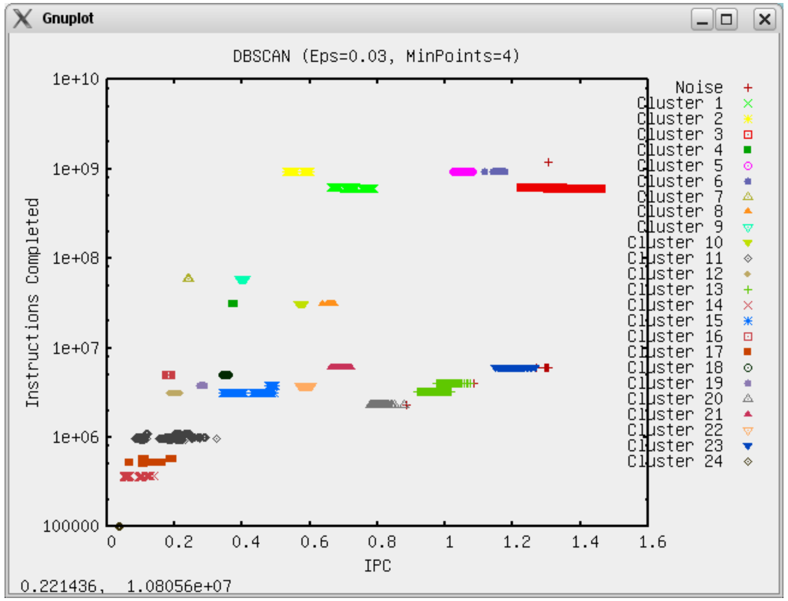
Comparison (parsek)

31515 points



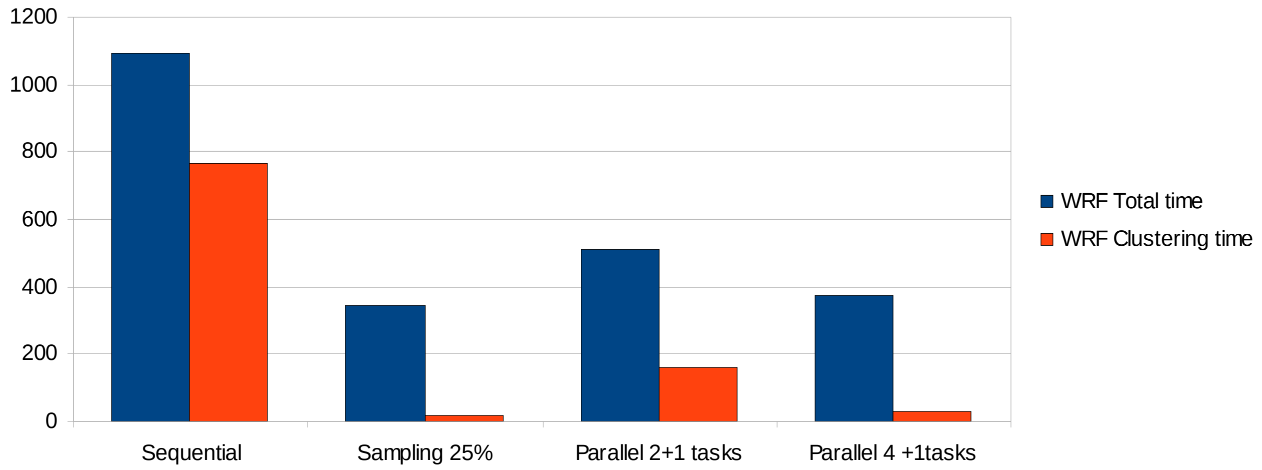
Sampling 25%



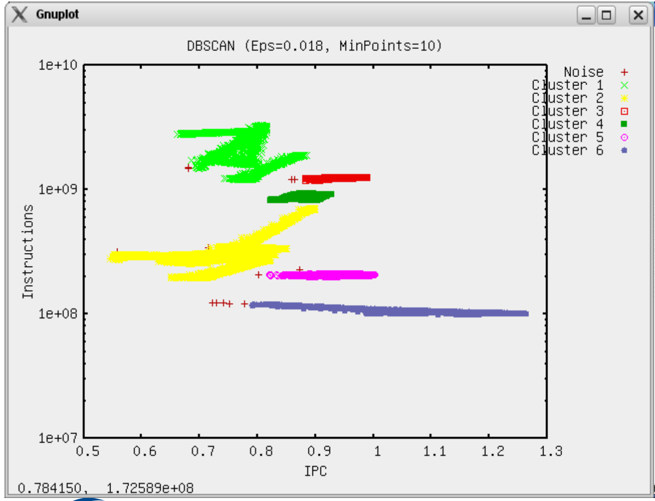


Comparison (WRF)

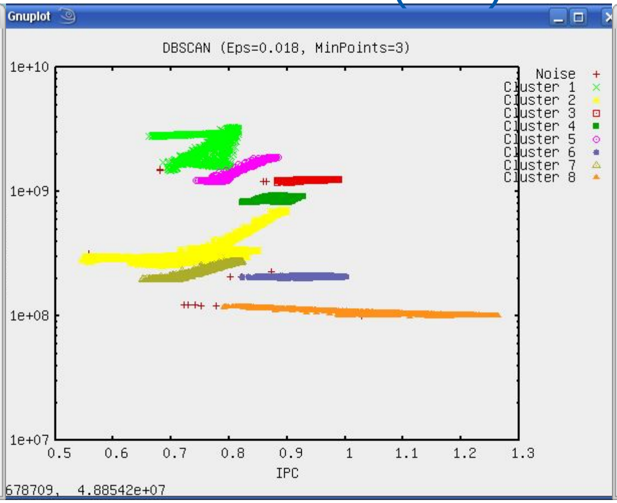
74240 points



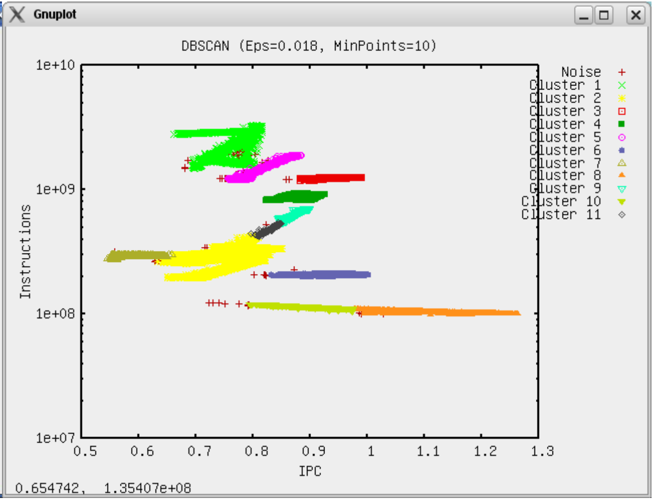
Sequential /Par (4+1)



Parallel (2+1)



Sampling 25%



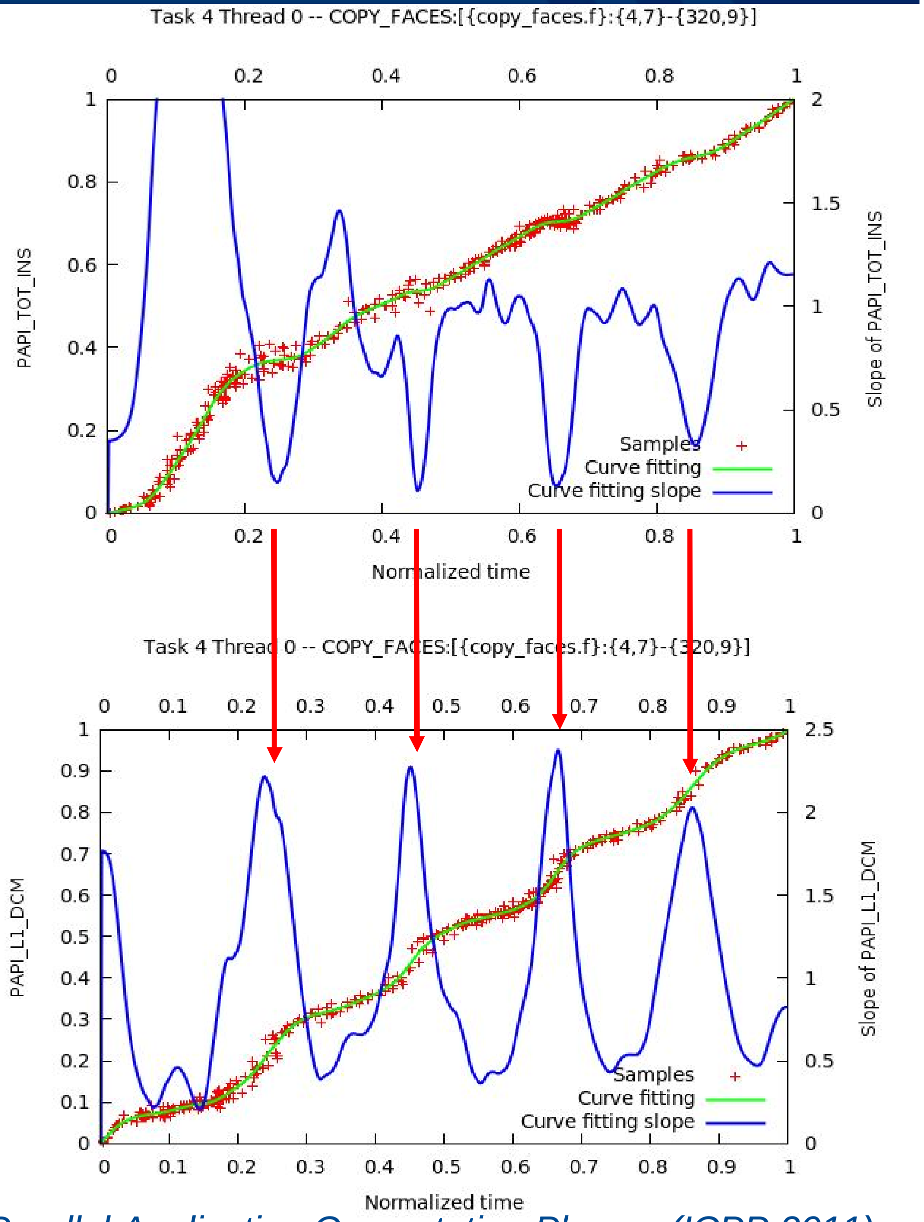
Outline

- Computation Structure detection
  - Short intro
  - Aggregative Refinement
  - Tracking program evolution
  - Scaling clustering algorithm

- **Instantaneous performance metric**
  - **Clustering + Folding**

Can I get very detailed perf. data with low overhead?

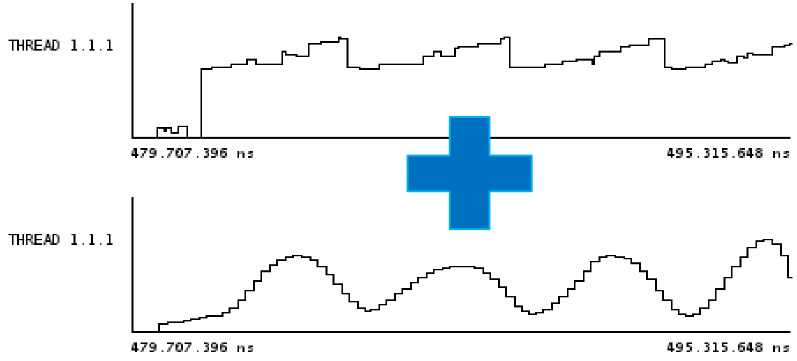
- Application granularity vs. detailed granularity
  - Samples: hardware counters + callstack
- Folding: based on known structure: iterations, routines, clusters;
  - Project all samples into one instance
- Extremely detailed time evolution of hardware counts, rates and callstack with minimal overhead
  - Correlate many counters
  - Instantaneous CPI stack models



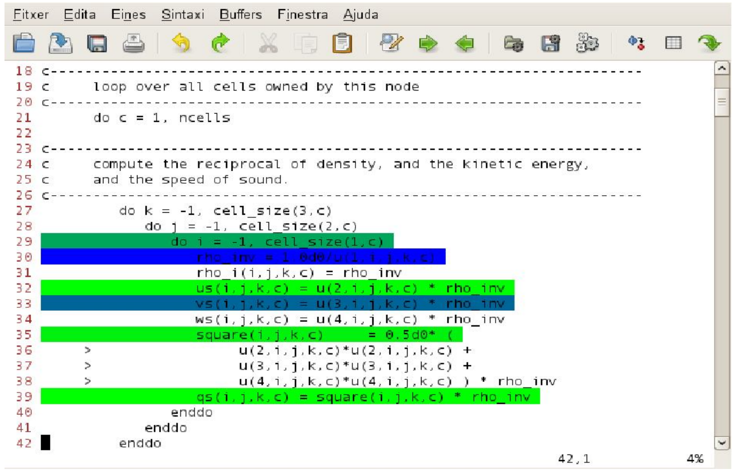
Unveiling Internal Evolution of Parallel Application Computation Phases (ICPP 2011)

Correlating with sources: which line should I look?

Folded source code line



Folded instructions

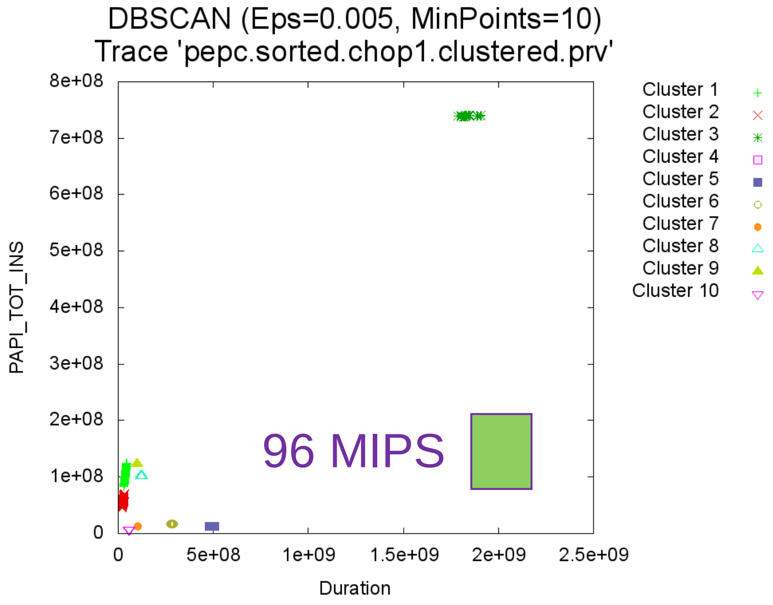


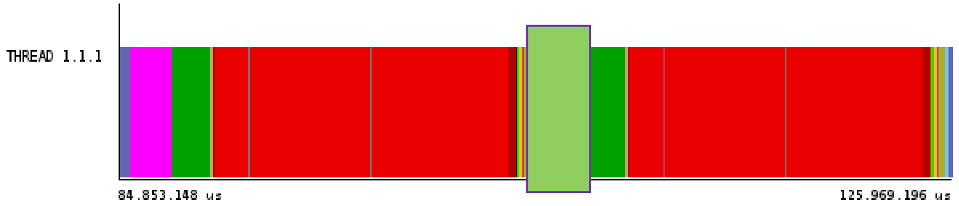
```
c-----------------------------------------------------------------------
c     loop over all cells owned by this node
c-----------------------------------------------------------------------
      do c = 1, ncells

c-----------------------------------------------------------------------
c     compute the reciprocal of density, and the kinetic energy,
c     and the speed of sound.
c-----------------------------------------------------------------------
         do k = -1, cell_size(3,c)
            do j = -1, cell_size(2,c)
               do i = -1, cell_size(1,c)
                  rho_inv = 1.0d0/u(1,i,j,k,c)
                  rho_i(i,j,k,c) = rho_inv
                  us(i,j,k,c) = u(2,i,j,k,c) * rho_inv
                  vs(i,j,k,c) = u(3,i,j,k,c) * rho_inv
                  ws(i,j,k,c) = u(4,i,j,k,c) * rho_inv
                  square(i,j,k,c)     = 0.5d0* (
     >                 u(2,i,j,k,c)*u(2,i,j,k,c) +
     >                 u(3,i,j,k,c)*u(3,i,j,k,c) +
     >                 u(4,i,j,k,c)*u(4,i,j,k,c) ) * rho_inv
                  qs(i,j,k,c) = square(i,j,k,c) * rho_inv
               enddo
            enddo
         enddo
```

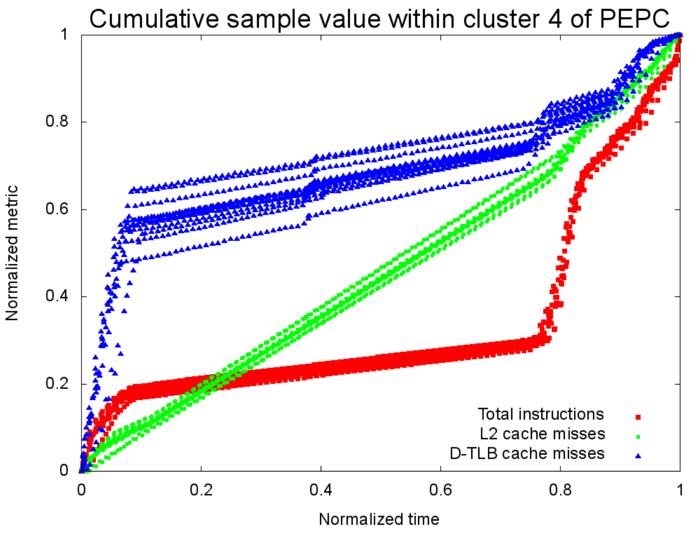
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

The “benefits” of Fortran 90 intrinsic (PEPC)





Performance metrics
16 MIPS
2.3 M L2 misses/s
0.1 M TLB misses/s

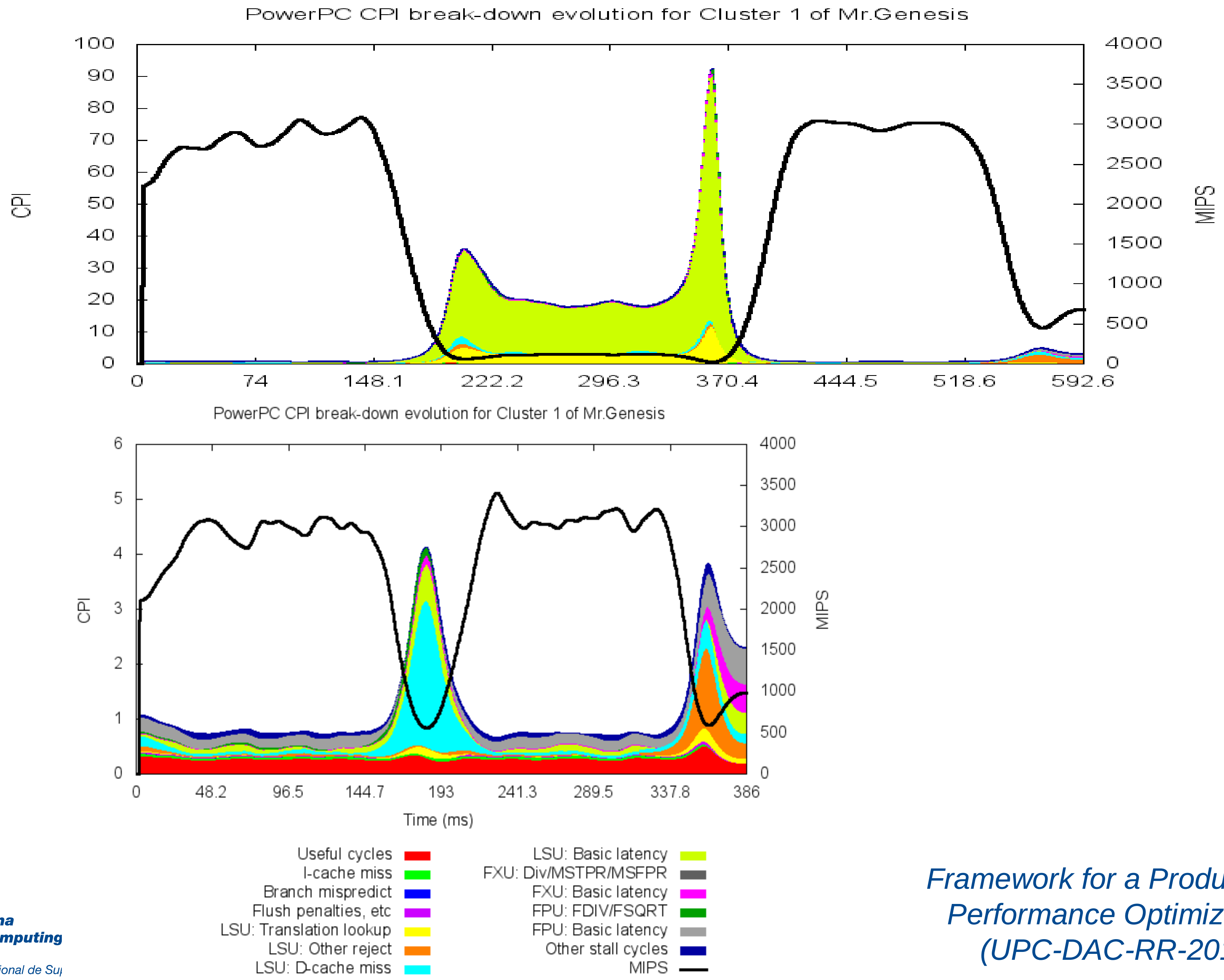


```
htable%node = 0
htable%key = 0
htable%link = -1
htable%leaves = 0
htable%childcode = 0
```

```
do i = 1, n
 htable(i)%node = 0
 htable(i)%key = 0
 htable(i)%link = -1
 htable(i)%leaves = 0
 htable(i)%childcode = 0
End do
```

Changes
-70% time
-18% instructions
-63% L2 misses
-78% TLB misses
253 MIPS (+163%)

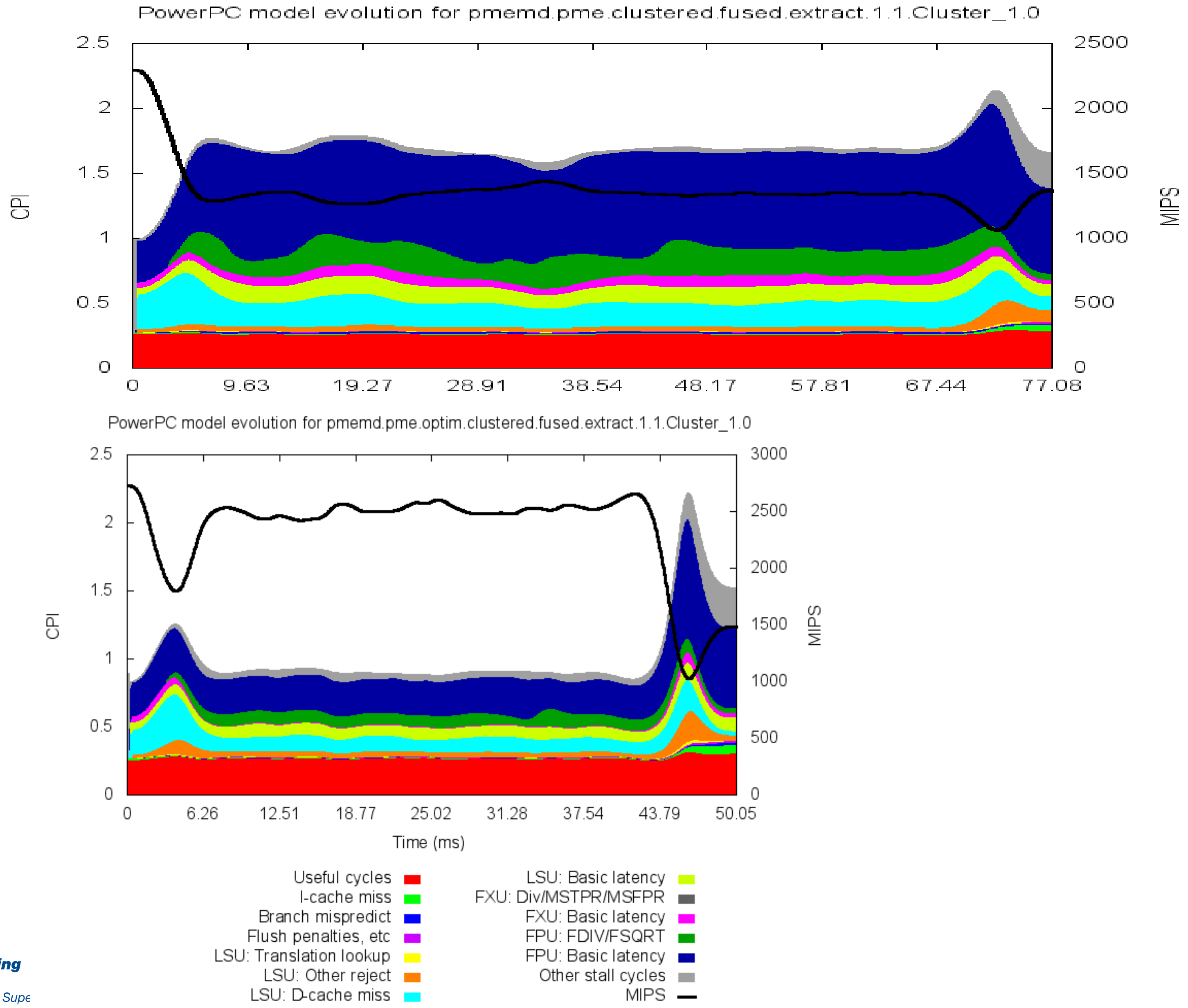
Interchanging loops (MR. GENESIS)



Barcelona
Supercomputing
Center
Centro Nacional de Su

Framework for a Productive Performance Optimization (UPC-DAC-RR-2012-2)

Pre-computing float data – loop split (PMEMD)



Conclusions

- Performance analytics
  - Data analytics applied to raw performance data
  - From data to insight
    - Information is on variability and distribution
  - Huge room for research

- Showed results of some techniques
  - Clustering enables focusing the analysis and open many different uses on the analysis
  - Folding makes possible to compute instantaneous performance metric functions with low overhead
  - Tracking helps detecting movement in the performance space
    - Sequence of “frames” along many factors (not just time)

www.bsc.es/paraver