HACC

Hardware Accelerated Cosmology Code

PPTM - Q2 2015

Professor: Jesus Labarta

Author: Cristóbal Ortega

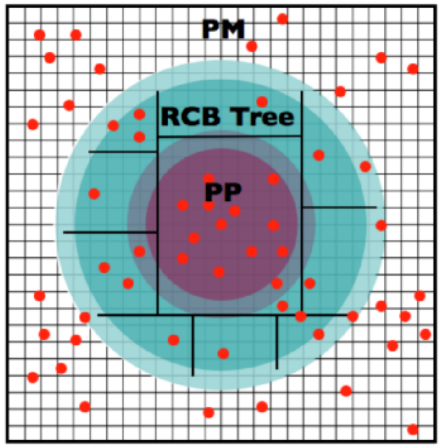
June 23, 2015

Outline

Overview[1]

- Getting the N-body problem to petaflop systems
  - "The N-body problem is the problem of predicting the individual motions of a group of celestial objects interacting with each other gravitationally"

- HACC uses a hybrid parallel algorithmic structure:
  - A grid-based for long/medium range (common in all these codes)
  - An architecture-tunable particle-based short/close-range solver



Green: medium/short range
Magenta: close range

- Coded to run in large systems:
  Tested on 1,572,864 cores with 90% parallel efficiency

[1] Salman Habib et al. "HACC: Extreme Scaling and Performance Across Diverse Architectures". In: **Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis**. SC '13. Denver, Colorado: ACM, 2013, 6:1–6:10. ISBN: 978-1-4503-2378-9.

Input

Important parameters to consider:

- N particles alive
- Steps
- Geometry:
  - Virtual topology of the network
  - D1xD2xD3
  - Each dimension must of the decomposition must be a deviser of the N value.
  - A Cartesian topology will be created based on it

Topology

In the presentation 2 types of topologies will be analyzed:

Distributed

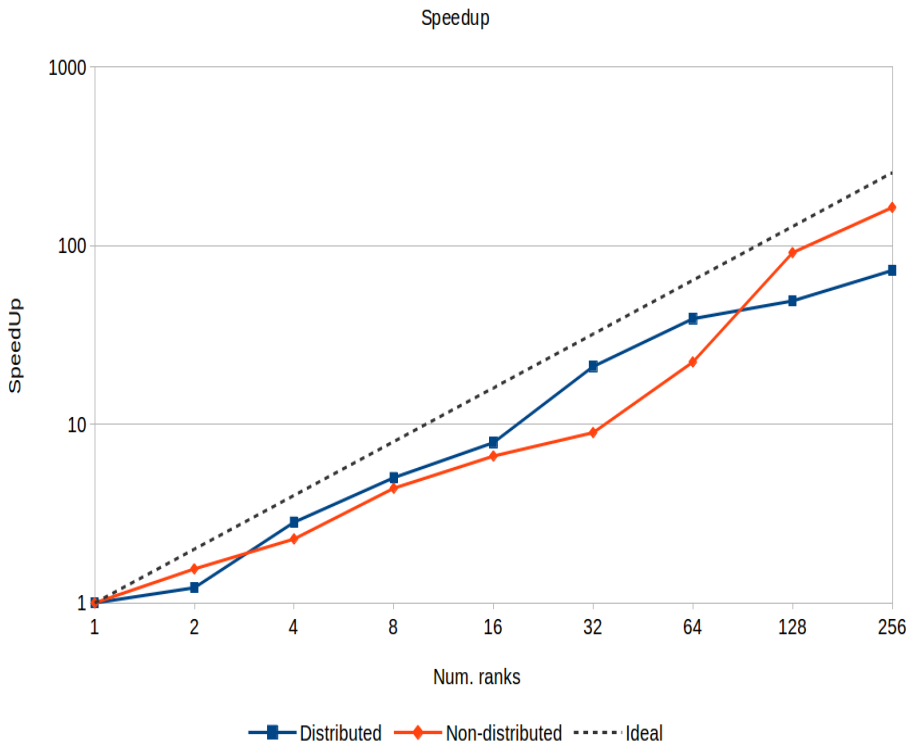
- Per a given number of MPI processors it will be decomposed in powers of 2 across all the dimensions
  - For 2 ranks $\rightarrow$ 2x1x1
  - For 4 ranks $\rightarrow$ 2x2x1
  - For 8 ranks $\rightarrow$ 2x2x2
  - For 64 ranks $\rightarrow$ 4x4x4
  - For 128 ranks $\rightarrow$ 8x4x4

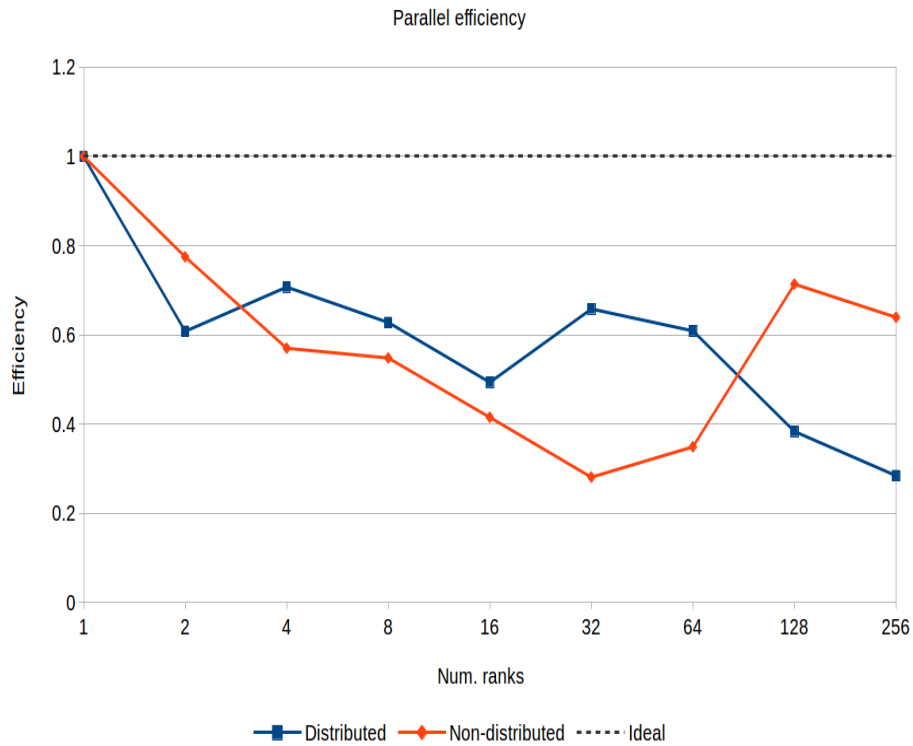
Non Distributed

- Per a given number of MPI processors all of them will correspond to the first dimension:
  - For 2 ranks $\rightarrow$ 2x1x1
  - For 4 ranks $\rightarrow$ 4x1x1
  - For 8 ranks $\rightarrow$ 8x1x1
  - For 64 ranks $\rightarrow$ 64x1x1
  - For 128 ranks $\rightarrow$ 128x1x1

Strong Scalability

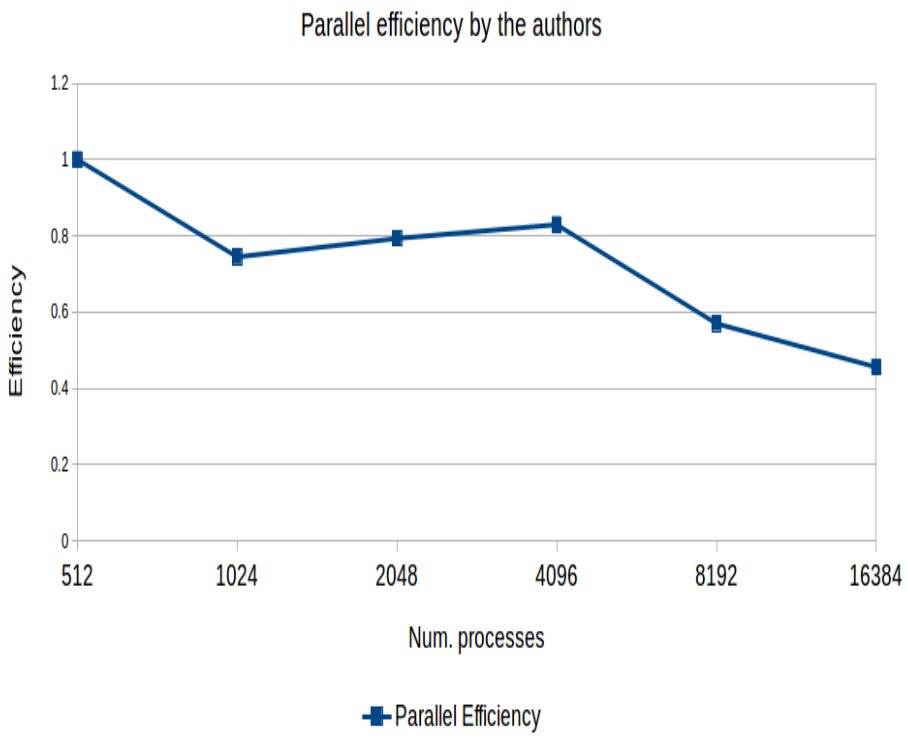
- Shown only for testing behaviour
- Will focus on the weak scalability
- Execution time for 1 process: 5 hours

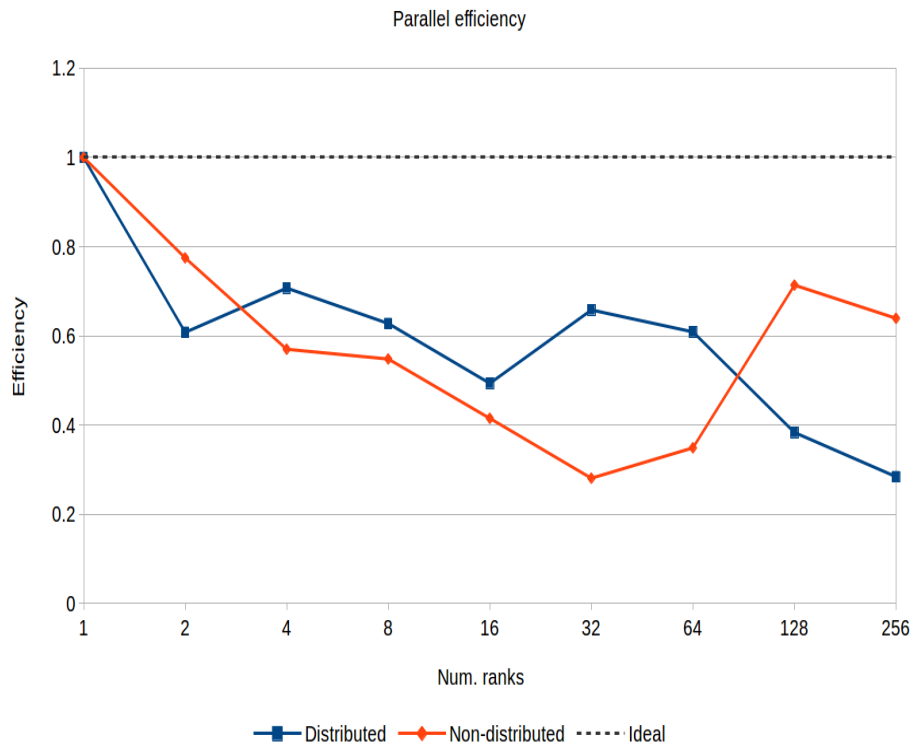




Strong Scalability (II)

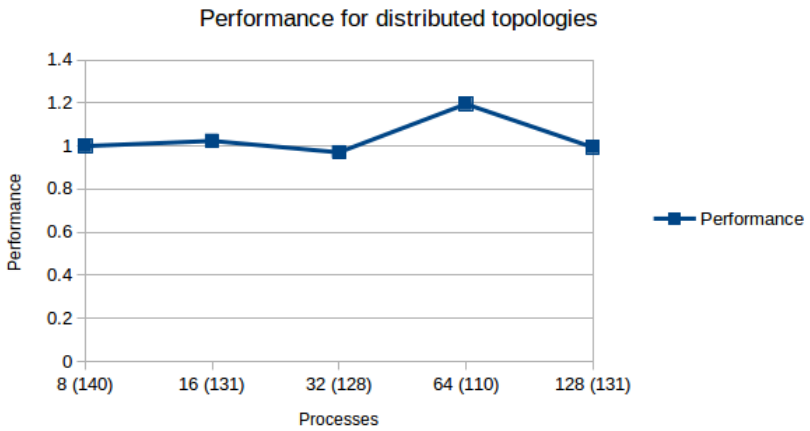
- Similar behaviour
- They run with 1.073.741.824 particles



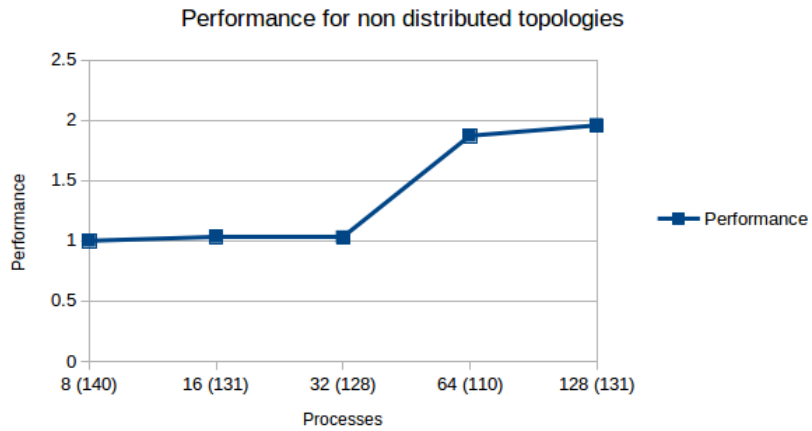


Weak Scalability

- A little imbalance by input
  Problem size has to be multiple of number of processors
- For a topology 4x4x2, the problem size has to be multiple of $4 \times 4 \times 2$
- $\text{Performance} = \frac{\text{Execution Time[8 MPI Rank]}}{\text{Execution Time[N MPI Ranks]}} \times \frac{\text{Problem Size[N MPI Rank]}}{\text{Problem Size[8 MPI Ranks]}}$

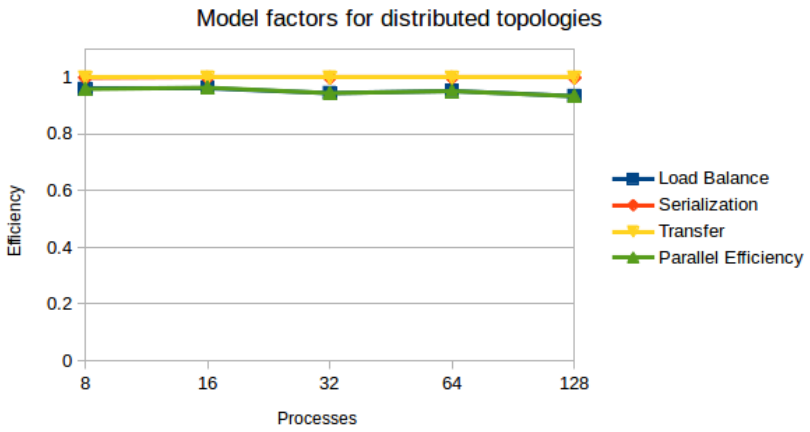


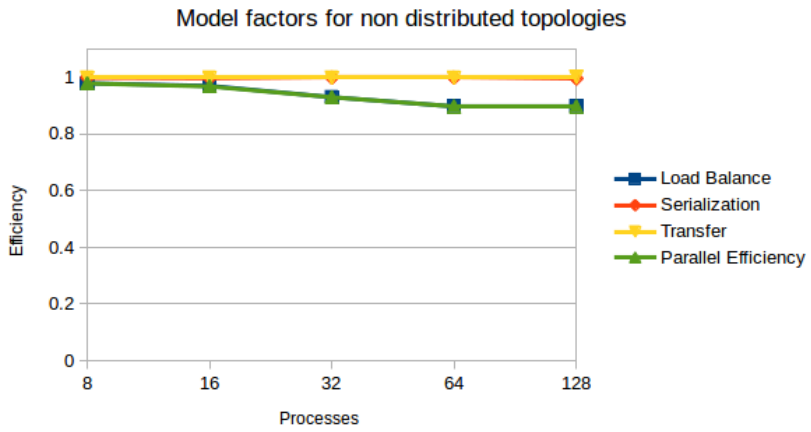
Performance for Distributed topologies

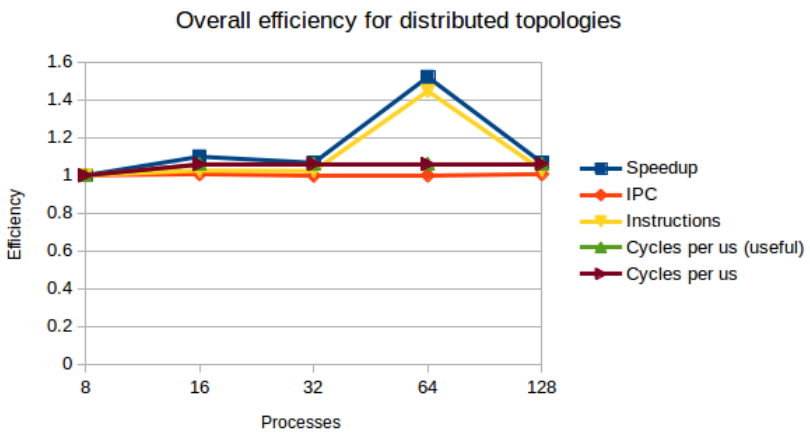


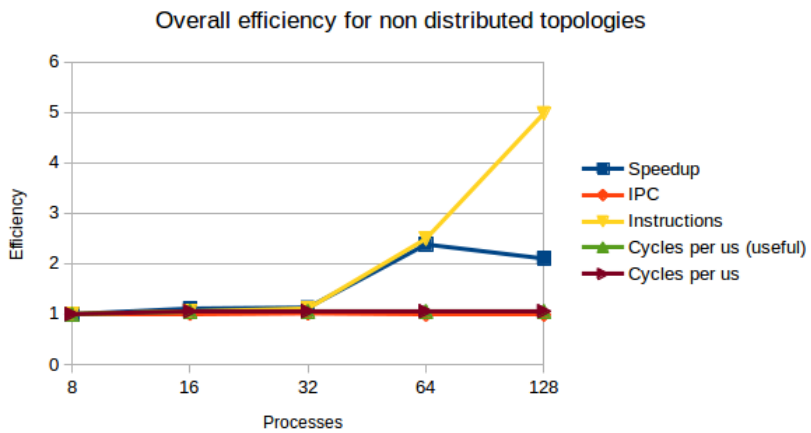
Performance for non Distributed topologies

Weak Scalability (II)



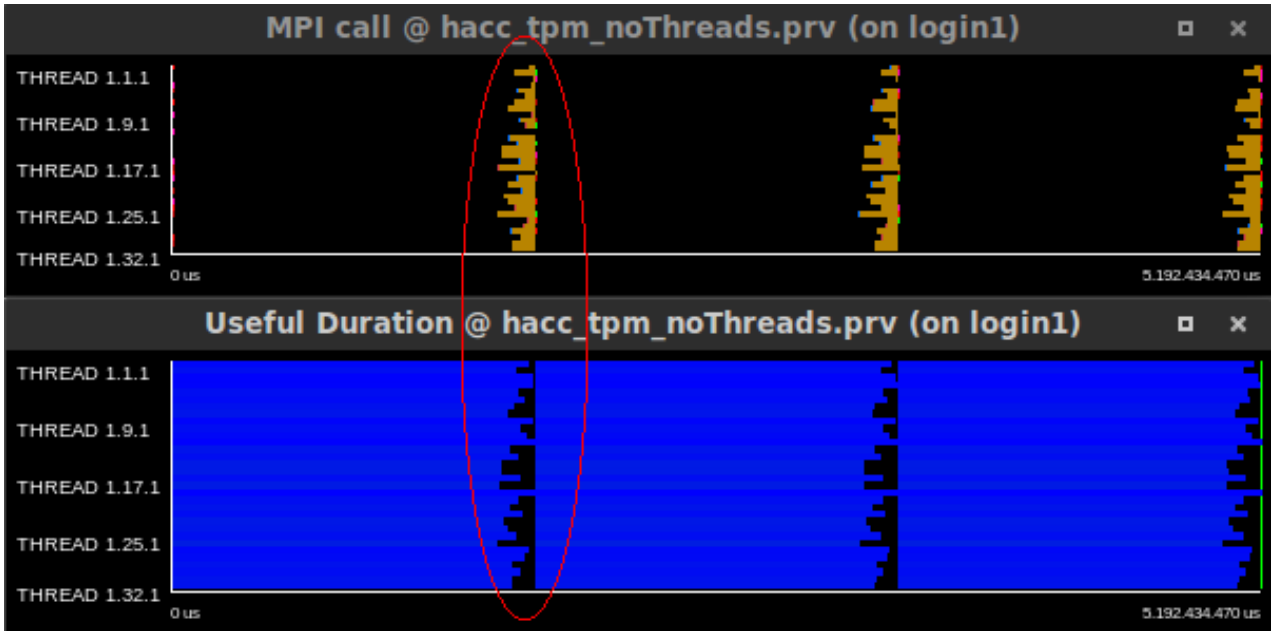






Overview

- 3 Steps
- 4x4x2 config
- MPI Calls
- Useful duration

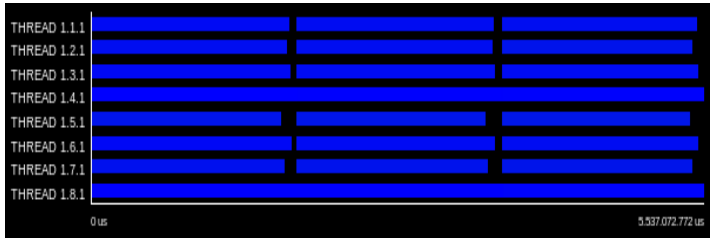


Imbalance

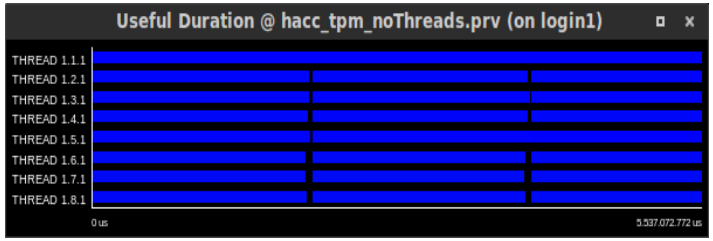
- Some process waiting in MPI to others to finalize

For different configurations

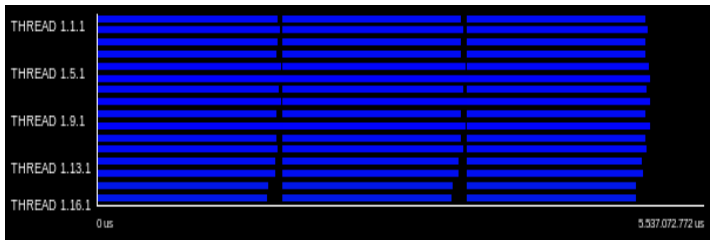
2x2x2



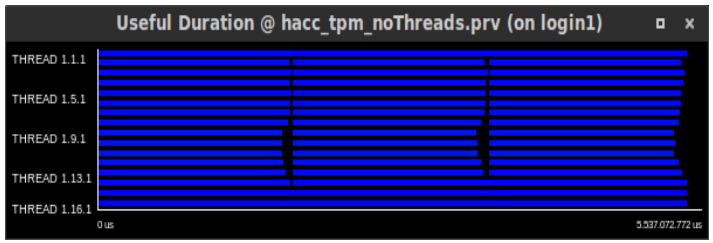
8x1x1



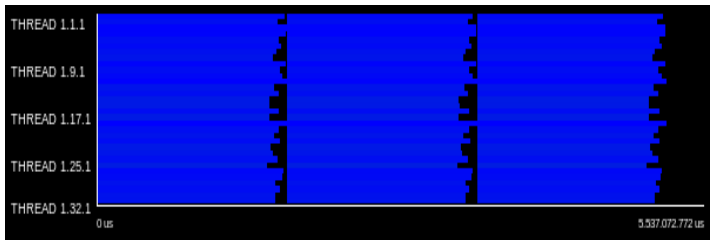
4x2x2



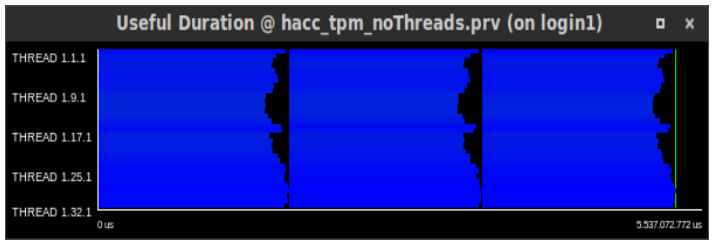
16x1x1



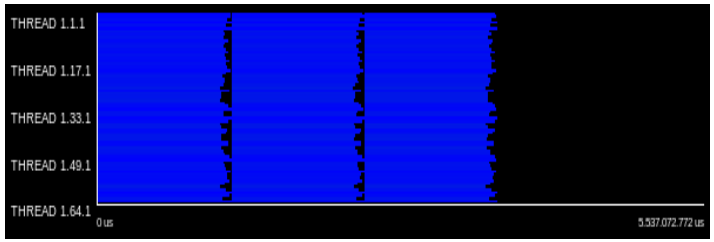
4x4x2



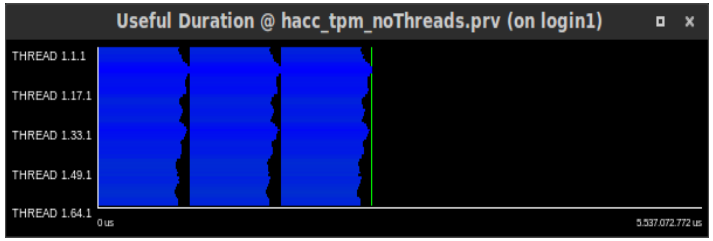
32x1x1



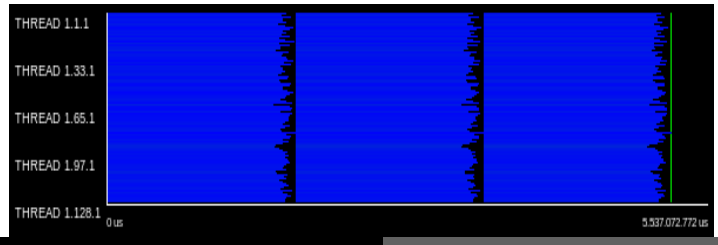
4x4x4



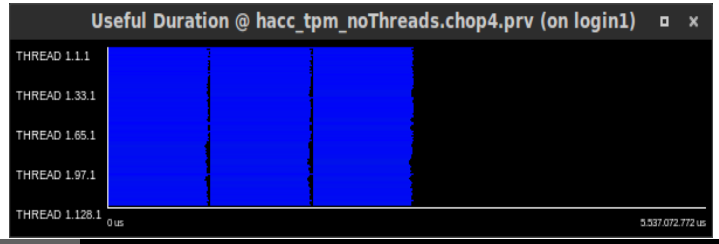
64x1x1



8x4x4



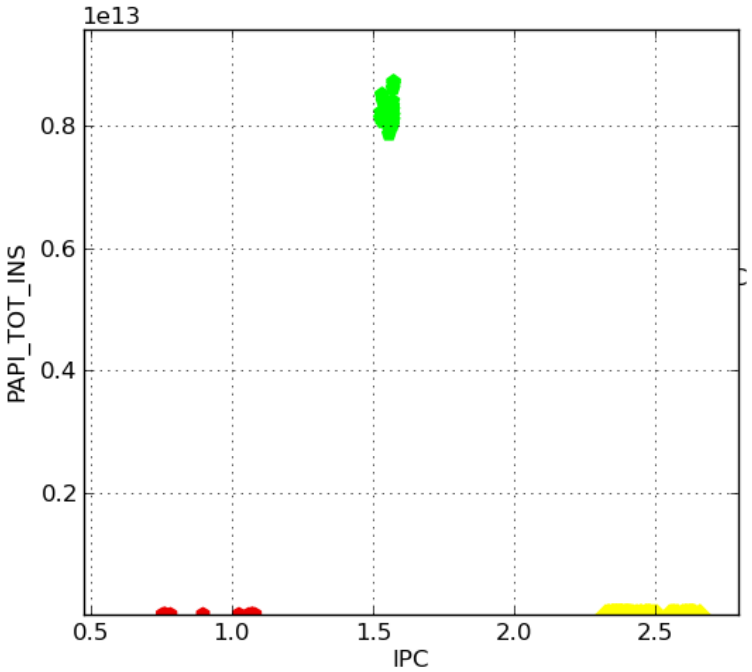
128x1x1



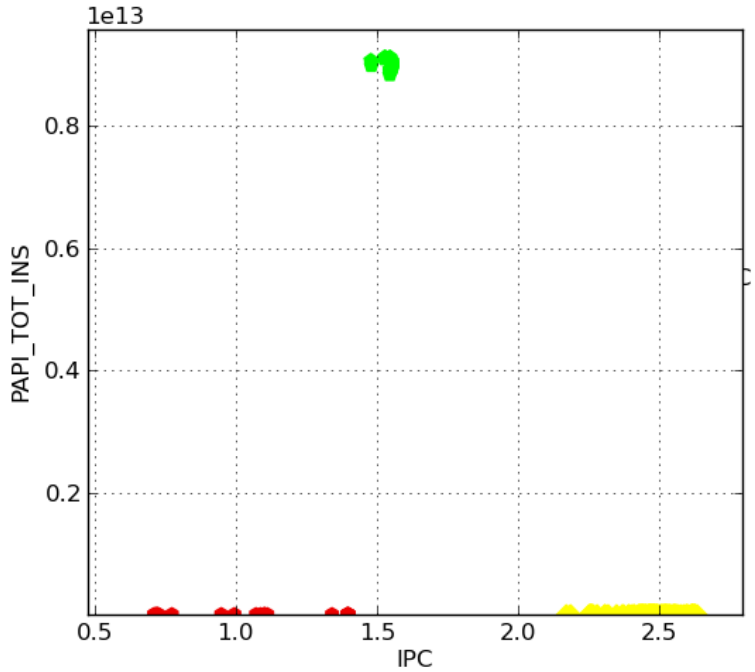
Notes

- From 8 processes up to 128
- Showing distributed and non-distributed topologies
  - Differences between number of instructions
- **64 processes** case has a smaller input data

Evolution

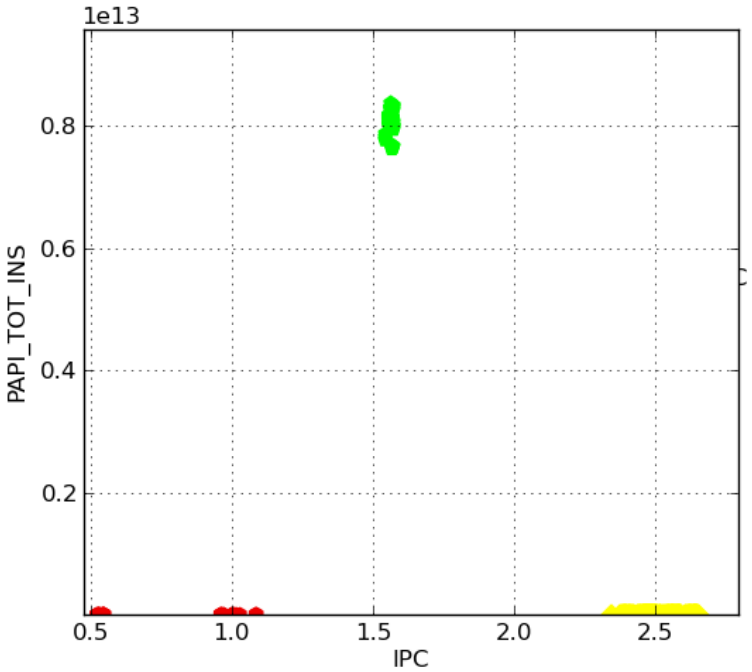


#Instructions vs IPC, 8 ranks, 2x2x2

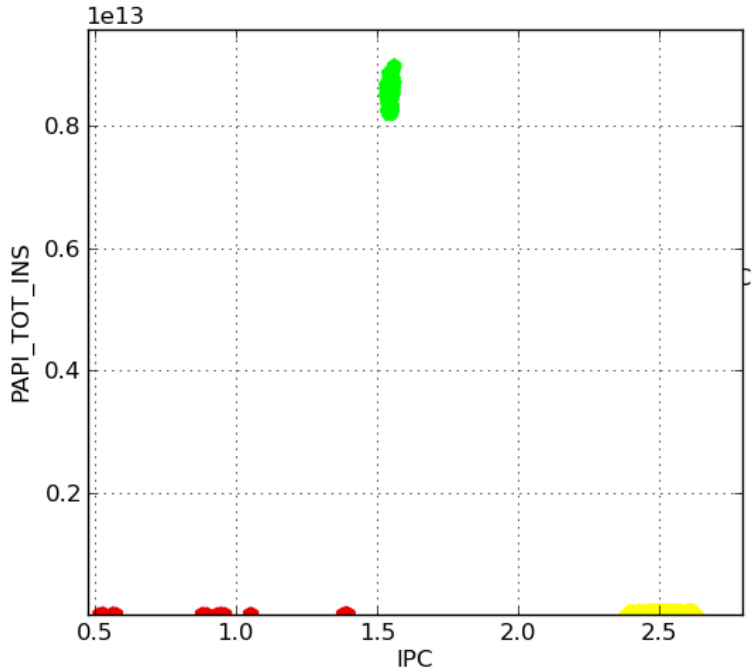


#Instructions vs IPC, 8 ranks, 8x1x1

Evolution

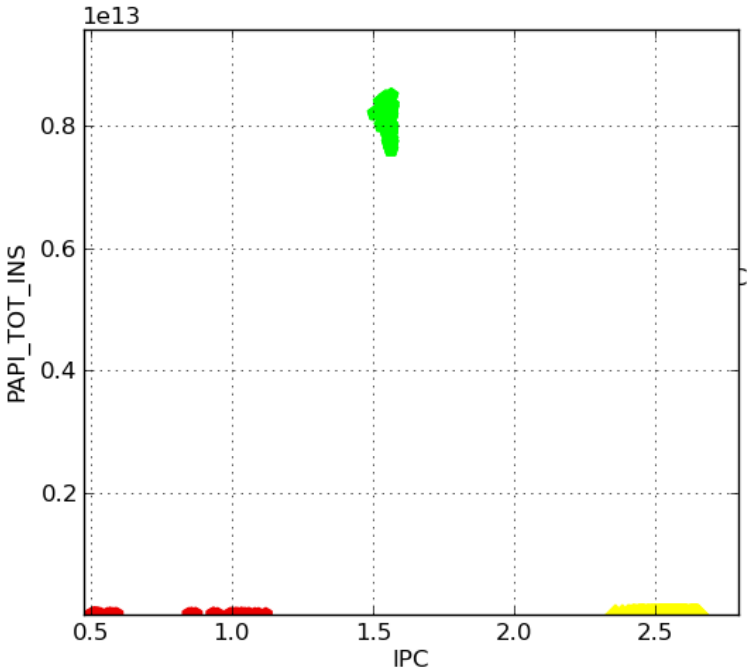


#Instructions vs IPC, 16 ranks, 4x2x2

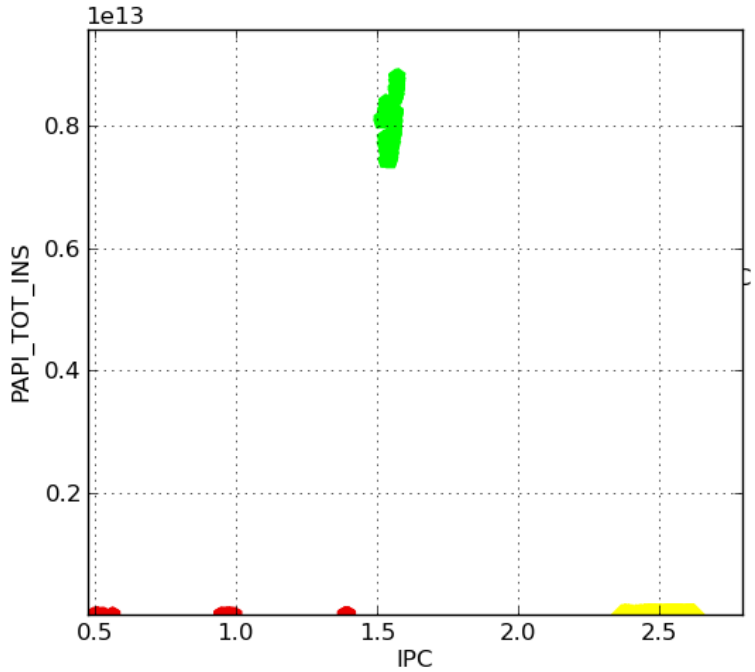


#Instructions vs IPC, 16 ranks, 16x1x1

Evolution

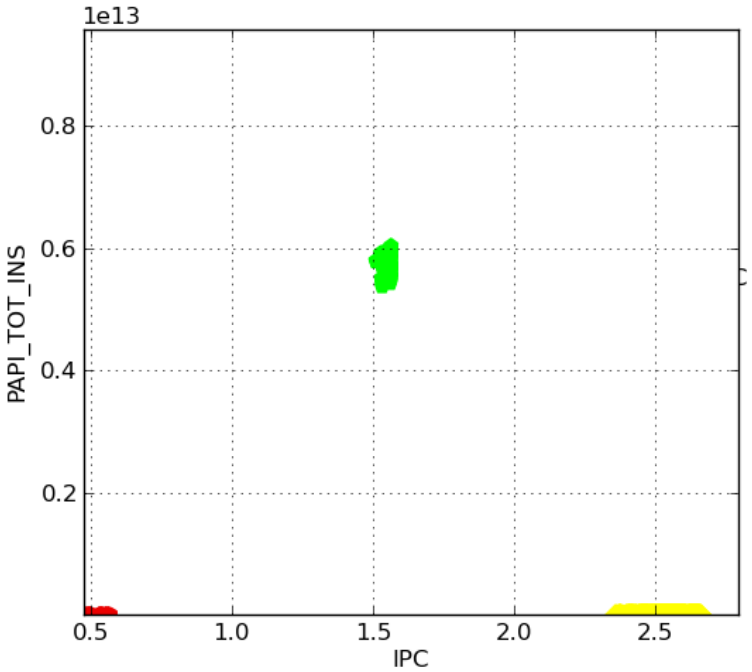


#Instructions vs IPC, 32 ranks, 4x4x2

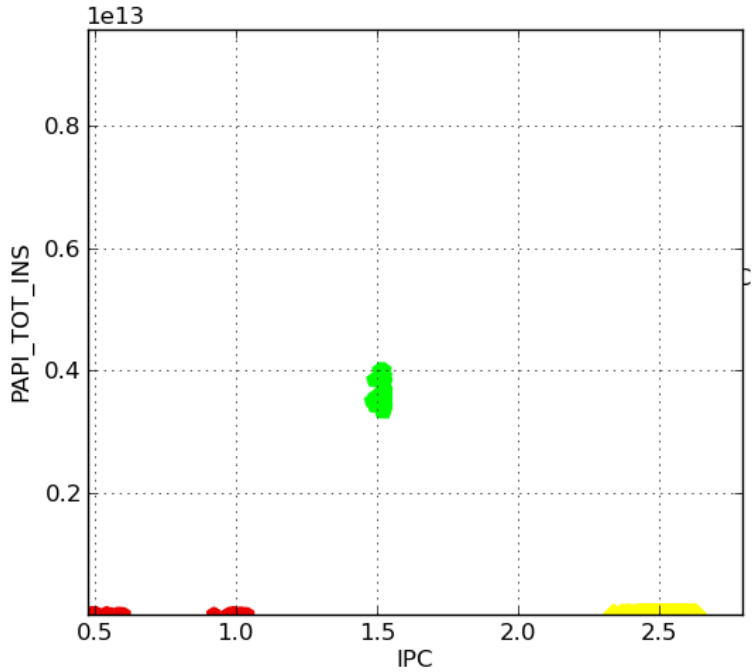


#Instructions vs IPC, 32 ranks, 32x1x1

Evolution

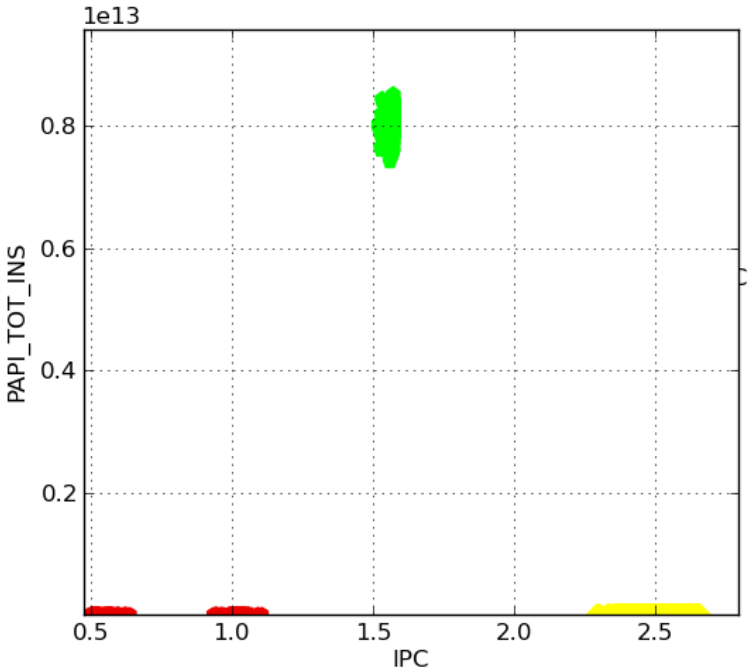


#Instructions vs IPC, 64 ranks, 4x4x4

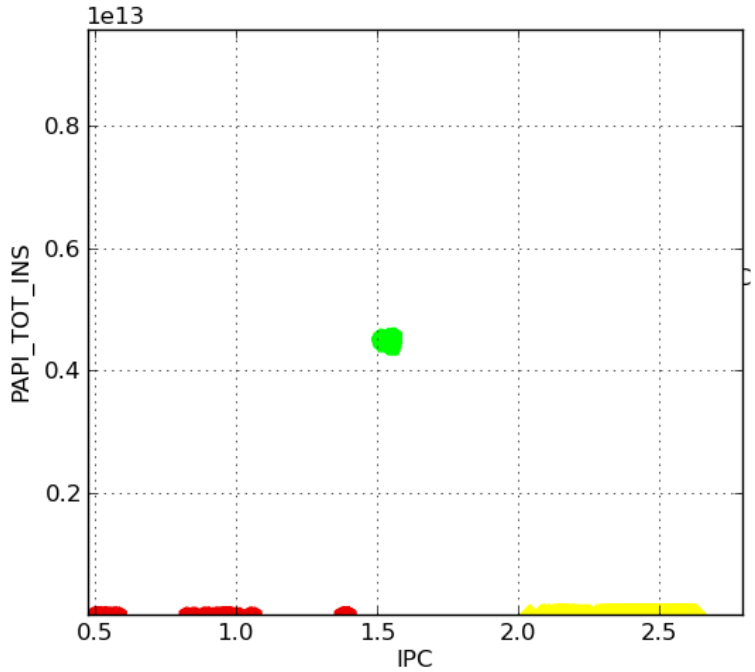


#Instructions vs IPC, 64 ranks, 64x1x1

Evolution



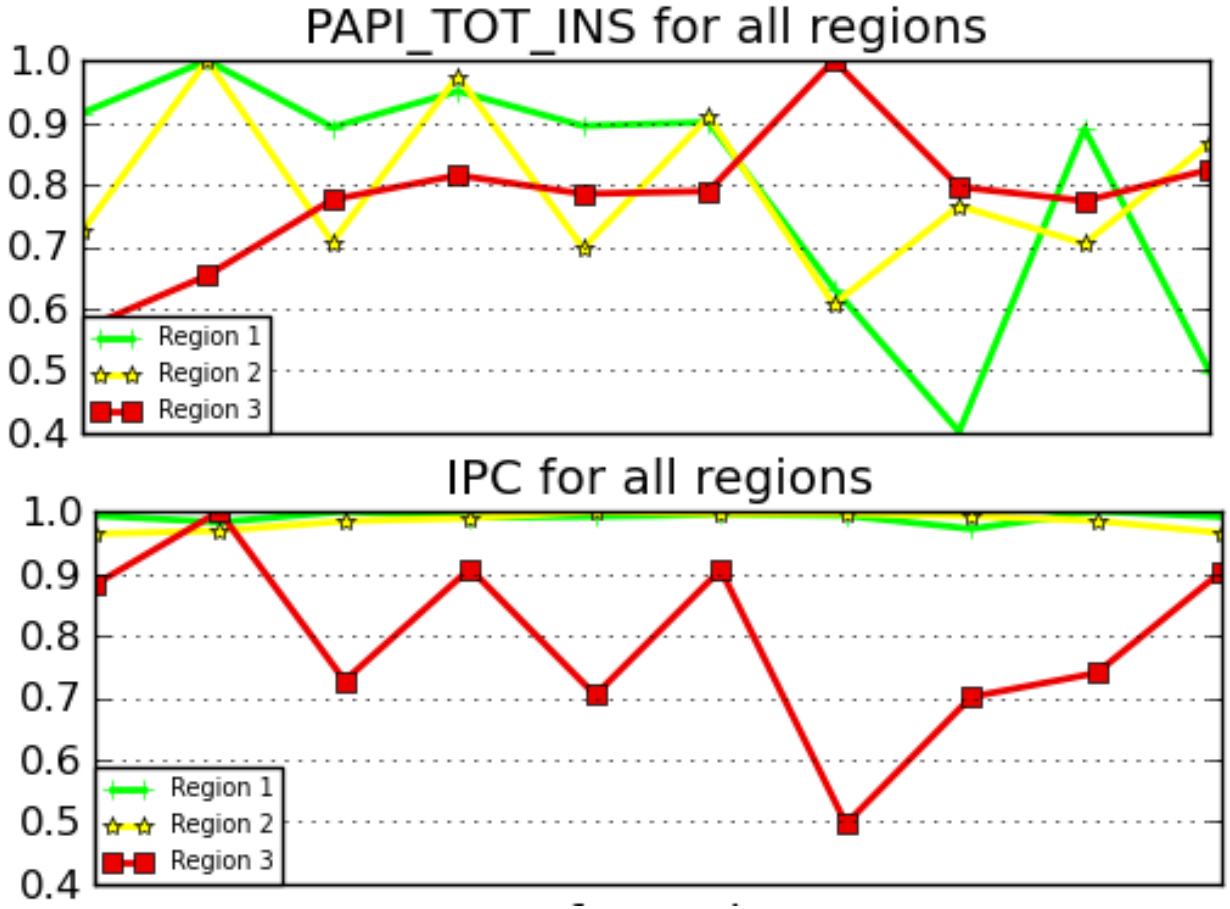
#Instructions vs IPC, 128 ranks, 8x4x4



#Instructions vs IPC, 128 ranks, 128x1x1

Metrics

- Instructions:
  - Drop in 64 ranks by the given input, no application problem!
  - Very similar between ranks (most imbalance around 5-10%)
- IPC:
  - Region 1. Computation: IPC keeps equal
  - Region 2 & 3. More imbalance if there is no distribution in the space.

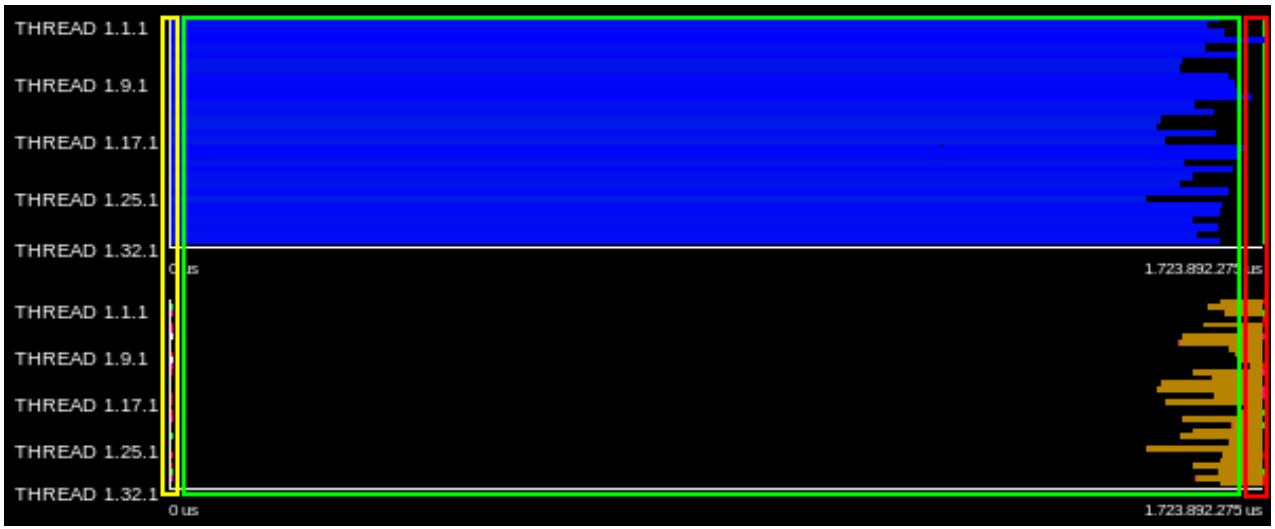


Overview

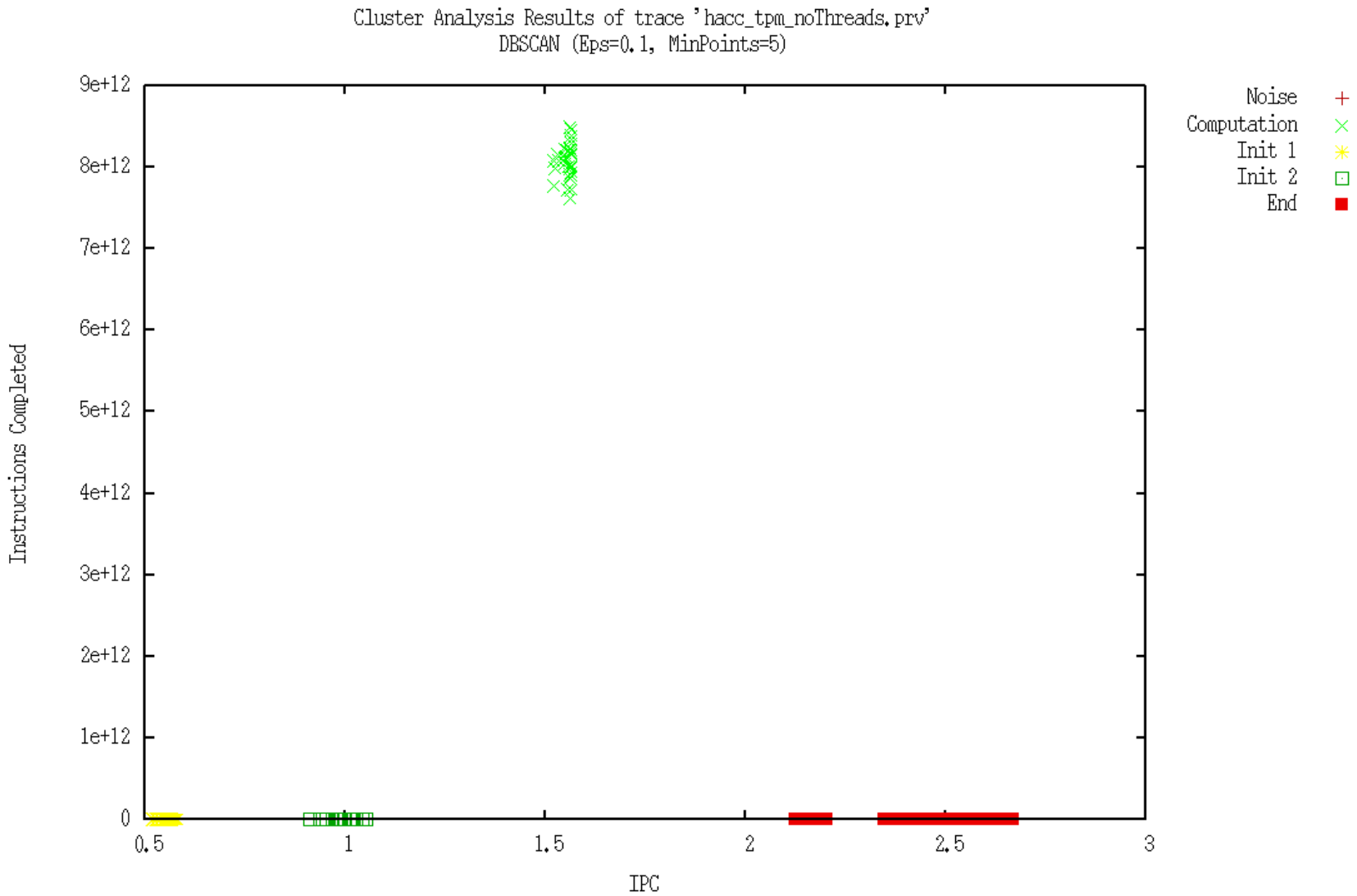
- 4x4x2 config
- 32 MPI Ranks

Phases

1. Init
2. Computation
3. End



Clustering


Cluster Analysis Results of trace 'hacc_tpm_noThreads.prv'
DBSCAN (Eps=0.1, MinPoints=5)
Instructions Completed
IPC
9e+12
8e+12
7e+12
6e+12
5e+12
4e+12
3e+12
2e+12
1e+12
0
0.5
1
1.5
2
2.5
3
Noise
Computation
Init 1
Init 2
End

Imbalance

- Cluster of computation shows vertical line
  $\rightarrow$ Difference in number of instructions
- Losing performance:

	Outside MPI
Avg	94,85%
Max	99,95%
Min	89,07%
Avg/Max	0,95%

4x4x2

	Outside MPI
Avg	92,14%
Max	99,93%
Min	86,70%
Avg/Max	0,95%

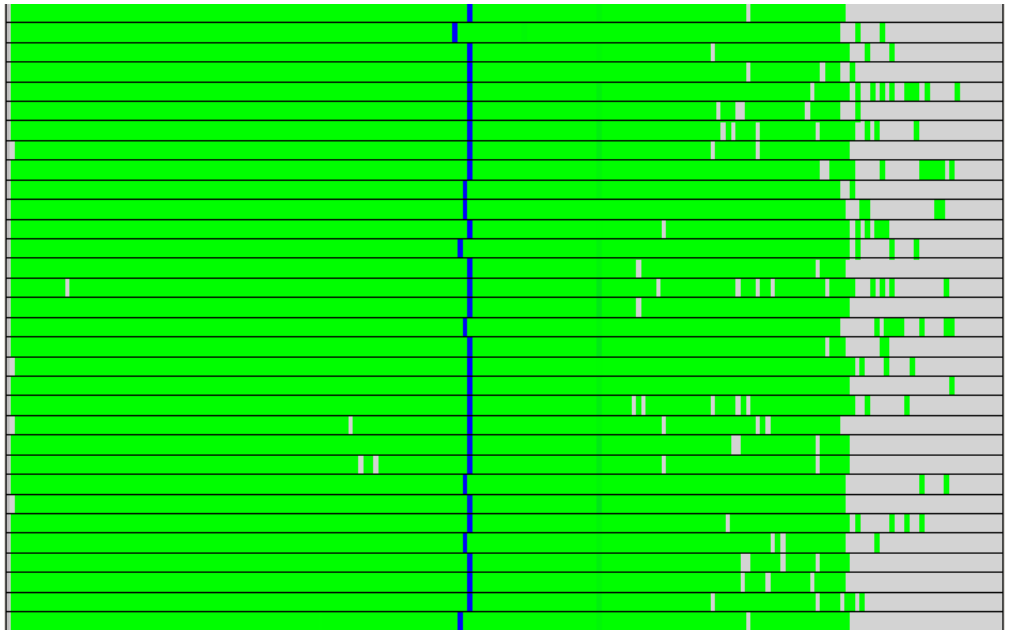
4x4x4

	Outside MPI
Avg	96,46%
Max	99,86%
Min	93,80%
Avg/Max	0,97%

128x1x1

IPC

- Computation phase with same IPC
- Blue phases are the important ones, computation phase

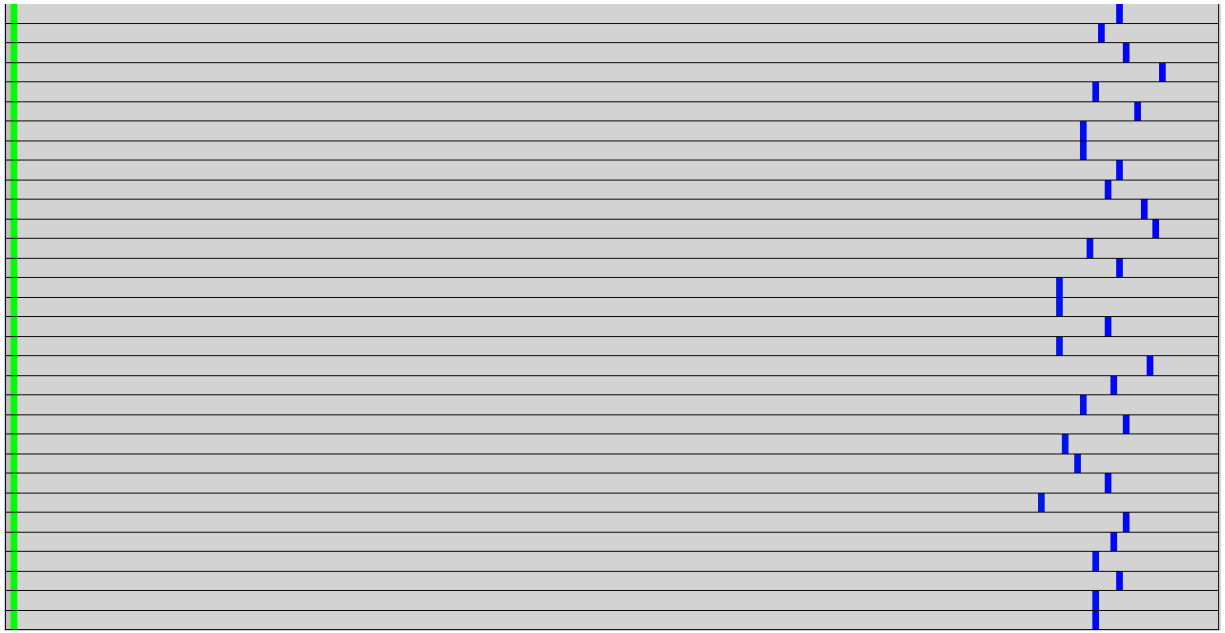


IPC Histogram

- Min. IPC: 1.50
- Max. IPC: 1.57

Instructions

- IPC = $\frac{\#Instructions}{\#Cycles}$
- Blue phases are the important ones, computation phase

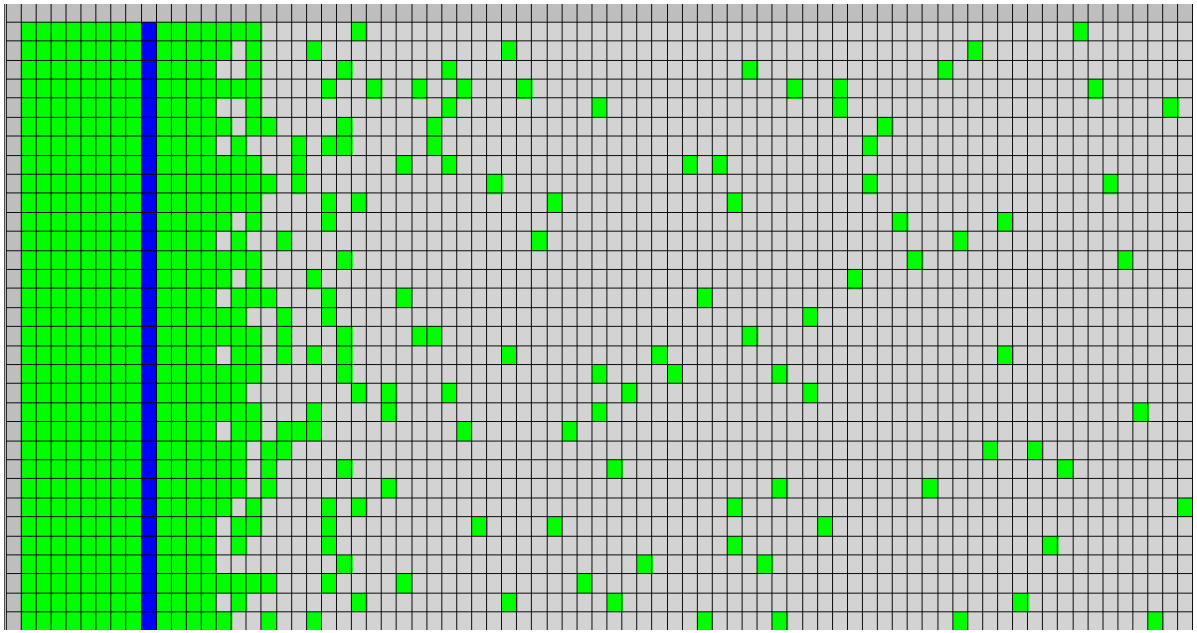


Instructions Histogram

- Min. Instructions: 7.5×10^{12}
- Max. Instructions: 8.5×10^{12}

Preemption

- Extra instructions come from preemption?
- Blue phases are the important ones, computation phase



Cycles per Us Histogram

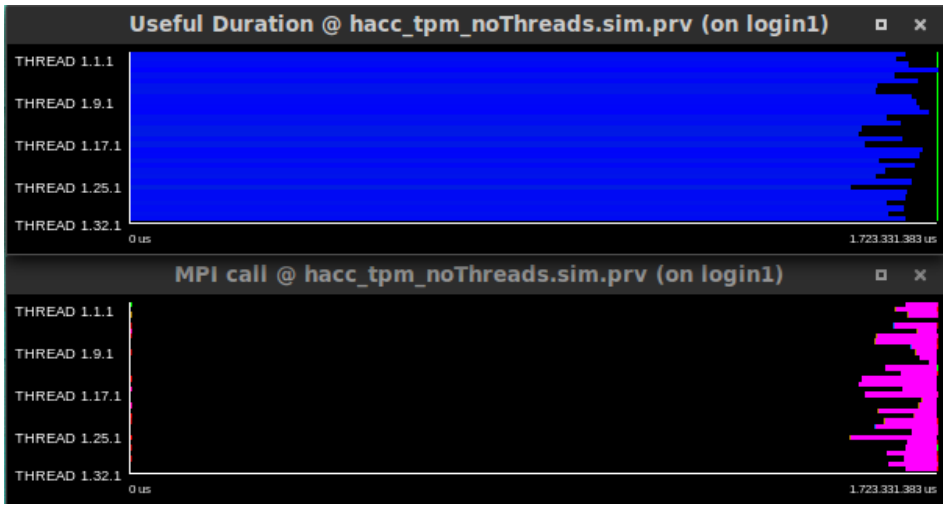
- No preemption in the computing phases

Network

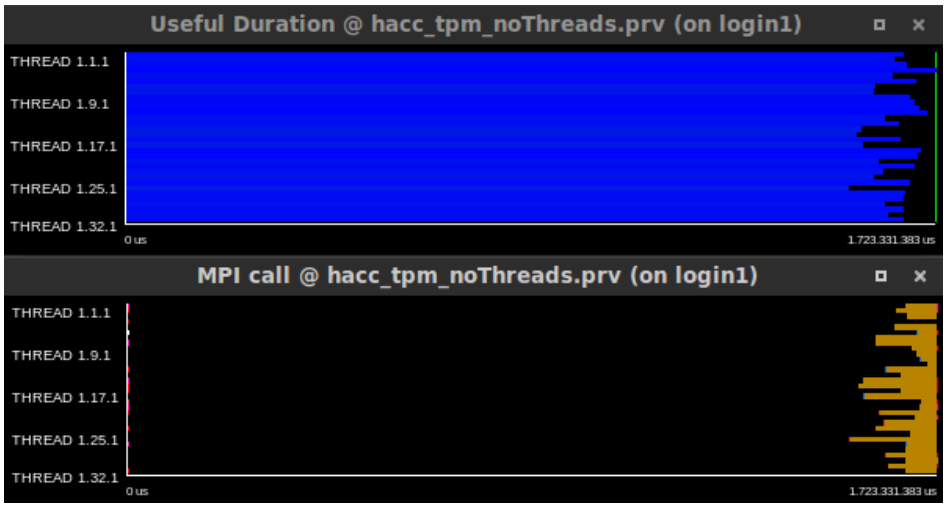
- Is the Network affecting the application?

Ideal network

Bandwidth: ∞ MB/s
Latency: 0s



Dimemas: ideal network execution



Real execution

Why less instructions

- 64 & 128 processes
  - Same input
  - Different virtual topology
  - Less instructions
- Focus on 128 processes case

Output

- 8x4x4
  Initial Particles 16777216
  - 131072 particles per process

  STEP 0: particles = 16777207

  STEP 1: particles = 16777215

  STEP 2: particles = 16777213

- 128x1x1
  Initial Particles 17307773
  - 135216 particles per process

  STEP 0: particles = 17332773

  STEP 1: particles = 17363514

  STEP 2: particles = 17404656

- Size of the generated traces:
  - 8x4x4: 200 MB
  - 128x1x1: 1 GB

Conclusions

- HACC is a bit imbalance (up to 10%)
- Seems to be cause of how the algorithm works
  - Input points are distributed among the different process
  - There are no hardware problems

- Possible solutions:
  - Try with a bigger problem
    → ¿better division of the problem?
  - Increase number of MPI process working
    → ¿better distribution of the problem?
  - Execute with more threads
    → dynamic scheduling could improve perfomance
  - Dynamic Load Balancing
    → assign more resources to those processes with more instructions

Problems

- Did not manage to get threads working → It could help to balance number of instructions
  - It seemed there was a race condition
- Spawned a lot of threads
  - There is a nested loop with OpenMP
  - With only OMP_NUM_THREADS=X, it will create more
  - OMP_THREAD_LIMIT=1 is needed

References

HACC

[1] Salman Habib et al. “HACC: Extreme Scaling and Performance Across Diverse Architectures”. In: **Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis**. SC ’13. Denver, Colorado: ACM, 2013, 6:1–6:10. ISBN: 978-1-4503-2378-9.

Tools

- Perfomance tools documentation: http://www.bsc.es/computer-sciences/performance-tools/documentation
- Constan documentation

Questions

Thank you!
Feel free to ask

Cristobal Ortega
cristobal.ortega@est.fib.upc.edu