Understanding applications with BSC Tools

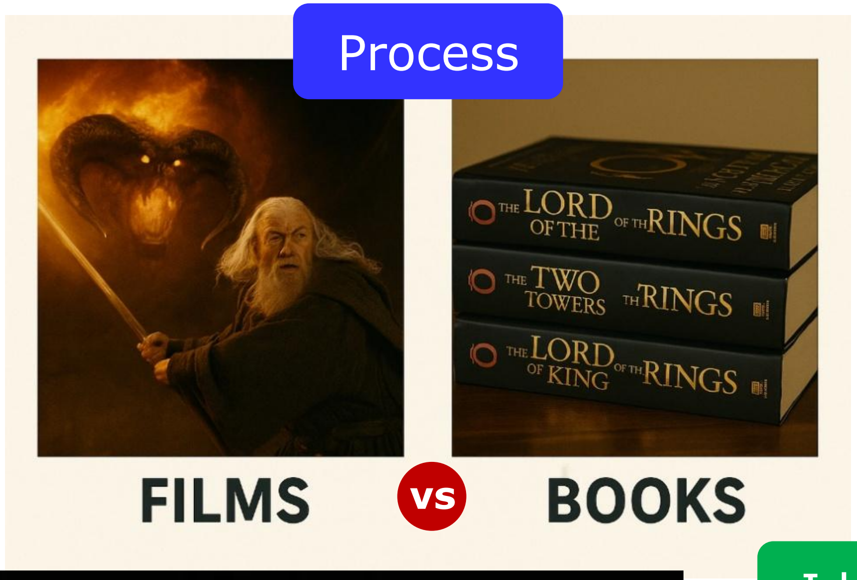
Lau Mercadal, Sandra Méndez, Germán Llort

tools@bsc.es

Barcelona Supercomputing Center

Performance Optimisation and Productivity 3 (101143931)

Process



Store

Identify

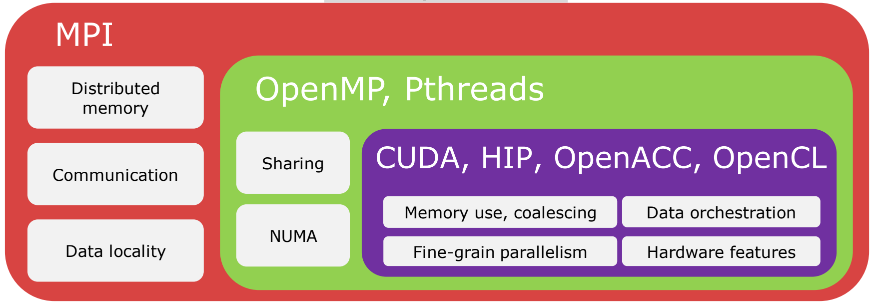
BSC Tools

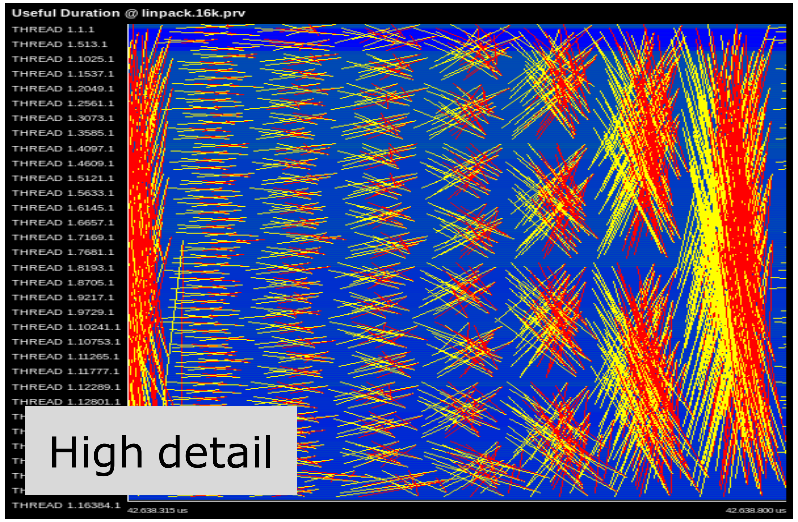
- Since 1991
- Based on traces
- Open Source
  - https://tools.bsc.es
- Core tools:
  - **Extrae** – instrumentation
  - **Paraver** – offline trace analysis
  - **Dimemas** – message passing simulator
- Focus
  - Detail, variability, flexibility
  - Behavioural structure vs. syntactic structure
  - Intelligence: Performance Analytics

Complex Apps

Complex Hw

Complex Sw





Paraver

Paraver – Performance data browser

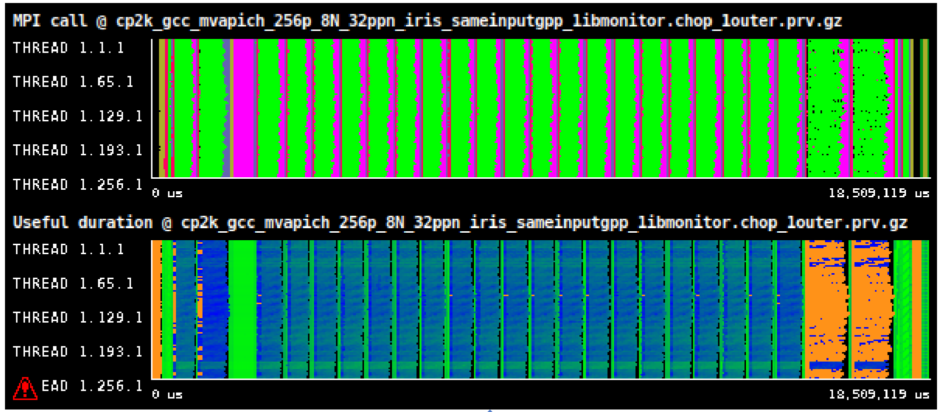
Raw data

```
1:151:1:147:1:294672917:294676549:1
2:151:1:147:1:294672917:50000001:0
3:151:1:147:1:294623693:294672917:158:1:154:1:294670636:297419889:4223536:8
```

Trace visualization/analysis

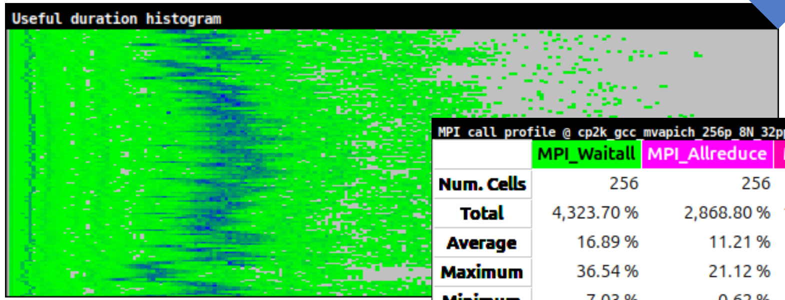
Trace manipulation

Timelines



Goal = Flexibility
No semantics
Programmable

2/3D Histograms



2/3D tables (statistics)

MPI call profile @ cp2k_gcc_mvapich_256p_8N_32ppn_iris_sameinputgpp_libmonitor.chop_1outer.prv.gz

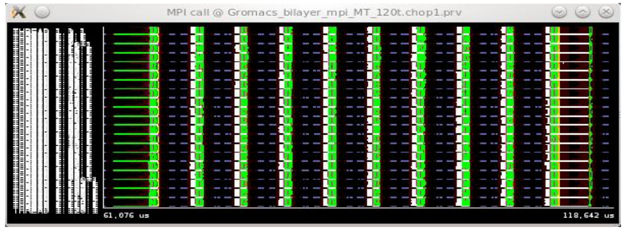
	MPI_Waitall	MPI_Allreduce	MPI_Irecv	MPI_Alltoallv	MPI_Alltoall	MPI_Isend
Num. Cells	256	256	256	256	256	256
Total	4,323.70 %	2,868.80 %	1,201.01 %	312.82 %	305.23 %	172.99 %
Average	16.89 %	11.21 %	4.69 %	1.22 %	1.19 %	0.68 %
Maximum	36.54 %	21.12 %	12.49 %	1.30 %	4.66 %	5.10 %
Minimum	7.03 %	0.62 %	0.09 %	1.12 %	0.12 %	0.10 %
StDev	5.76 %	4.13 %	3.25 %	0.04 %	0.52 %	0.72 %
Avg/Max	0.46	0.53	0.38	0.94	0.26	0.13

Comparative analyses
Multiple traces
Synchronize scales

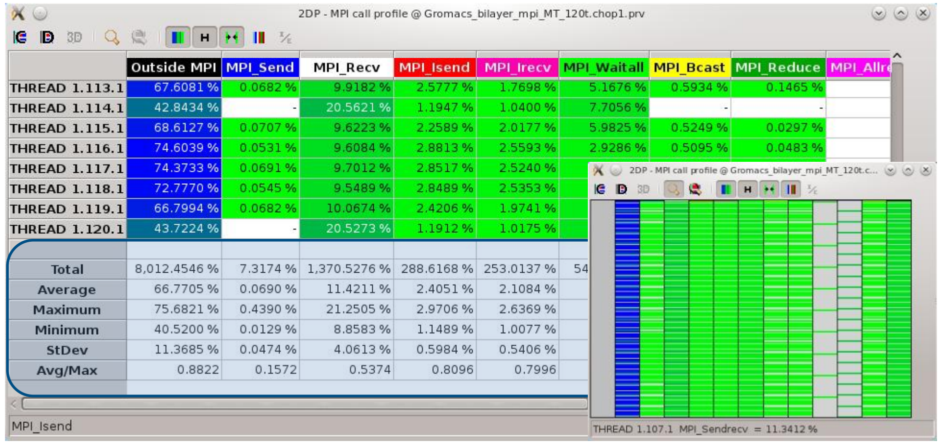
From timelines to tables

- Categorical metrics → colours
  - Each category is assigned a distinct colour

MPI calls



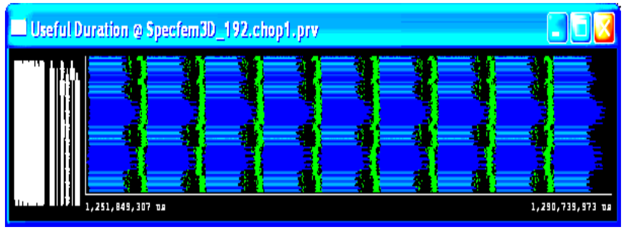
MPI calls profile



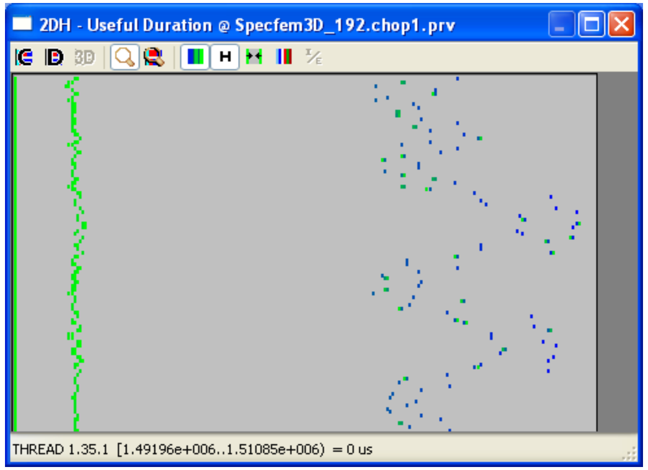
	Outside MPI	MPI_Send	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Bcast	MPI_Reduce	MPI_Allre
THREAD 1.113.1	67.6081 %	0.0682 %	9.9182 %	2.5777 %	1.7698 %	5.1676 %	0.5934 %	0.1465 %	
THREAD 1.114.1	42.8434 %	-	20.5621 %	1.1947 %	1.0400 %	7.7056 %	-	-	
THREAD 1.115.1	68.6127 %	0.0707 %	9.6223 %	2.2589 %	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 %	2.5240 %				
THREAD 1.118.1	72.7770 %	0.0545 %	9.5489 %	2.8489 %	2.5353 %				
THREAD 1.119.1	66.7994 %	0.0682 %	10.0674 %	2.4206 %	1.9741 %				
THREAD 1.120.1	43.7224 %	-	20.5273 %	1.1912 %	1.0175 %				
Total	8,012.4546 %	7.3174 %	1,370.5276 %	288.6168 %	253.0137 %	54			
Average	66.7705 %	0.0690 %	11.4211 %	2.4051 %	2.1084 %				
Maximum	75.6821 %	0.4390 %	21.2505 %	2.9706 %	2.6369 %				
Minimum	40.5200 %	0.0129 %	8.8583 %	1.1489 %	1.0077 %				
StDev	11.3685 %	0.0474 %	4.0613 %	0.5984 %	0.5406 %				
Avg/Max	0.8822	0.1572	0.5374	0.8096	0.7996				

- Continuous metrics → gradients
  - **Green** to **blue** for **low** to **high** values

Useful duration

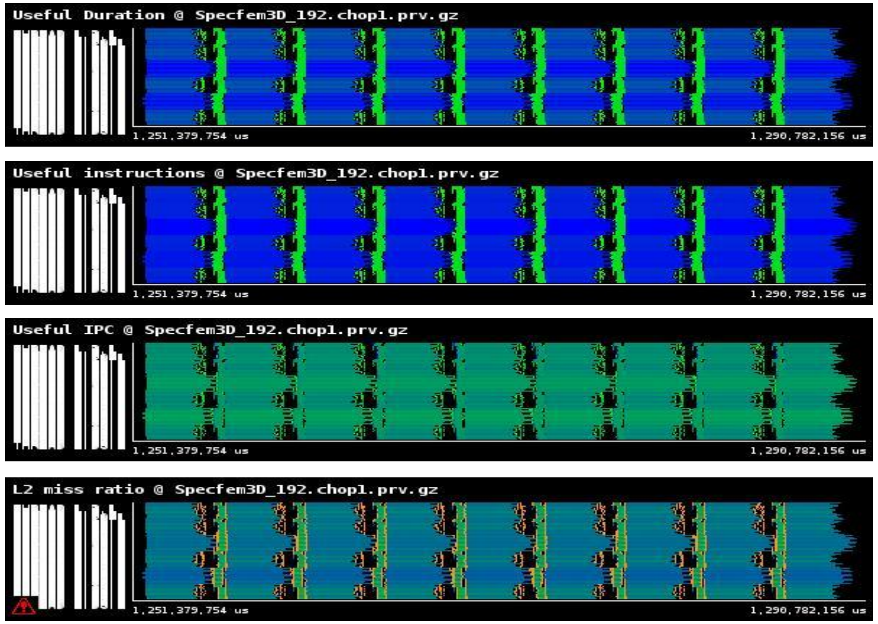


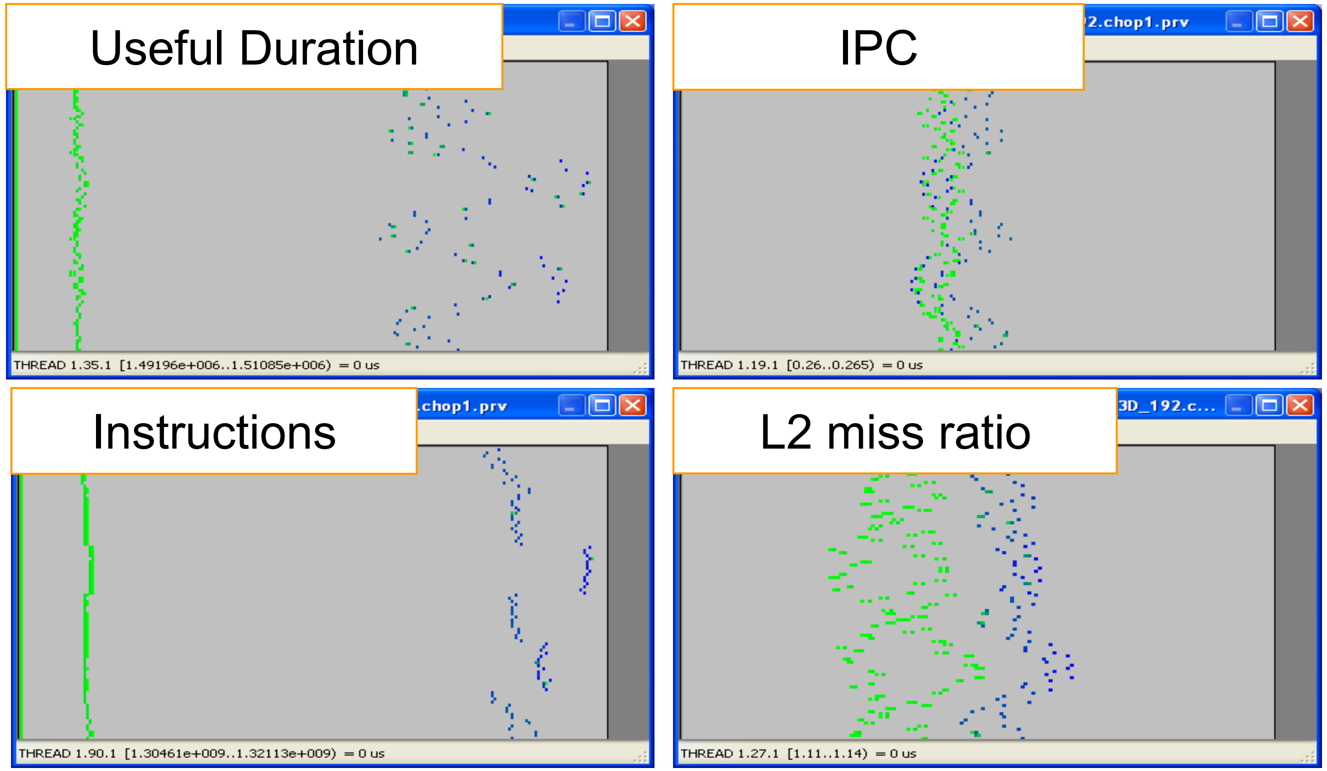
Compute-time histogram



Analyzing variability through timelines and histograms

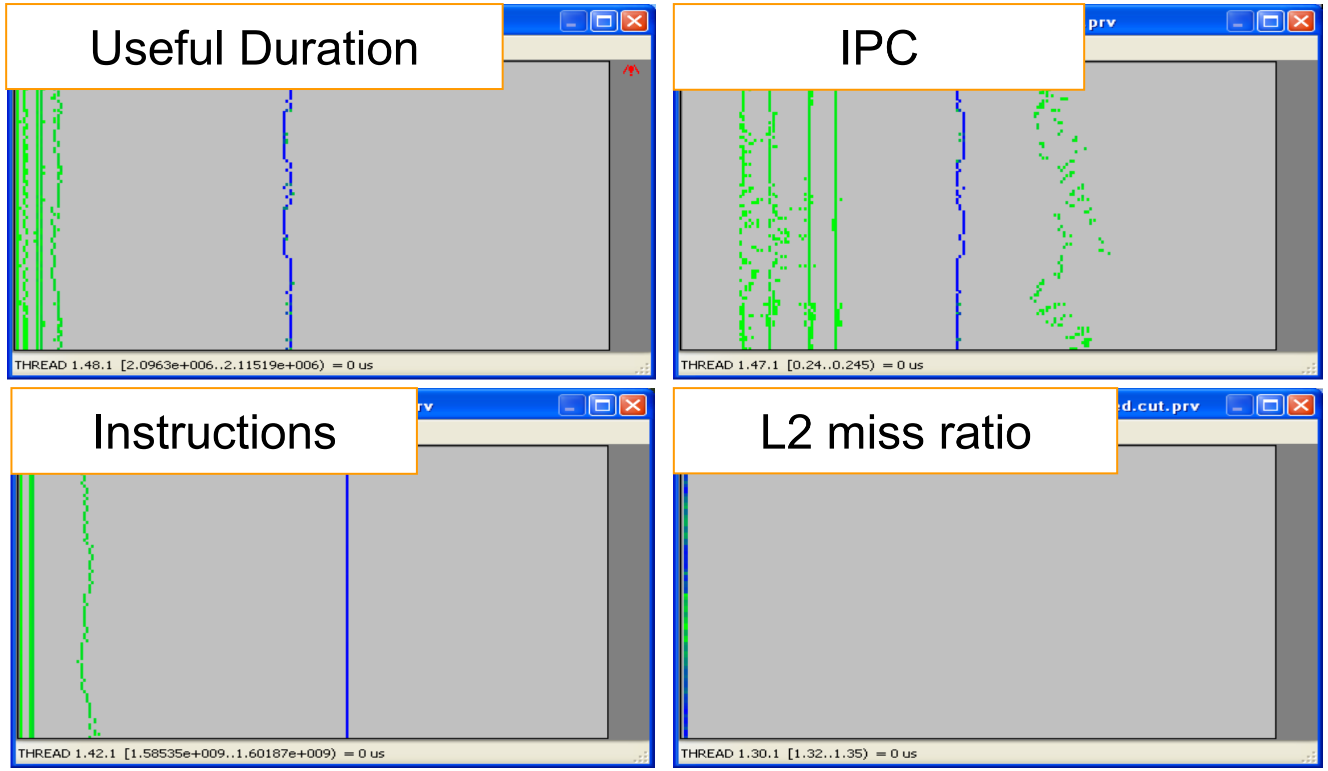
Courtesy Dimitri Komatitsch





Analyzing variability through histograms and timelines

- By the way: six months later…

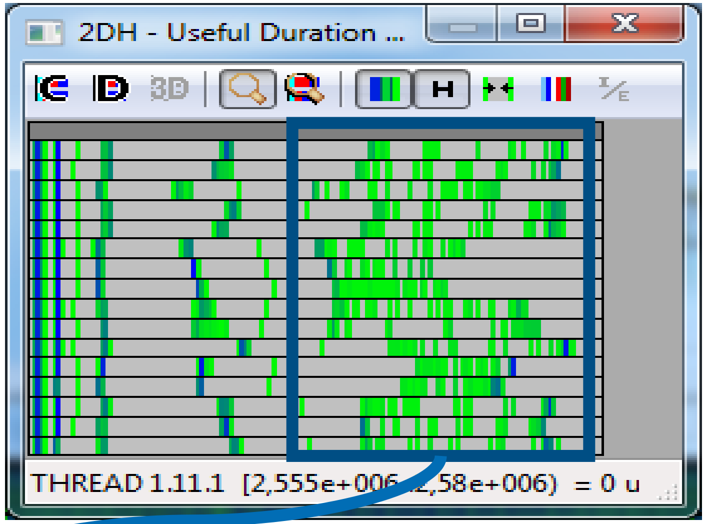


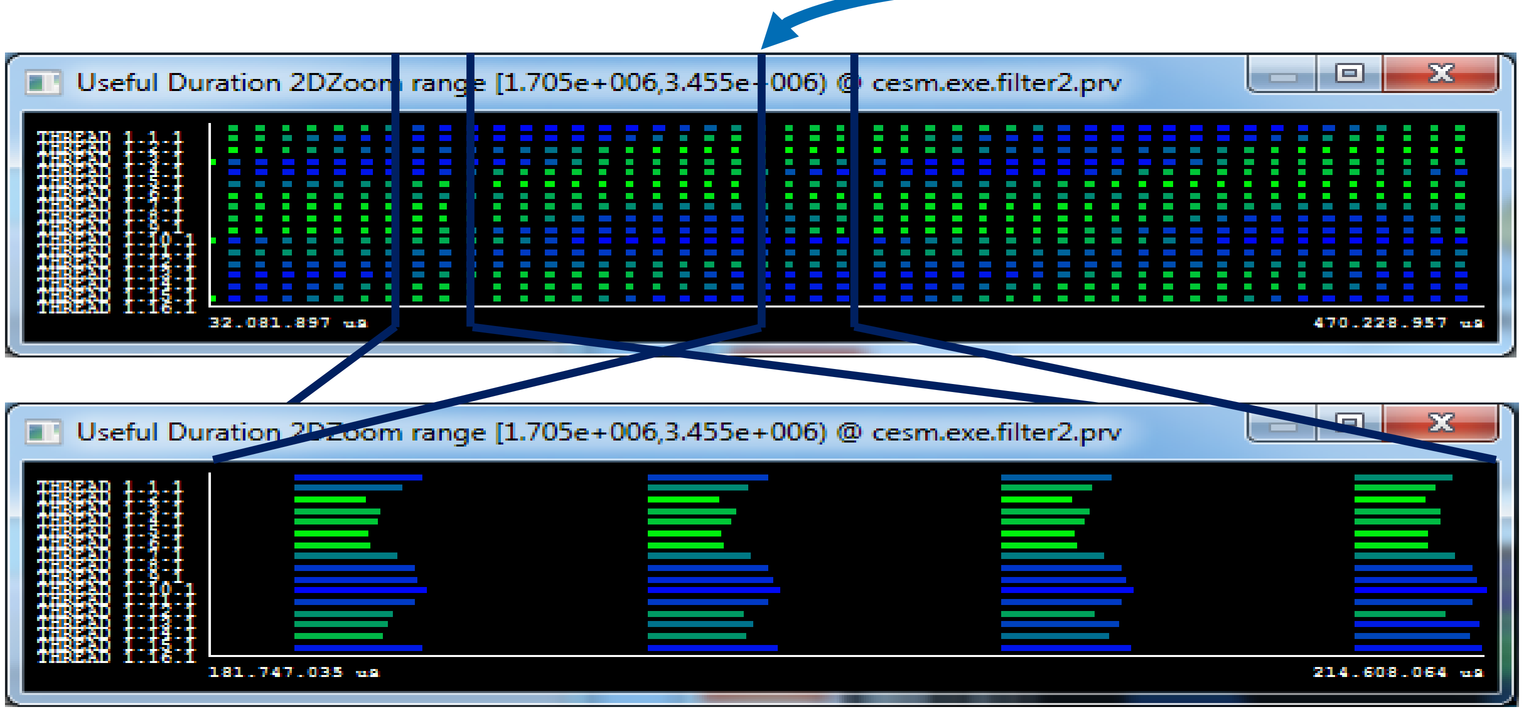
Focus optimization where it matters

From tables to timelines

- CESM: 16 processes, 2-day simulation

- High variability in useful computation duration
- How is it distributed?
- Dynamic imbalance
  - In space and time
  - Day and night
  - Season?





Why traces?

Performance data trade-offs

Measure	
Instrumentation	Sampling

Accuracy vs. Overhead

Store	
Traces	Profiles

Detail vs. Size

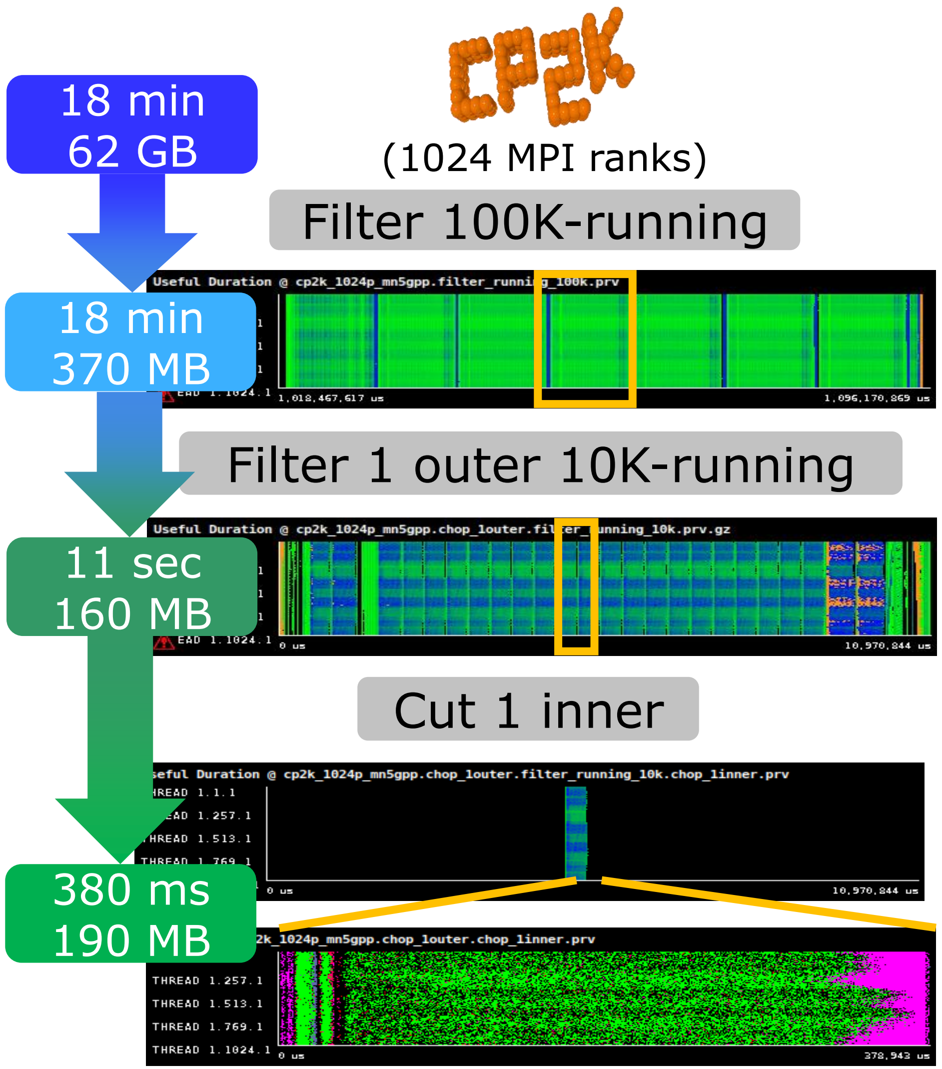
See	
Timelines	Statistics

Insight vs. Complexity

Highly detailed traces can be quite large…

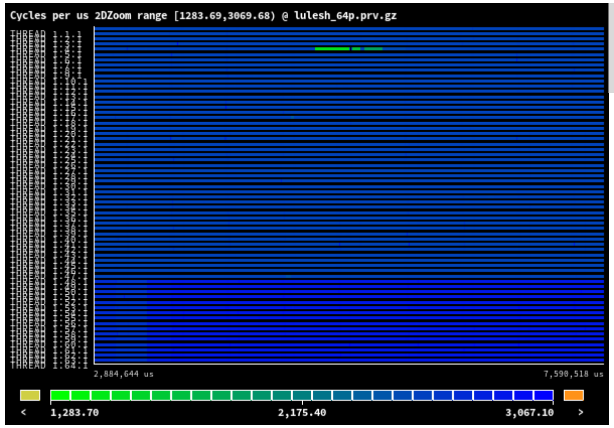
Manipulating big traces

- Data processing & summarization
  - **Filtering**
    - Subset of records from the original trace
    - By duration, type, value…
  - **Cutting**
    - All records within a time interval
    - Selected processes only
  - **Software counters**
    - Aggregated metrics as new events
      - MPI call count, HW totals…
  - Remains a Paraver-compatible trace for analysis with the same CFGs (if needed data is kept)
- Automatic analysis → **Performance Analytics**



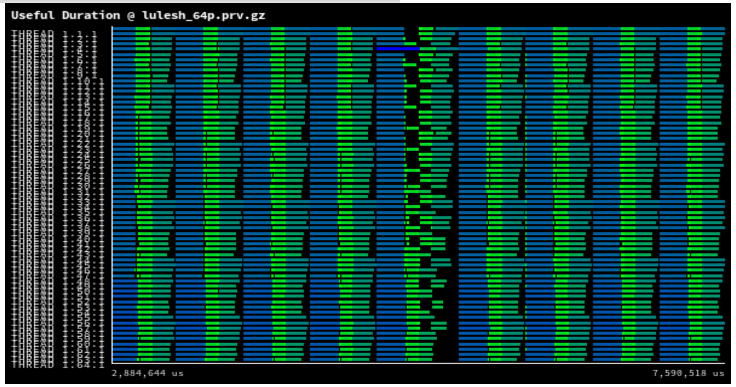
The Butterfly Effect…

- A system preemption reduces the cycles assigned to one of the processes for a small interval

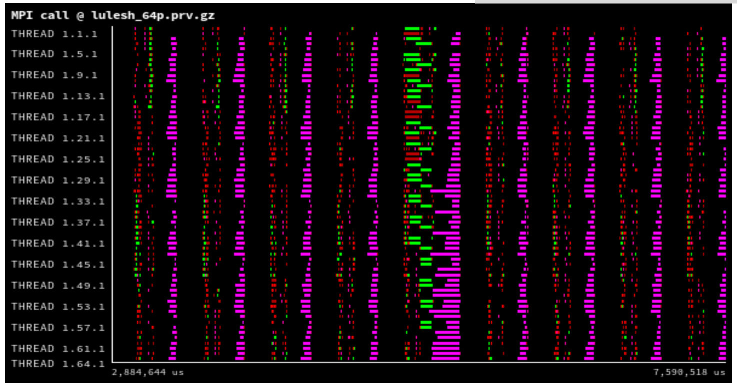


Cycles/us

Useful duration



MPI calls

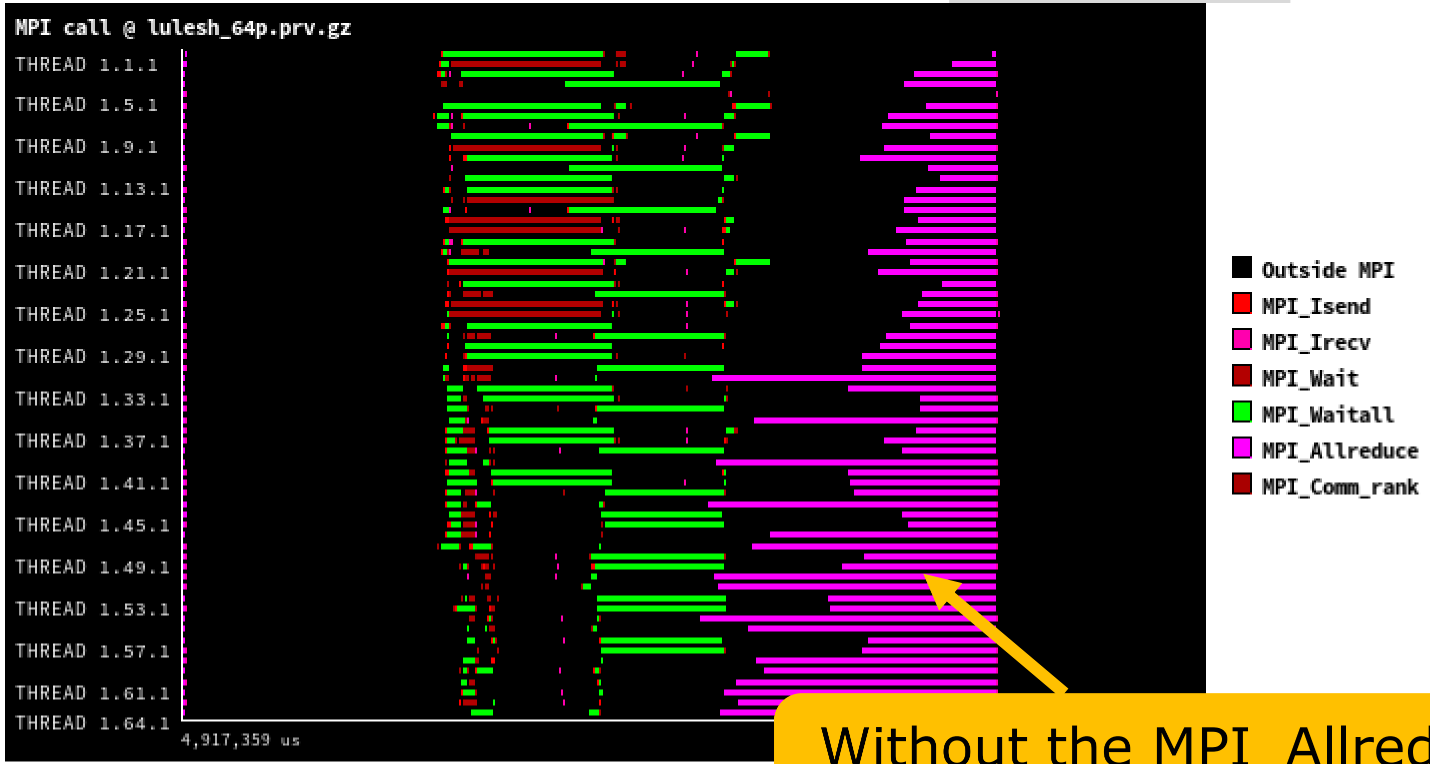


Affects only one process, but disturbs all processes

… flying through the MPI pattern

- Produces a wave effect that reflects and interferes across all partners

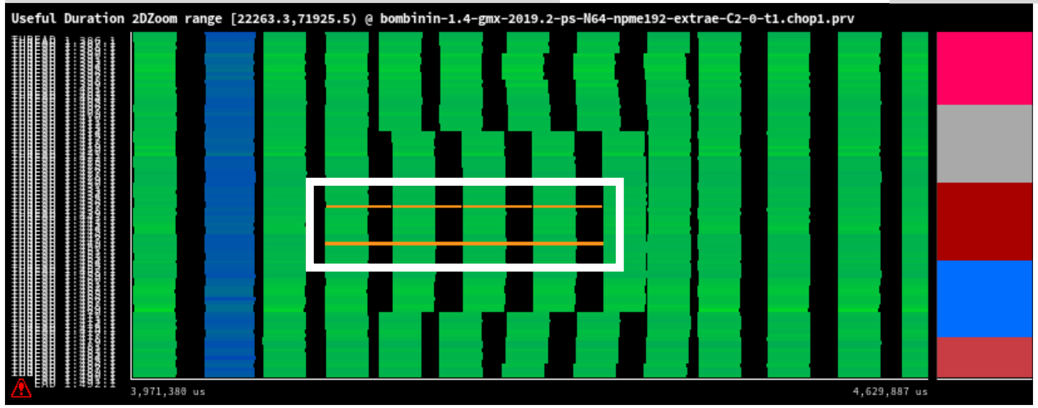
MPI calls



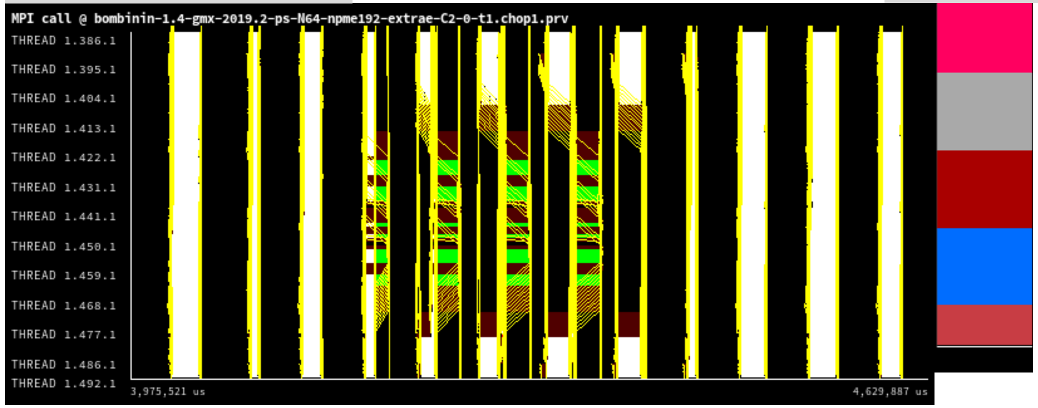
Without the MPI_Allreduce it would perturb multiple iterations

… flying to other nodes

Useful duration **Node**



MPI calls **Node**

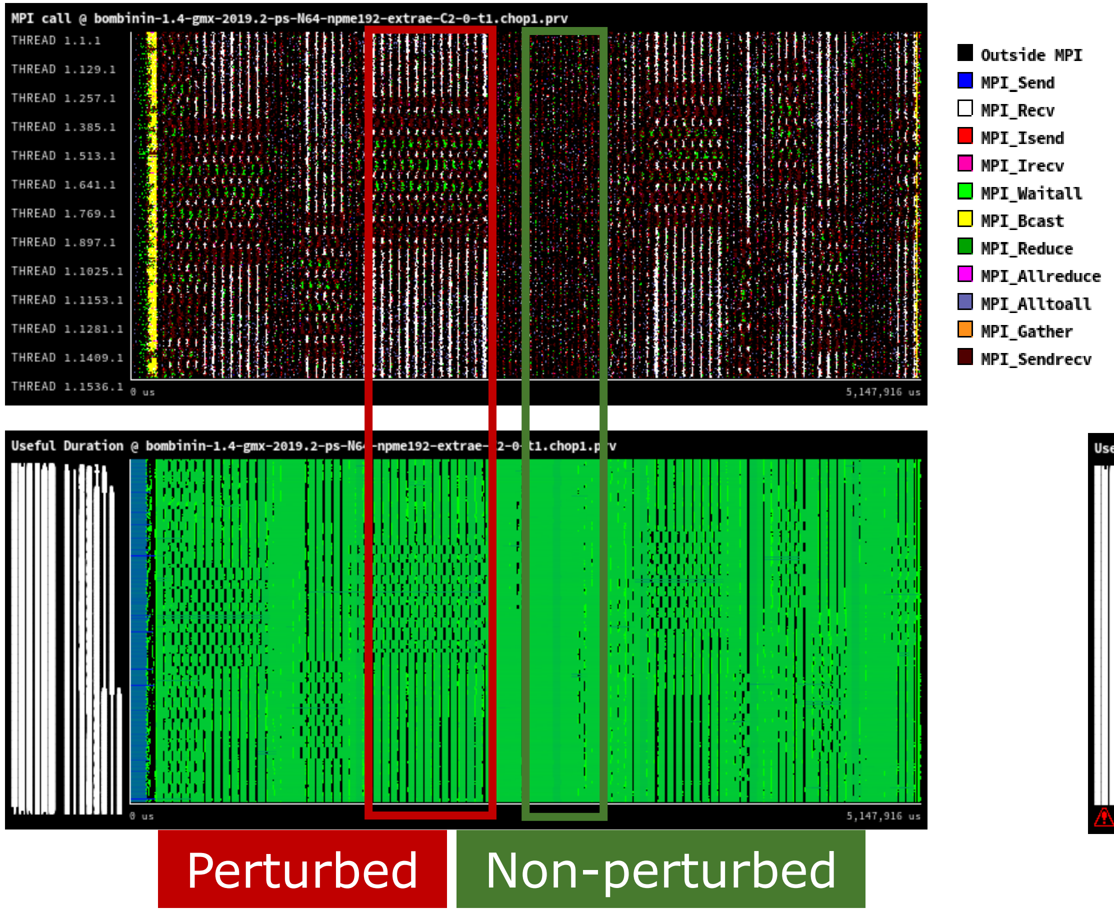


- Outside MPI
- MPI_Recv
- MPI_Isend
- MPI_Waitall
- MPI_Sendrecv

- 2 processes perturbed in the same interval
- On the same node

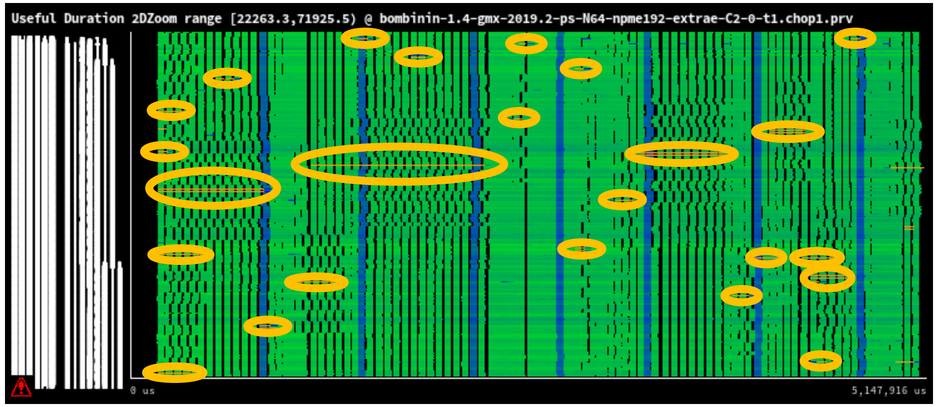
Affects only one node, but disturbs many nodes

Thousands of butterflies?



Microscopic effects with large global impact

- Rescaling the gradient…

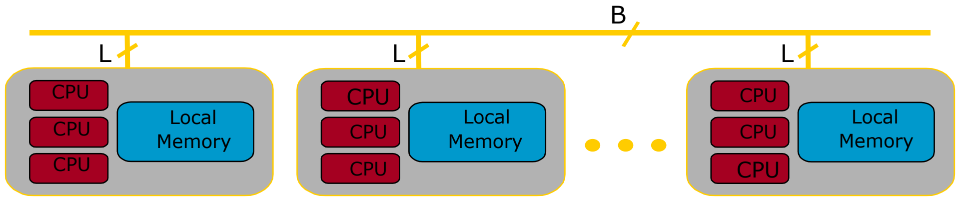


Pinpoint bursts larger than expected

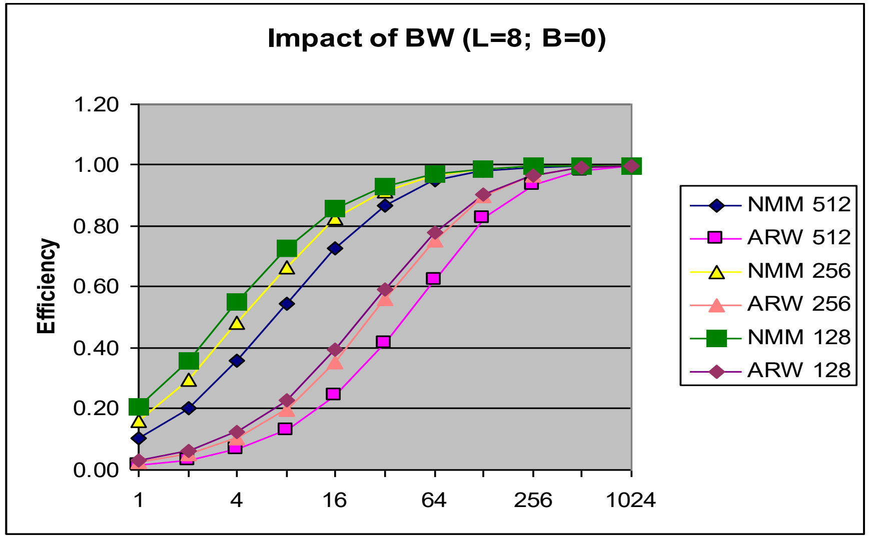
Dimemas

Dimemas: Coarse-grain, Trace-driven simulation

- **Fast simulation of an abstract interconnect**
  - SMP nodes with local memory for intra-node comms
  - Interconnect defined by L (links) and B (buses)
    - B → Limits concurrent messages (contention)
    - L → Limits per-node traffic (connectivity)
  - Local/remote Latency/Bandwidth



- **Parametric sweeps**
  - On abstract architectures
  - On application computing regions
- **"What-if…" analysis**
  - Ideal machine (instantaneous network)
  - Would benefit from asynchronous communications?
  - Are all regions equally sensitive to the network?

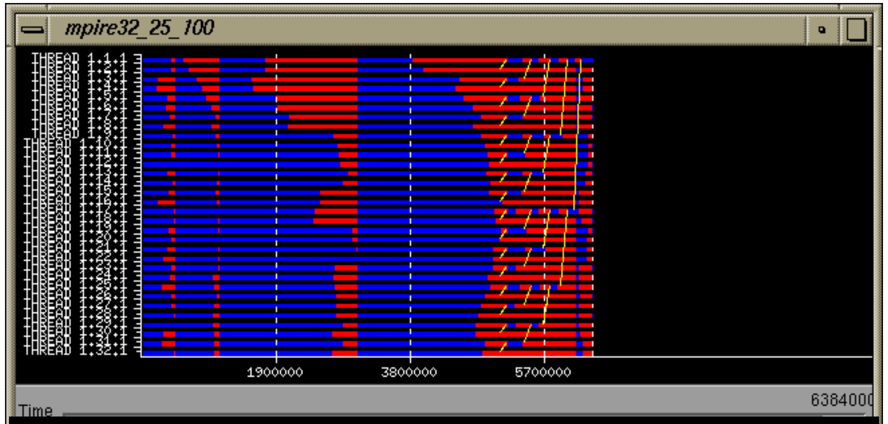


- **MPI sanity check**
  - Nominal modeling
- **Paraver + Dimemas tandem**
  - Analysis and prediction
  - What-if from selected time window

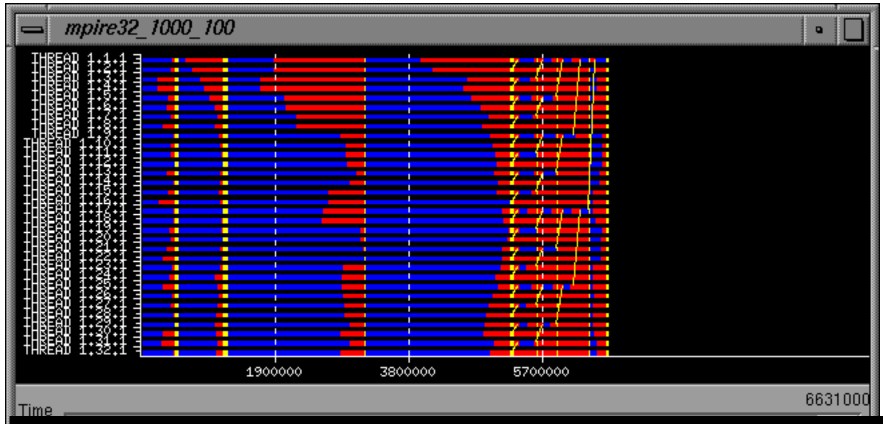
Simulation generates a trace → Detailed feedback

Network sensitivity

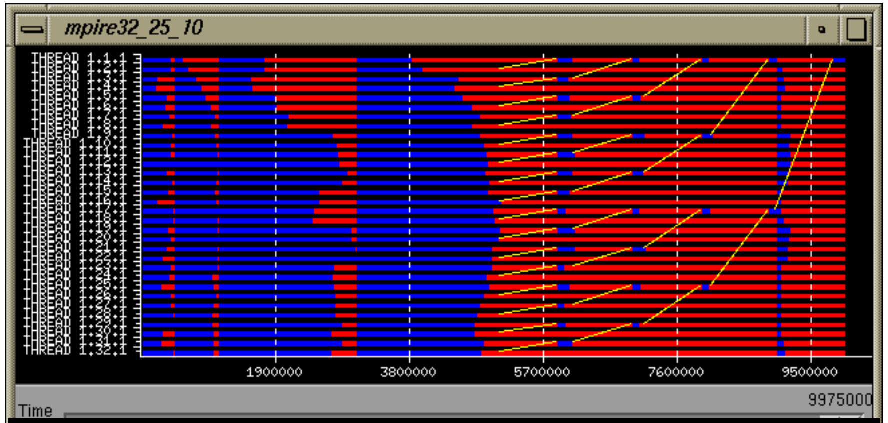
- MPIRE 32 tasks, no contention → Sensitive to BW or Latency?



L = 5µs – BW = 1 GB/s



L = 1000µs – BW = 1 GB/s



L = 5µs – BW = 100MB/s

Higher latency doesn't hurt

Lower bandwidth kills the P2P phase

What if… we had asynchronous communications?

Courtesy Dimitri Komatitsch

Ideal simulation shows not benefit

Perturbation only appears below 10 MB/s

Async comms won't help

Real

Ideal

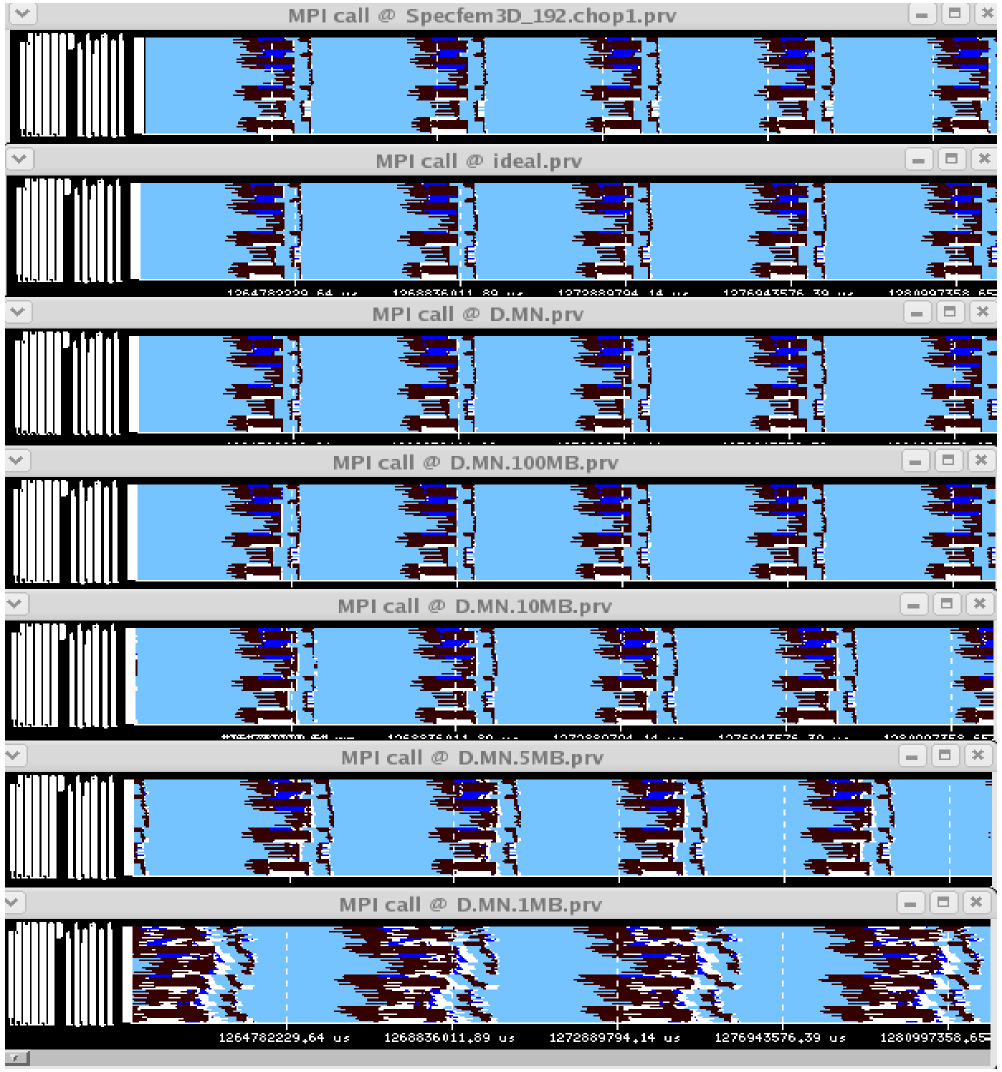
Prediction MN

Prediction 100MB/s

Prediction 10MB/s

Prediction 5MB/s

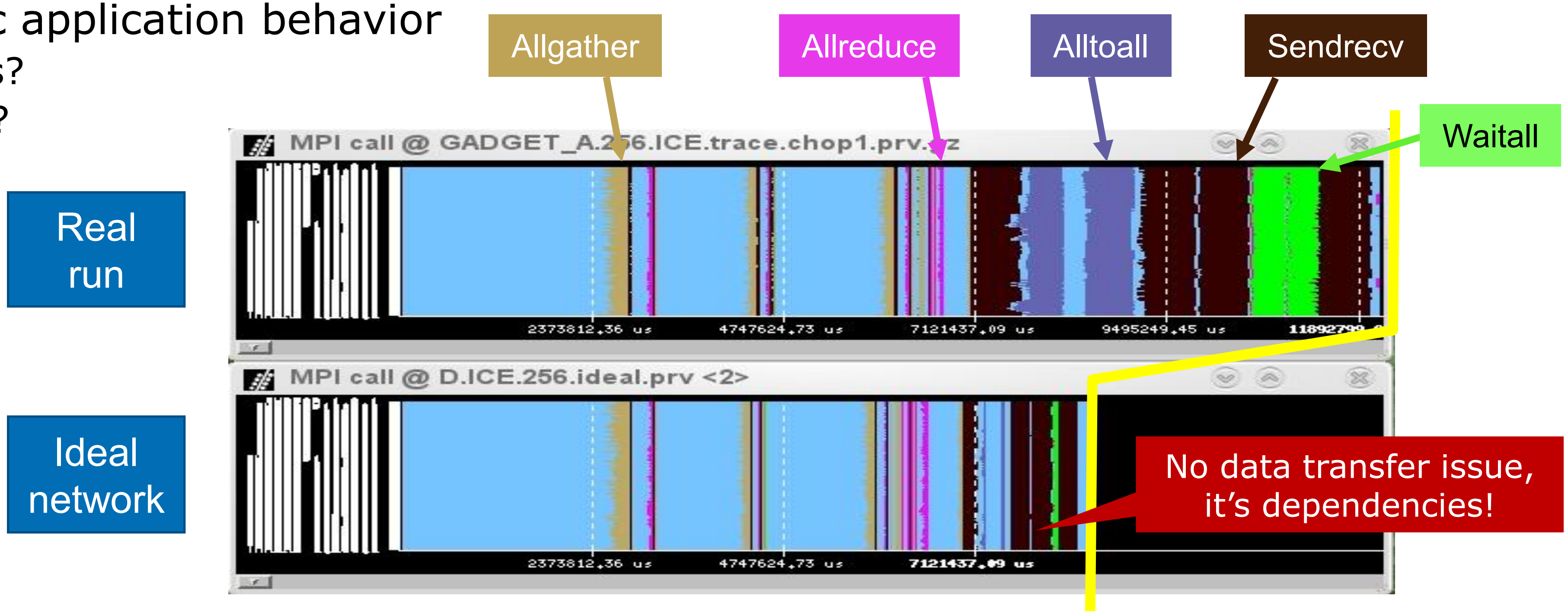
Prediction 1MB/s



The Ideal Machine

- **BW = ∞, L = 0**
  - Data transfer would be instantaneous → MPI time should vanish. Why not?
  - Characterizes intrinsic application behavior
    - Load balance problems?
    - Dependency problems?

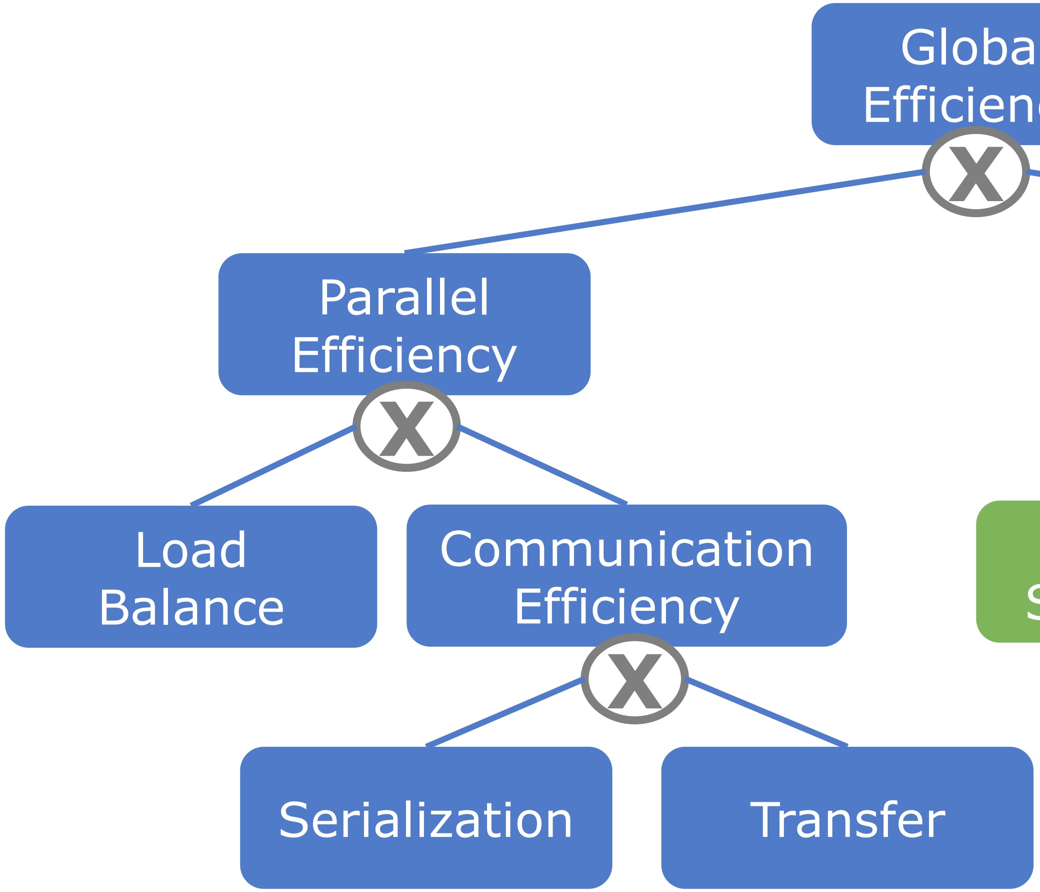
GADGET @ Nehalem
256 processes



Impact on real machines?
If it doesn't improve on the ideal, it won't on a faster real one

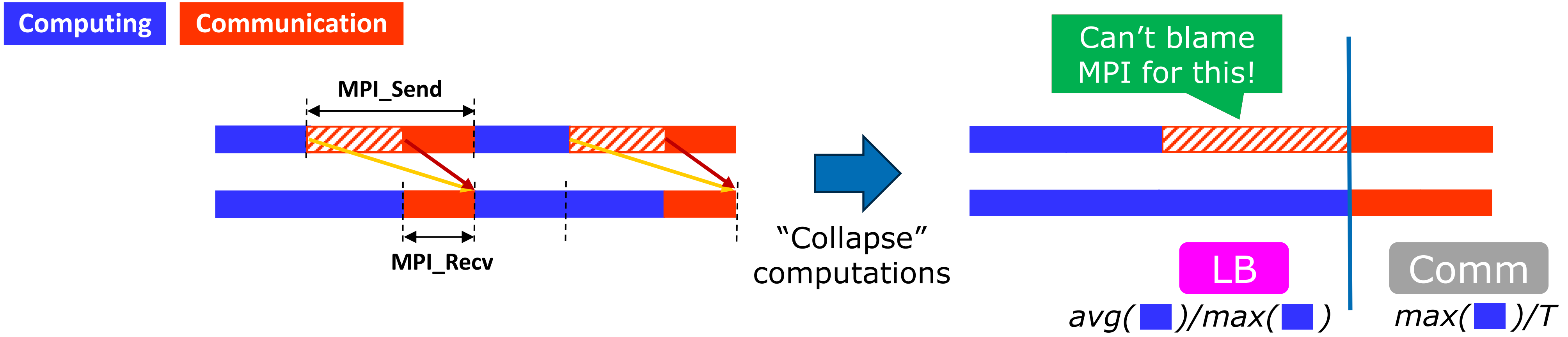
Efficiency Model

Efficiency Metrics



- Hierarchical model
- Multiplicative metrics
- 1 to 100% scale
- Two kinds of metrics:
  - Efficiency metrics (absolute)
  - Scalability metrics (relative to a base case)
- MPI, Hybrid MPI+OpenMP/CUDA

Parallel Efficiency model



Parallel Efficiency = **LB** * **Comm**

How effectively all system resources stay active doing useful work

A low value means most time isn't used for useful work

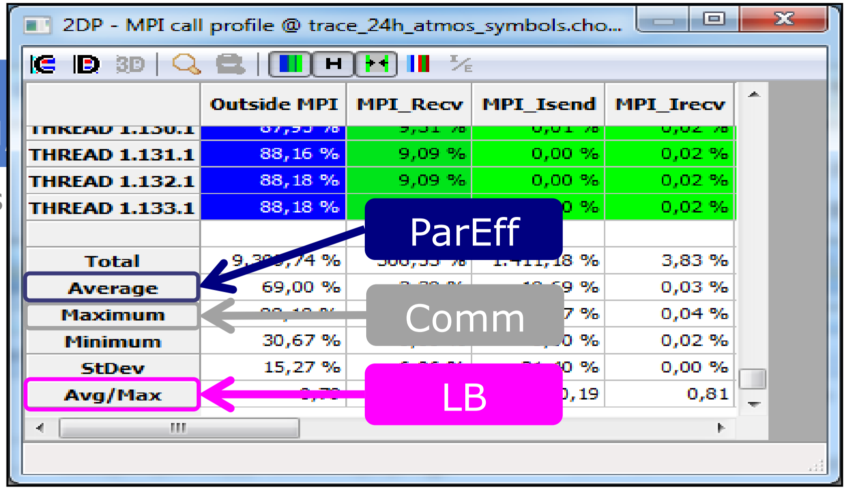
Efficiency loss from data communication, processes and communication overhead

A low value means inter-process interactions

Can be measured directly in Paraver over a profile of Computing Time

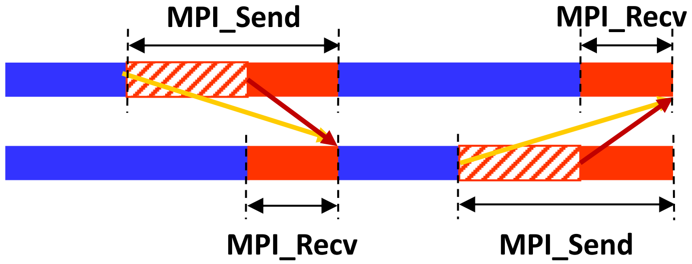
Efficiency loss from the global distribution of work among processes

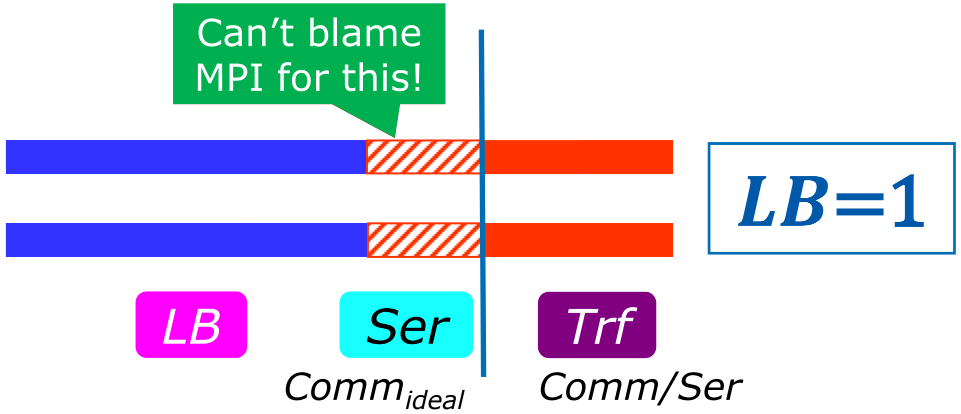
A low value means heavily loaded processes keep others idle for long periods



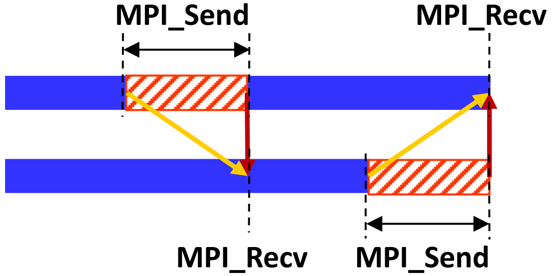
Communication Efficiency refinement

Computing | Communication





- Splitting Serialization (*Ser*) and Transfer (*Trf*)
  - Simulate with Dimemas an ideal network
    - Instantaneous data transfer ➔ ■ vanishes (*Trf*=100%)



Parallel Efficiency = LB * *Ser* * *Trf*

Inefficiencies from circular dependences or non-uniform imbalances

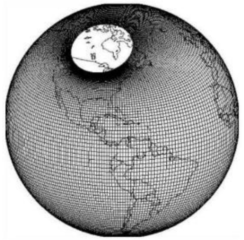
Inefficiencies from communication delays, including transfer

Can be measured in Paraver with the real and simulated trace, and directly with BasicAnalysis

A low value means algorithmic dependency chains, varying load imbalance, or noise

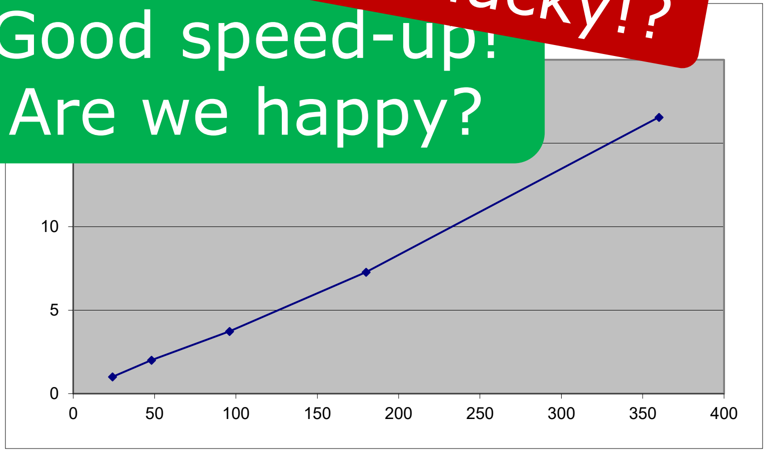
Why scaling?

- CGPOP ocean modelling
  - Intrinsically unbalanced problem



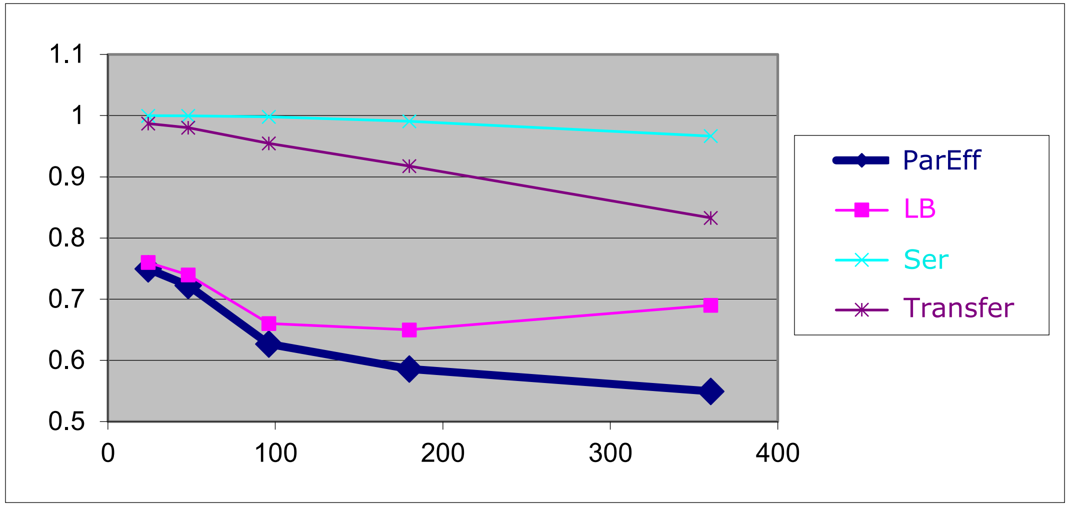
Or just lucky!?

Good speed-up!
Are we happy?

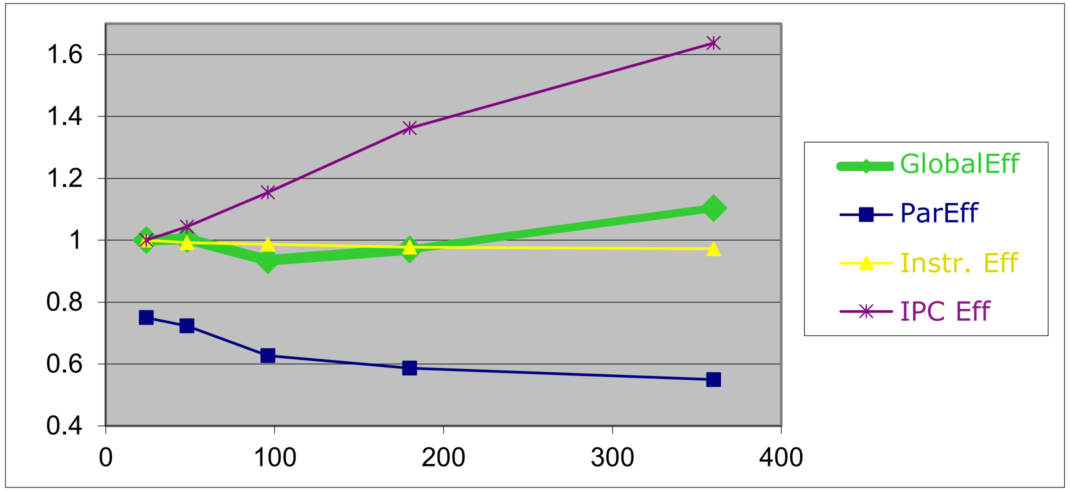


- Transfer Efficiency ↓
- IPC helps… for now!
- **Comms will become a bottleneck**

ParEff = *LB* * *Ser* * *Trf*

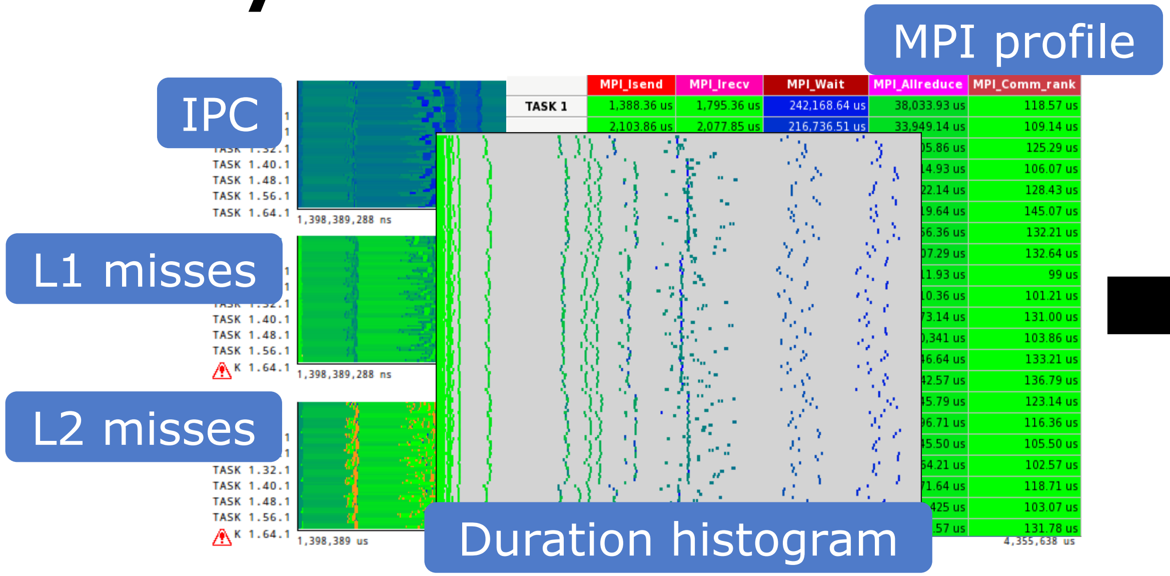


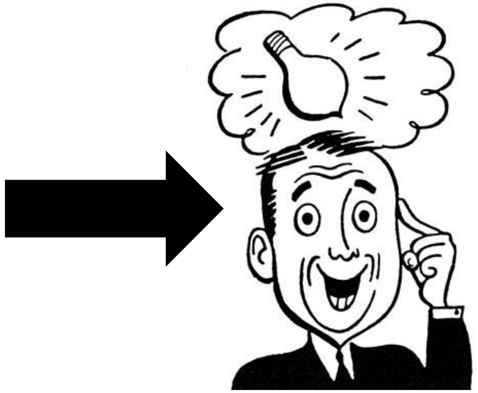
GlobalEff = *ParEff* * *Instr* * *IPC* * *Freq*



Analytics

Analysis → Analytics





- Dominant practice
  - Lots of data captured
    - But presentation goes from raw data to too general statistics
- Need for performance analytics
  - Leveraging techniques from data science, image processing, signal processing, etc.
  - Towards insight and models

Basic Analysis

- Automatically compute the efficiency metrics from a Paraver trace (or many for scalability studies)
- Dig down from global to detailed efficiencies

What to look for?

- Low values
- Trends
- High values

Comparing scales

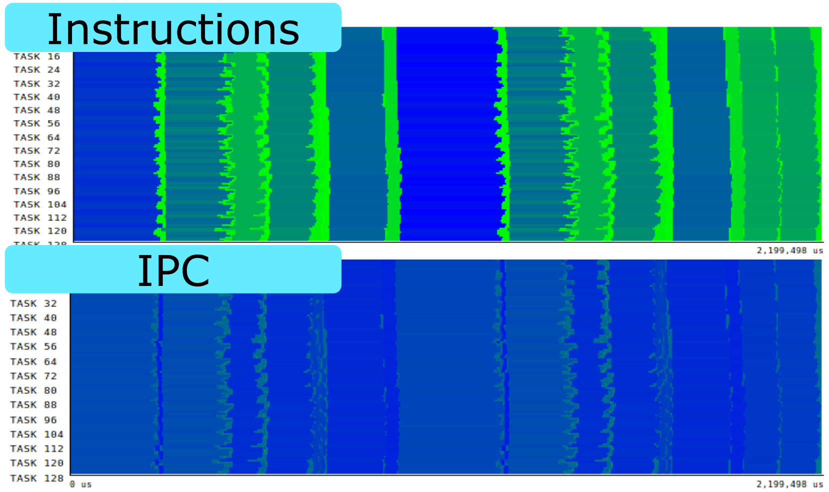
	256	512	1024
Global efficiency	84.32	83.48	84.28
-- Parallel efficiency	84.32	82.79	83.15
-- Load balance	93.51	93.33	93.30
-- Communication efficiency	90.17	88.71	89.11
-- Serialization efficiency	90.79	89.26	89.76
-- Transfer efficiency	99.32	99.38	99.28
-- Computation scalability	100.00	100.83	101.36
-- IPC scalability	100.00	101.02	101.43
-- Instruction scalability	100.00	99.91	99.86
-- Frequency scalability	100.00	99.90	100.07

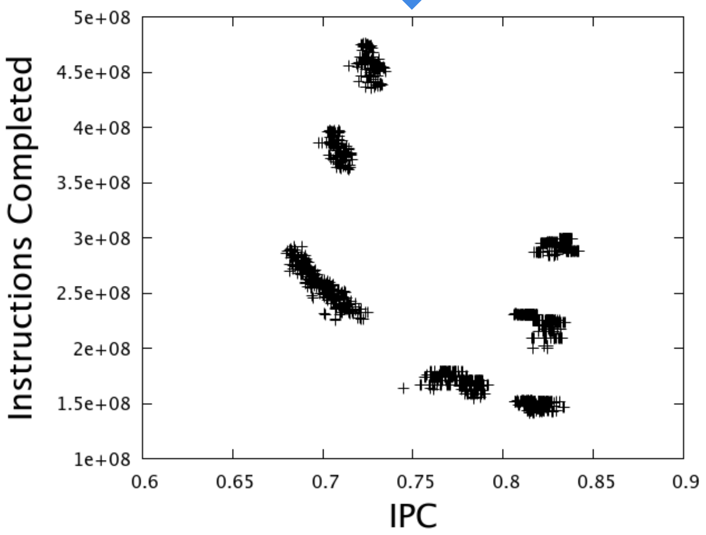
Percentage(%): 0 – 20 – 40 – 60 – 80 – 100

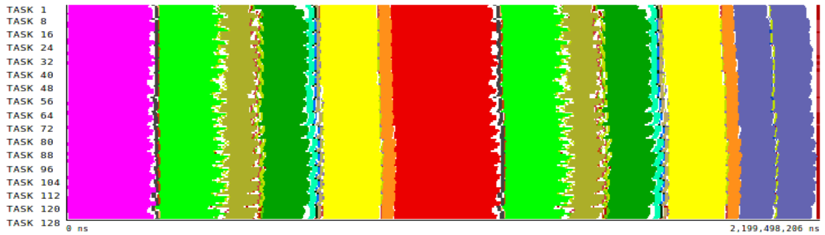
Comparing phases

	512[1]	512[2]	512[3]	512[4]
Parallel efficiency	85.27	44.77	81.89	72.25
-- Load balance	86.08	88.13	82.82	74.97
-- Communication efficiency	99.05	50.80	98.88	96.38
-- Serialization efficiency	99.19	51.07	98.89	97.68
-- Transfer efficiency	99.86	99.47	99.99	98.66

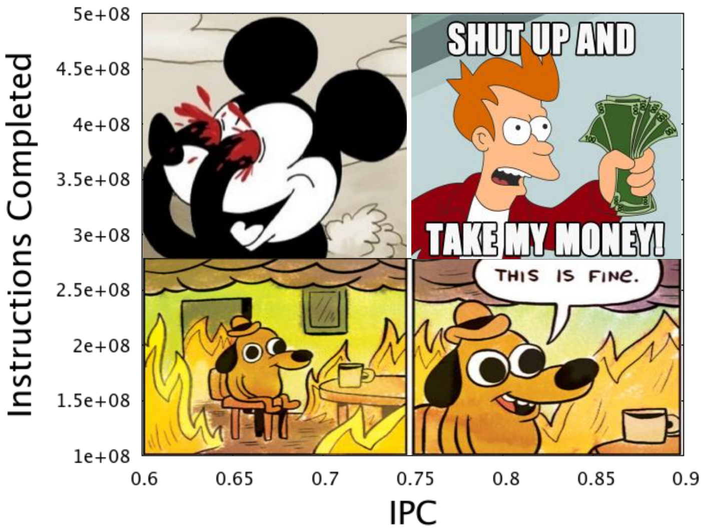
Clustering to identify structure



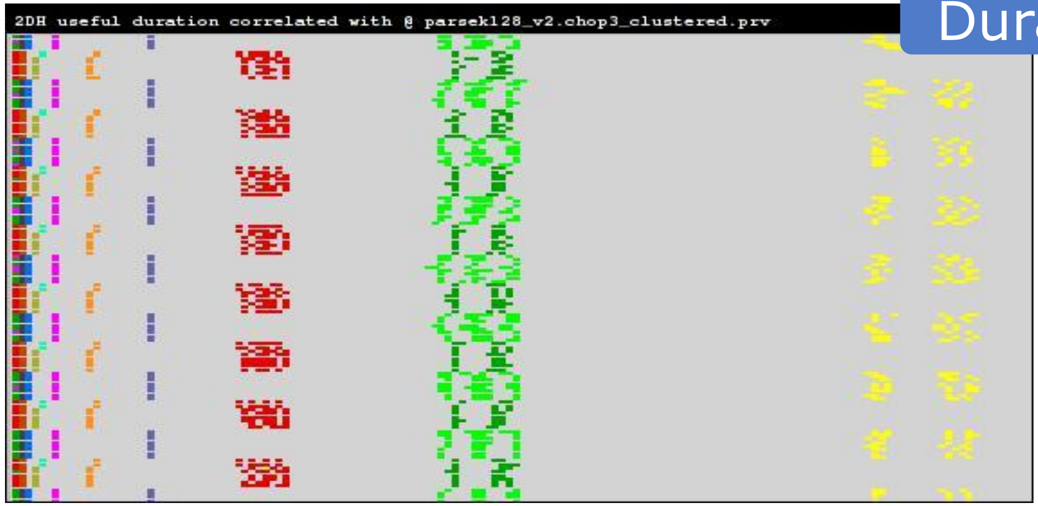




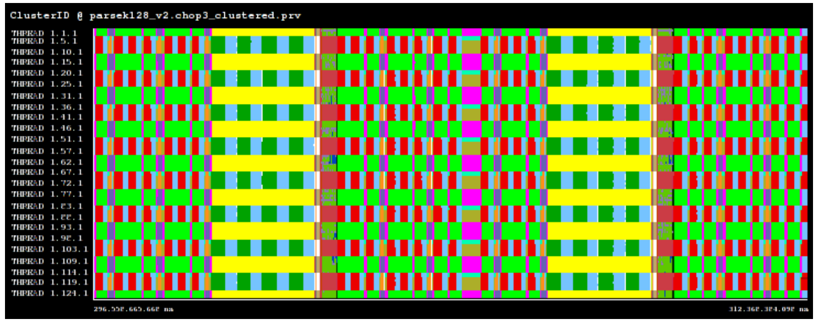
Quick insight into program's behavior

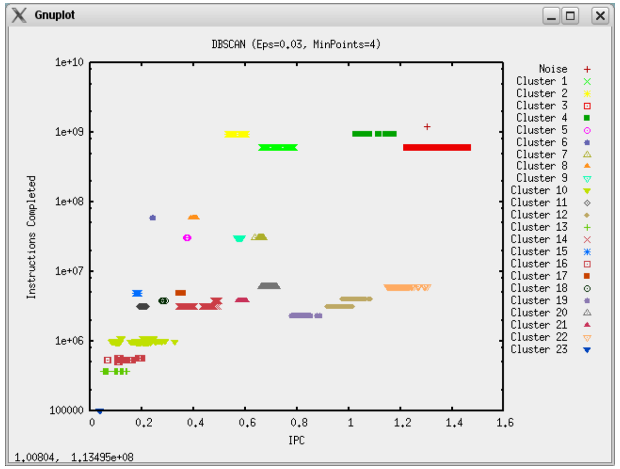


Correlate clusters, histograms & timelines



Duration vs. Cluster



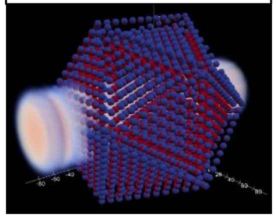


Instructions vs. Cluster

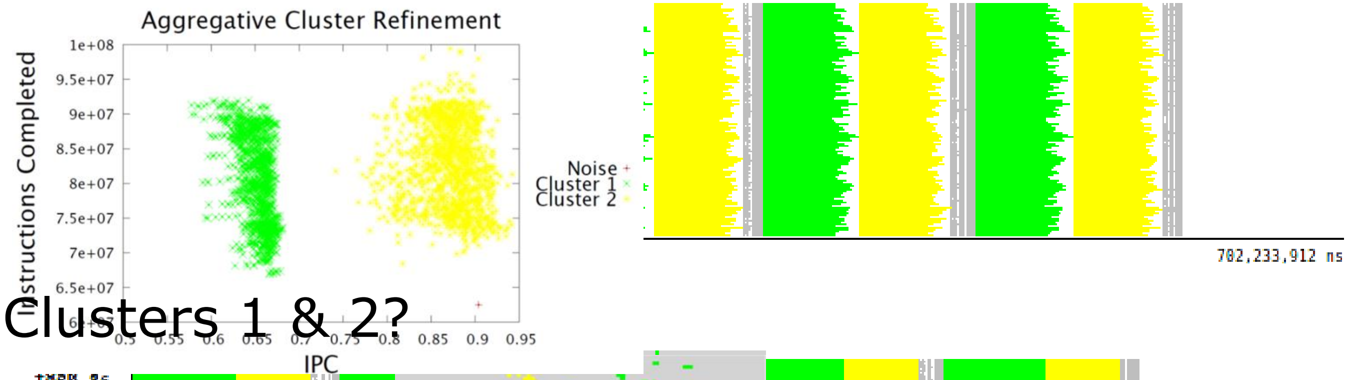
Integrating models and analytics

- What if…

PEPC

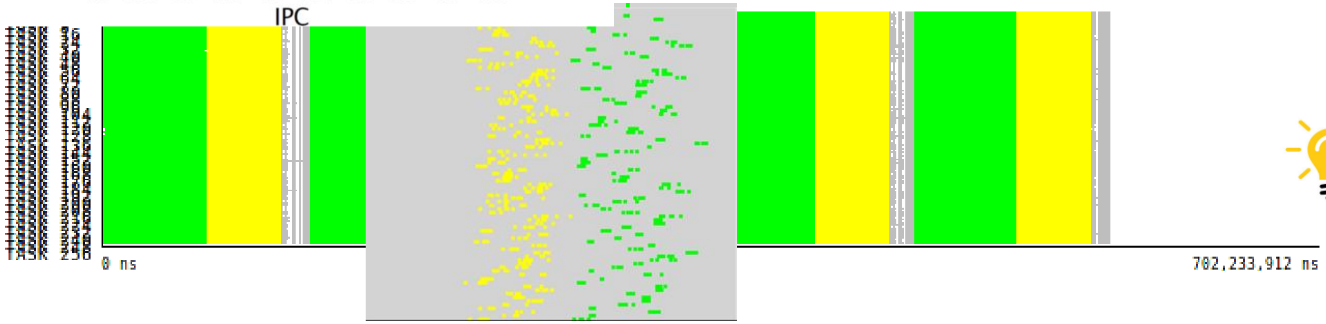


… we increase the IPC of Cluster1?



13%

… we balance Clusters 1 & 2?

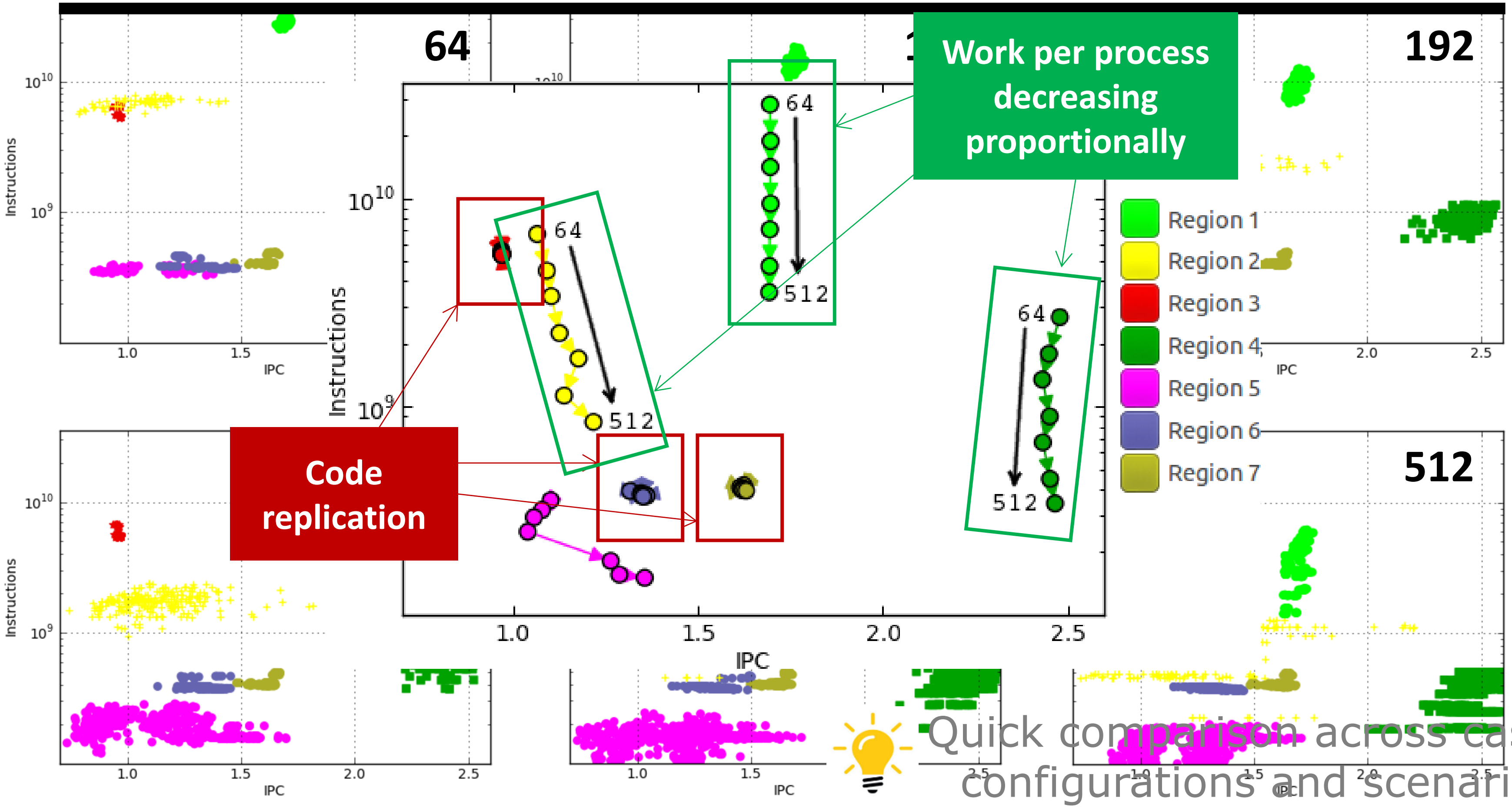


19%

Know where effort pays off.

Tracking scalability through clustering

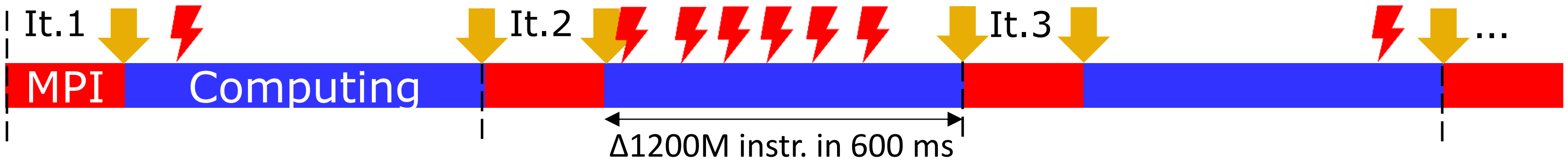
- Analyze scalability of computing regions across 64 – 512 tasks

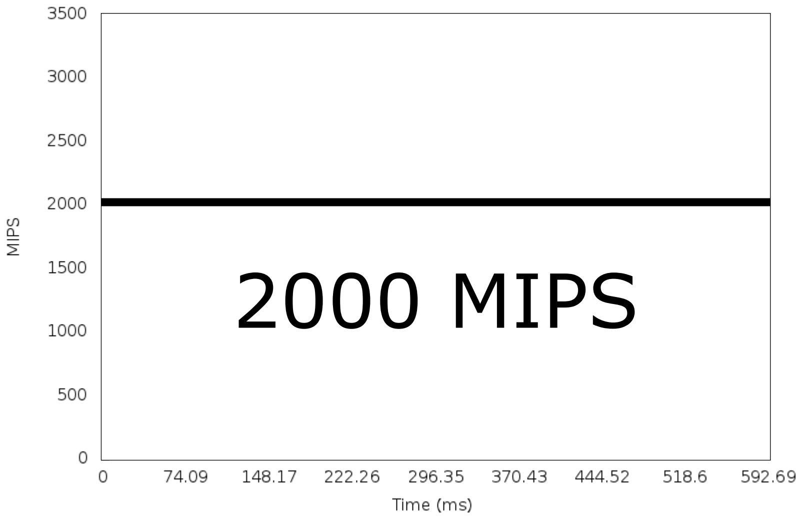


Quick comparison across cases, configurations and scenarios

Folding to increase details

- What is the performance of a single serial region?





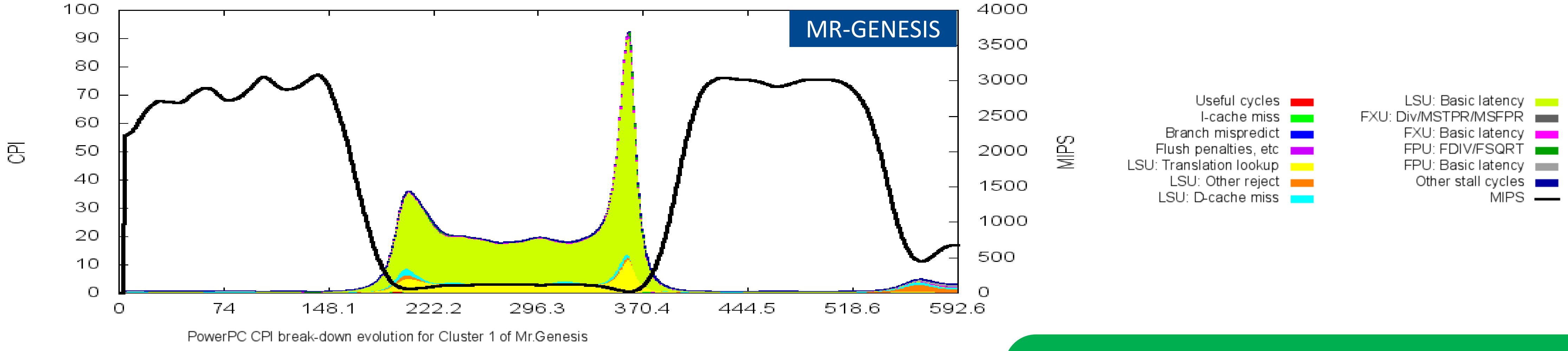
Instrumentation

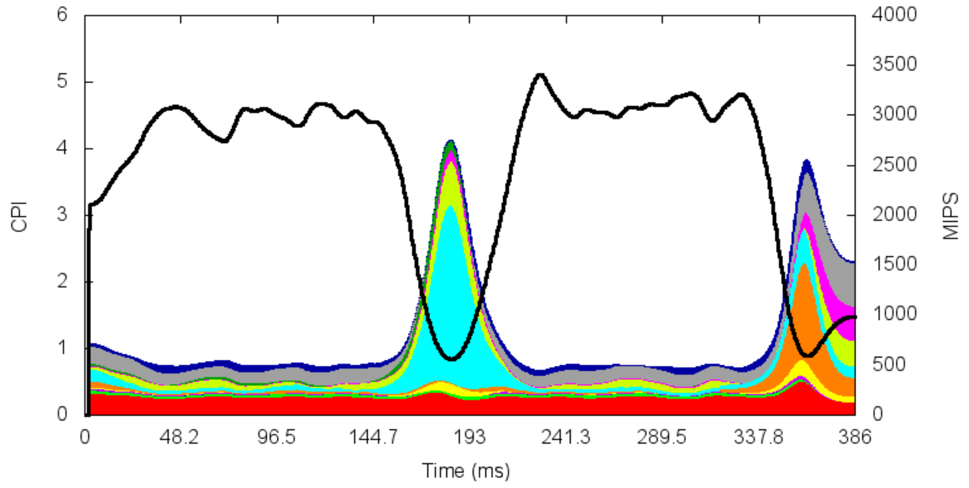
+ Sampling

Reveals sub-phases in high detail

"Fold" similar iterations into a single, highly detailed synthetic iteration

Folding: CPI and HWC stack models



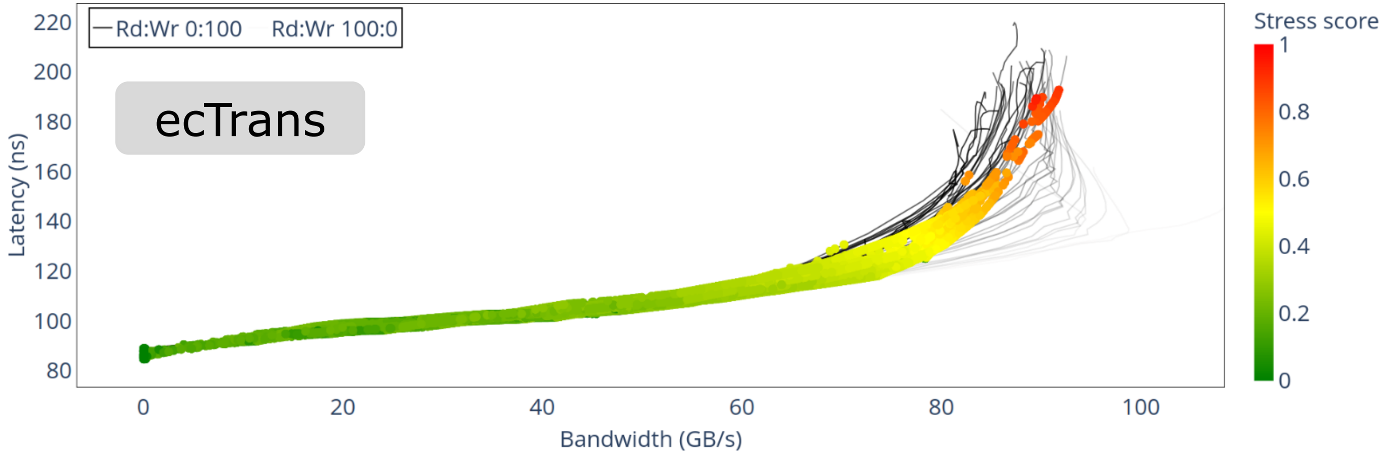


- Trivial fix (loop interchange)
- 1-line code fix → 25% boost
- Easy to locate

- Availability of CPI stack models for production processors

Understanding memory influence

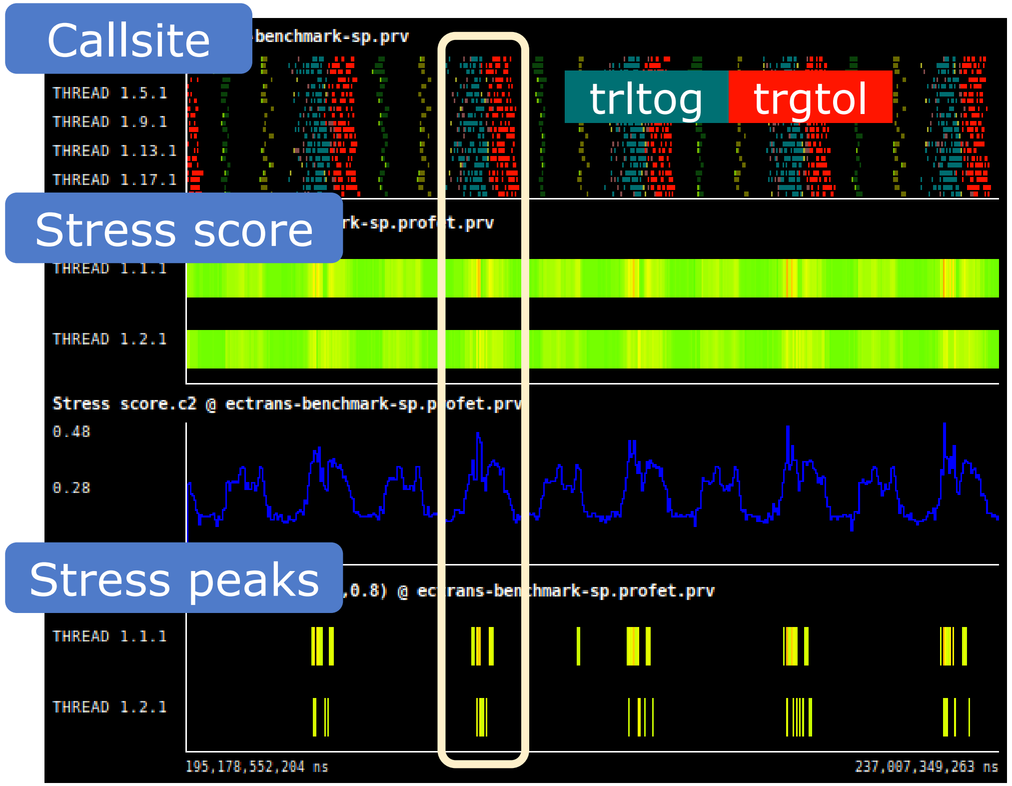
- Mess: Bandwidth-latency curves describe memory performance from unloaded to fully saturated states
- Integration with Paraver: Easily identify where memory stress is highest



Memory stress @ ectrans-benchmark-sp.profet.prv

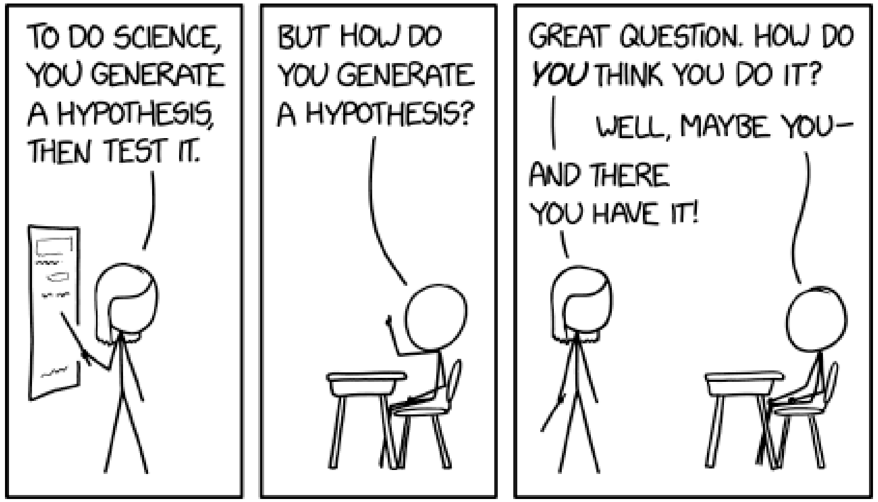
	[0.00..0.10)	[0.10..0.20)	[0.20..0.30)	[0.30..0.40)	[0.40..0.50)	[0.50..0.60)	[0.60..0.70)	[0.70..0.80)	[0.80..0.90)	[0.90..1.00]
THREAD 1.1.1	0.46 %	22.92 %	45.57 %	28.06 %	2.50 %	0.34 %	0.03 %	0.01 %	0.11 %	0.00 %
THREAD 1.2.1	0.38 %	20.48 %	55.90 %	22.57 %	0.53 %	0.03 %	0.00 %	0.02 %	0.09 %	0.00 %
Average	0.42 %	21.70 %	50.73 %	25.31 %	1.52 %	0.18 %	0.02 %	0.01 %	0.10 %	0.00 %
Avg/Max	0.92	0.95	0.91	0.90	0.61	0.54	0.51	0.77	0.94	0.60

- Prediction of future ones (PROFET)
- Cache-Aware Roofline Model (CARM)



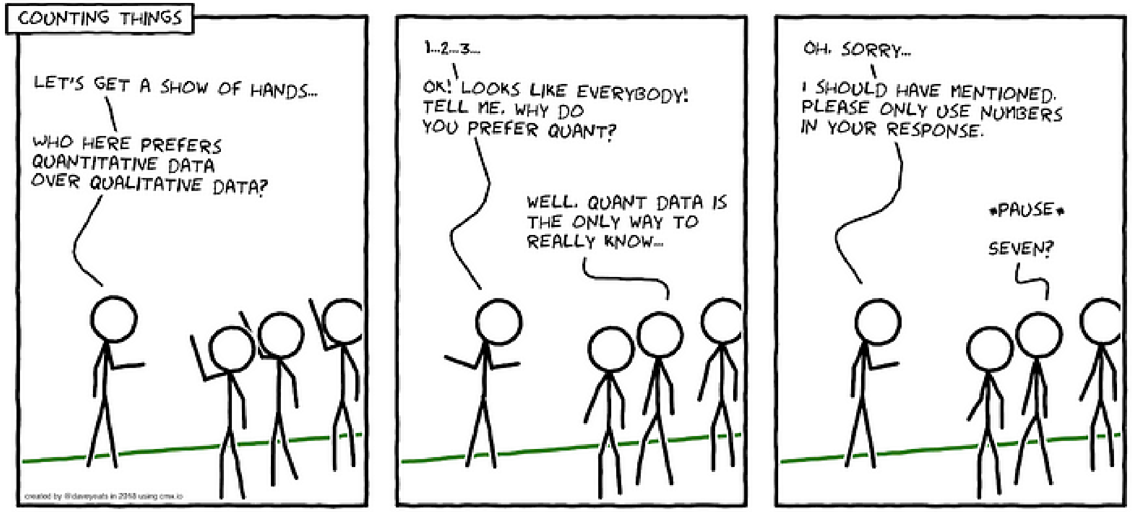
Methodology

BSC Performance Tools aim



- Be your copilot in the process of generating & validating hypotheses
  - The tool is the “wheel” and you “drive it”

- Provide quantitative (how much) + qualitative (why/how)
  - Patterns, shapes, structure… beyond raw numbers “for a better understanding”



First steps of analysis

- **Parallel Efficiency: Parallel resources are mostly doing useful work?**
  - Load Balance → Work (programmer's fault)? Performance (machine's fault)?
  - Serialization → Dependency chains? Sequence of MPI calls? Noise?
  - Transfer → Bandwidth or Latency? Simulations with Dimemas
- **Serial Efficiency: How far from peak performance?**
  - IPC → Cache effects? Correlate with other counters
  - Frequency → Cycles per us? Multi-core sharing? Dynamic scaling? Power?
- **Scalability: Benefit from additional resources?**
  - Code replication → Total instructions?
- **Variability?**
- **Behavioral Structure → Analytics**

Paraver Tutorials: Introduction to Paraver & Dimemas methodology

BSC Tools Website & Contact

- https://tools.bsc.es

Open Source

Downloads

- Sources & Binaries

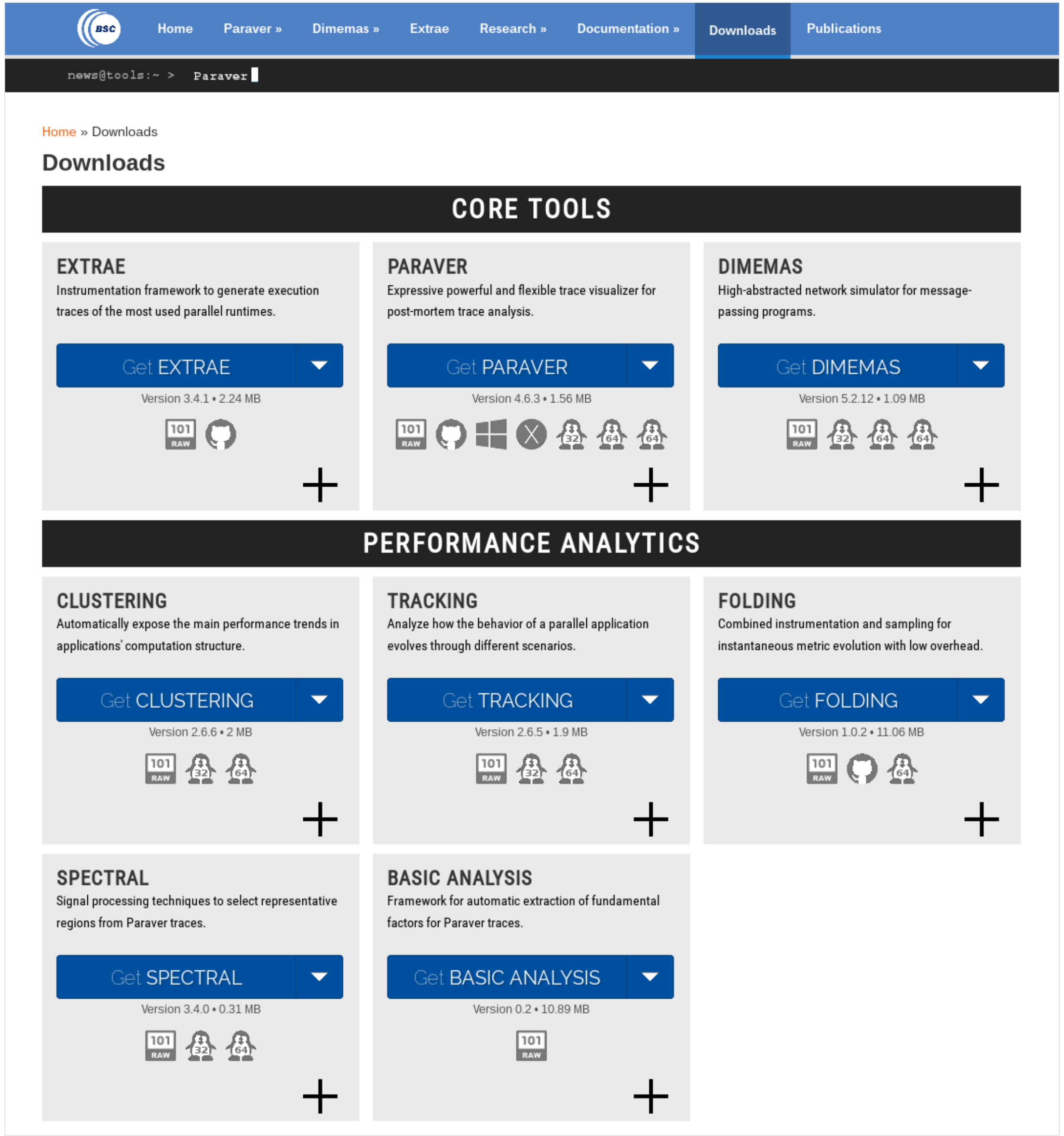
Documentation

- Training guides
- Tutorial slides

tools@bsc.es

Quick Start

- Start wxparaver
- Help → Tutorials
- Follow training guides



Takeaway

- The importance of understanding
  → **Keep asking questions**
- Use your brain
  → **Use visual tools**
- The devil is in the details
  → **Do not miss them**
- Don’t over-theorize about your code
  → **Look at it**

Performance Optimisation and Productivity 3

A Centre of Excellence in HPC

- Free performance assessment services for European users
- Request your assessment here

Contact:

https://www.pop-coe.eu
pop@bsc.es
youtube.com/POPHPC

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101143931. The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, France, Portugal and the Czech Republic.