Understanding applications using the BSC performance tools

Judit Gimenez (judit@bsc.es), Lau Mercadal

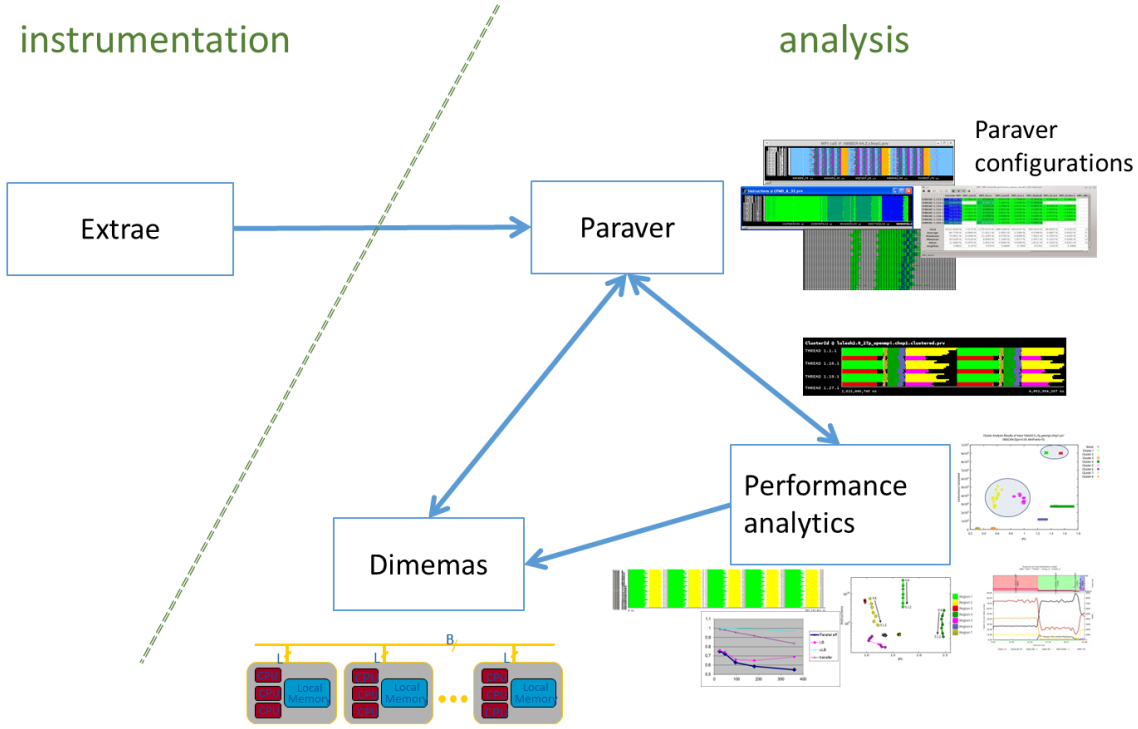
Barcelona Supercomputing Center

Humans are visual creatures

- Films or books? PROCESS
  - Two hours vs. days (months)
- Memorizing a deck of playing cards STORE
  - Each card translated to an image (person, action, location)
- Our brain loves pattern recognition IDENTIFY
  - What do you see on the pictures?

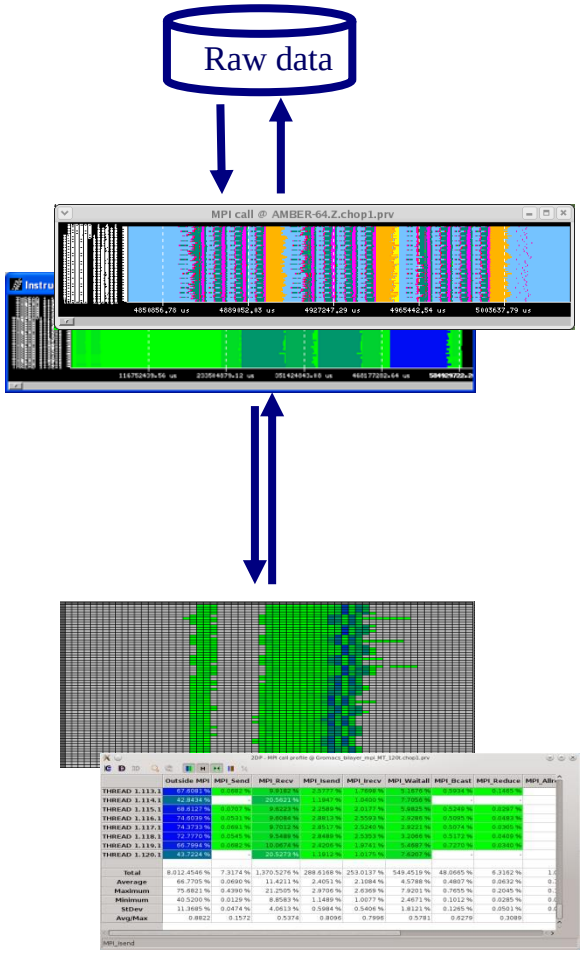
Our tools

- 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
  - Key factors
  - Visual analysis
  - Intelligence: Performance Analytics
  - Behavioral structure vs. syntactic structure



Paraver

Paraver: Performance data browser



Timelines

2/3D tables
(Statistics)

Trace visualization/analysis

+ trace manipulation

Goal = Flexibility

No semantics

Programmable

Comparative analyses

Multiple traces

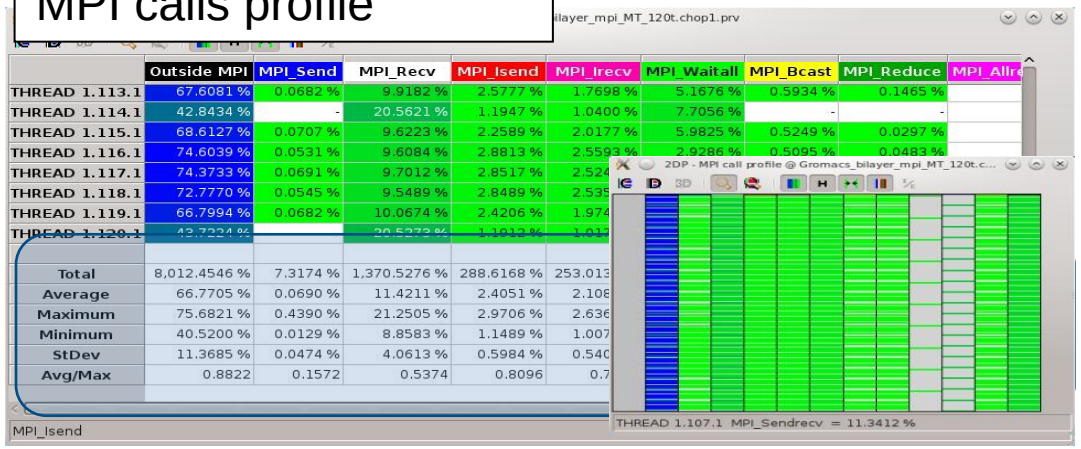
Synchronize scales

Tables: Profiles, histograms, correlations

- From timelines to tables

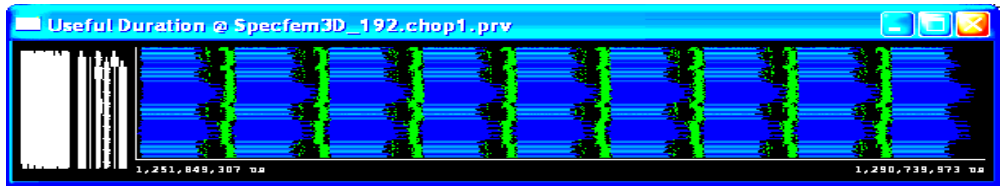
MPI calls

MPI calls profile

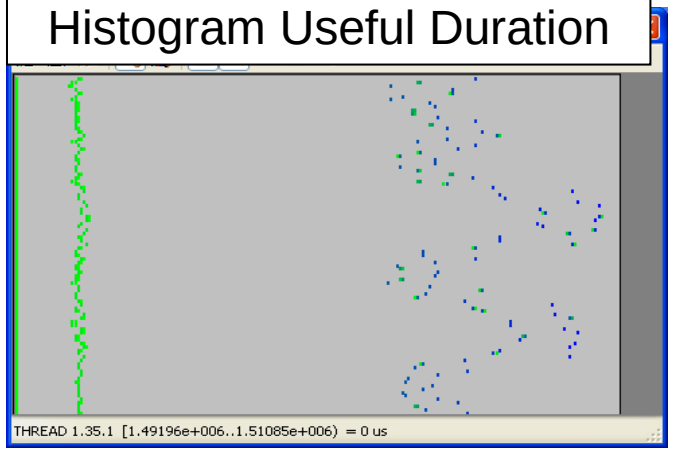


	Outside MPI	MPI_Send	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Bcast	MPI_Reduce
THREAD 1.113.1	67.6081 %	0.0682 %	9.9182 %	2.5777 %	1.7698 %	5.1676 %	0.5934 %	0.1465 %
THREAD 1.114.1	42.8434 %	-	20.5621 %	1.1947 %	1.0400 %	7.7056 %	-	-
THREAD 1.115.1	68.6127 %	0.0707 %	9.6223 %	2.2589 %	2.0177 %	5.9825 %	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 %	2.524			
THREAD 1.118.1	72.7770 %	0.0545 %	9.5489 %	2.8489 %	2.535			
THREAD 1.119.1	66.7994 %	0.0682 %	10.0674 %	2.4206 %	1.974			
THREAD 1.120.1	43.7234 %		20.5273 %	1.1012 %	1.013			
Total	8,012.4546 %	7.3174 %	1,370.5276 %	288.6168 %	253.013			
Average	66.7705 %	0.0690 %	11.4211 %	2.4051 %	2.108			
Maximum	75.6821 %	0.4390 %	21.2505 %	2.9706 %	2.636			
Minimum	40.5200 %	0.0129 %	8.8583 %	1.1489 %	1.007			
StDev	11.3685 %	0.0474 %	4.0613 %	0.5984 %	0.540			
Avg/Max	0.8822	0.1572	0.5374	0.8096	0.7			

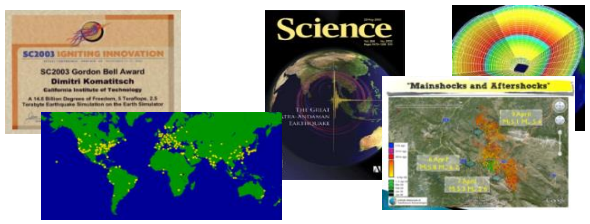
Useful Duration



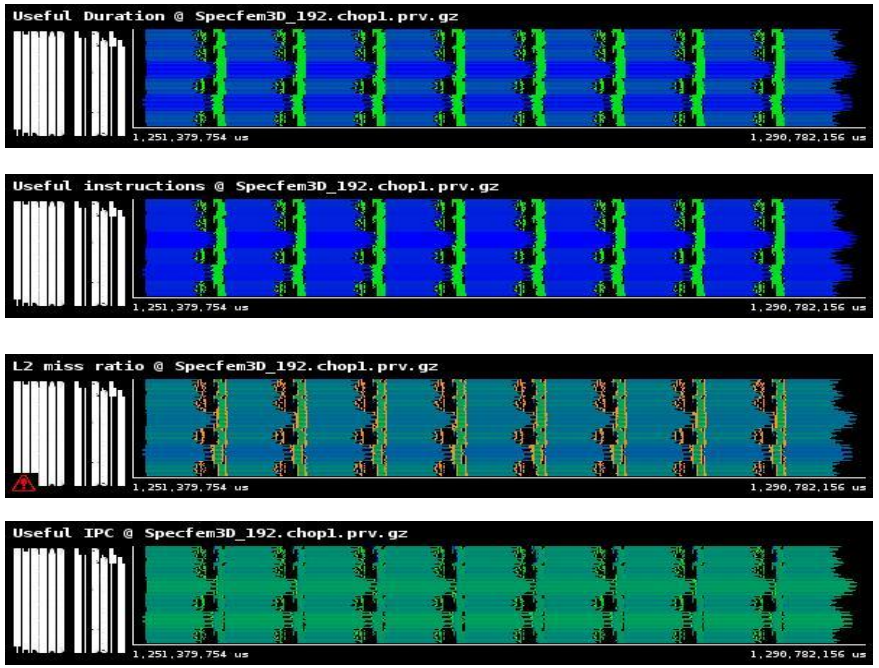
Histogram Useful Duration

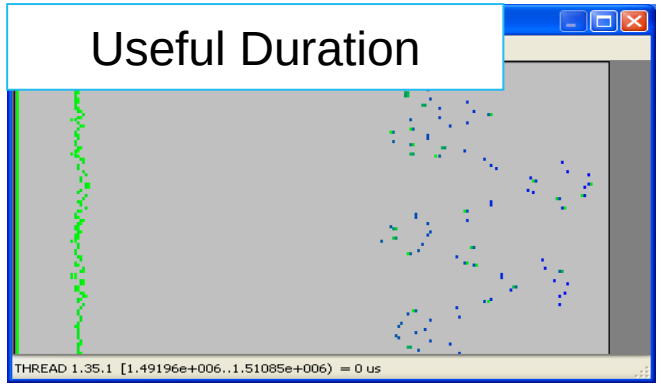


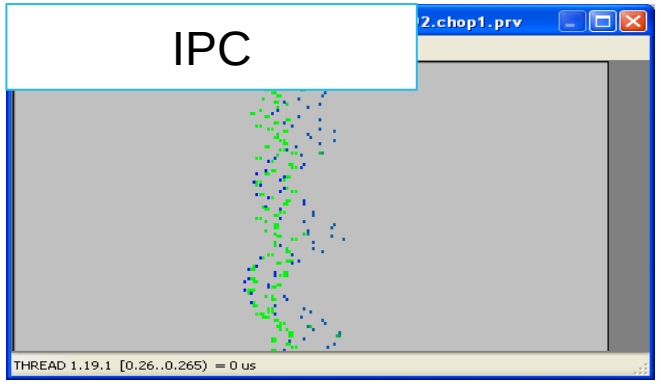
Analyzing variability through histograms and timelines

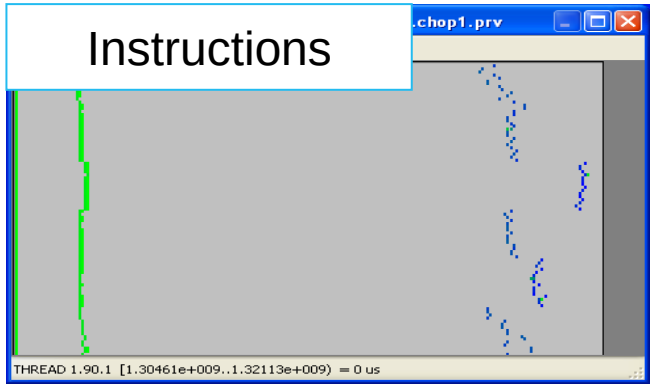


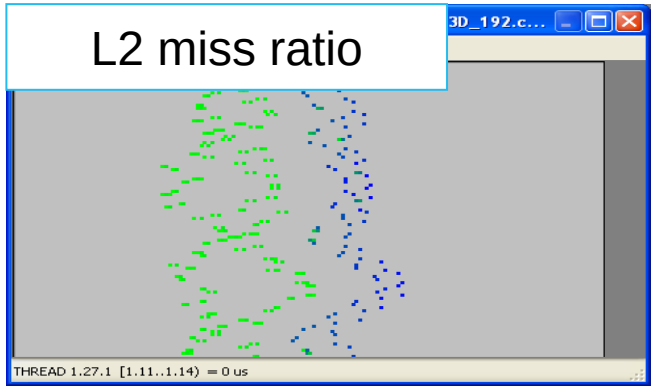
SPECFEM3D





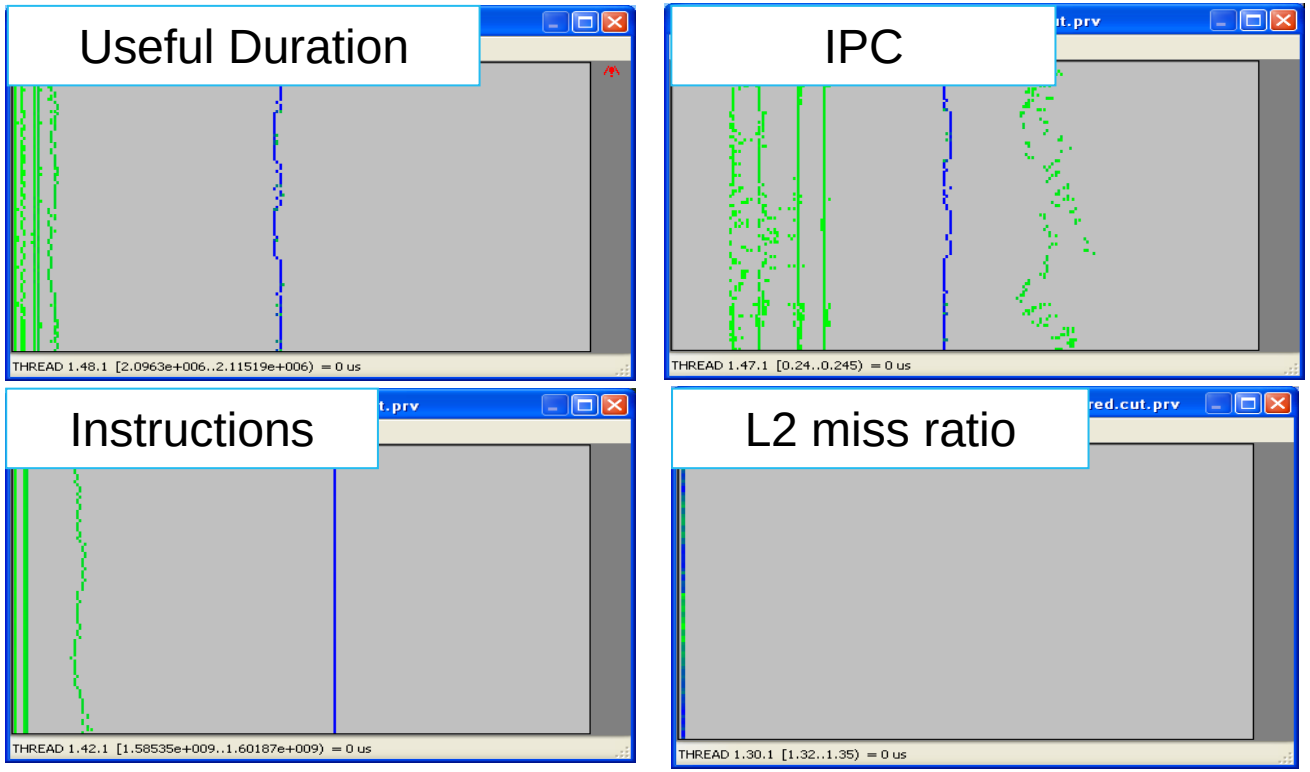






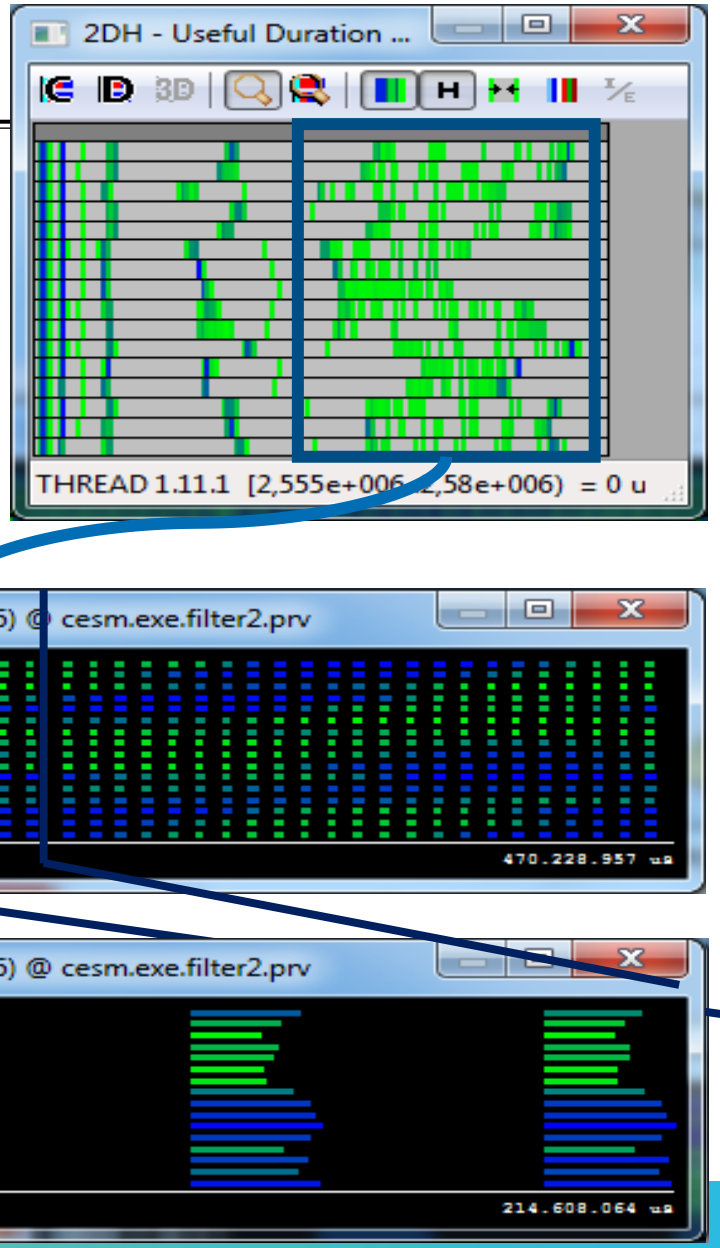
Analyzing variability through histograms and timelines

- By the way: six months later ….



Variability … is everywhere

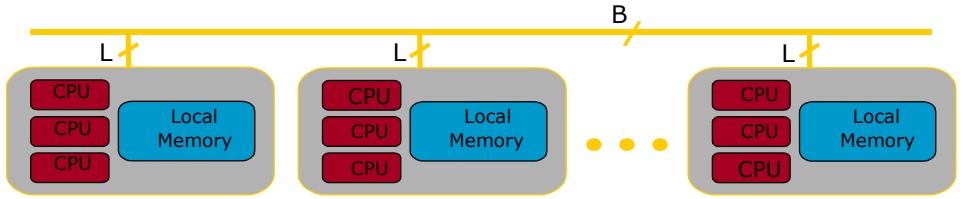
- CESM: 16 processes, 2 simulated days
- Histogram useful computation duration shows high variability
- How is it distributed?
- Dynamic imbalance
  - In space and time
  - Day and night

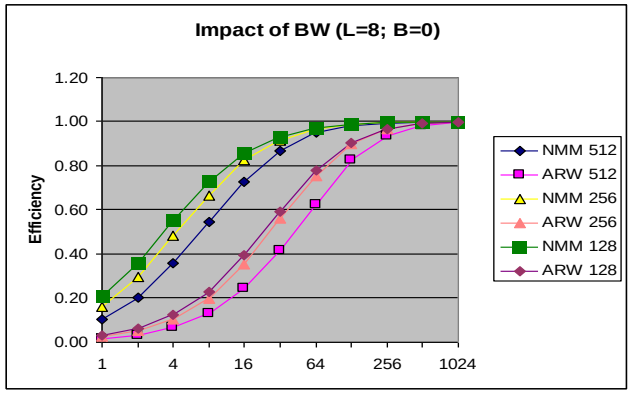


Dimemas

Dimemas: Coarse grain, Trace driven simulation

- Simulation: Highly non linear model
  - MPI protocols, resources 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 of an app. equally sensitive to network?
- MPI sanity check
  - Modeling nominal
- Paraver – Dimemas tandem
  - Analysis and prediction
  - What-if from selected time window

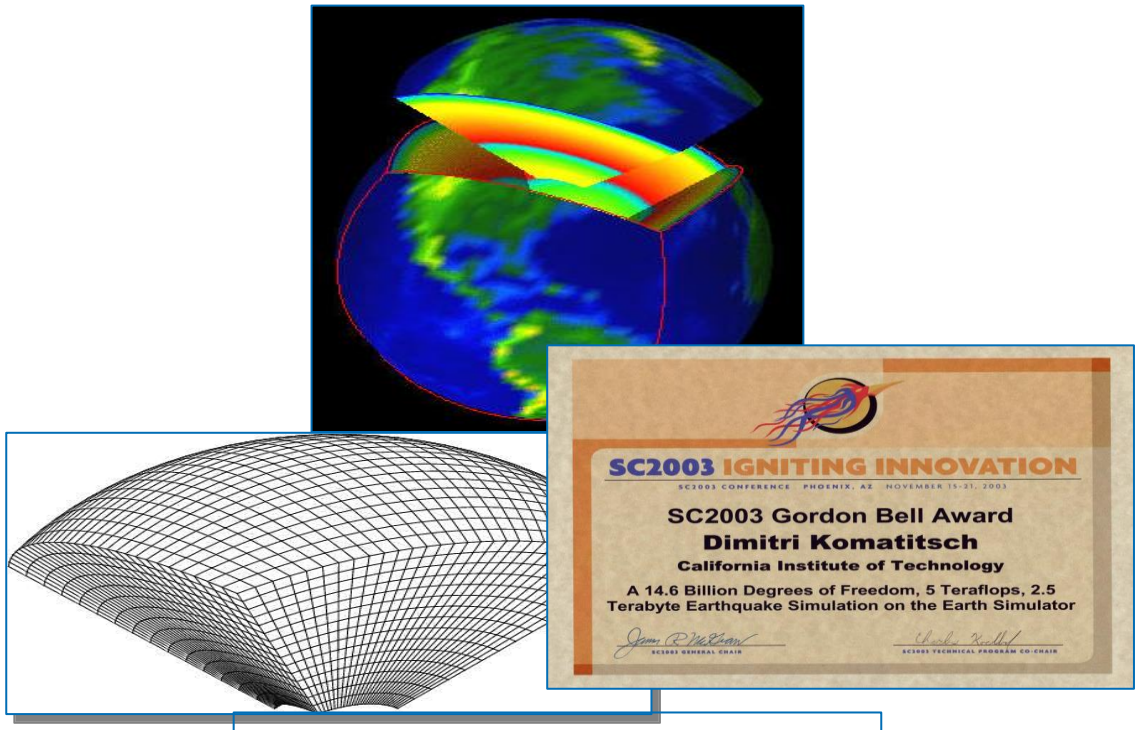




Detailed feedback on simulation (trace)

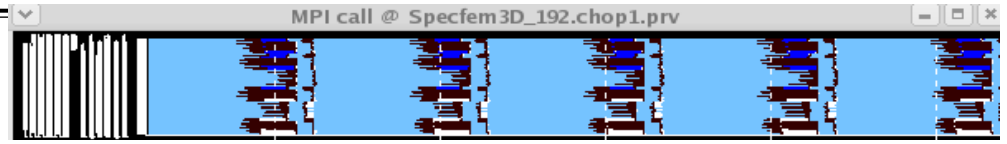
Would I will benefit from asynchronous communications?

- SPECFEM3D



Courtesy Dimitri Komatitsch

Real



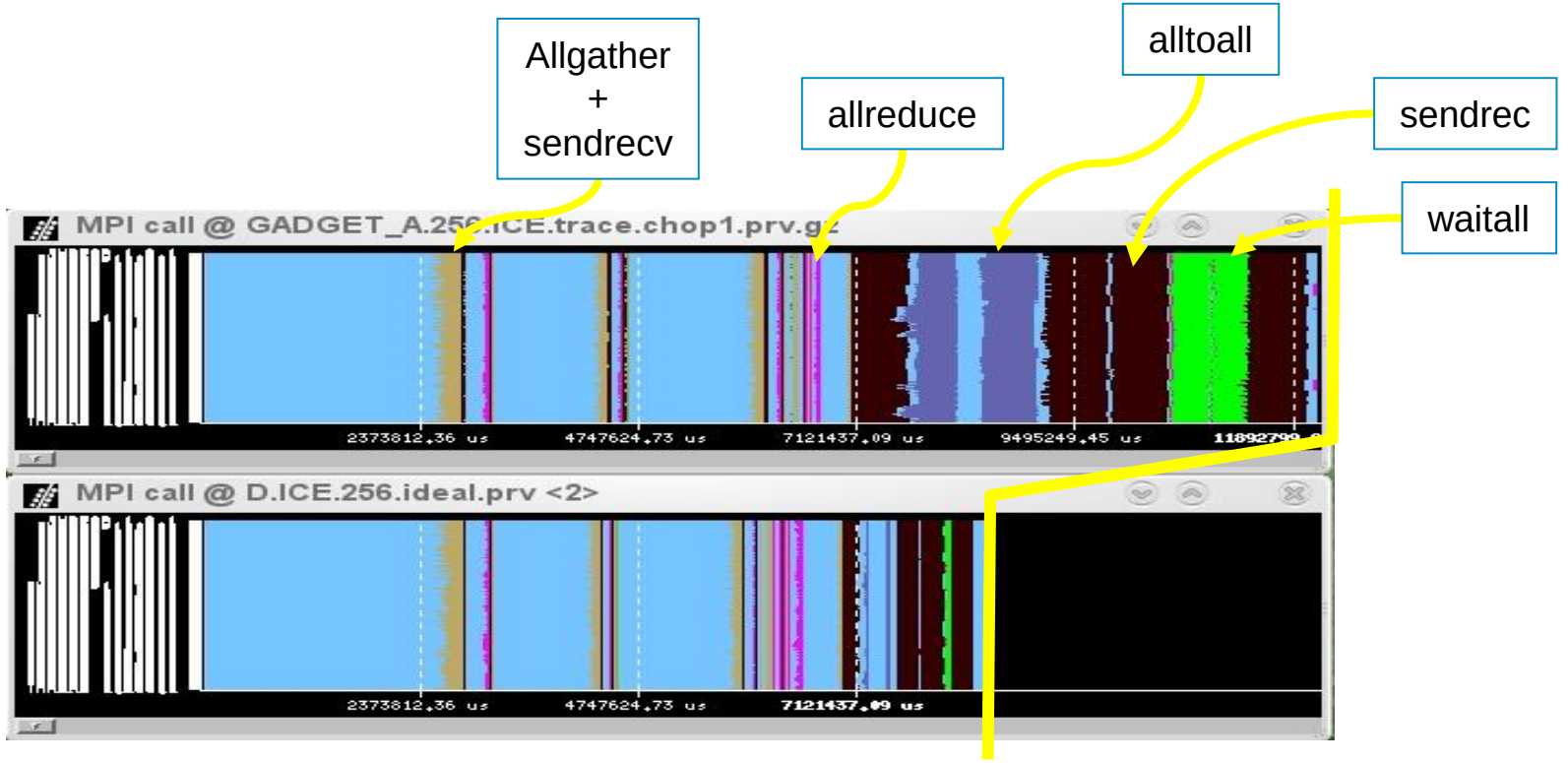
Ideal machine

- The impossible machine: BW = ∞, L = 0
  - Actually describes/characterizes Intrinsic application behavior
    - Load balance problems?
    - Dependence problems?

GADGET @ Nehalem cluster
256 processes

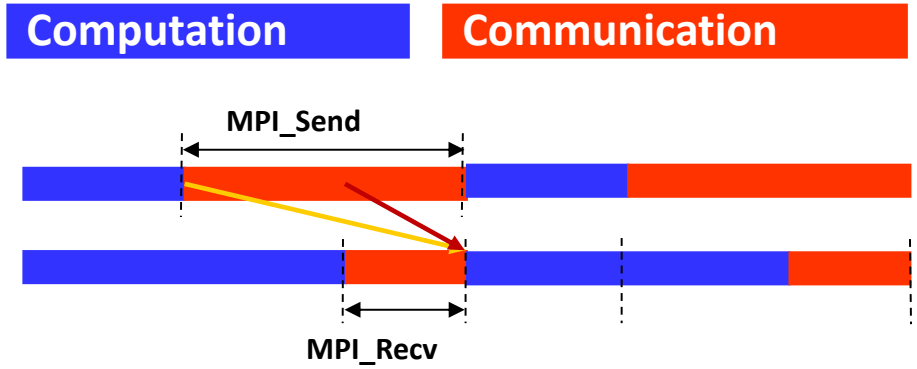
Real run

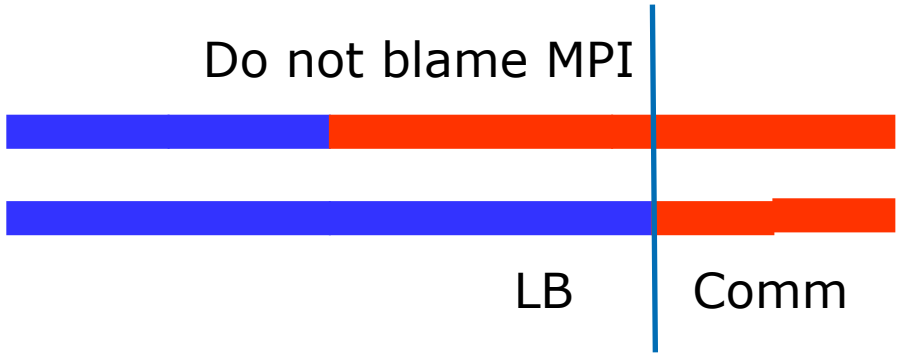
Ideal network



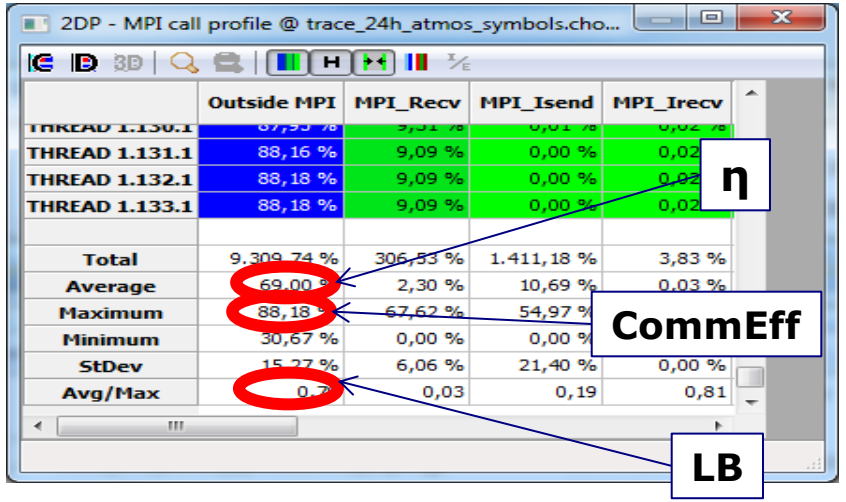
Efficiency Model

Parallel efficiency model

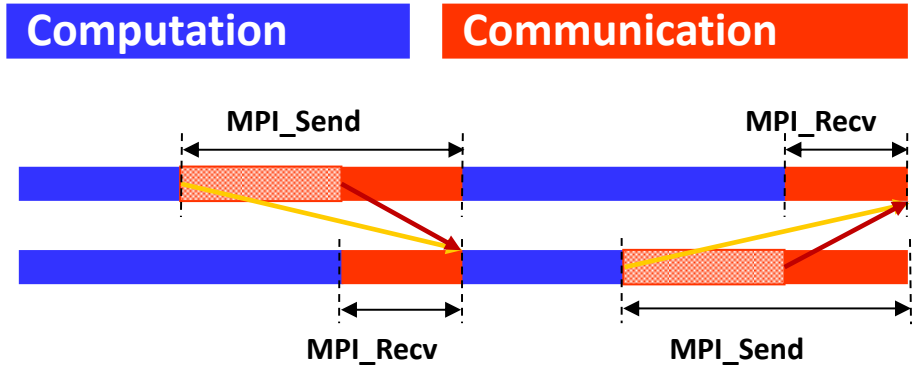


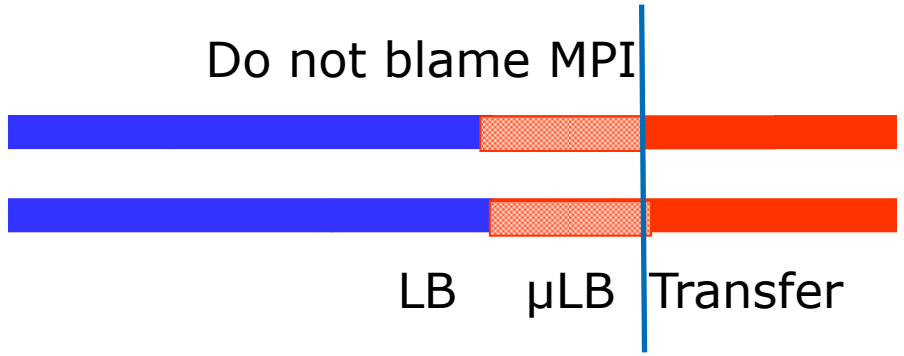


- Parallel efficiency = LB eff * Comm eff



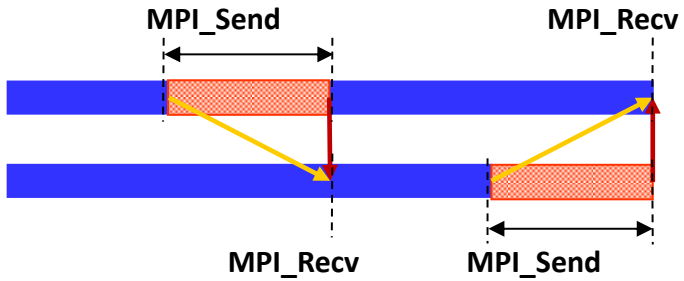
Parallel efficiency refinement: LB * µLB * Transfer





$$LB=1$$

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

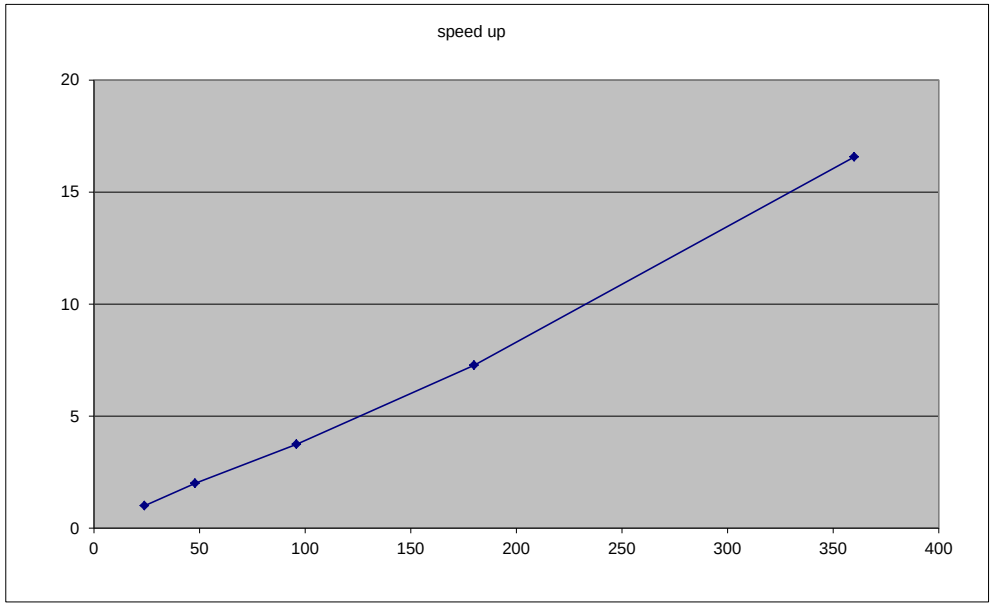


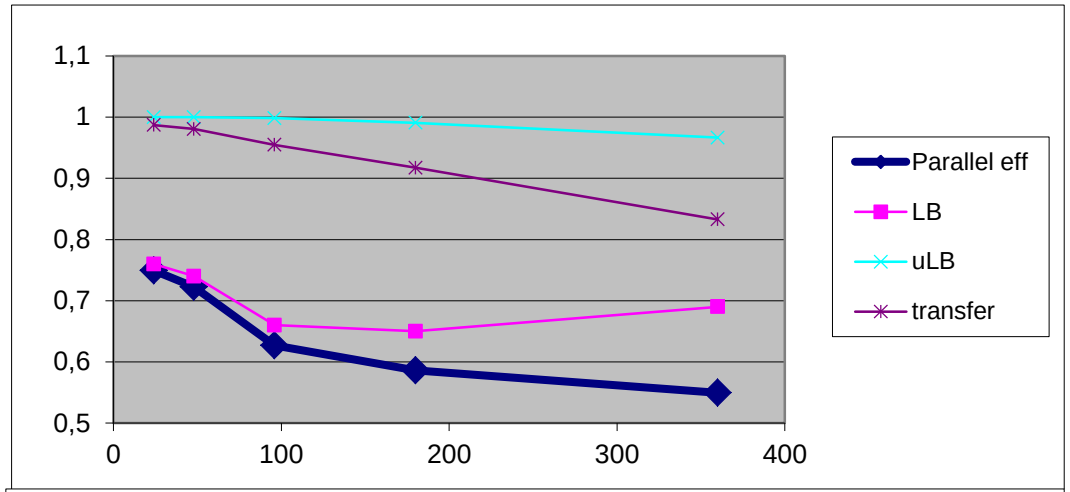
Why scaling?

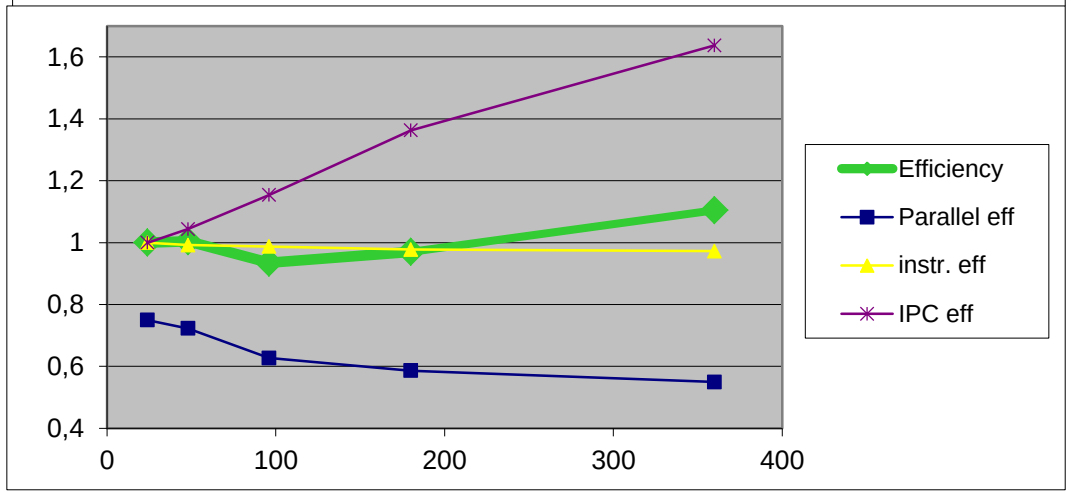
CG-POP mpi2s1D - 180x120

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

Good scalability !!
Should we be happy?





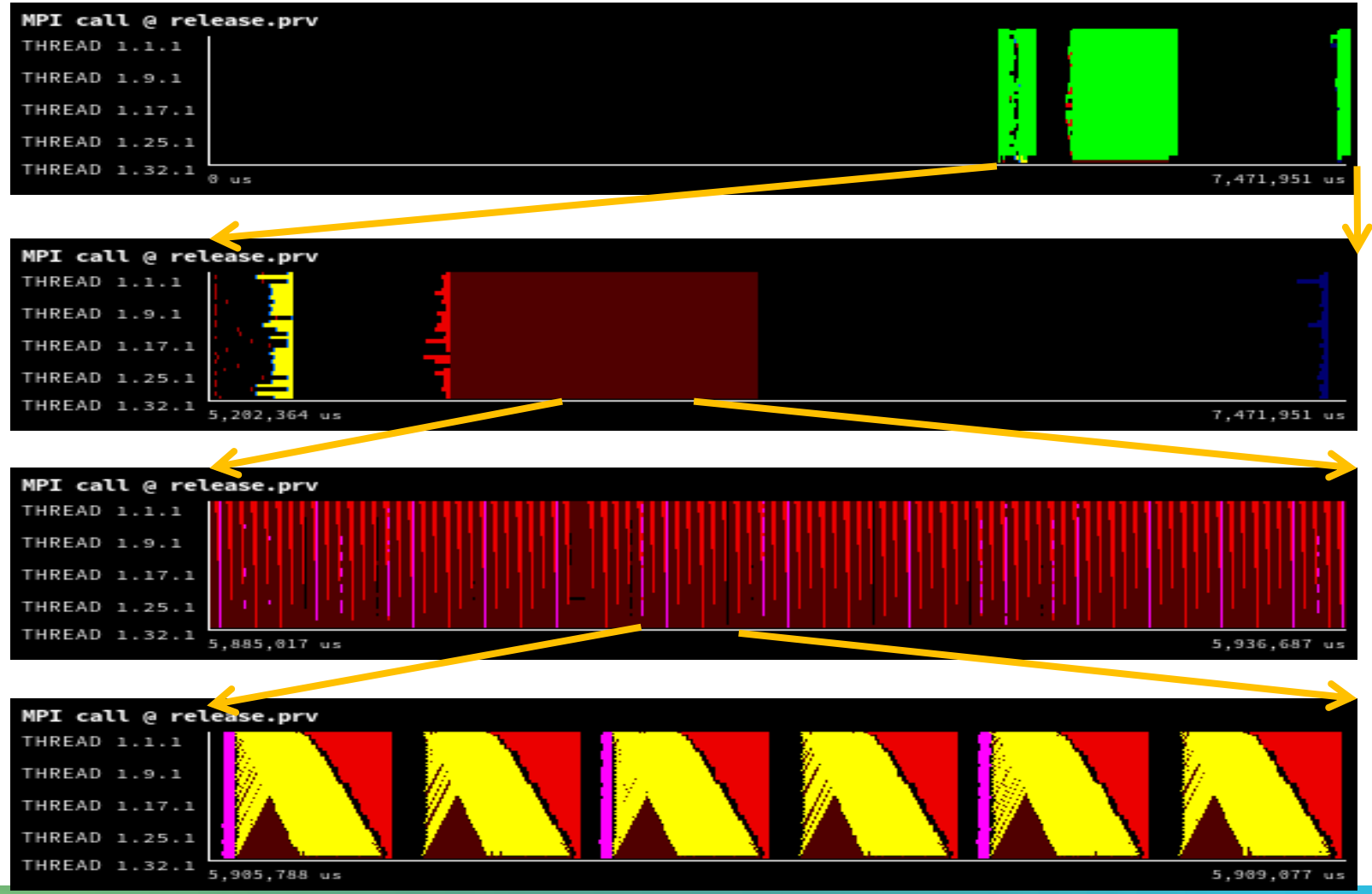


Why efficient?

Parallel efficiency =93.28
Communication = 93.84

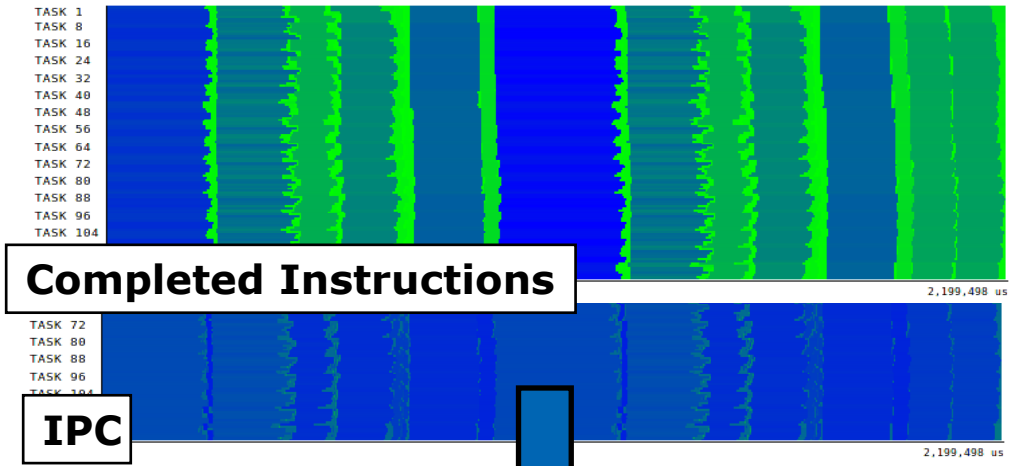
Parallel efficiency = 77.93
Communication = 79.79

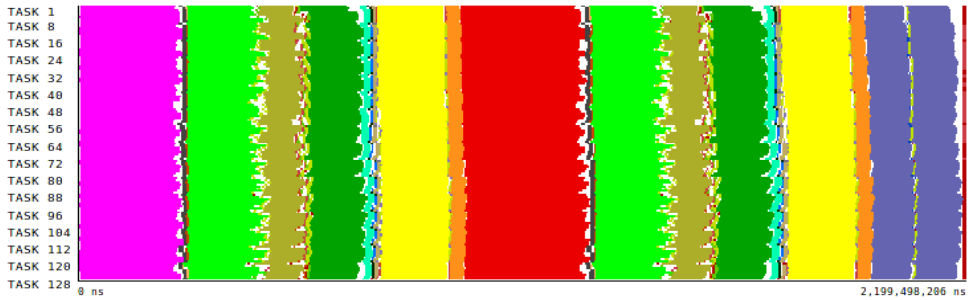
Parallel efficiency = 28.84
Communication eff = 30.42

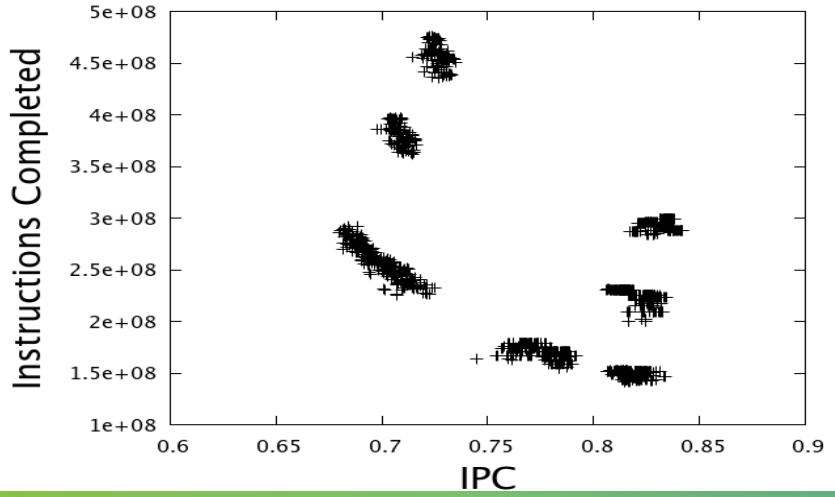


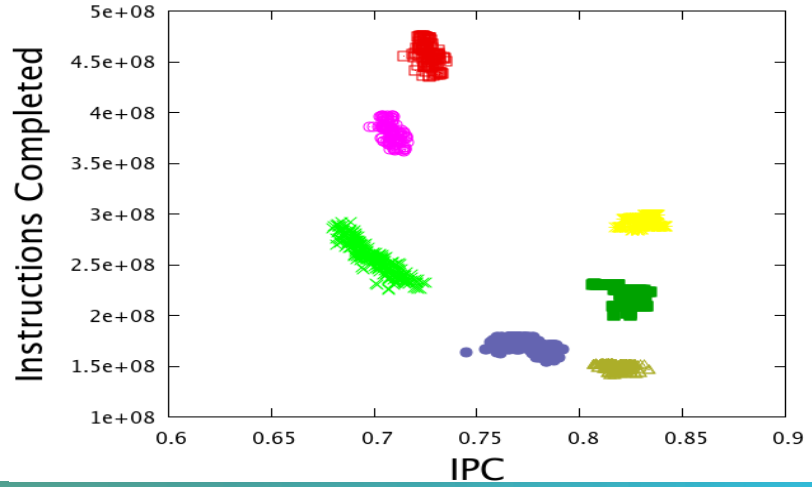
Clustering

Using Clustering to identify structure



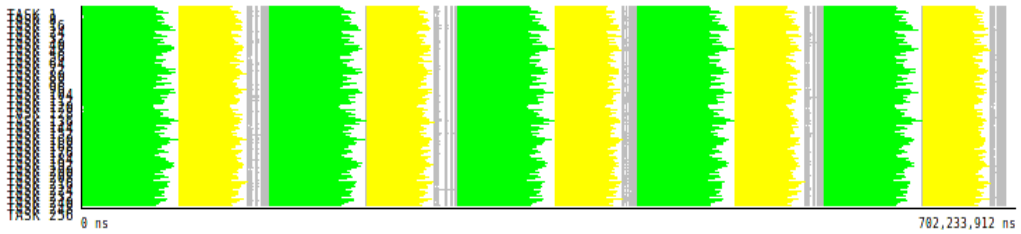






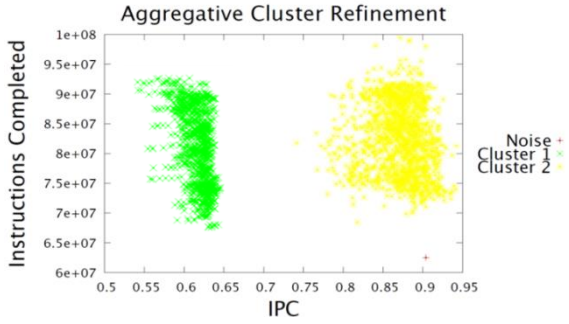
Integrating models and analytics

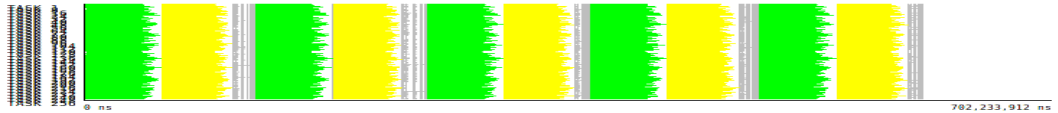
What if ….



PEPC

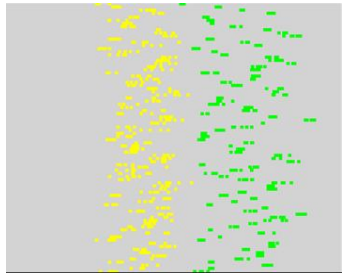
… we increase the IPC of Cluster1?





13% gain

… we balance Clusters 1 & 2?



19% gain

Methodology

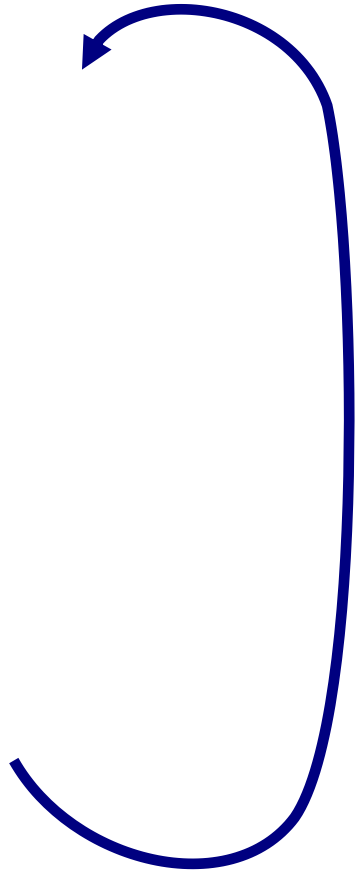
Performance analysis tools objective

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?

Paraver Tutorial:
Introduction to Paraver and Dimemas methodology

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

Paraver Demo

Same code, different behaviour

- Lulesh 2.0
  - Easy to install
  - Requires a cube number of MPI ranks

- What about 27? Check how the system reacts to a "weird" request

Code	Parallel efficiency	Communication eff.	Load Balance eff.
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@inti	88.16	98.65	89.36
lulesh@archer	88.01	97.95	89.86
lulesh@romeo	89.56	99.01	90.45
lulesh@mn4	**91.02**	98.38	**92.52**
lulesh@ stampede2 (skl)	85.76	97.63	87.84
lulesh@ stampede2 (knl)	89.21	98.42	90.64
lulesh@isambard	90.32	97.16	92.96
lulesh@hawk (openmpi)	87.82	98.28	89.35
lulesh@meggie	89.44	96.80	92.00
lulesh@IvyMUC	86.59	98.16	88.21

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