Understanding applications with BSC Tools

Germán Llort, Lau Mercadal

✉ tools@bsc.es

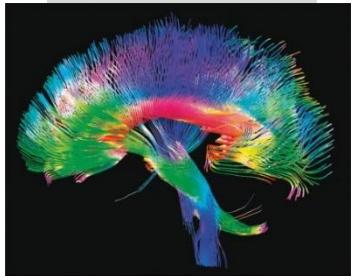
Barcelona Supercomputing Center

Performance Optimisation and Productivity 3 (101143931)

BSC Tools

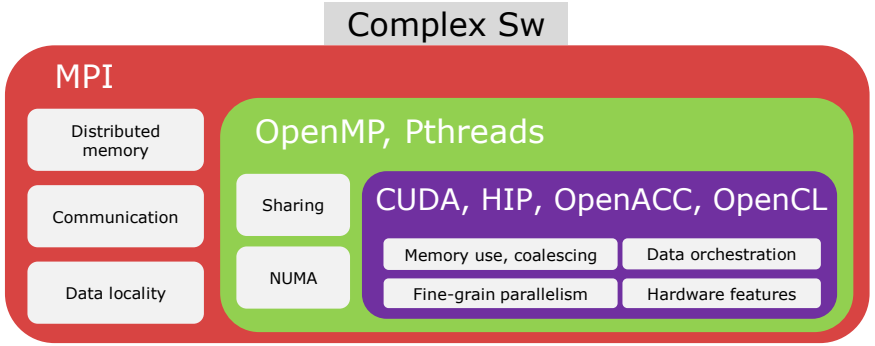
- Since 1991
- Based on traces
- Open Source
  - https://tools.bsc.es
- Core tools:
  - **Extrae** – instrumentation
  - **Paraver** – offline trace analysis
  - **Dimemas** – message passing simulator
- + Performance Analytics → From data to insight
- Focus
  - Detail, variability, flexibility
  - Behavioural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

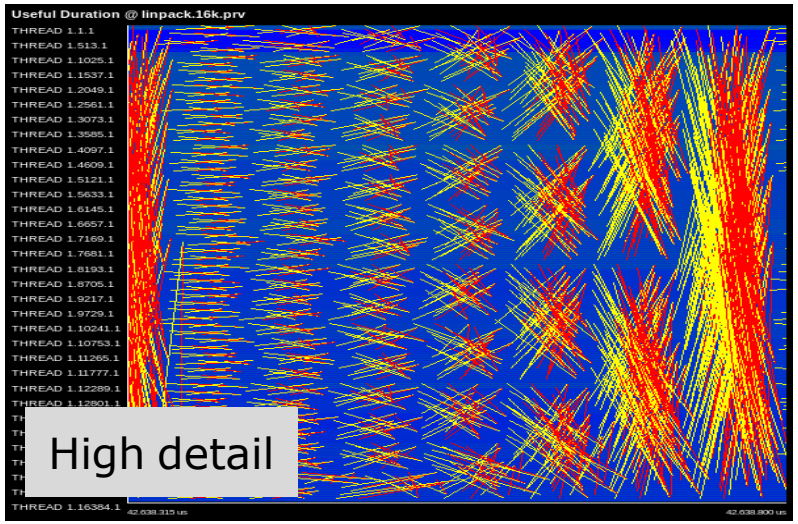
Complex Apps



Complex Hw

Complex Sw

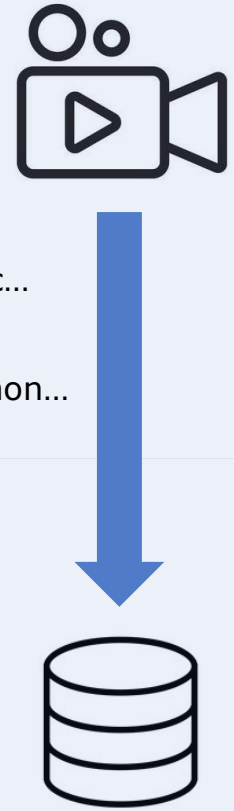




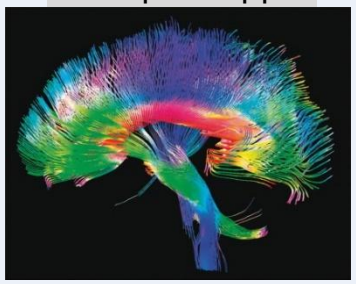
Extrae

Extrae – Tracing framework

- Records parallel program execution behaviour over time
- Generates Paraver traces
  - Platforms
    - Intel, ARM, RISC-V, Cray, BlueGene, MIC, Android, Fujitsu Sparc…
  - Parallel programming models
    - MPI, OpenMP, pthreads, OmpSs, CUDA, HIP, OpenCL, Java, Python…
  - Hardware counters
    - Using PAPI interface
  - Link to source code
    - Callstack at MPI routines
    - OpenMP outlined routines
    - Selected user functions (Dyninst, compiler instrumentation)
  - Periodic sampling
  - User events anywhere in your program (Extrae API)

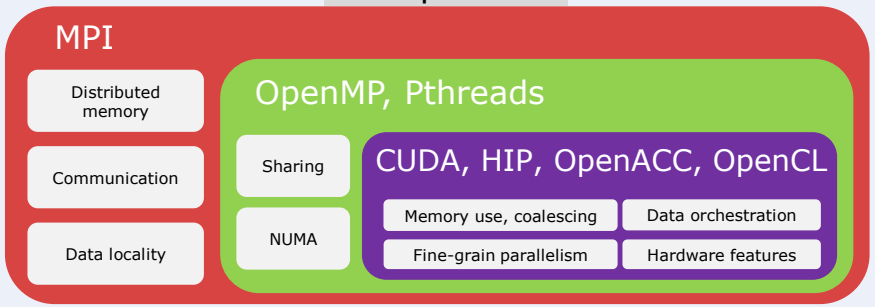


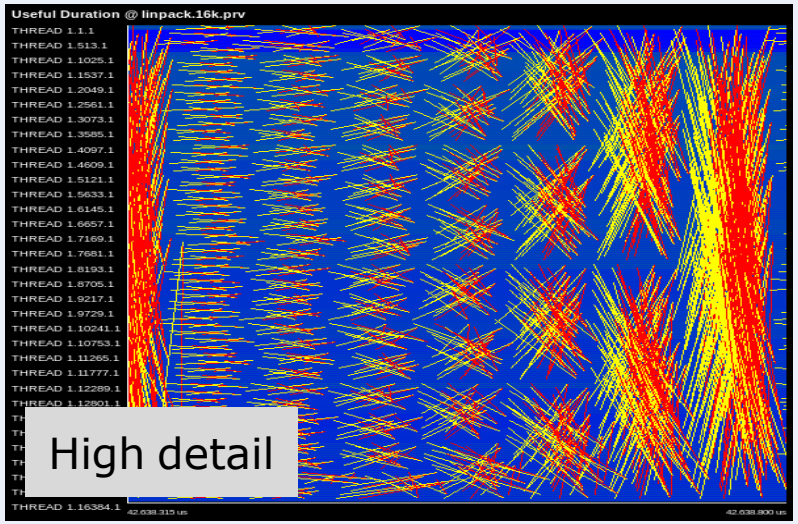
Complex Apps



Complex Hw

Complex Sw





Extrae – Tracing framework

- Records parallel program execution behaviour over time
- Generates Paraver traces
  - Platforms
    - Intel, ARM, RISC-V, Cray, BlueGene, MIC, Android, Fujitsu Sparc…
  - Parallel programming models
    - MPI, OpenMP, pthreads, OmpSs, CUDA, HIP, OpenCL, Java, Python…
  - Hardware counters
    - Using PAPI interface
  - Link to source code
    - Callstack at MPI routines
    - OpenMP outlined routines
    - Selected user functions (Dyninst, compiler instrumentation)
  - Periodic sampling
  - User events anywhere in your program (Extrae API)

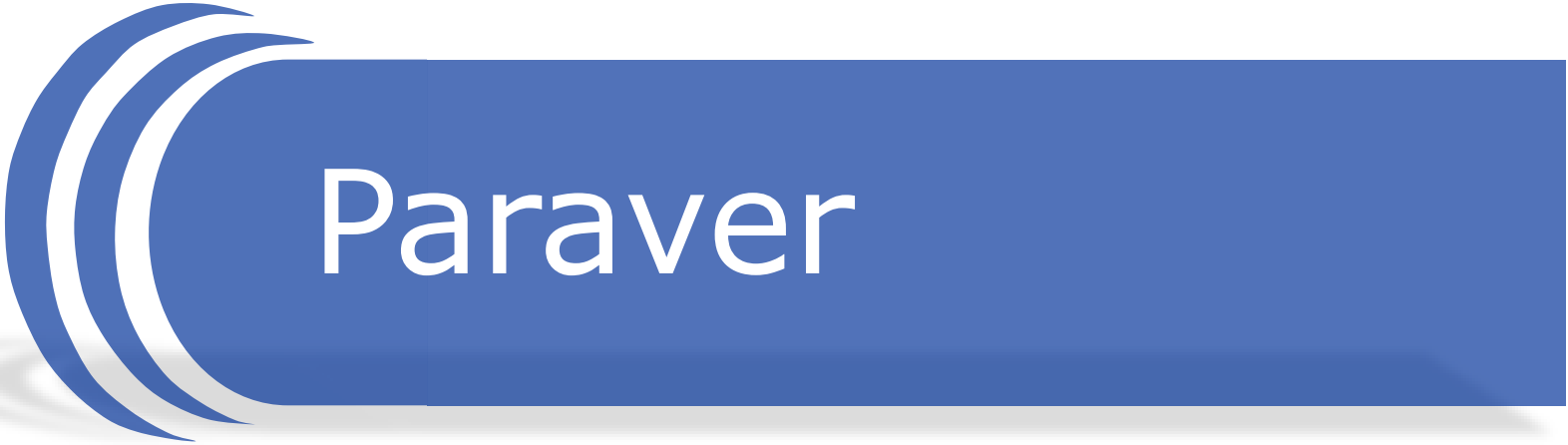
Fully automatic on production binaries: No need to recompile nor relink!

States
Events
Comms

```
#Paraver (12/06/2026 at
10:00):15000000000_ns:...

1:151:1:147:1:294672:294676:1
2:151:1:147:1:294672:50000001:0
3:151:1:147:1:294623:294672:158:1:154:1:294670
:297419:4223536:8
...
```

ASCII, open format: Easy to generate, parse and script in workflows

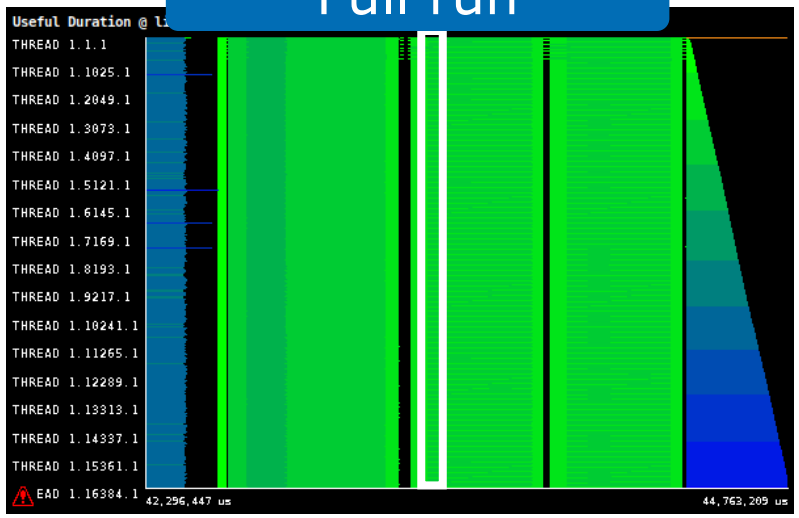


Paraver

Paraver – Performance data browser

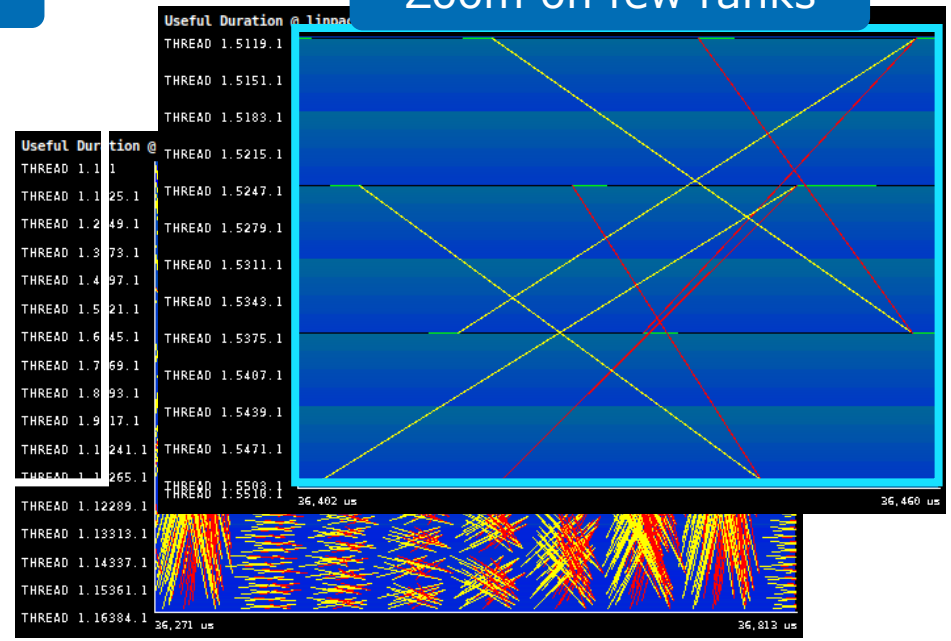
- Visualization, navigation, selection, zoom → analysis from macro to micro

Full run



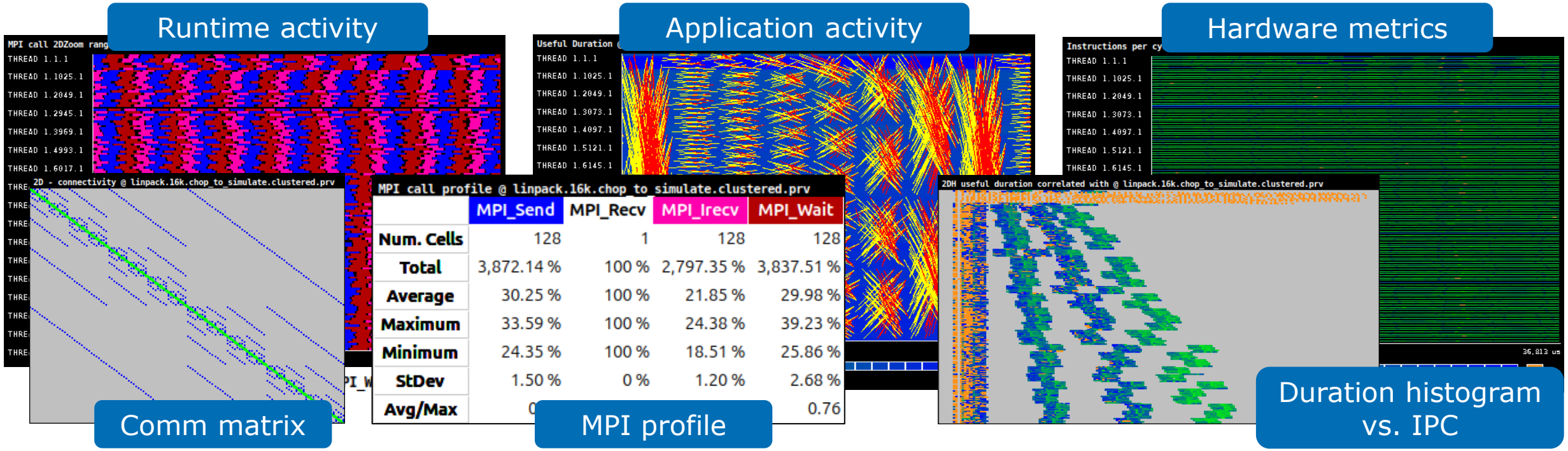
Zoom on 1 iteration

Zoom on few ranks



Paraver – Performance data browser

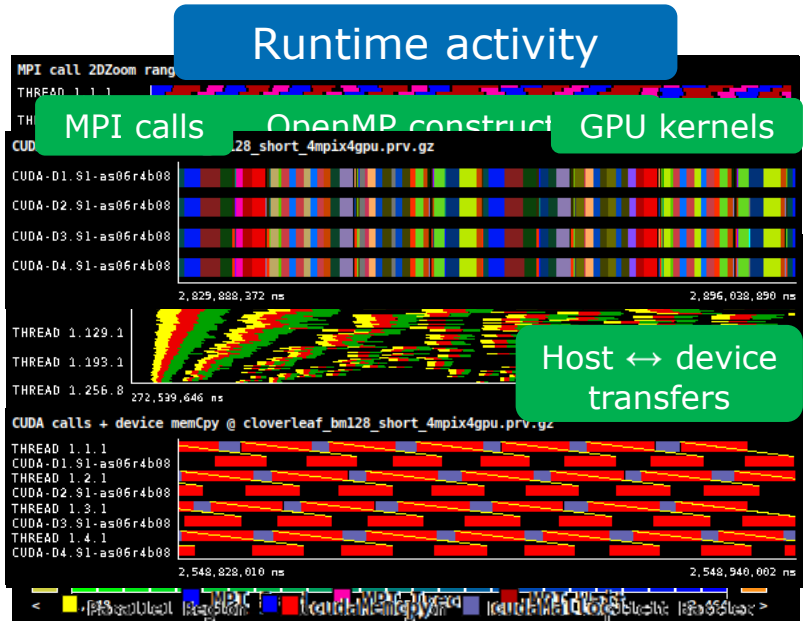
- Visualization, navigation, selection, zoom → analysis from macro to micro
- Multi-spectral view of application, runtime, and hardware activity
- Timelines, 2/3D tables, statistics and histograms

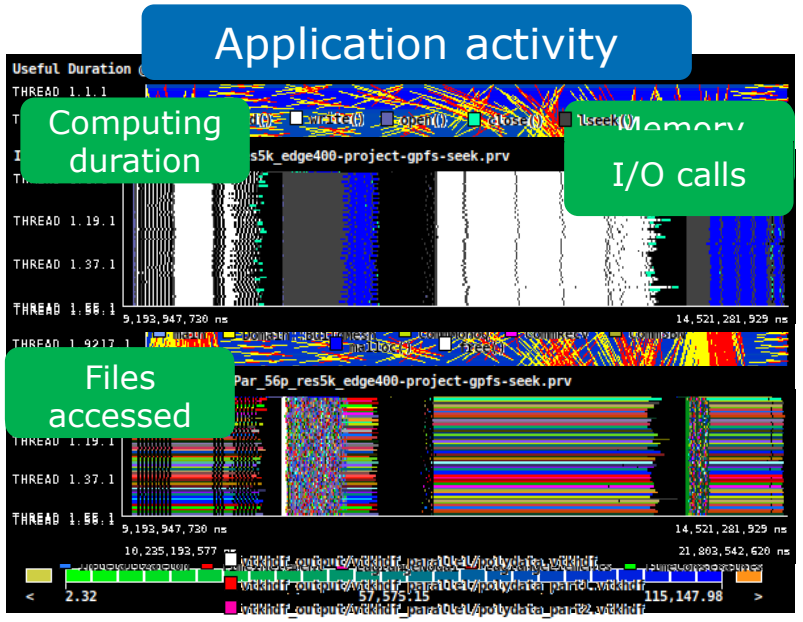


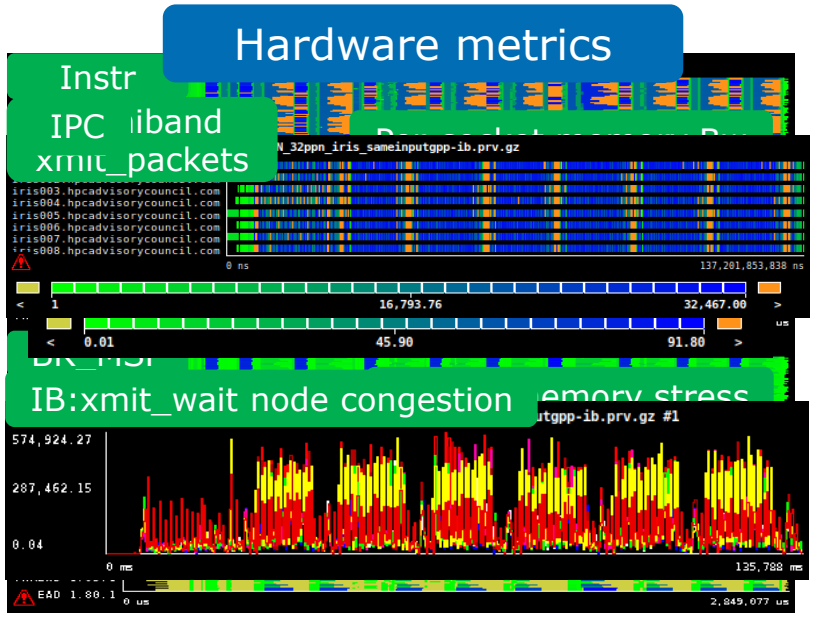
	MPI_Send	MPI_Recv	MPI_Irecv	MPI_Wait
Num. Cells	128	1	128	128
Total	3,872.14 %	100 %	2,797.35 %	3,837.51 %
Average	30.25 %	100 %	21.85 %	29.98 %
Maximum	33.59 %	100 %	24.38 %	39.23 %
Minimum	24.35 %	100 %	18.51 %	25.86 %
StDev	1.50 %	0 %	1.20 %	2.68 %
Avg/Max	0			0.76

Paraver – Performance data browser

- Visualization, navigation, selection, zoom → analysis from macro to micro
- Multi-spectral view of application, runtime, and hardware activity
- Timelines, 2/3D tables, statistics and histograms
- Any trace, any semantics → Unified visualization & analysis across domains





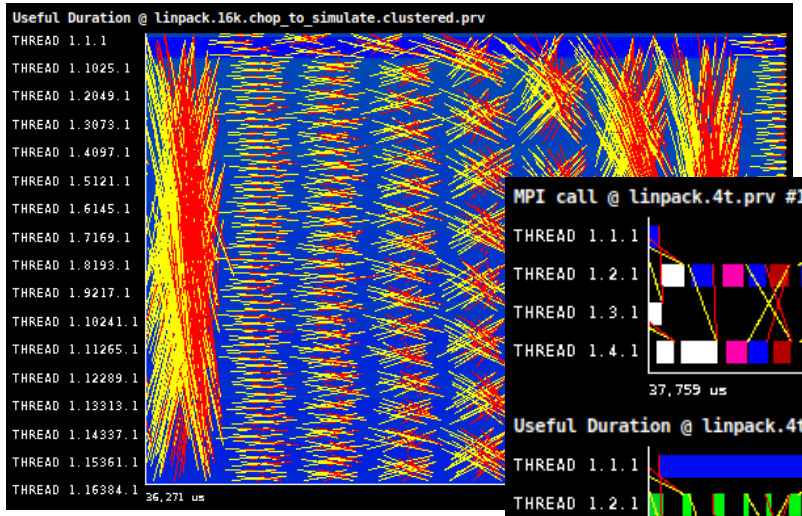


Paraver – Performance data browser

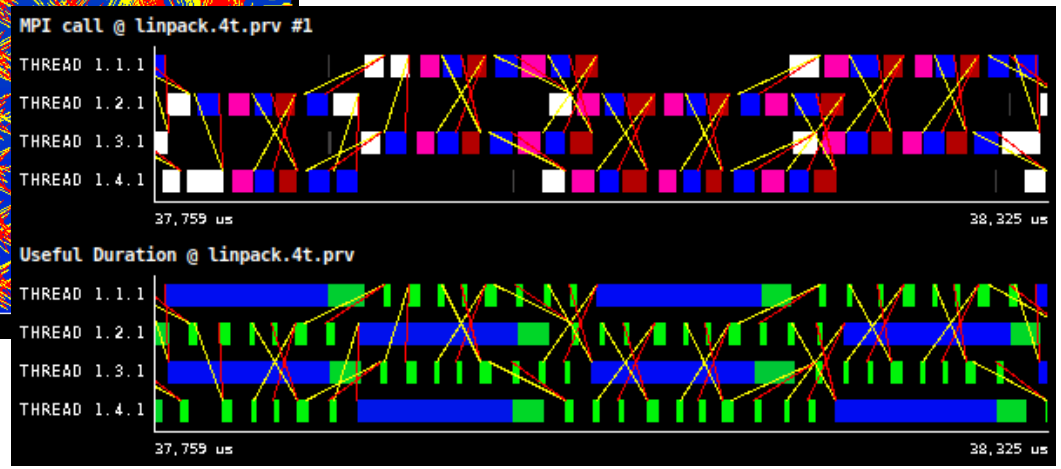
- Visualization, navigation, selection, zoom → analysis from macro to micro
- Multi-spectral view of application, runtime, and hardware activity
- Timelines, 2/3D tables, statistics and histograms
- Any trace, any semantics → Unified visualization & analysis across domains
- Comparative analysis across traces, from tiny to massive

16,384 vs 4 ranks
over **sync time span**

Keep synchronized: time, objects, size, duration, semantic, axis…



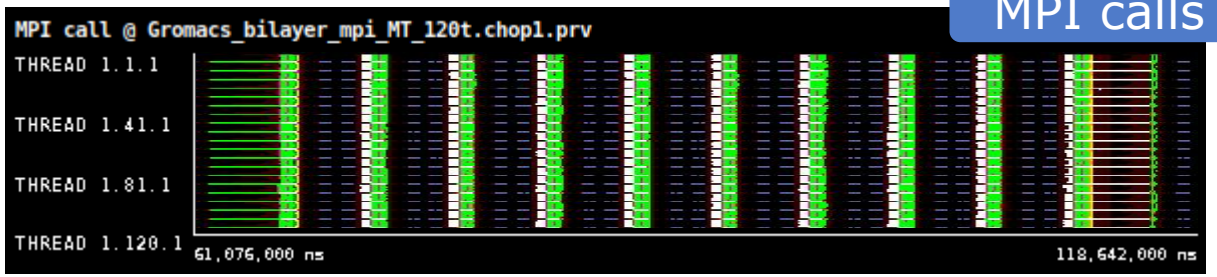
Computing vs MPI calls
over **sync time interval**



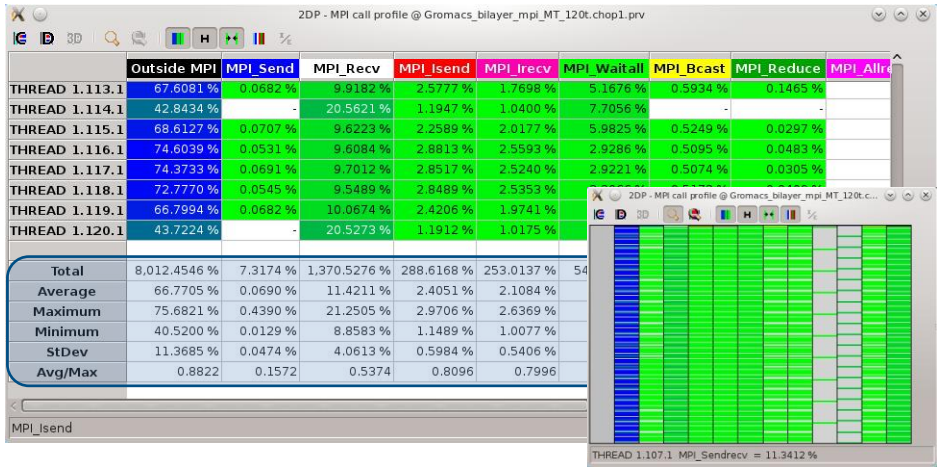
From timelines to tables

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

MPI calls

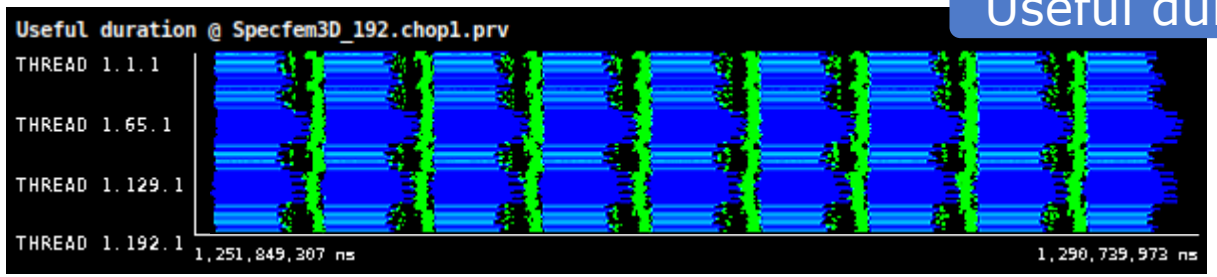


MPI calls profile

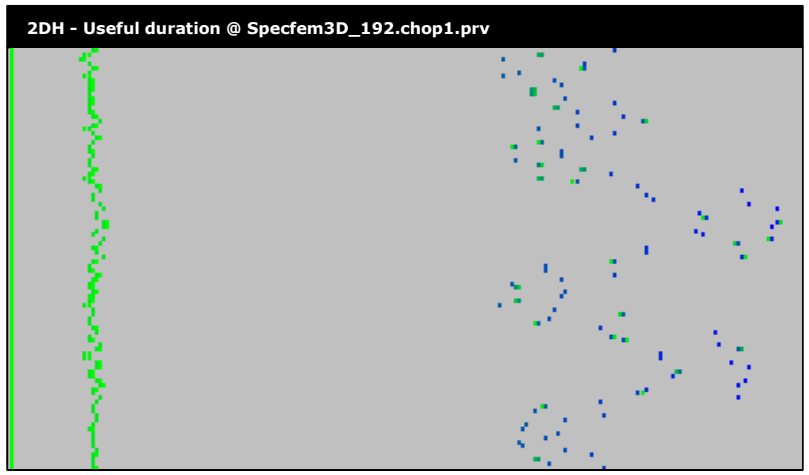


- Continuous metrics → gradients
  - **Green** to **blue** for **low** to **high** values

Useful duration

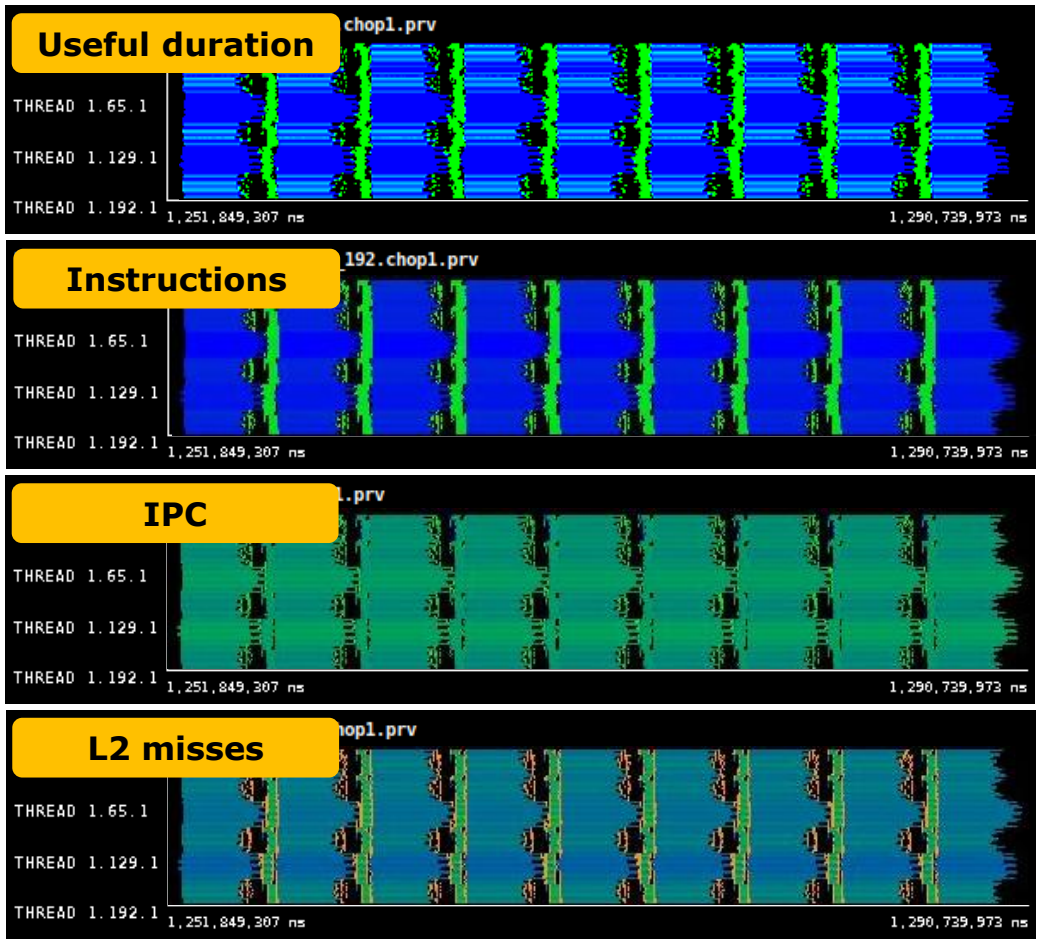


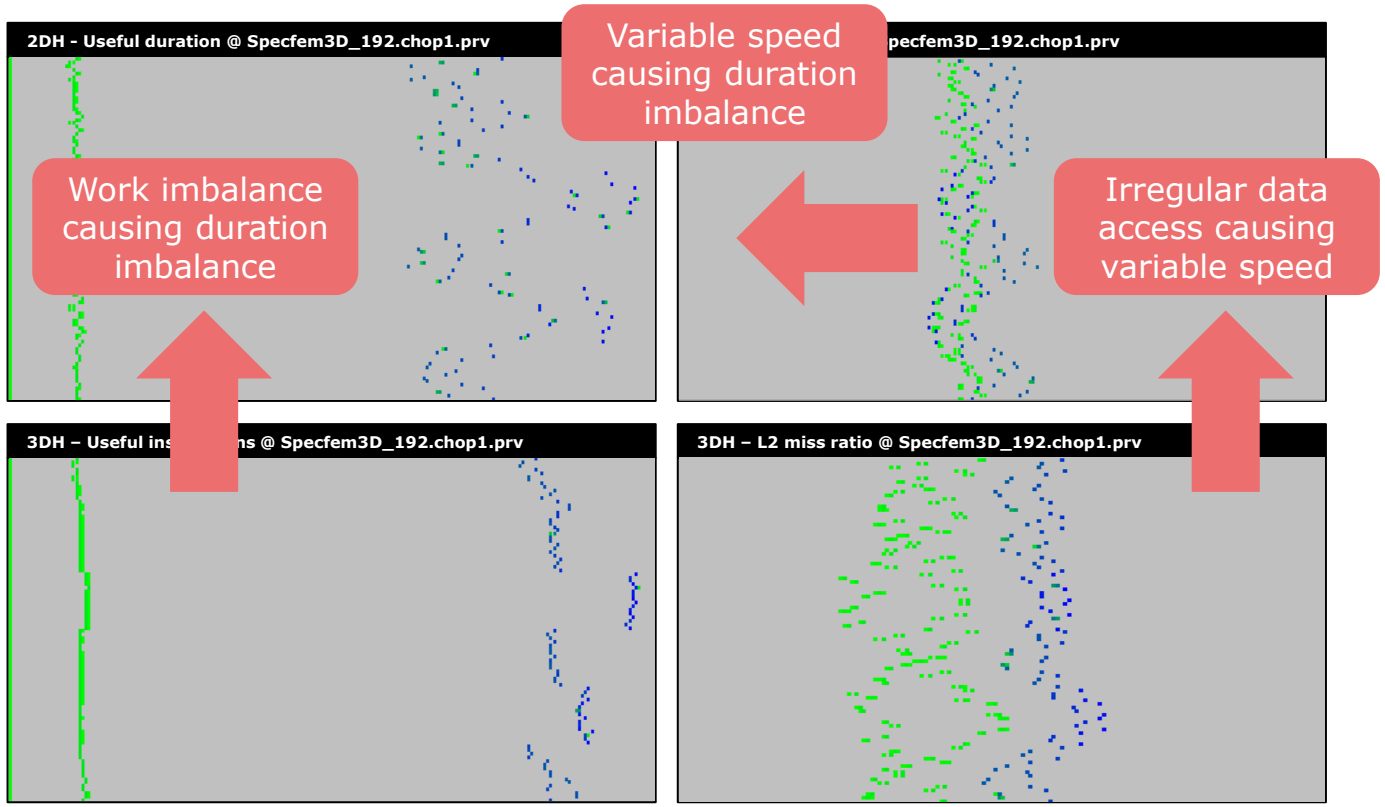
Compute-time histogram



Analyzing variability through timelines and histograms

Courtesy Dimitri Komatitsch



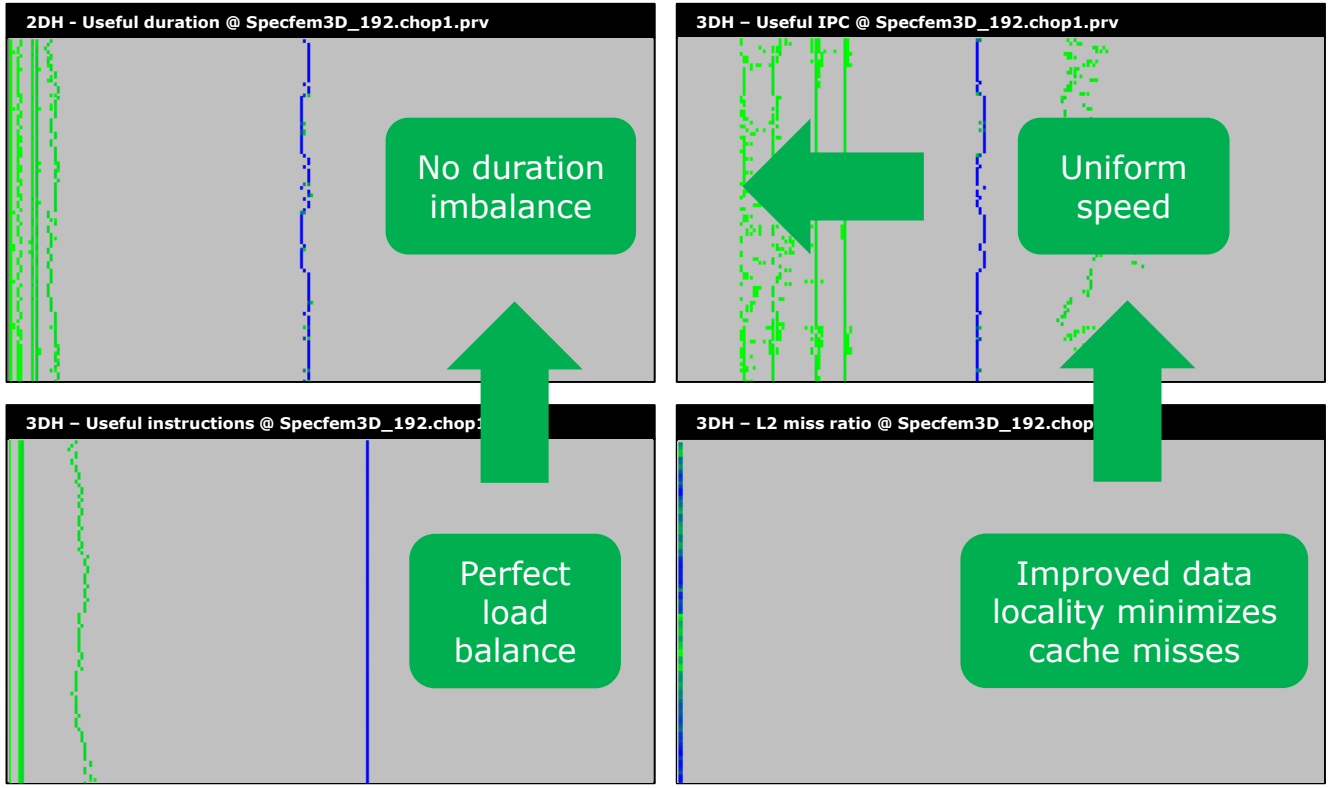


Analyzing variability through histograms and timelines

Courtesy Dimitri Komatitsch

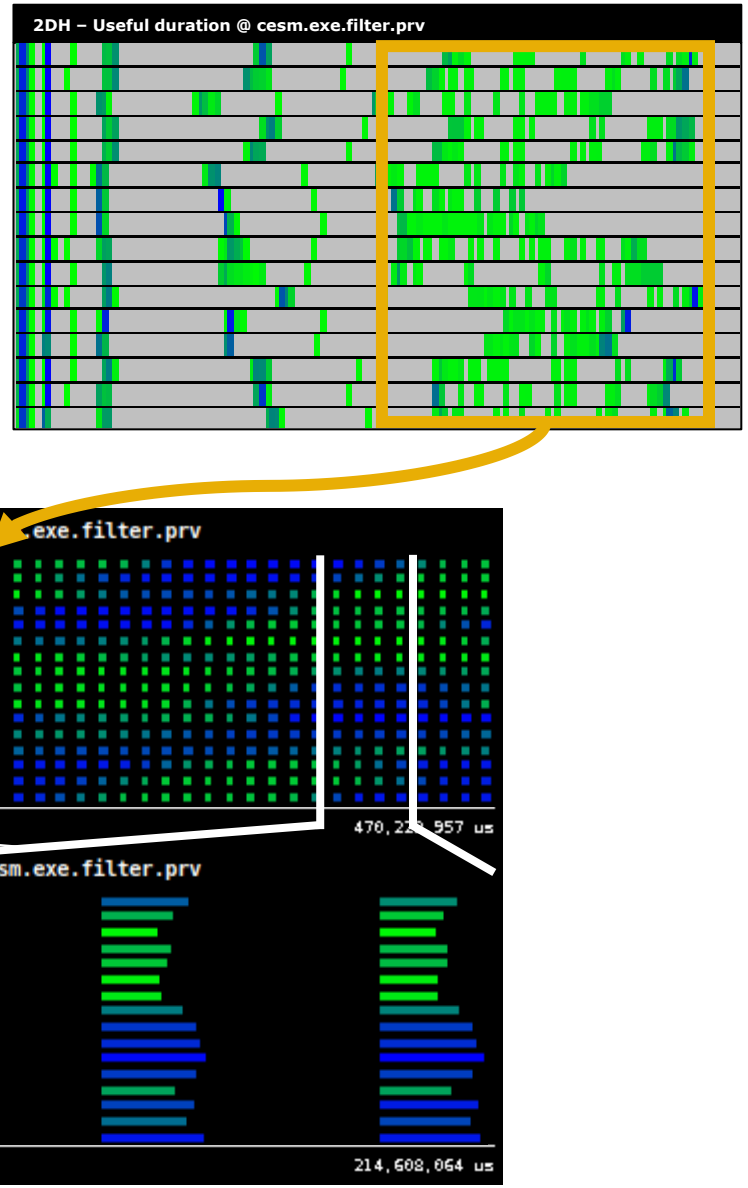
- A few months later…

Focus optimization where it matters



From tables to timelines

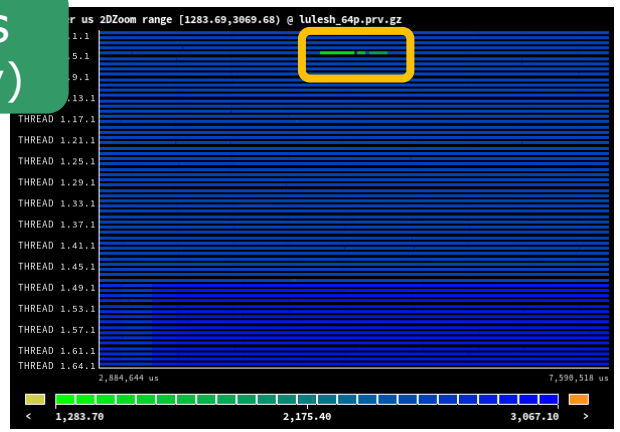
- CESM: 16 processes, 2-day simulation
- High variability in useful computation duration
- How is it distributed?
- Dynamic imbalance
  - In space and time
  - Day and night
  - Season?



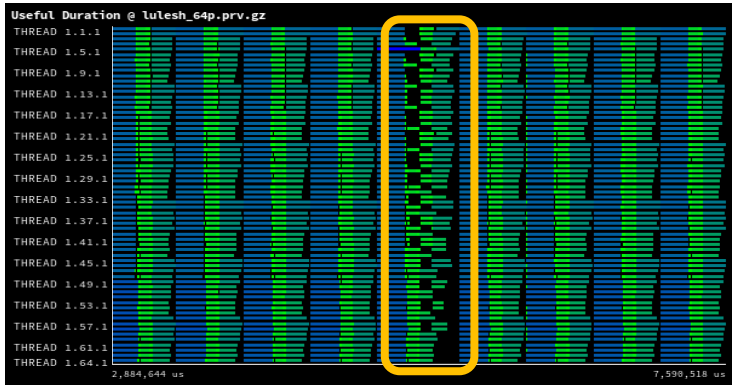
The Butterfly Effect…

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

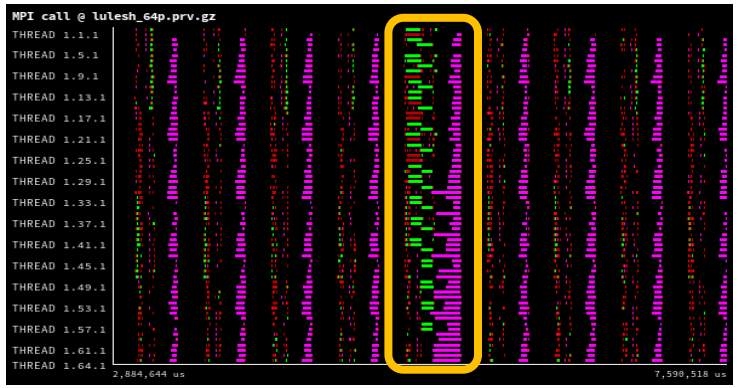
Cycles/us (Frequency)



Useful duration



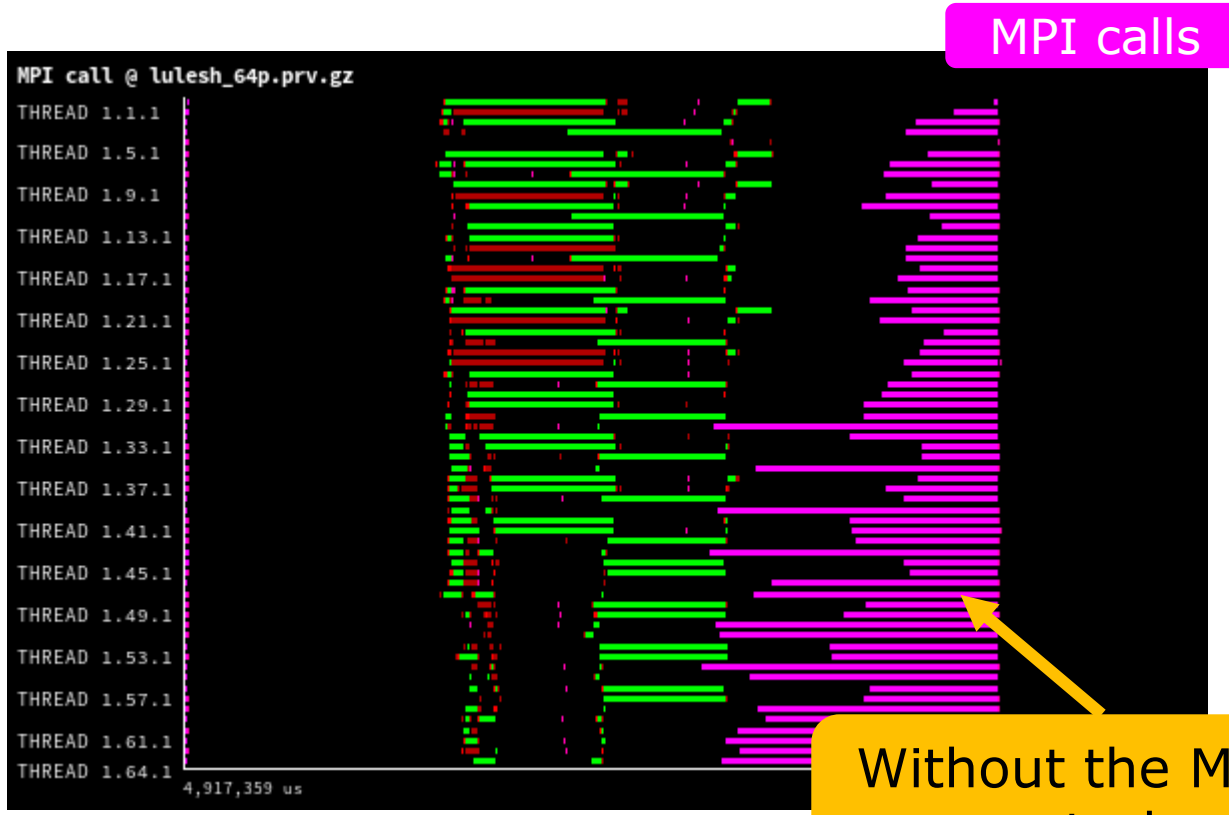
MPI calls



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

Outside MPI
MPI_Isend
MPI_Irecv
MPI_Wait
MPI_Waitall
MPI_Allreduce
MPI_Comm_rank

Microscopic effects with large global impact

Without the MPI_Allreduce it would perturb multiple iterations

From big to small traces

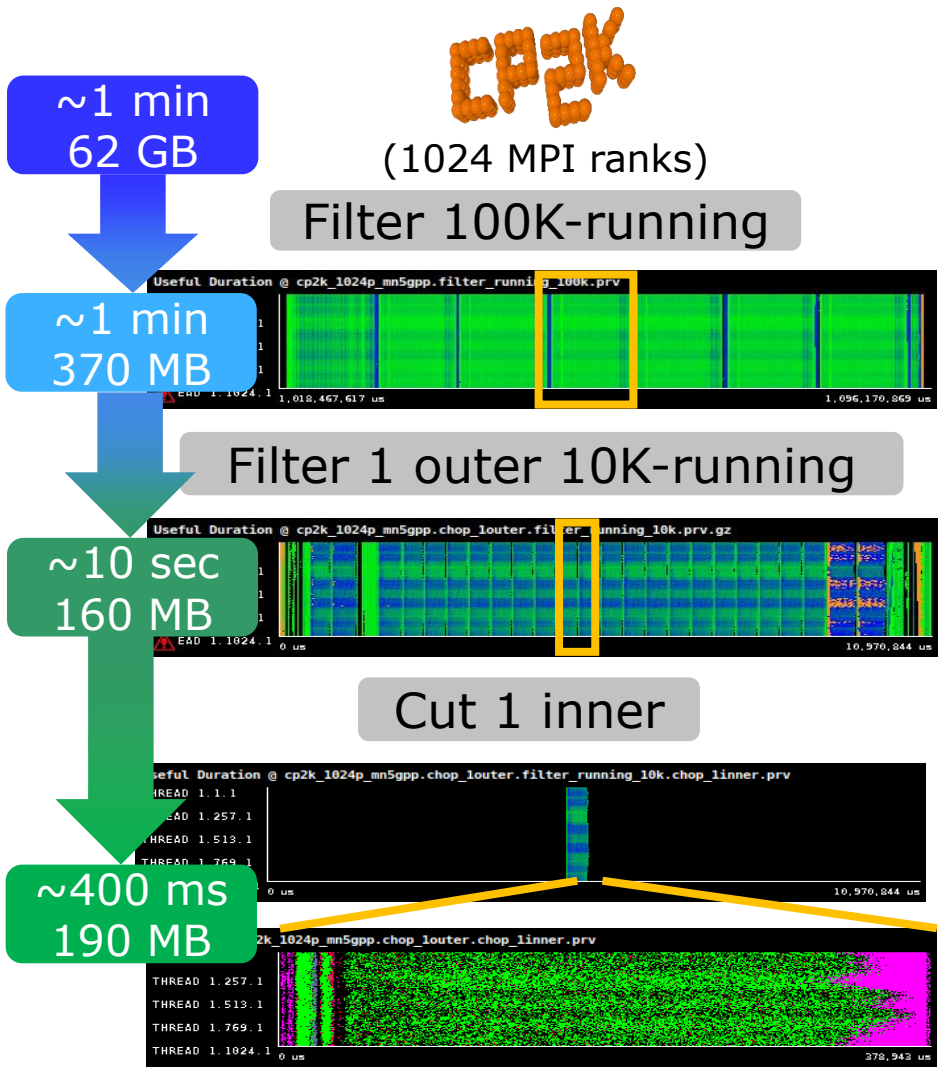
Post-mortem

…r: 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 s… CFGs (if needed data is kept)

On-line

…de: **Burst mode** keeps long computations + stats

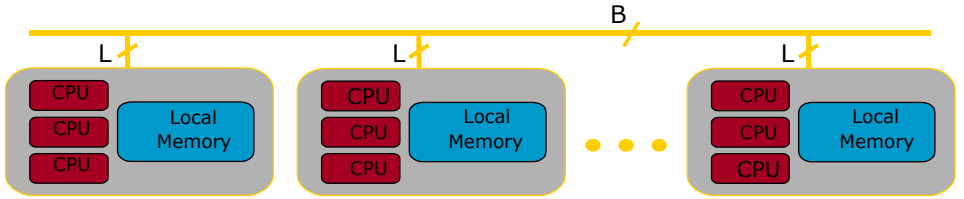


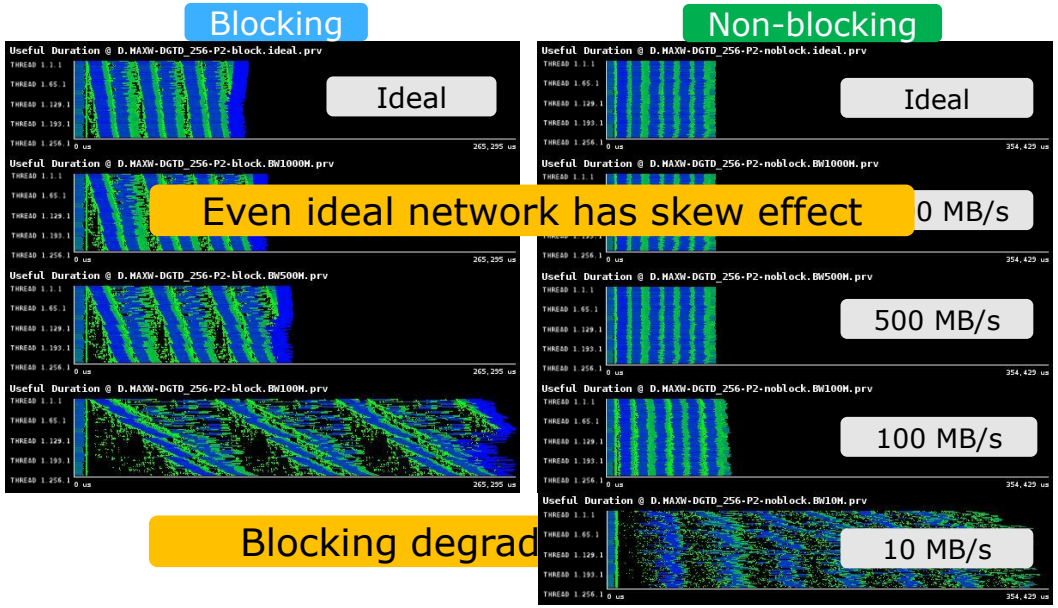
Dimemas

Dimemas: Coarse-grain, Trace-driven simulation

- **Fast simulation of an abstract interconnect**
  - SMP nodes with local memory for intra-node comms
  - Buses (B) model contention (concurrent messages)
  - Links (L) model connectivity (per-node traffic)
  - Local/remote Latency/Bandwidth
- **Ideal network: Bw=∞ Lat=0**
  - Highlights serializations
- **"What-if…" analysis**
  - Code & machine changes
- **Parametric sweeps**
  - Sistematic range exploration
- **Dimemas + Paraver**
  - Analysis + Detailed prediction

Simulation generates a trace → Detailed feedback





What if… we had asynchronous communications?

Courtesy Dimitri Komatitsch

Ideal simulation shows no benefit

Real

Ideal

Predict MN

Predict 100 MB/s

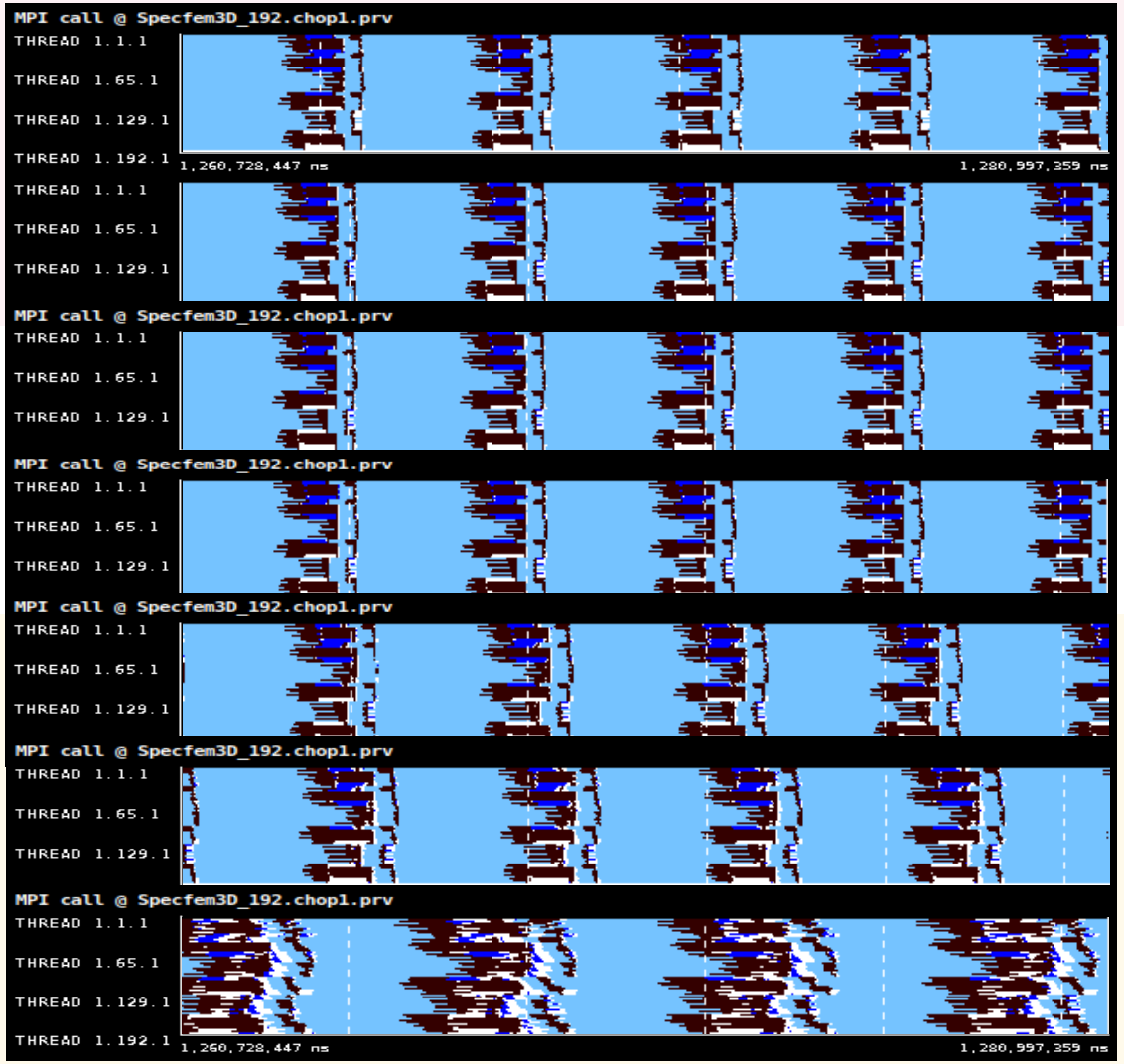
Predict 10 MB/s

Predict 5 MB/s

Predict 1 MB/s

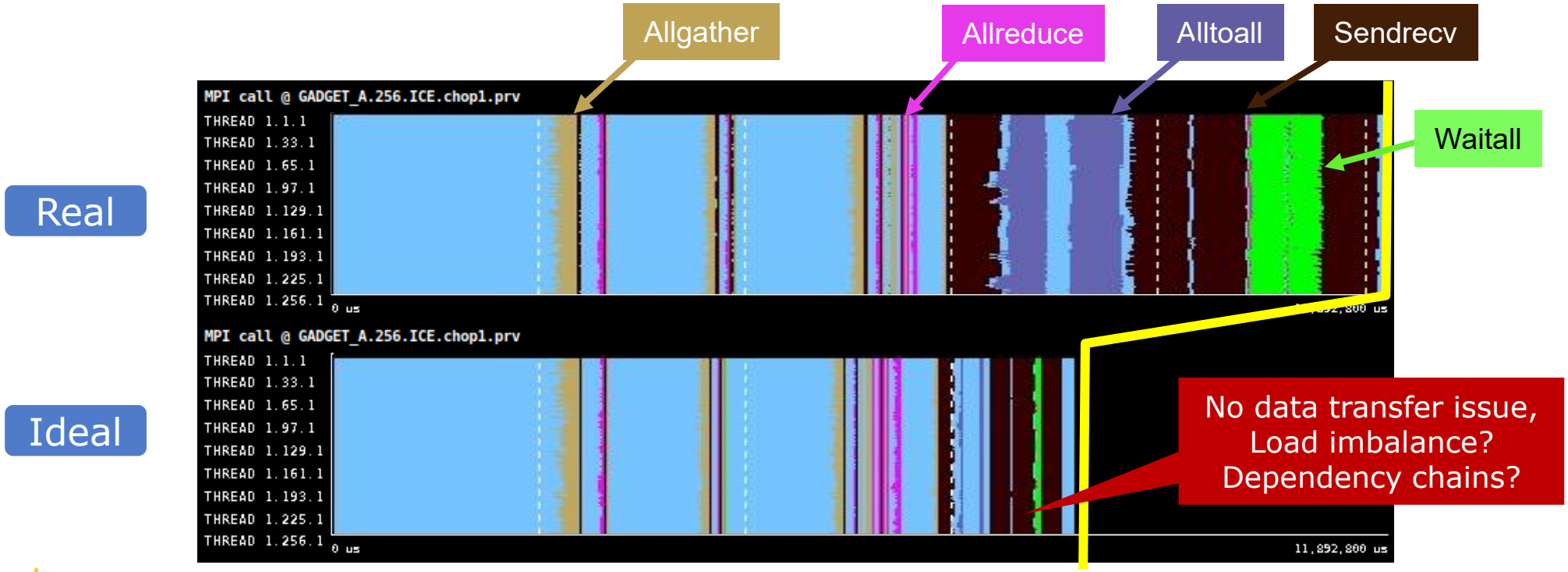
Async comms won't help

Insensitive to bandwidth until it falls below 10 MB/s



The Ideal Machine

- **BW = ∞, L = 0**
  - If data transfers are instantaneous ➔ MPI time should vanish. Why not?
  - Characterizes intrinsic application behavior

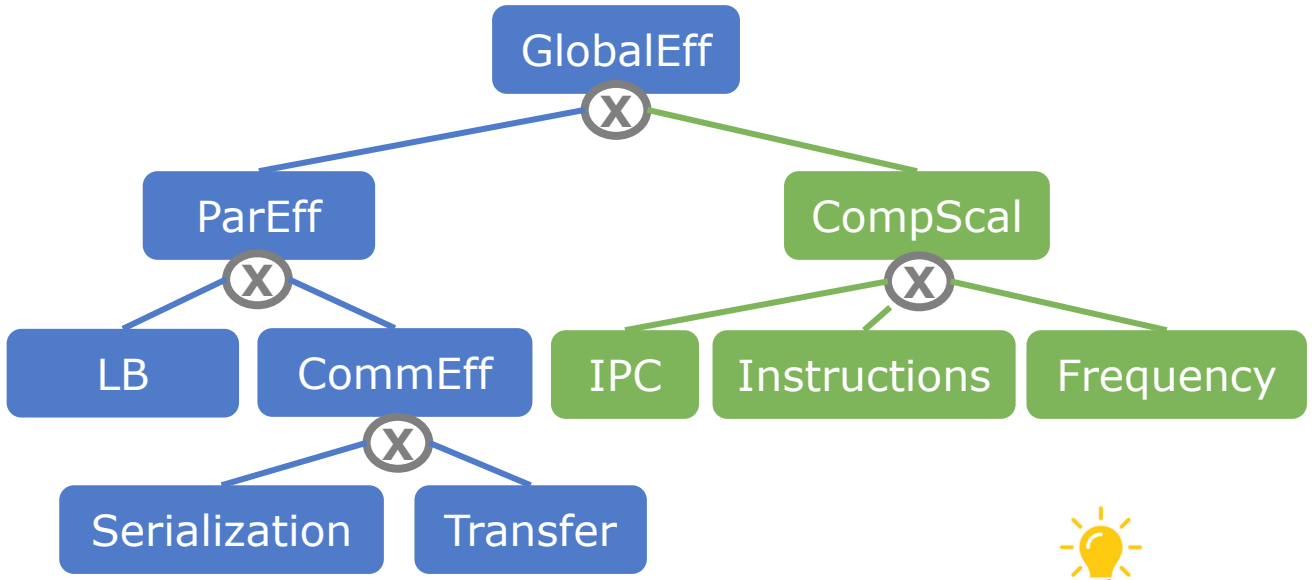


Impact on real machines? No gain in an ideal case ➔ No gain from faster hardware!

Analytics

BasicAnalysis – Assessing key performance factors

- Explain application's performance with few fundamental factors
- Automatically computed from: **Paraver trace + Dimemas simulation**
- Dig down from global to detailed efficiencies



	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

Comparing scales

	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

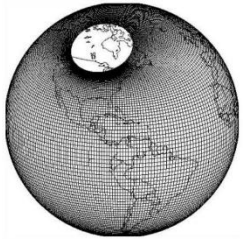
Comparing phases

What to look for?

- Low values
- Trends
- High values

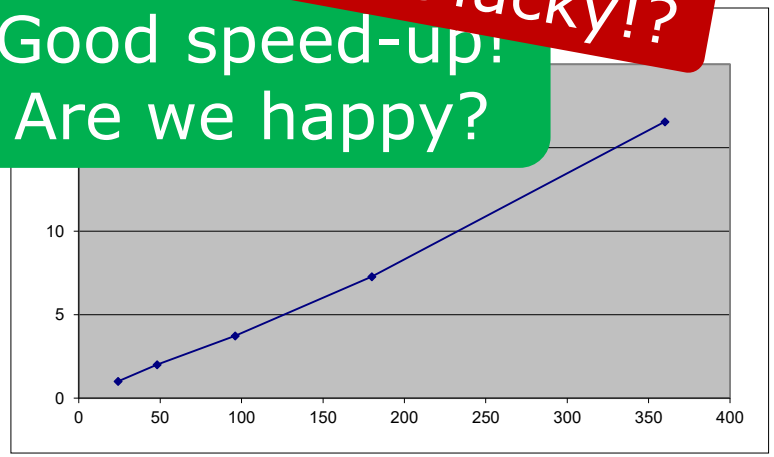
Why scaling?

- CGPOP ocean modelling
  - Intrinsically unbalanced problem



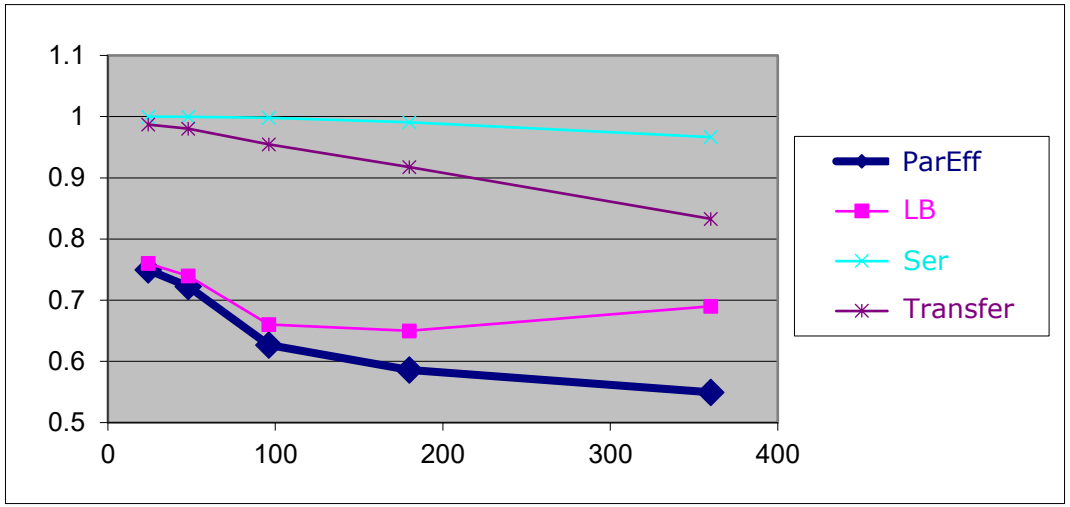
Or just lucky!?

Good speed-up!
Are we happy?

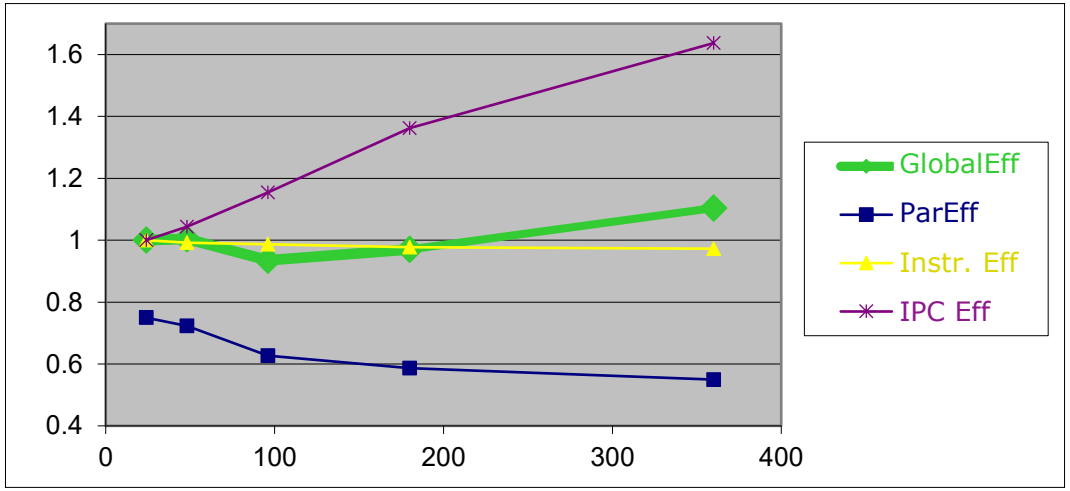


- Transfer Efficiency ↓
- IPC helps… for now!
- **Comms will become a bottleneck**

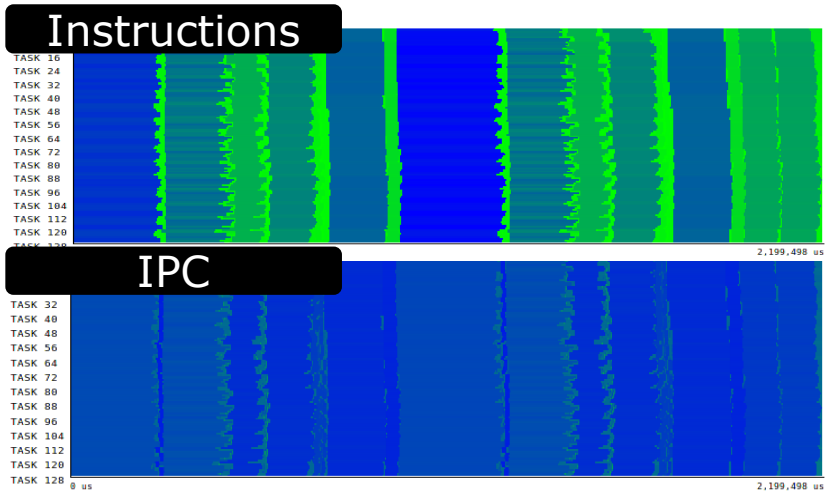
ParEff = *LB* * *Ser* * *Trf*

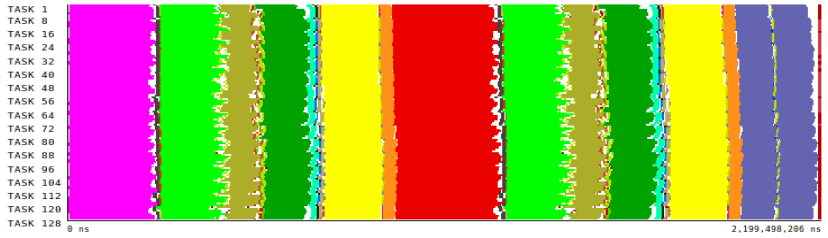


GlobalEff = *ParEff* * *Instr* * *IPC* * *Freq*

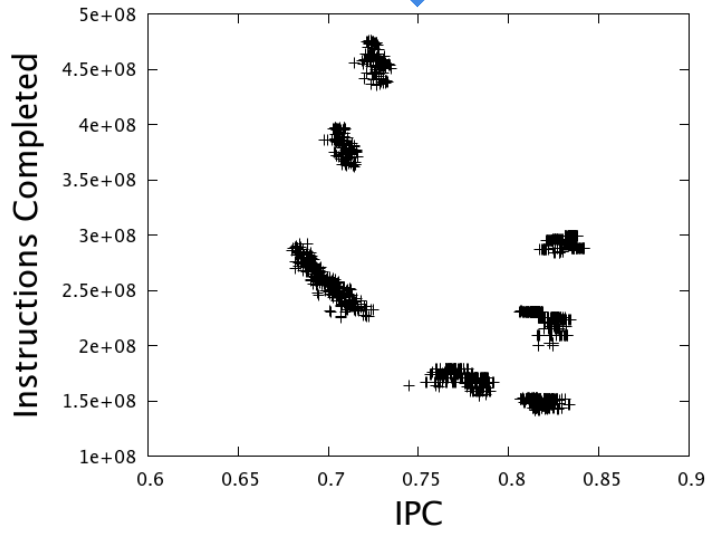


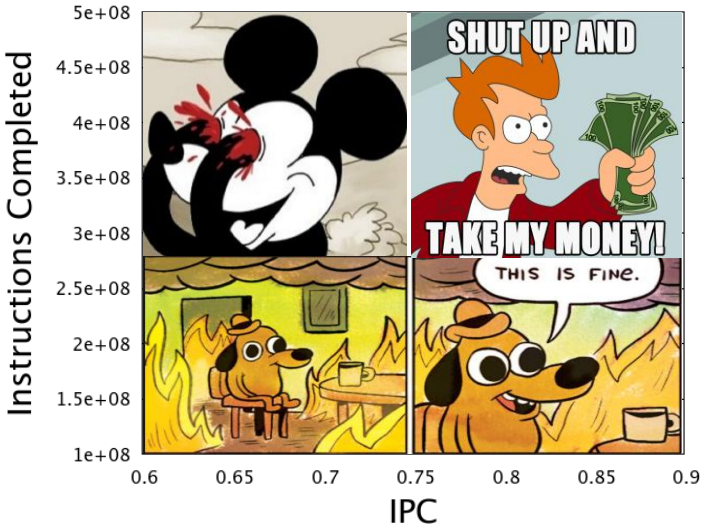
Clustering to identify structure





Quick insight into program's behavior

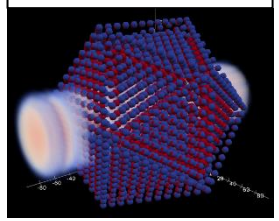




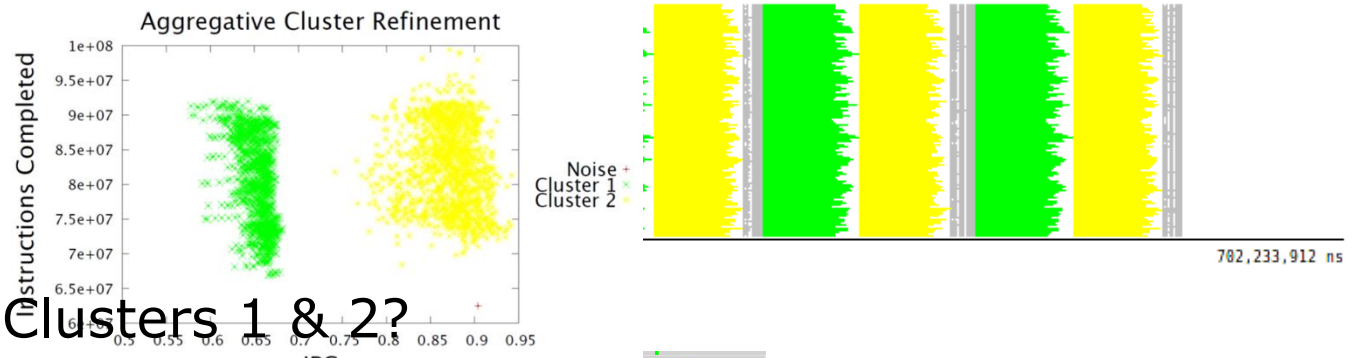
Integrating models and analytics

- What if…

PEPC

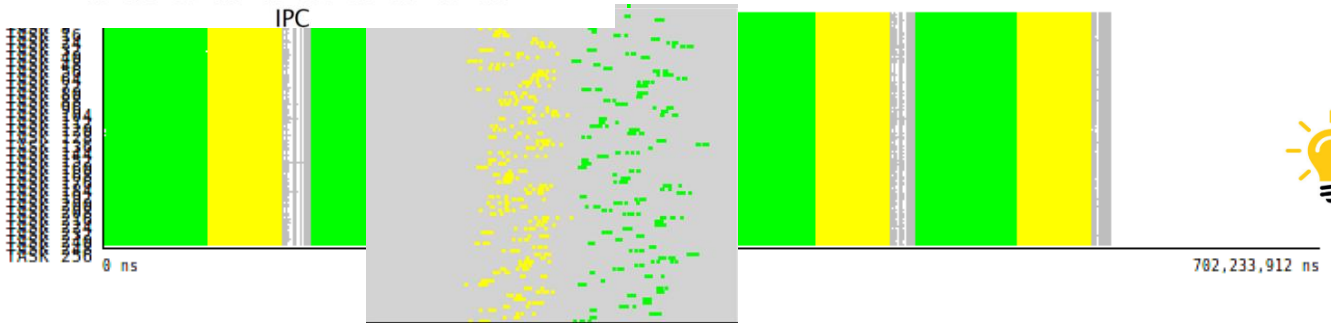


… we increase the IPC of Cluster1?



13%

… we balance Clusters 1 & 2?

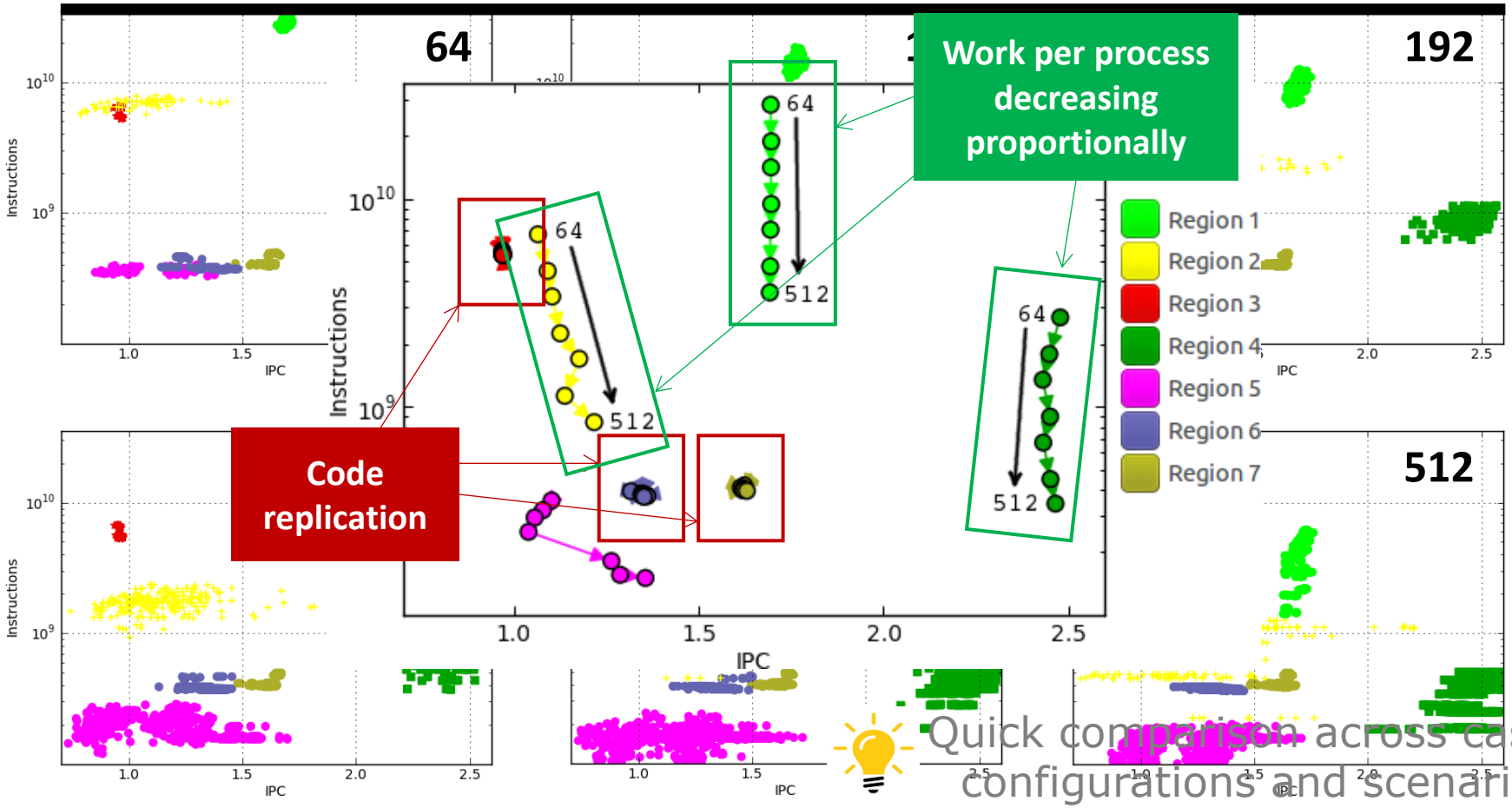


19%

Know where effort pays off.

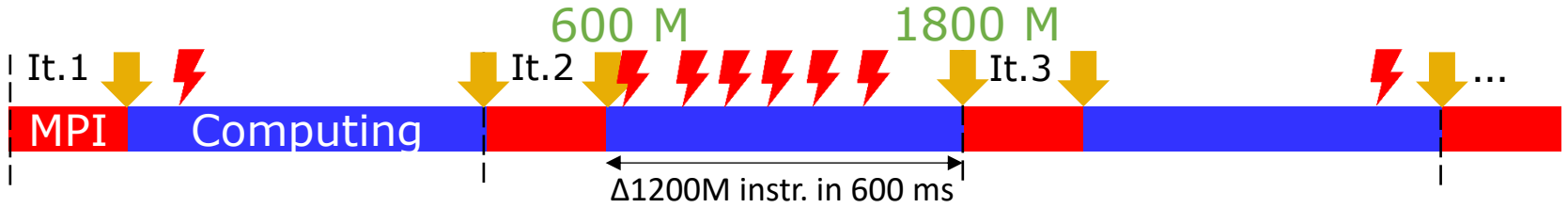
Tracking scalability through clustering

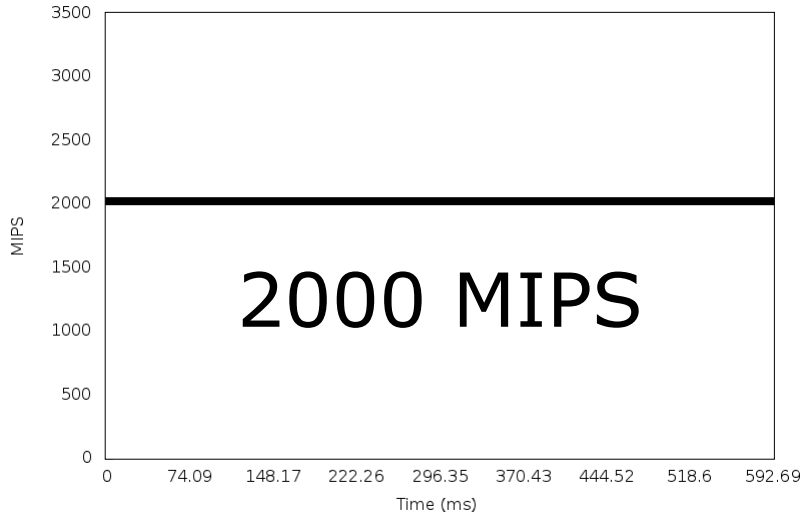
- Analyze scalability of computing regions across 64 – 512 tasks



Folding to increase details

- What is the performance of a single serial region?





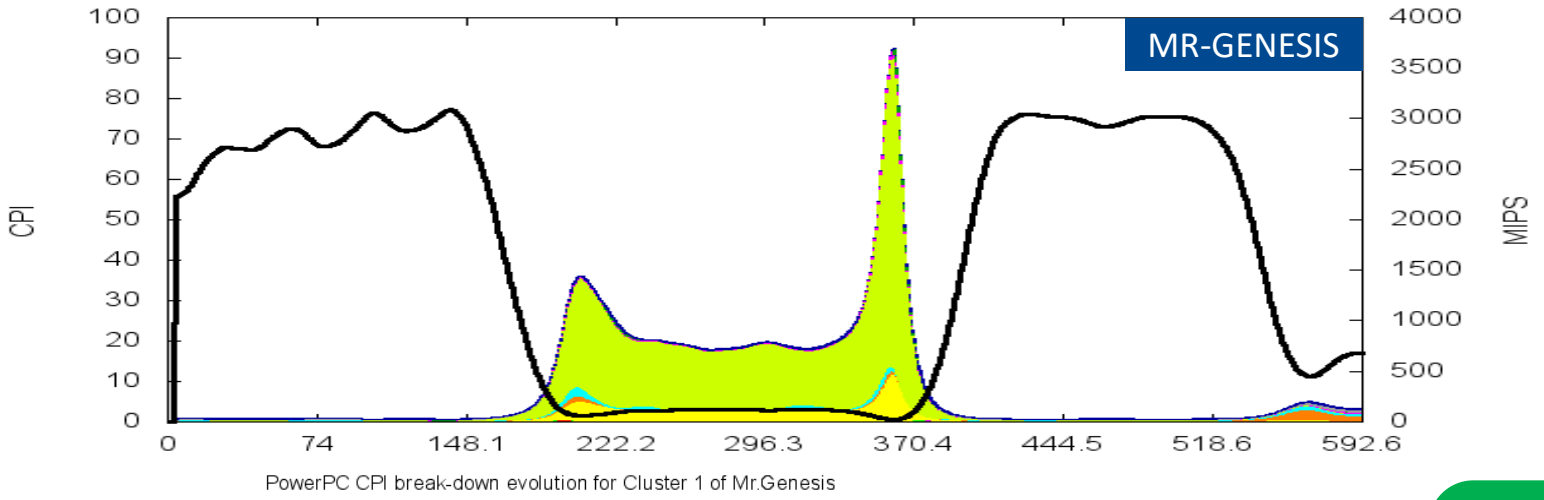
Instrumentation

+ Sampling

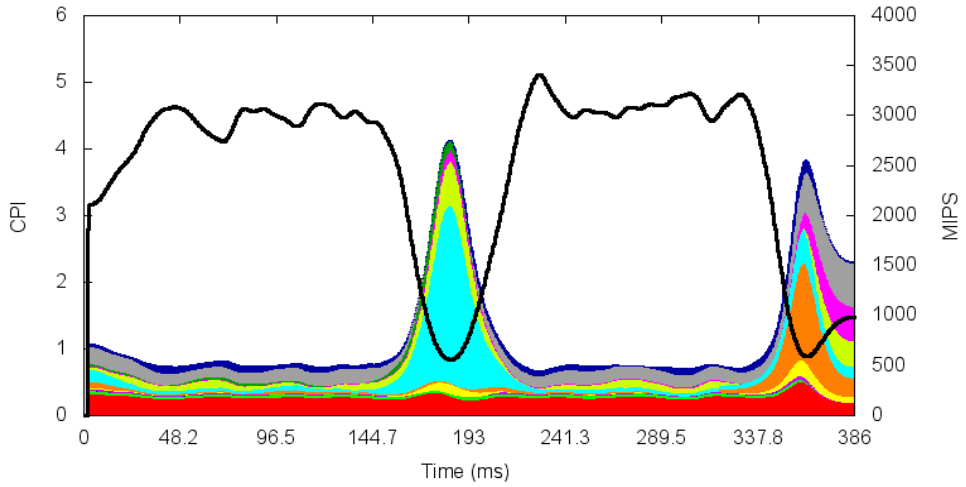
Reveals sub-phases in high detail

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

Folding: CPI and HWC stack models



PowerPC CPI break-down evolution for Cluster 1 of Mr.Genesis

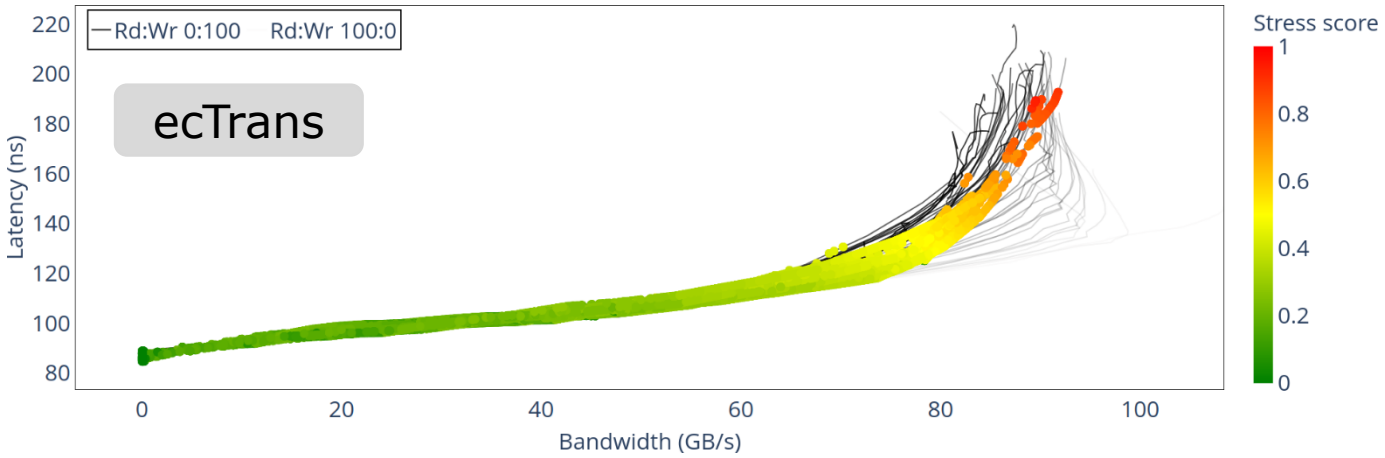


- Trivial fix (loop interchange)
- 1-line code fix → 25% boost
- Easy to locate

- Availability of CPI stack models for production processors

Understanding memory influence

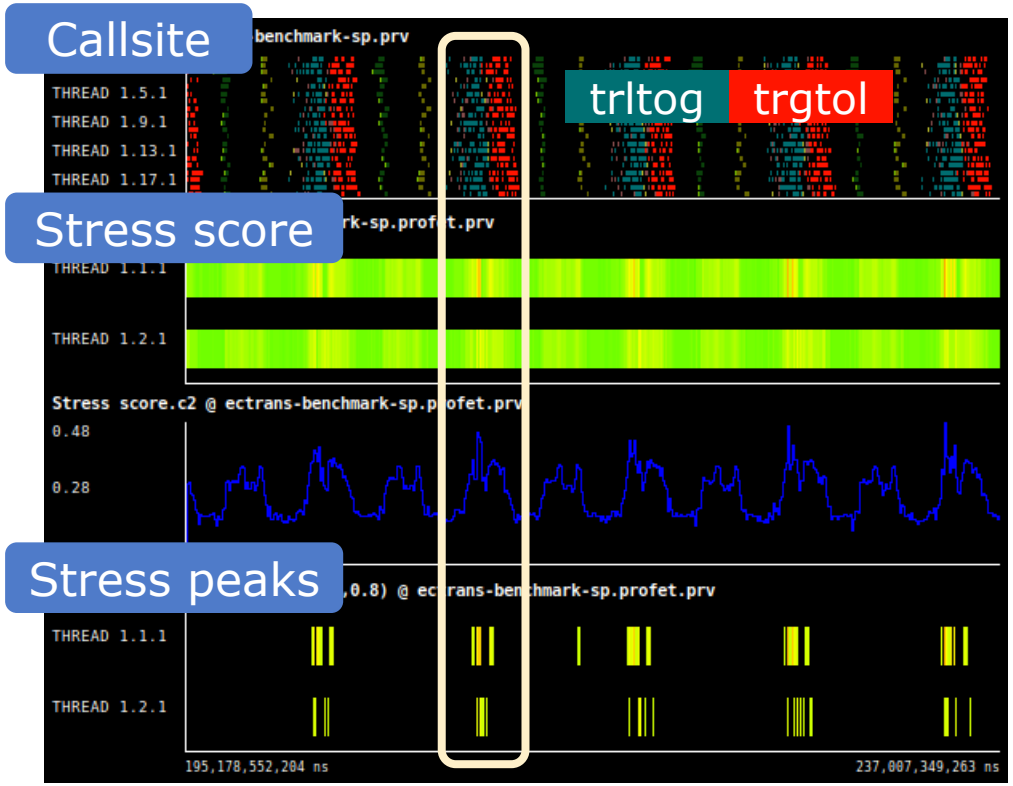
- Mess: Bandwidth-latency curves describe memory performance from unloaded to fully saturated states
- Integration with Paraver: Easily identify where memory stress is highest and correlate with sources



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

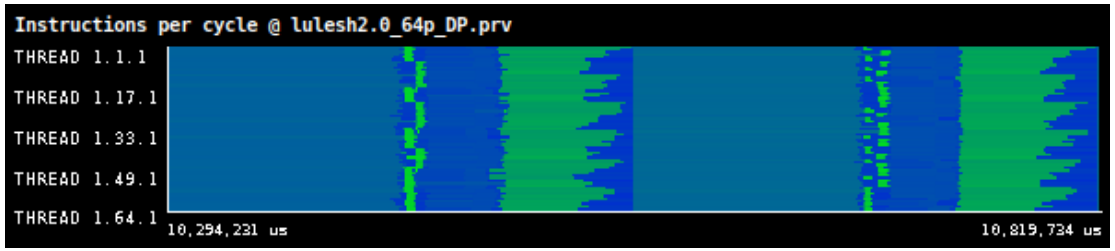
	[0.00..0.10)	[0.10..0.20)	[0.20..0.30)	[0.30..0.40)	[0.40..0.50)	[0.50..0.60)	[0.60..0.70)	[0.70..0.80)	[0.80..0.90)	[0.90..1.00]
THREAD 1.1.1	0.46 %	22.92 %	45.57 %	28.06 %	2.50 %	0.34 %	0.03 %	0.01 %	0.11 %	0.00 %
THREAD 1.2.1	0.38 %	20.48 %	55.90 %	22.57 %	0.53 %	0.03 %	0.00 %	0.02 %	0.09 %	0.00 %
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

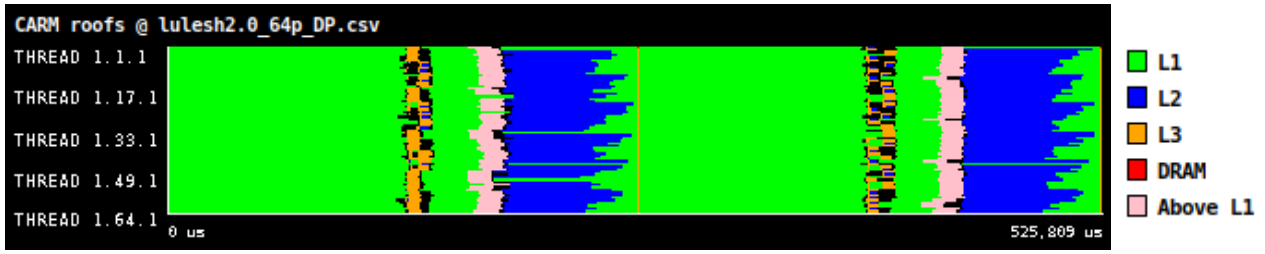
- Prediction of future ones (PROFET)



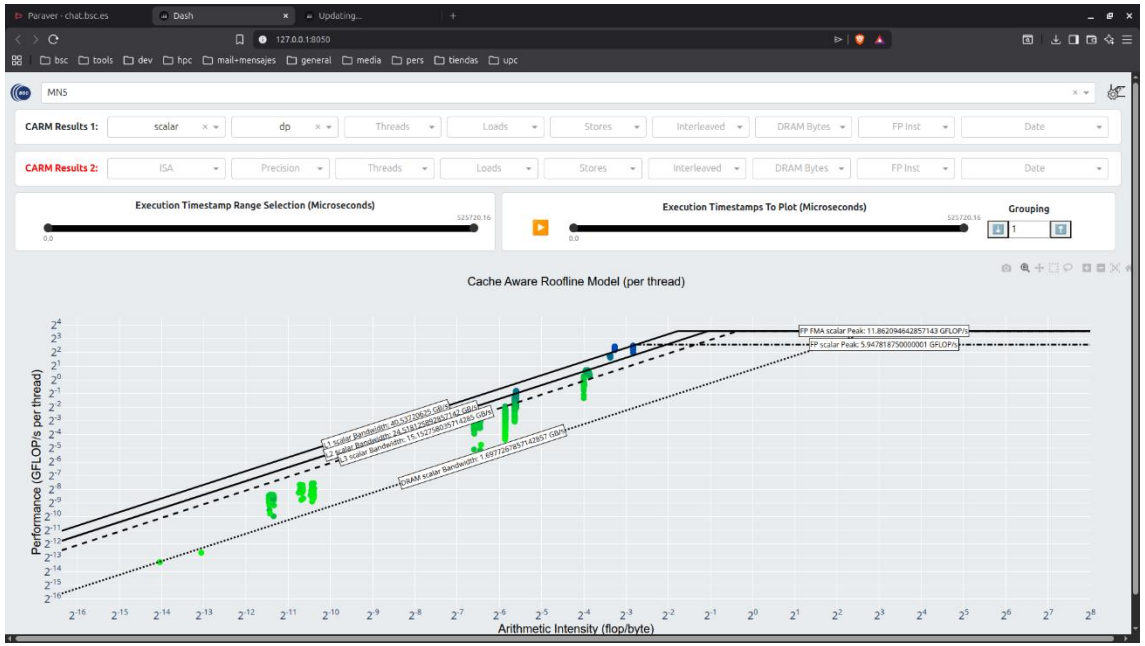
Accounting for all memory hierarchy levels

- CARM: Cache-Aware Roofline Model
- Unlike the original Roofline Model, identifies bottlenecks related to specific cache levels





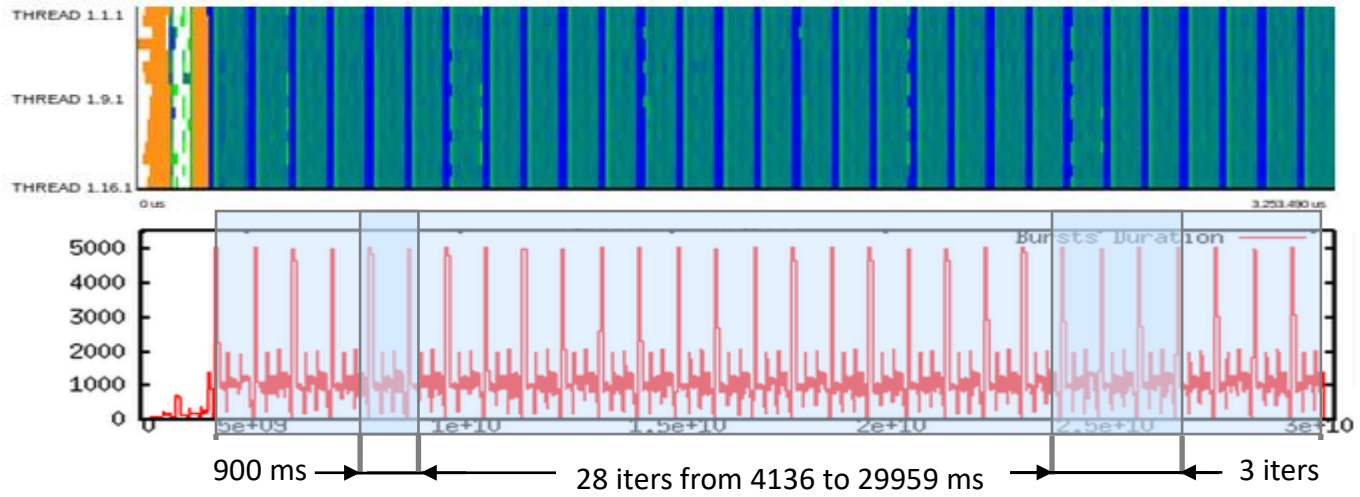
Paraver feeds computing bursts and hardware counter metrics



CARM feeds back the “Roofs” for each computing phase

Spectral Analysis: Finding the FoA

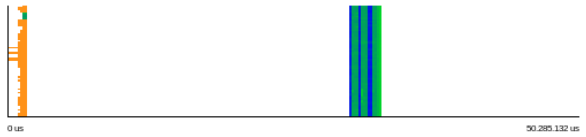
- Signal processing techniques to detect patterns in traces
  - Morphological filters, Wavelet, FFT, Autocorrelation
- Performance metrics expressed as a signal
  - A function of time
  - At the application level (sum of all tasks)
    - Tasks computing, tasks in MPI, instructions, L3 misses, **useful duration**, etc.



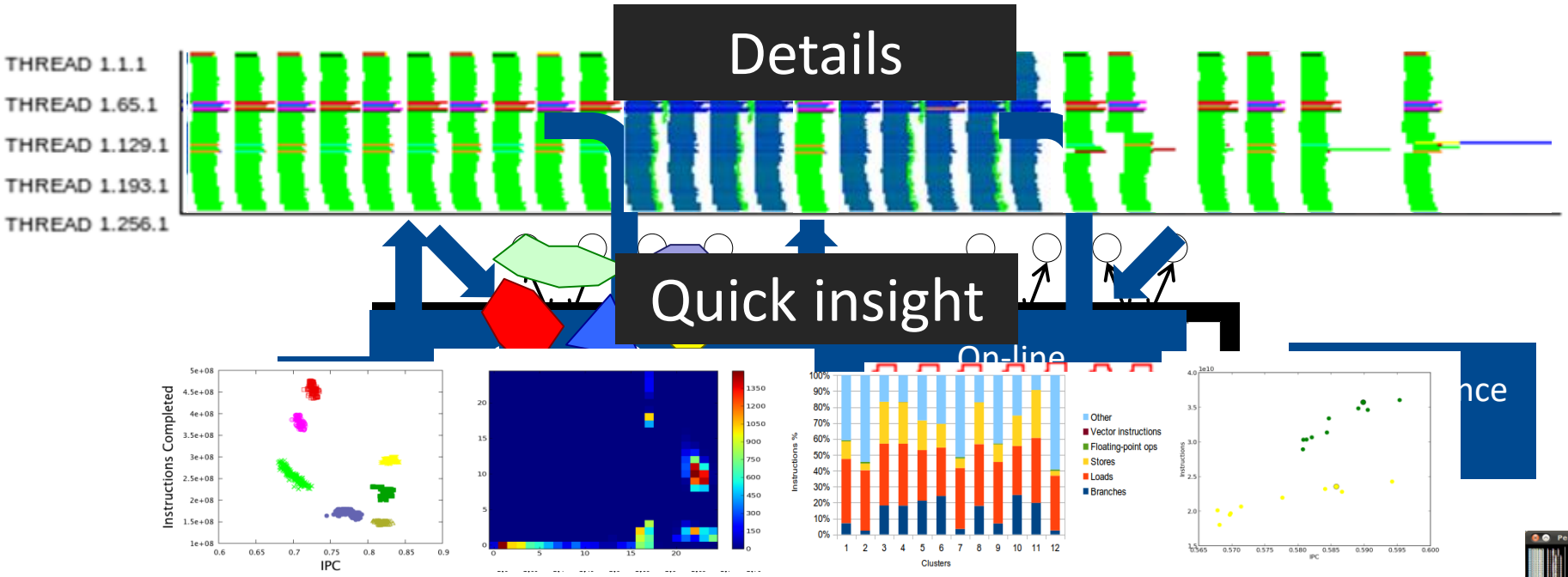
Long bursts

- Global periodic region
- Period length
- Representative iterations

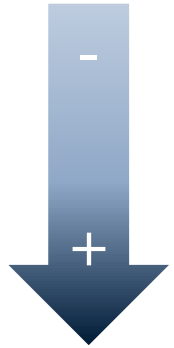
Large amplitudes

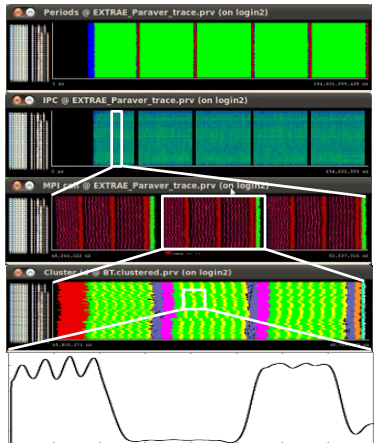


On-line Analysis: One Ring to Rule Them All



Towards smart traces with dynamic levels of detail and high-level insights





BSC Tools Website & Contact

- https://tools.bsc.es

Open Source

Downloads

- Sources & Binaries

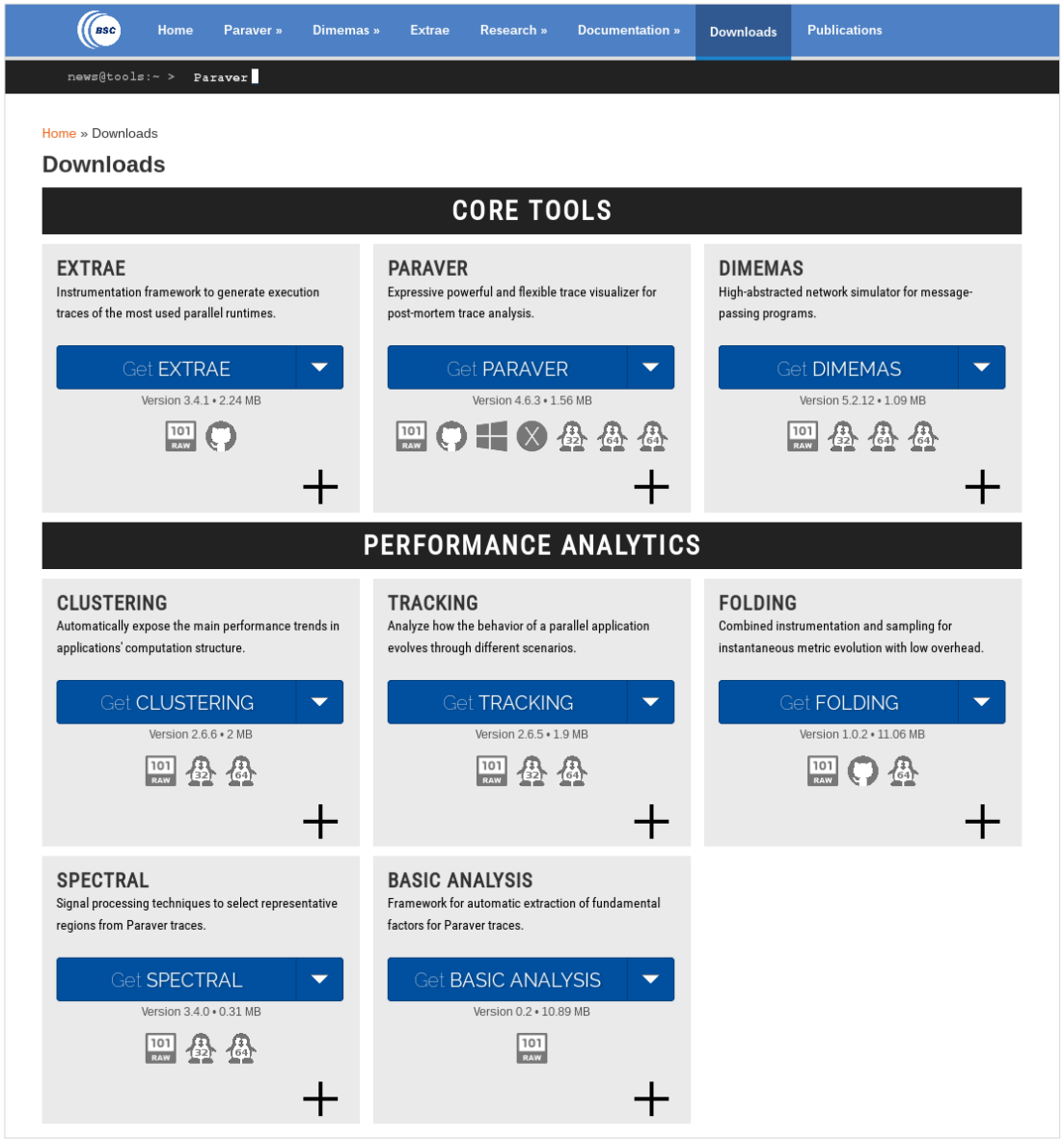
Documentation

- 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**