www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Methodology to Predict Scalability of Parallel Applications

Claudia Rosas, Judit Giménez and Jesús Labarta
crosas@bsc.es

Barcelona, 4 May 2015

Outline

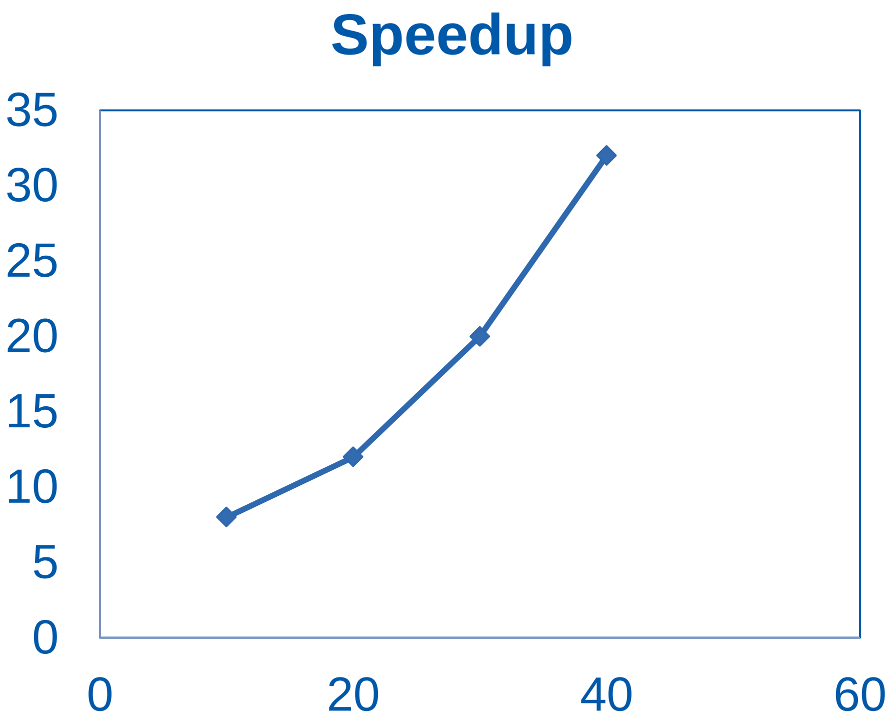
- Research Goal;
- Description of the methodology;
- Evaluated Scenarios and Results;
- Conclusions and future steps.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

RESEARCH GOAL

Research Goal

"Take advantage of existent techniques and tools to infer expected behavior of real parallel programs when executed at larger scale"

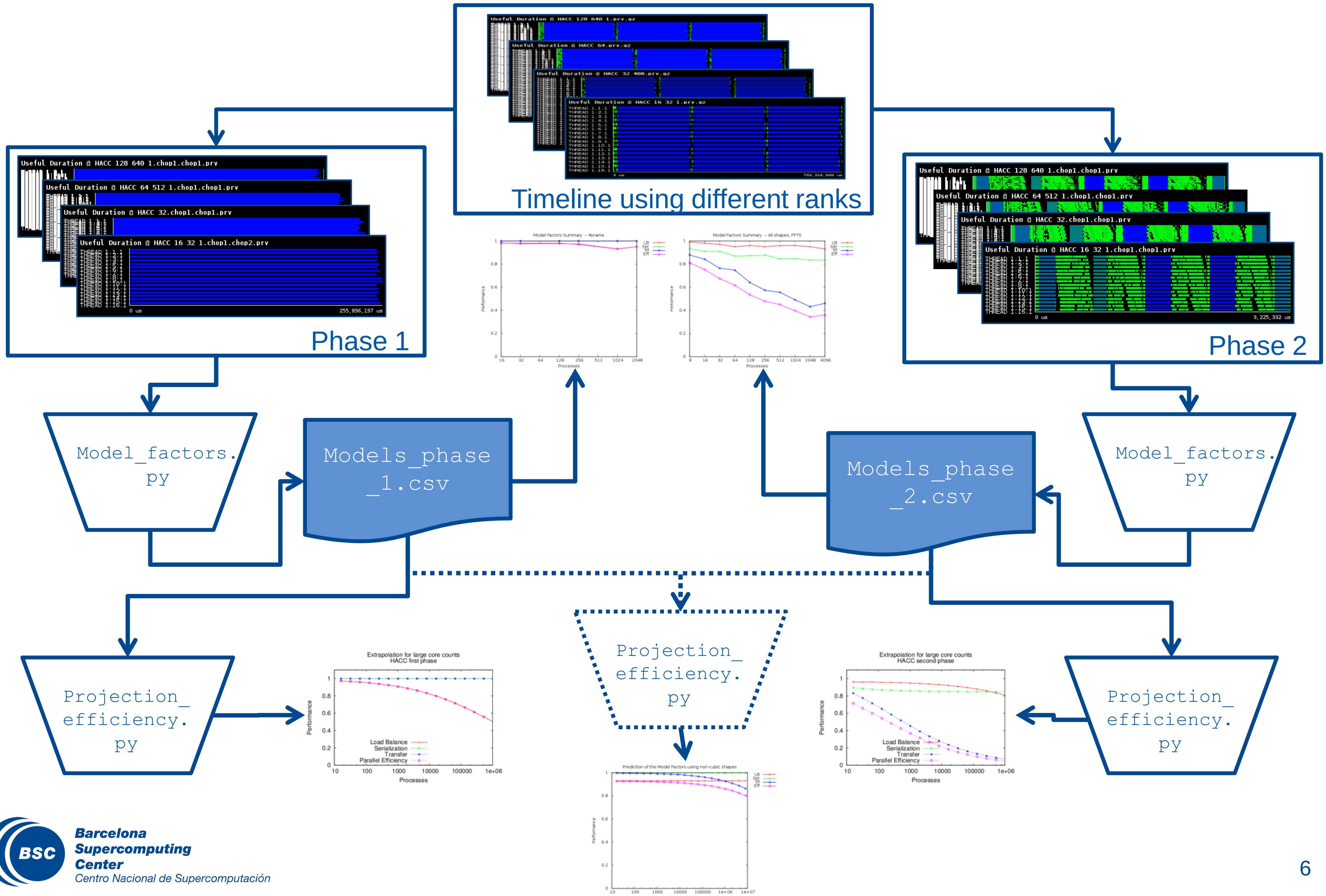


- Efficiency extrapolation
  - From few executions @ low core counts
- Fit based on
  - Reasonable fundamental behavioral models
  - Guided by observed internal application behavior (tools)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

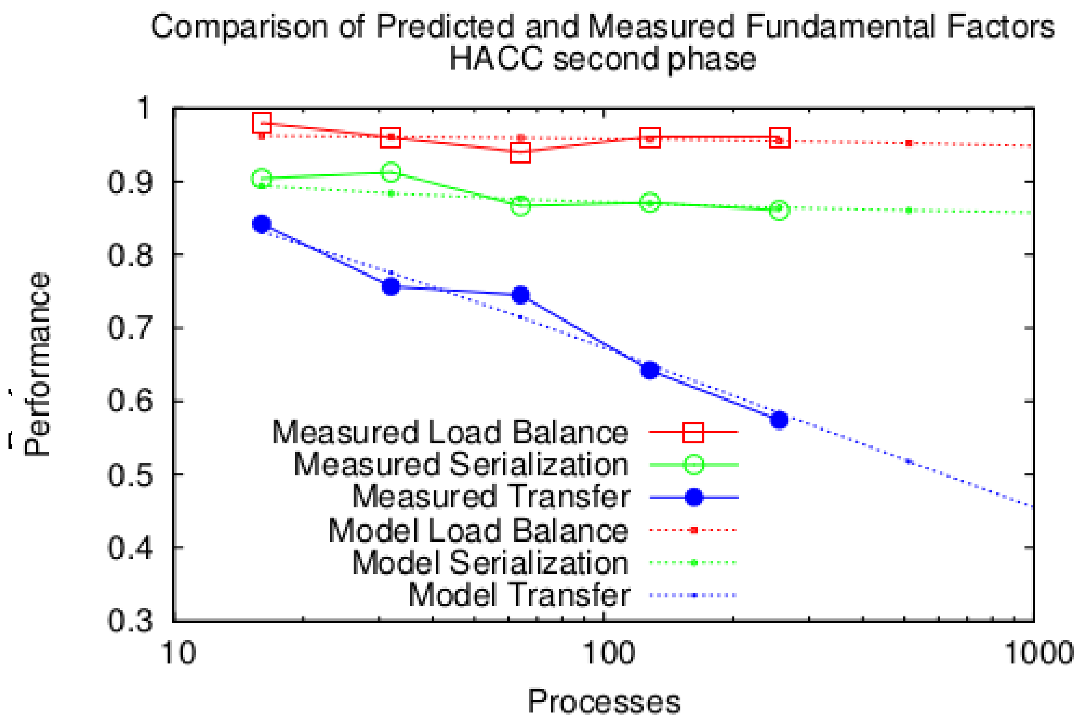
METHODOLOGY

Methodology to apply automatic framework



Scalability prediction

- Model based on the knowledge from the application:
  - Based on the parallel efficiency model
  - Constant behaviors
  - Different fittings
    - Amdahl’s Law
    - Pipeline
    - …
  - Extrapolation from few executions using a low core count
  - Fast and cheap!



Performance Analysis: Model Factors

Basic Analysis

– Quantitative summary of the performance of the identified phases;
– Measurements collected from real traces;
– Decomposes parallel efficiency as the product of normalized values between 0 (very bad) and 1 (perfect)

Parallel Efficiency:

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

Load Balance:

$$LB = \frac{\sum_i t_i}{P * \max(t_i)}$$

Serialization:

$$Ser = \max(ideal\ (eff_i))$$

Transfer:

$$Trf = \frac{Comm_{eff}}{Ser}$$

$$Comm_{eff} = \frac{\max(t_i)}{T}$$

T: phase time span

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

EXPERIMENTAL RESULTS

Parallel codes in strong and weak scaling

- CORAL Benchmarks: HACC, Nekbone and AMG2013
- CFD application AVBP
- Traces obtained in MareNostrum III, Juqueen, and Juropa
- Runs in weak and strong scaling
- Ranks used for each parallel code:

Weak Scaling	Ranks
HACC	16, 32, 64, 128, 256, 512, 1024, 2048, 4096
Nekbone	2, 4, 8, 16, 32, 64, 128, 256, 512

Strong Scaling	Ranks
AMG2013	32, 64, 96, 128, 192, 256, 384
AVBP*	16, 32, 64, 96, 128, 192, 256, 520, 768, 1024, 1040, 1280, 1536

*Traces were obtained in Juropa

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Rosas, C.; Giménez, J. & Labarta, J. *Scalability prediction from fundamental performance factors.* Supercomputing Frontiers and Innovations, 2014, 1, 4-19

HACC

- CORAL Benchmark, MareNostrum - 8 processes per node
  Indat:
    Number of Particles (NP) = 128 (128^3) particles per rank
    Number of Grids (NG) = NP
    Box size = Proportional to NP

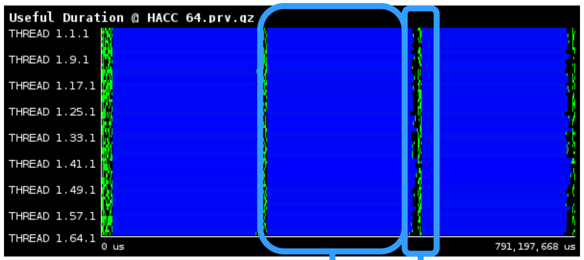
  Compiled using:
    impi/4.1.1.036, intel/13.1.0, MKL/13.1.3 and FFTW 3.3.3
    OpenMP support not used

Weak Scaling	Ranks
HACC	**16, 32, 64, 128, 256**, 512, 1024, 2048, 4096

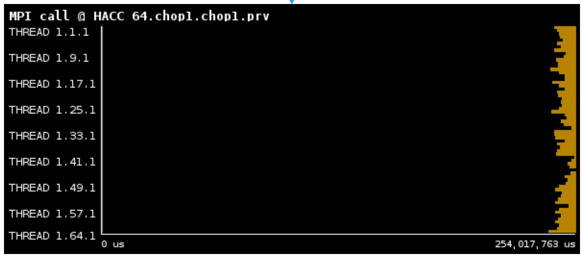
Identify structure (HACC)

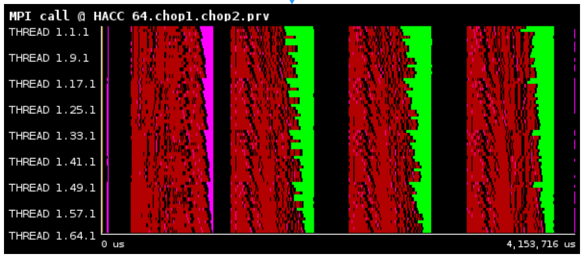
Original timeline

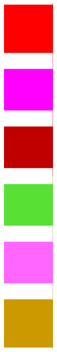
64 ranks



MPI calls





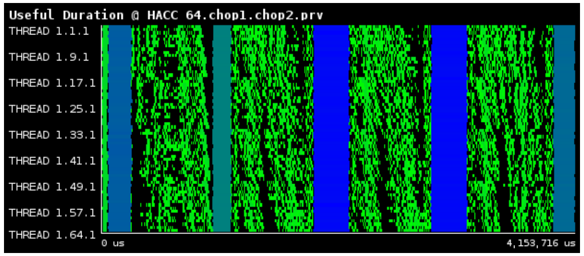


- MPI_Isend
- MPI_Irecv
- MPI_Wait
- MPI_Waitall
- MPI_Allreduce
- MPI_Cart_create

Phase 1 – 1it

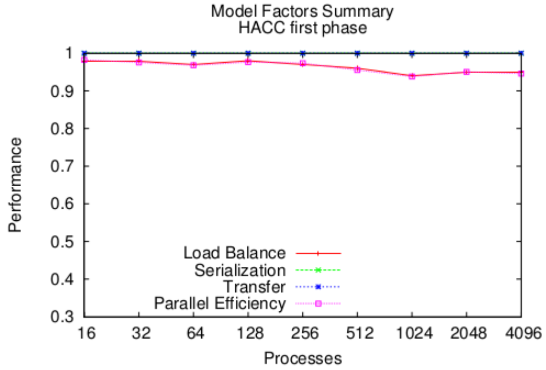
Phase 2 – 1it

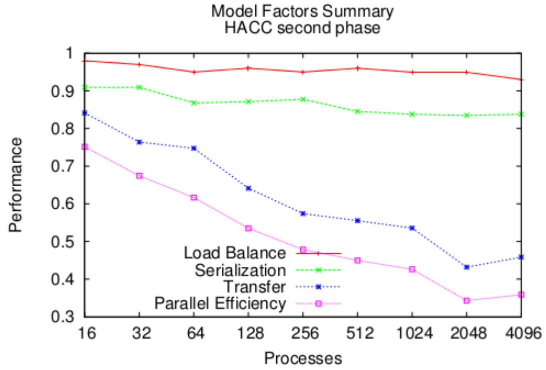
Useful duration



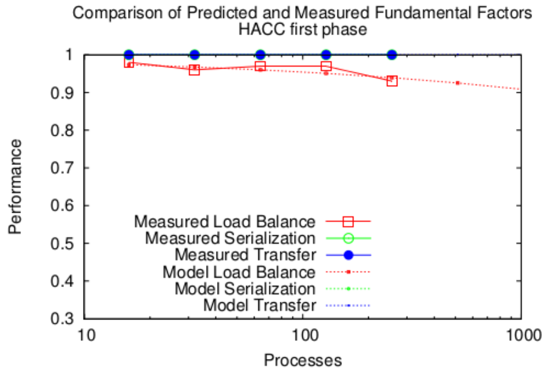
Efficiency model and extrapolation

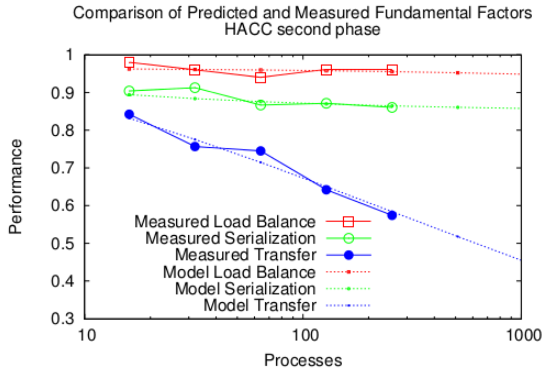
measured factors



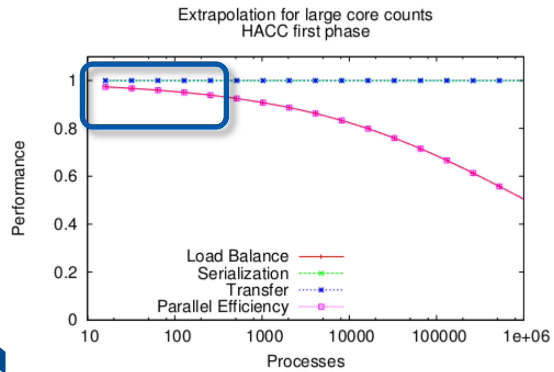


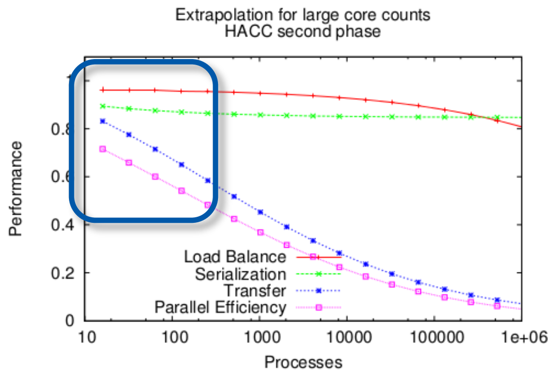
fitting



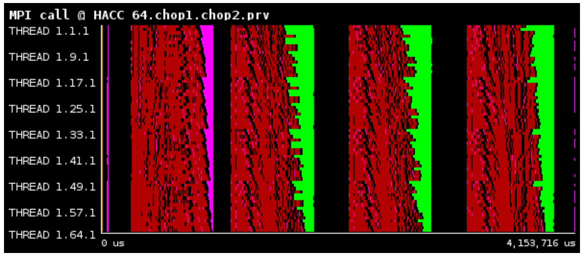


extrapolation





Phase 1 – 1it

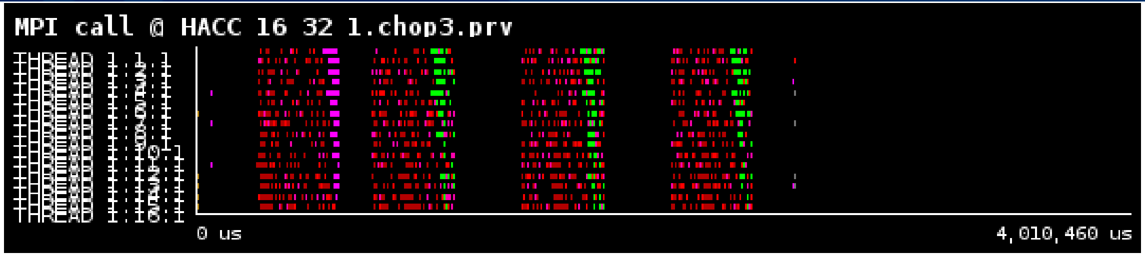


Phase 2 – 1it

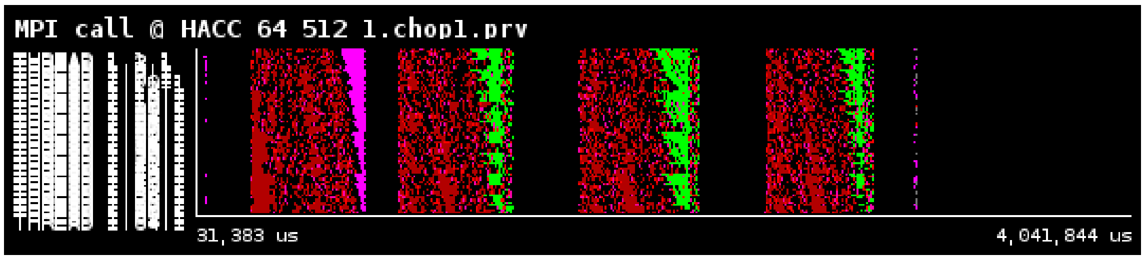
Fitting:
Serialization →Pipeline
Transfer → Amdahl
LB → Amdahl

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

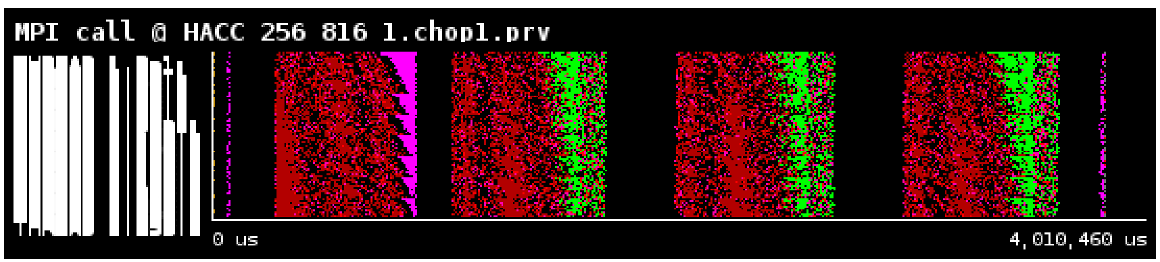
Scalability of communication (phase 2)



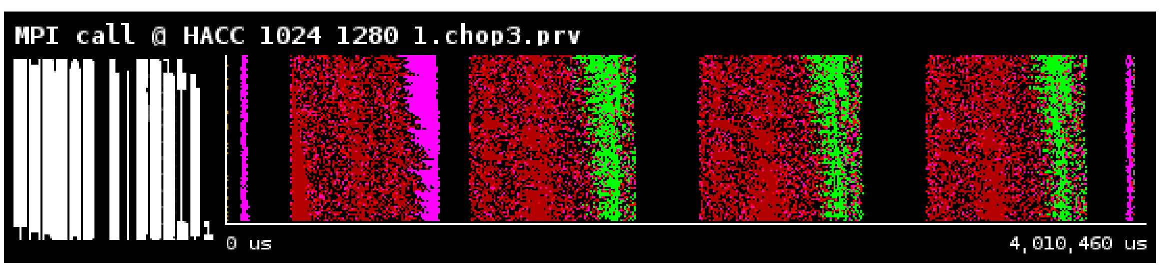
16 MPI



64 MPI

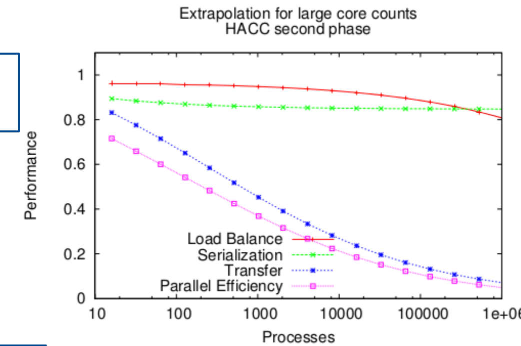


256 MPI



1024 MPI

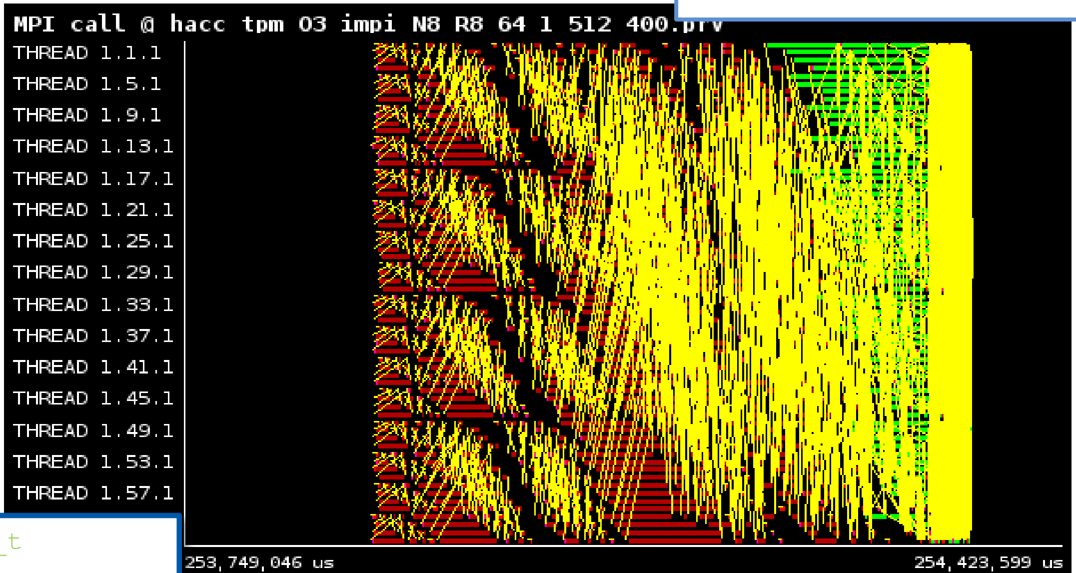
- MPI_Isend
- MPI_Irecv
- MPI_Wait
- MPI_Waitall
- MPI_Allreduce
- MPI_Cart_create

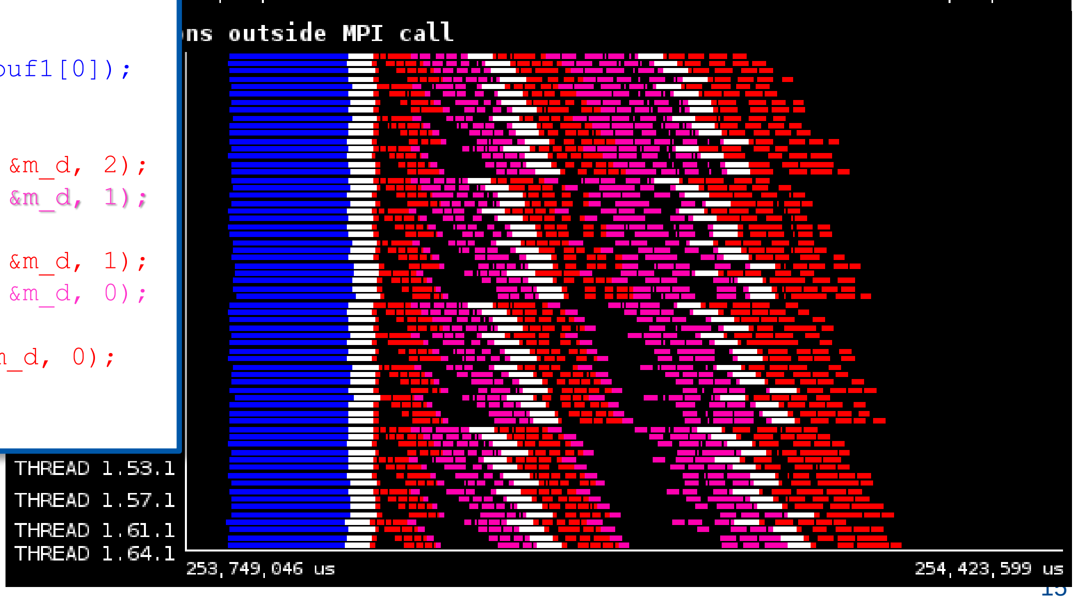


Main functions - Comm phase (64 MPIs)

Functions inside solve backward gradient

All communications





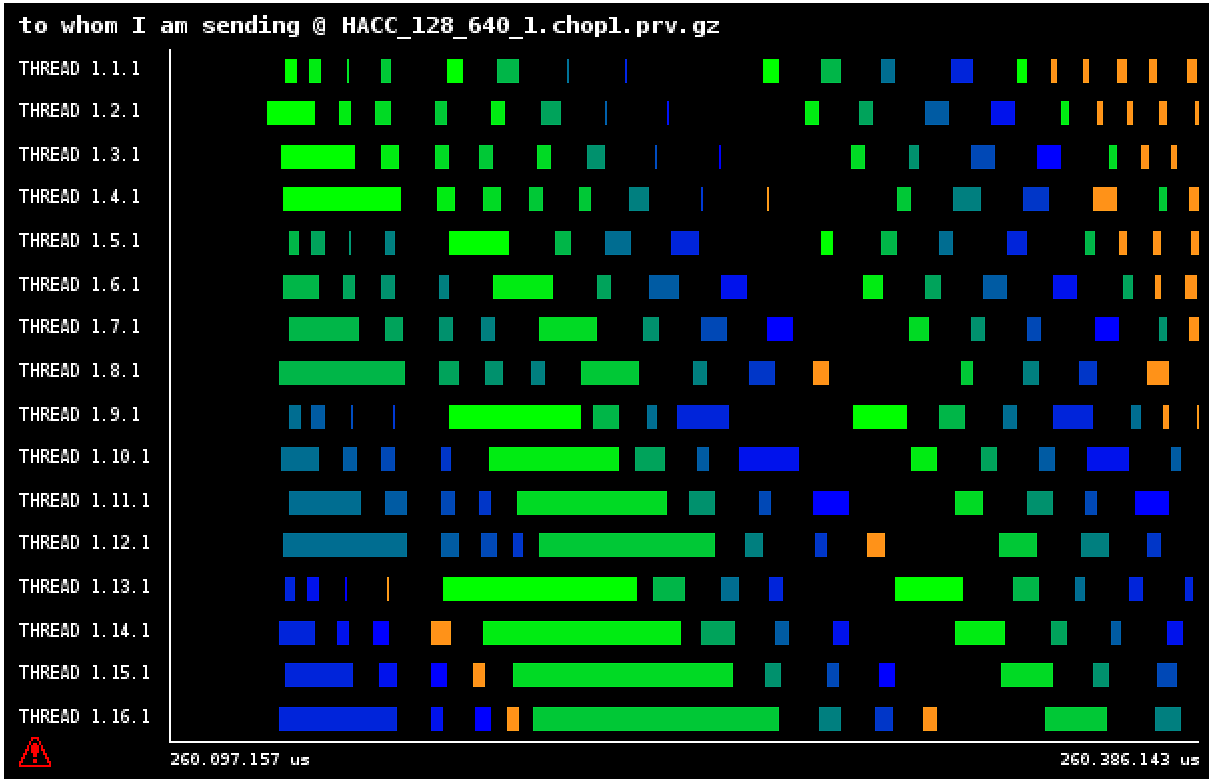

```
 void backward_solve_gradient(int axis, complex_t
*grad_phi)
  {
    kspace_solve_gradient(axis, &m_buf2[0], &m_buf1[0]);
#if defined(FFTW3) && defined(PENCIL)

    distribution_2_to_3(&m_buf3[0], &m_buf1[0], &m_d, 2);
    distribution_3_to_2(&m_buf1[0], &m_buf3[0], &m_d, 1);

    distribution_2_to_3(&m_buf1[0], &m_buf3[0], &m_d, 1);
    distribution_3_to_2(&m_buf3[0], &m_buf1[0], &m_d, 0);

    distribution_2_to_3(&m_buf3[0], grad_phi, &m_d, 0);
…
  }
```

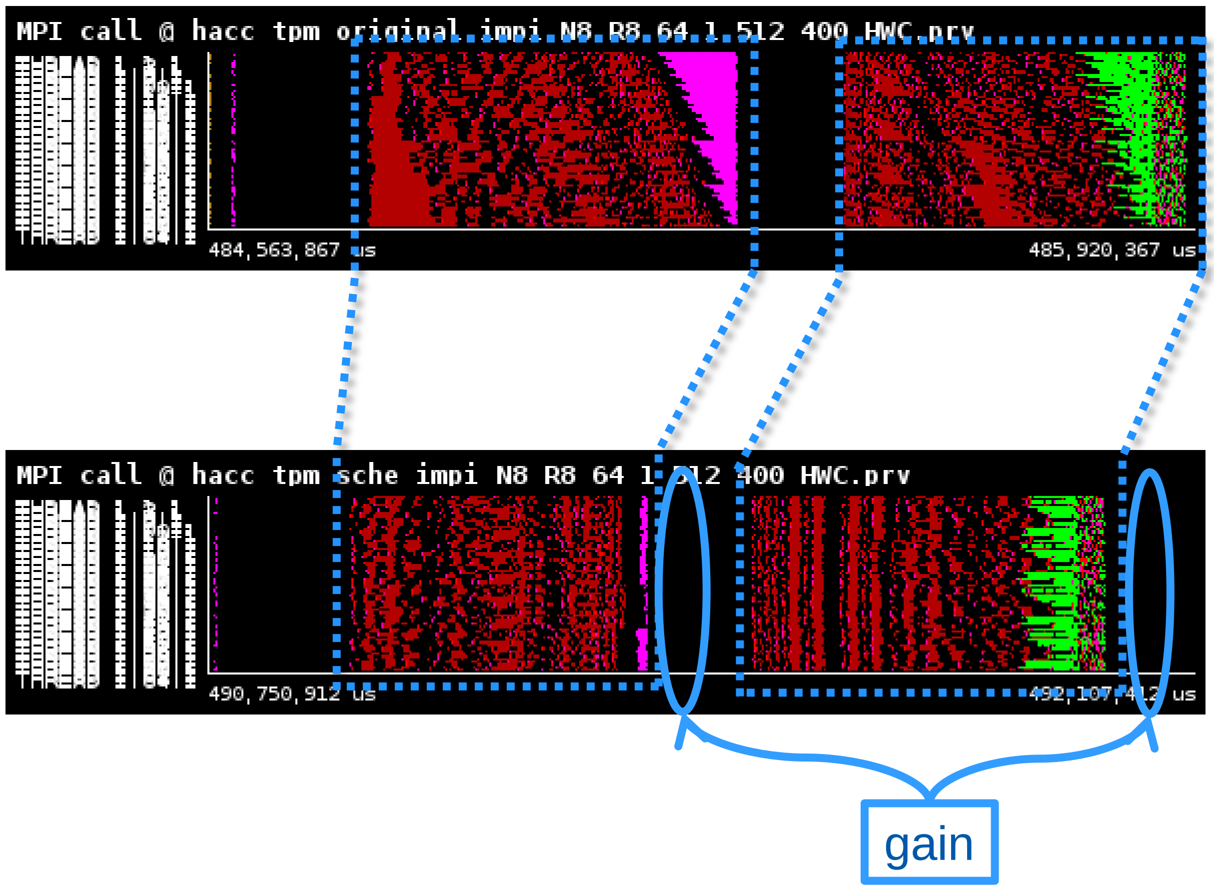
Original communication pattern



Communication reschedule

Original

Rescheduled



Nekbone executed in Juqueen

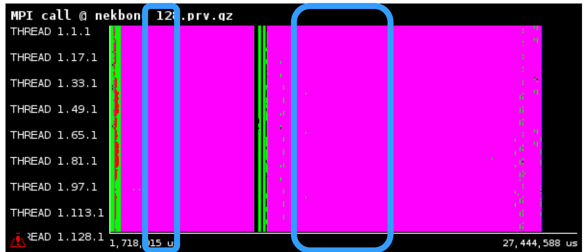
- CORAL Benchmark: Nekbone
- Traces obtained in Juqueen
- Runs in weak scaling
- Small Problem Size: 64 particles per process

Weak Scaling Runs	Ranks
Traces	**64, 128, 256, 512, 1024**, 2048, 4096
Times	64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144

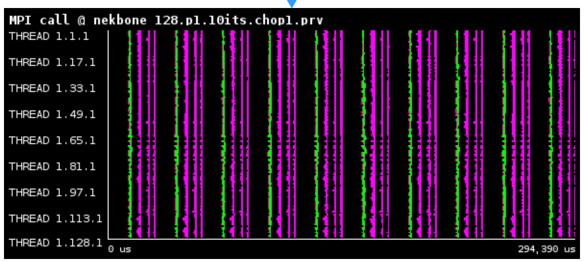
Identify structure (Nekbone)

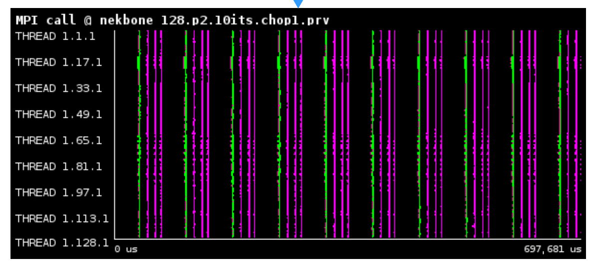
Original timeline

128 ranks



MPI calls



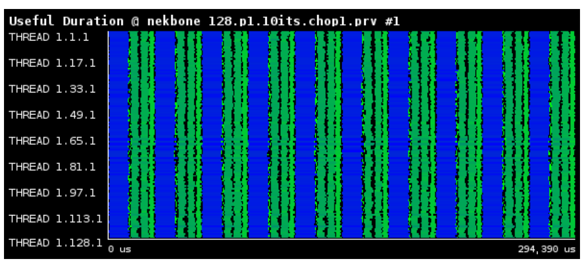


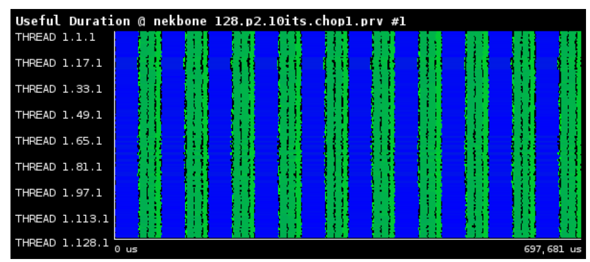
- MPI_Isend
- MPI_Irecv
- MPI_Wait
- MPI_Waitall
- MPI_Allreduce
- MPI_Cart_create

Phase 1 – 10its

Phase 2 – 10its

Useful duration

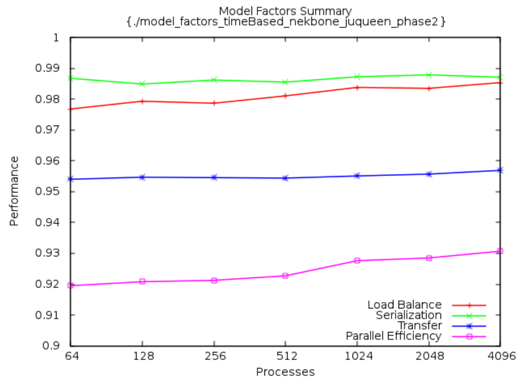


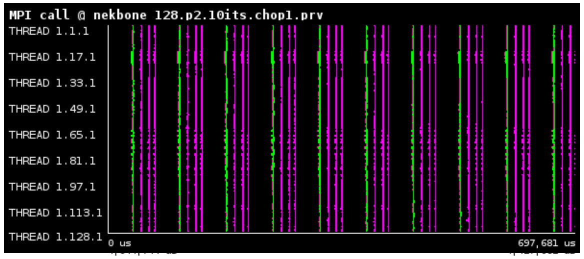


Factors and extrapolation

measured factors

4096 processes

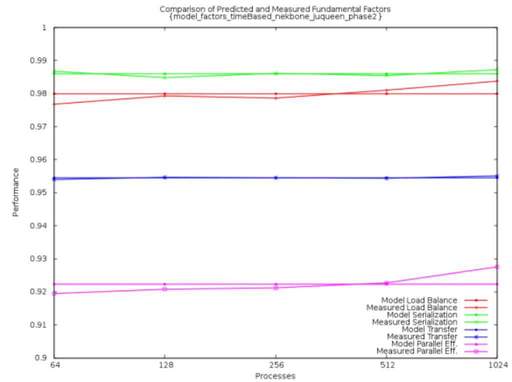




Phase 2 – 10its

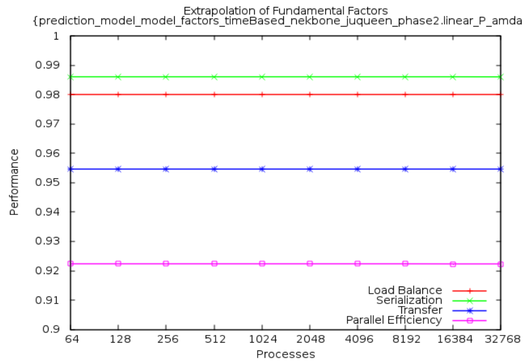
fitting

1024 processes



extrapolation

32768 processes

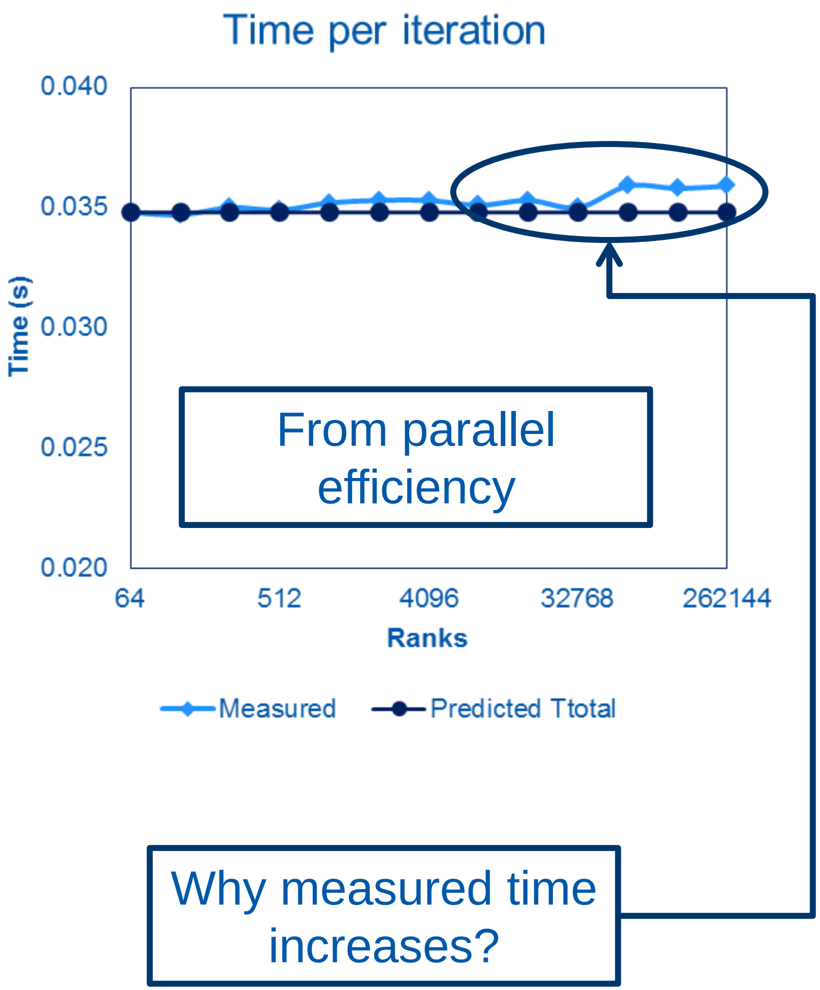


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Predicting execution time

- Expected execution time calculated as the sum of the computational phase plus communication phase;

- From calculated efficiency one can expect a constant time per iteration when increasing the core count.

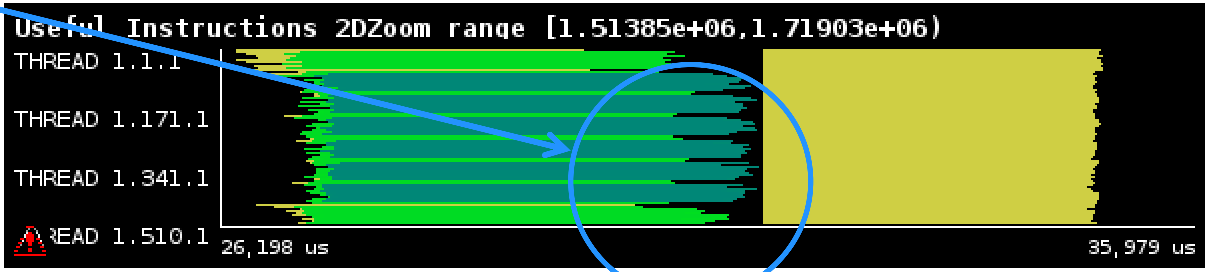


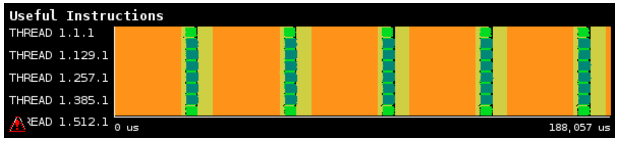
Why the time per iteration increases?

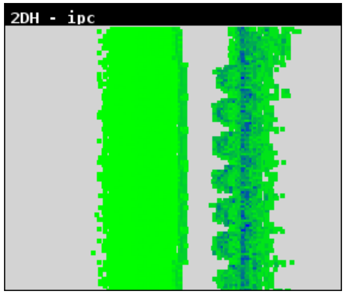
Load Imbalance

Computational region between communications

512

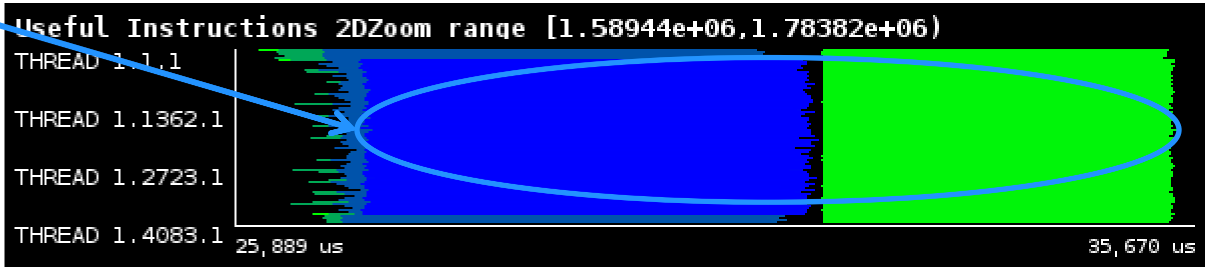


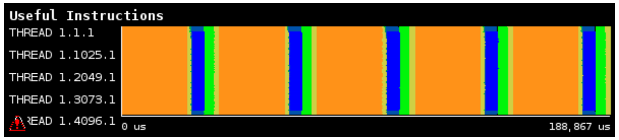


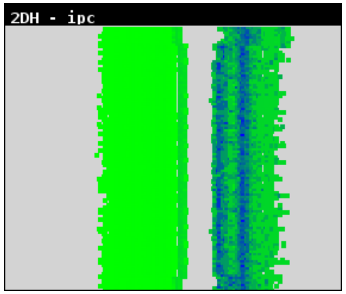


More Instructions

4096







Barcelona Supercomputing Center
Centro Nacional de Supercomputación

CONCLUSIONS AND FUTURE STEPS

Conclusions and Future Steps

- Current methodology serves:
  - To easily collect performance measurements in parallel applications;
  - To identify potential issues when scaling in a fast and not-expensive way;
  - To generate good-quality extrapolations from runs with low number of cores;
  - To improve the parallel application, because the factors point out where the performance is being lost.

- Current work and future steps:
  - The sensibility to noise in the machine;
    - Reduce the effect of noise
      - In the network: using Dimemas
      - In the computation: eliminate preemptions in the trace translation process
    - Use multiple instances
      - “Average”: integrated in analytics script
      - Eliminate outliers: manual experiments
  - Include machine noise as an additional factor in the model.

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you! Questions?

For further information please contact
crosas@bsc.es