POP Studies of Earth Sciences Codes

Jesús Labarta, BSC

EU H2020 Centre of Excellence (CoE)

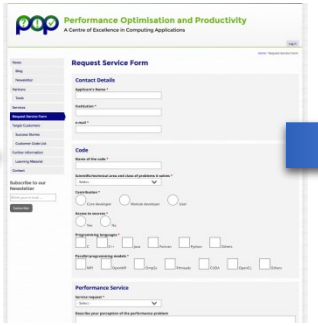
1 December 2018 – 30 November 2021

Grant Agreement No 824080

POP services

Online application

https://pop-coe.eu/request-service-form



User or developer

Who will obtain the traces?

expert

Client provides code, input and instructions on how to compile and run.

User or developer

POP expert provides instructions on how to install tracing tools and obtain traces.

traces

Parallel Application Performance Assessment

expert

Report

The report includes suggestions to improve performance

User or developer

Proof of concept (PoC)

expert

POP expert implements optimizations

code

Report

Code developer implements optimizations

Follow on audit

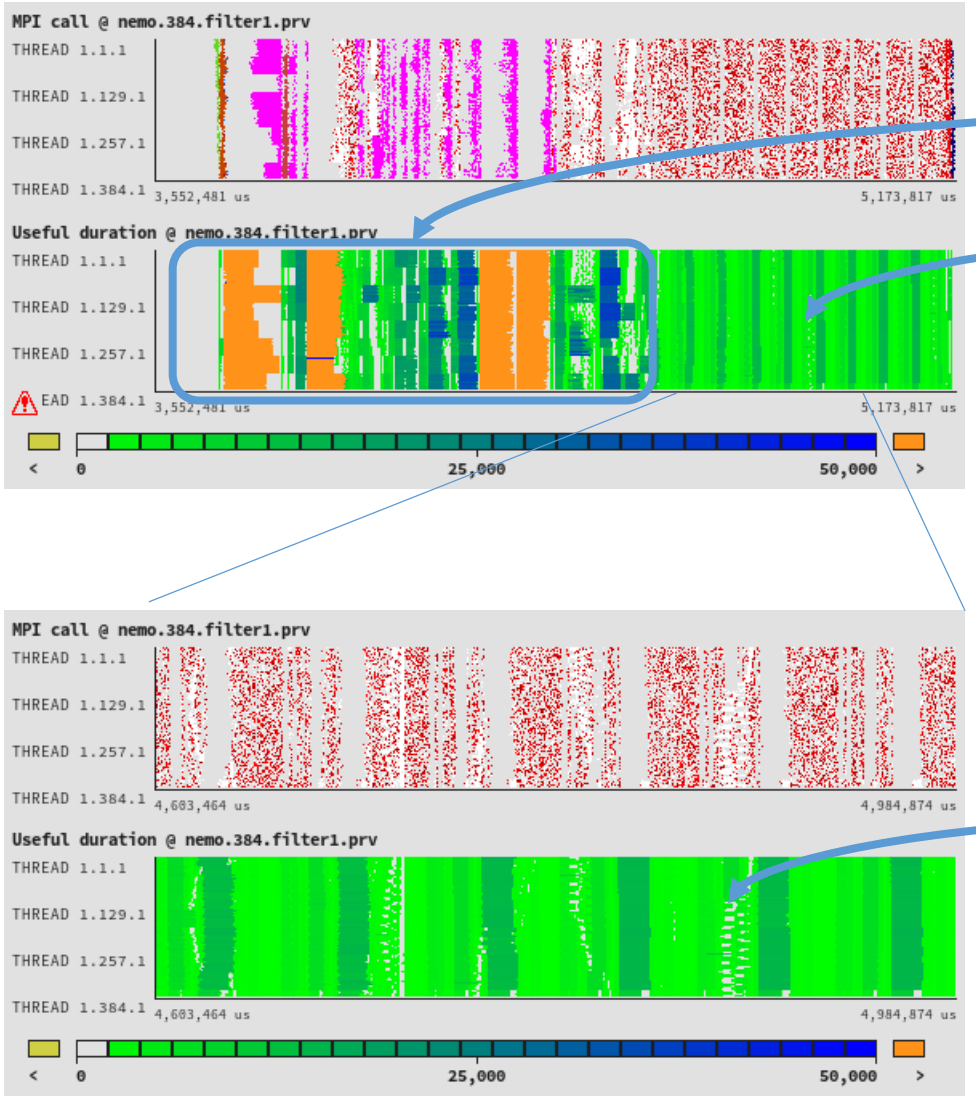
POP and ES

- Analyzed several codes …
  - IFS, FVM
  - NEMO
  - MONARCH
  - ICON

- Towards Best Practices in
  - Performance Analysis Methodology and Tools
  - Programming Practices

Structure

384 cores

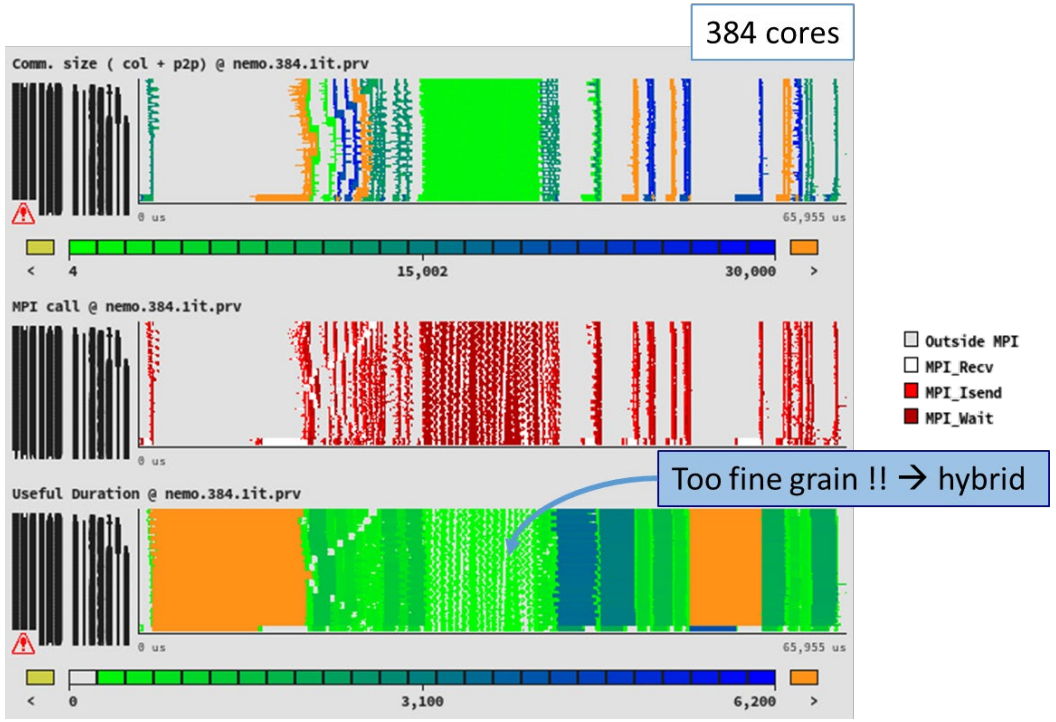


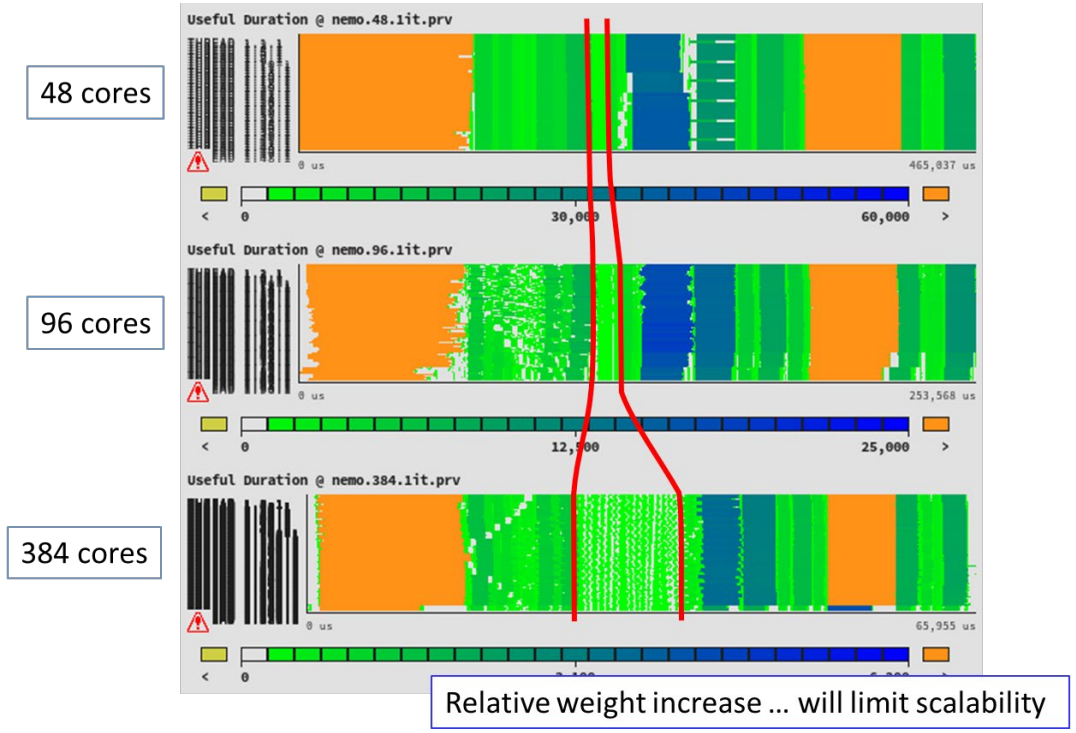
Initialization

Iterative structure

perturbation

Structure


384 cores
Comm. size (col + p2p) @ nemo.384.1it.prv
0 us
65,955 us
< 4 15,002 30,000 >
MPI call @ nemo.384.1it.prv
0 us
Useful Duration @ nemo.384.1it.prv
0 us
65,955 us
< 0 3,100 6,200 >
Outside MPI
MPI_Recv
MPI_Isend
MPI_Wait
Too fine grain !! → hybrid


48 cores
96 cores
384 cores
Useful Duration @ nemo.48.1it.prv
0 us
465,837 us
< 0 30,000 60,000 >
Useful Duration @ nemo.96.1it.prv
0 us
253,568 us
< 0 12,500 25,000 >
Useful Duration @ nemo.384.1it.prv
0 us
65,955 us
< 0
Relative weight increase … will limit scalability

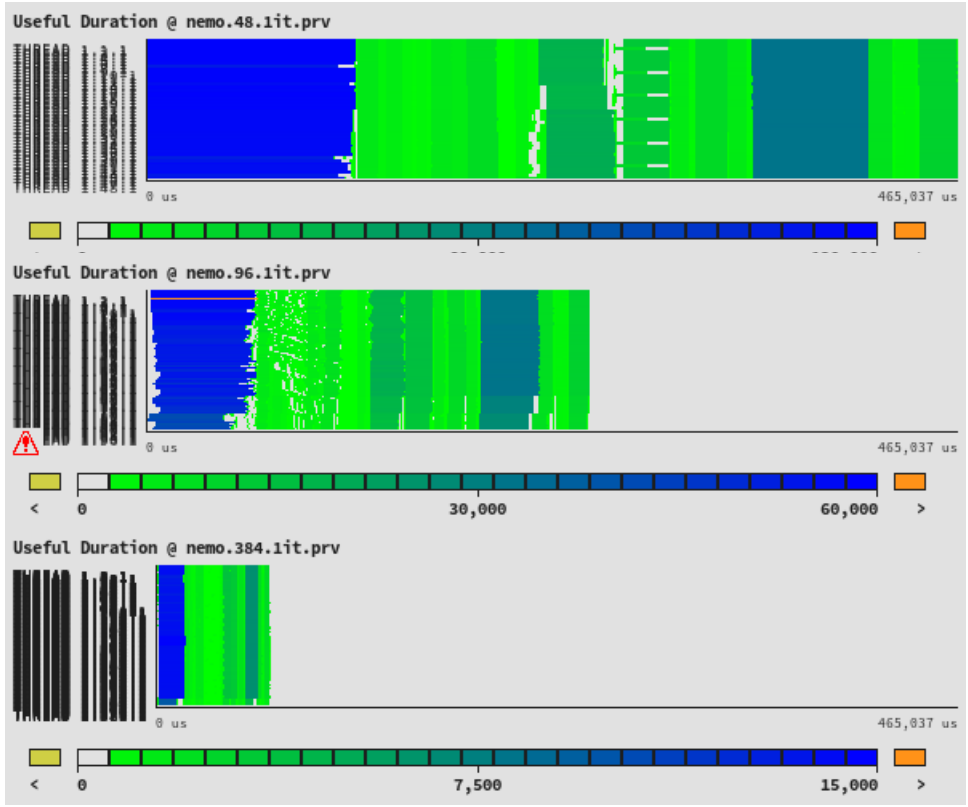
Scaling

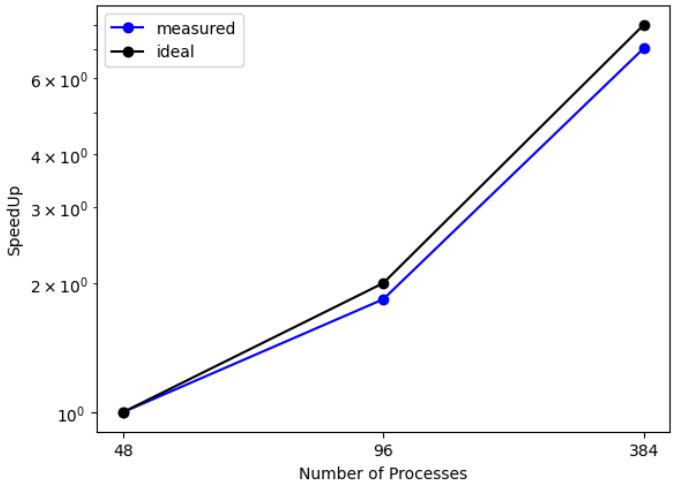
T(48) =465 ms

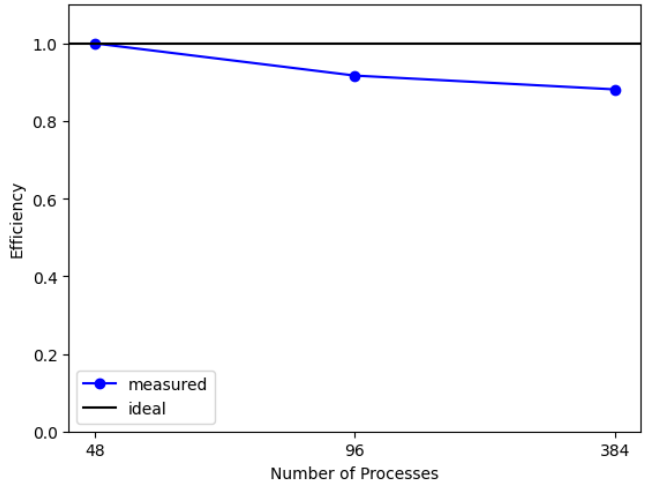
48

96

384

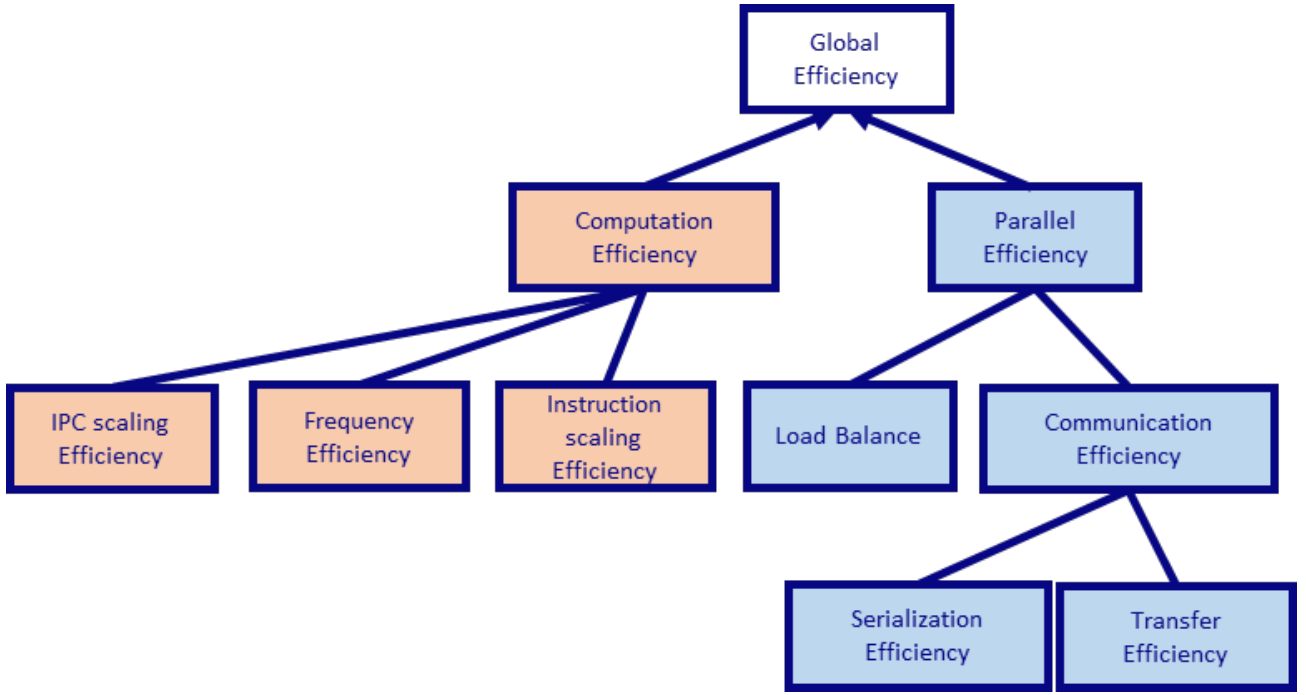






Hierarchical Performance Model

Efficiencies: ~ (0,1]
Multiplicative model



$$CompEff = Ieff * IPCeff * Feff$$

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

$CommEff$

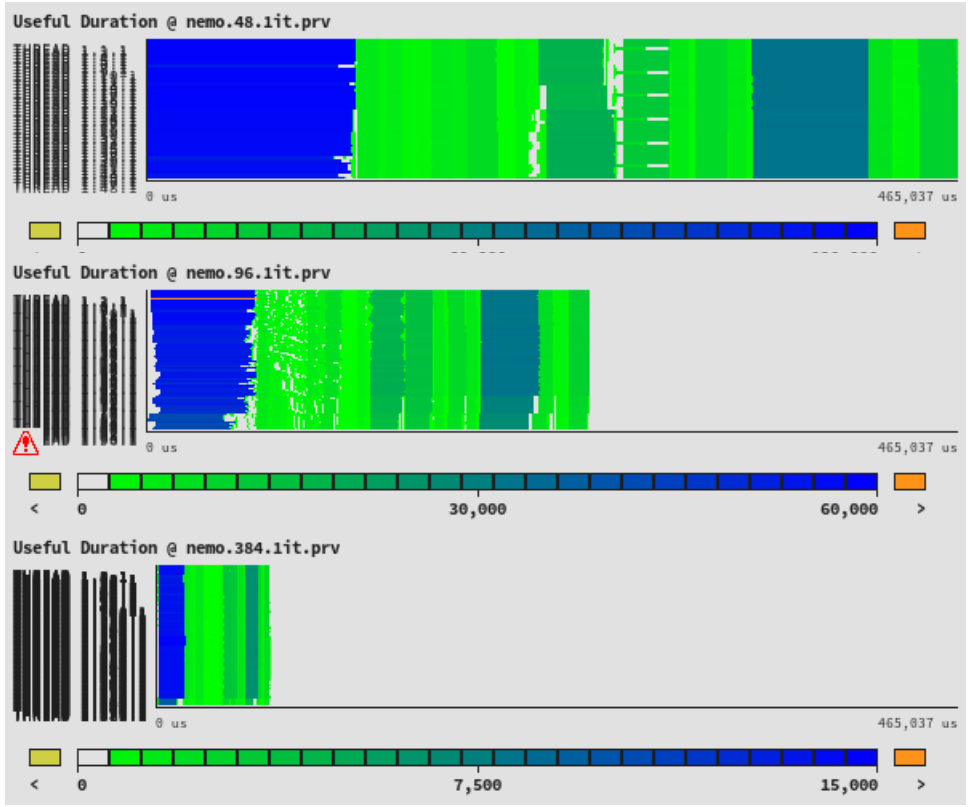
M. Casas et al, "Automatic analysis of speedup of MPI applications". ICS 2008.

Efficiency model

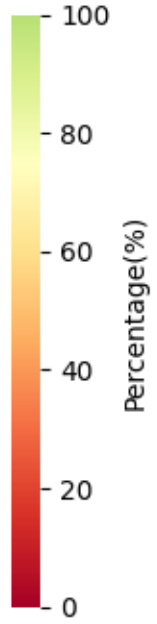
48

96

384



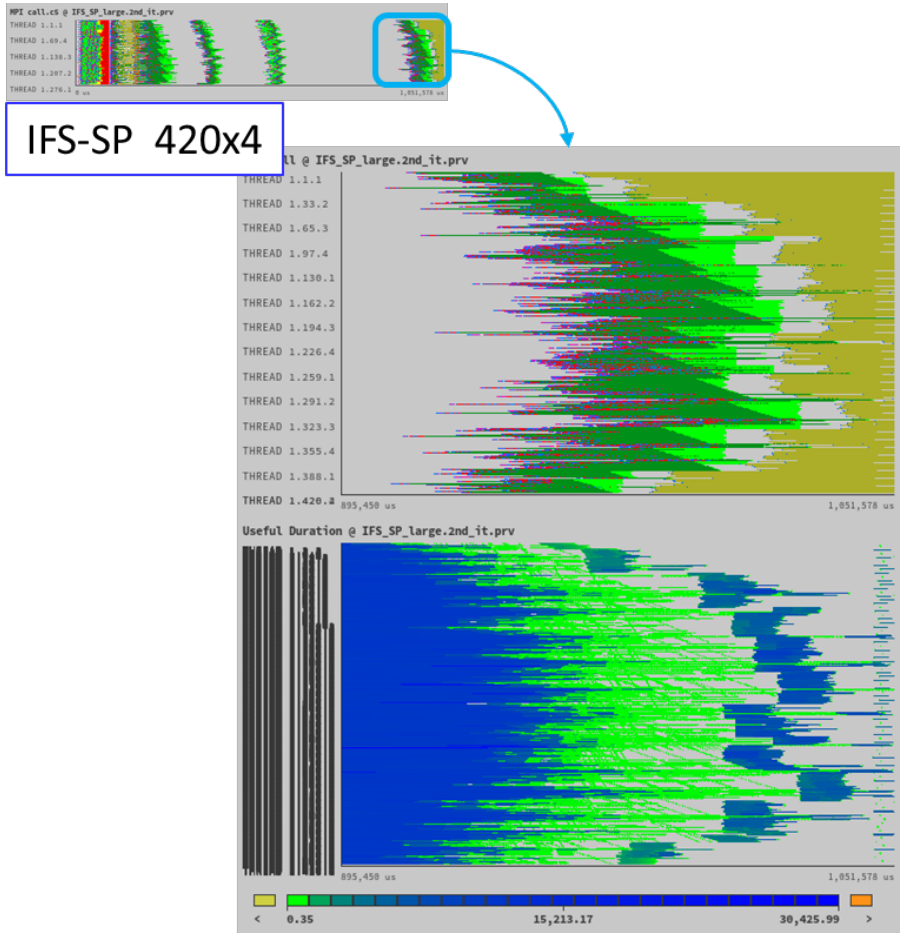
	48	96	384
Global efficiency	94.16	86.35	82.99
-- Parallel efficiency	94.16	84.11	74.80
-- Load balance	98.67	95.34	95.77
-- Communication efficiency	95.43	88.22	78.10
-- Serialization efficiency	97.86	93.05	89.61
-- Transfer efficiency	97.52	94.81	87.15
-- Computation scalability	100.00	102.66	110.95
-- IPC scalability	100.00	115.98	182.39
-- Instruction scalability	100.00	91.37	62.62
-- Frequency scalability	100.00	96.87	97.13



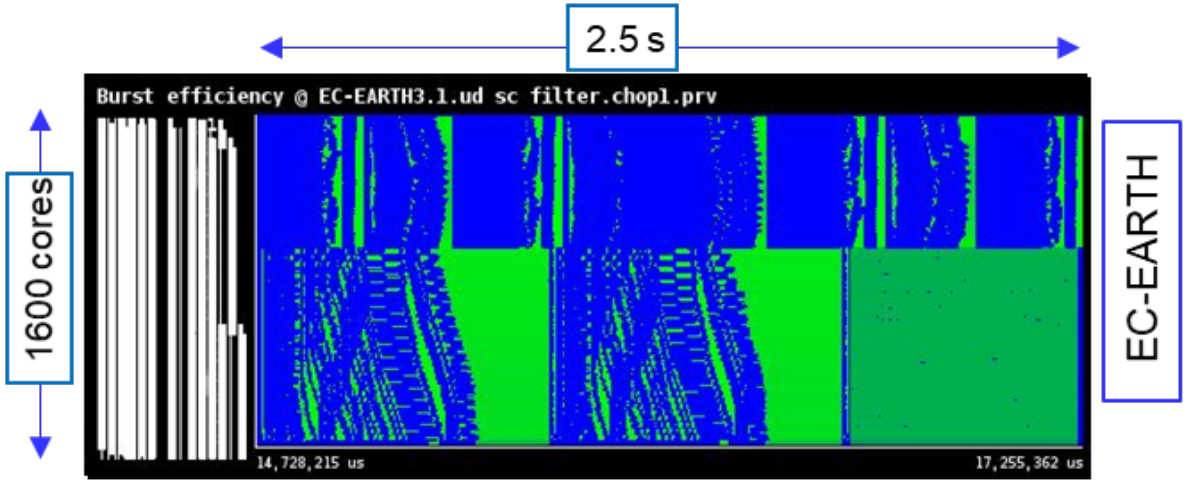
Avg Useful IPC(48) =0.67

Avg Useful Frequency(48) =2.061 GHz

Load Balance



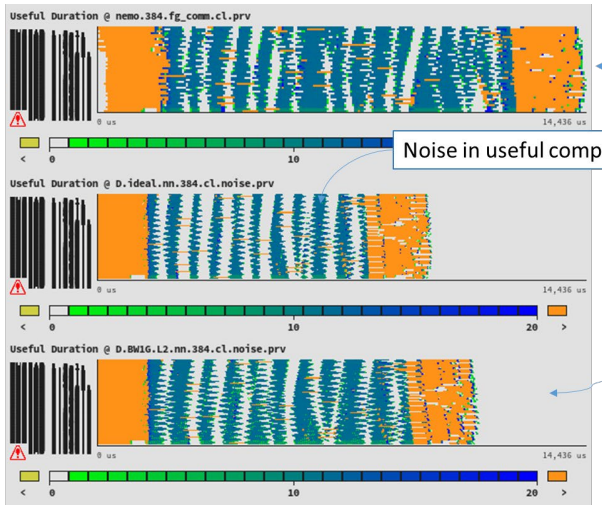
Within a model …



26.7MB trace
Eff: 0.43; LB: 0.52; Comm:0.81

… and coupled runs

Communication analysis



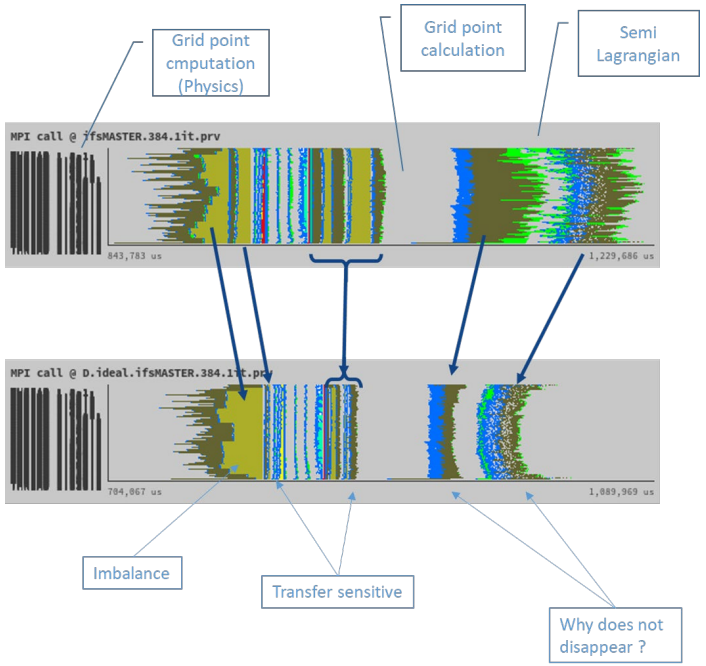
Actual run

Ideal Network

"Nominal" Network

Noise in useful computation in fine grain phase

Difference → there was noise in the internals of MPI implementation



Grid point cmputation (Physics)

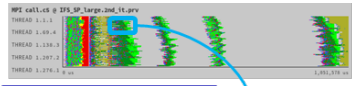
Grid point calculation

Semi Lagrangian

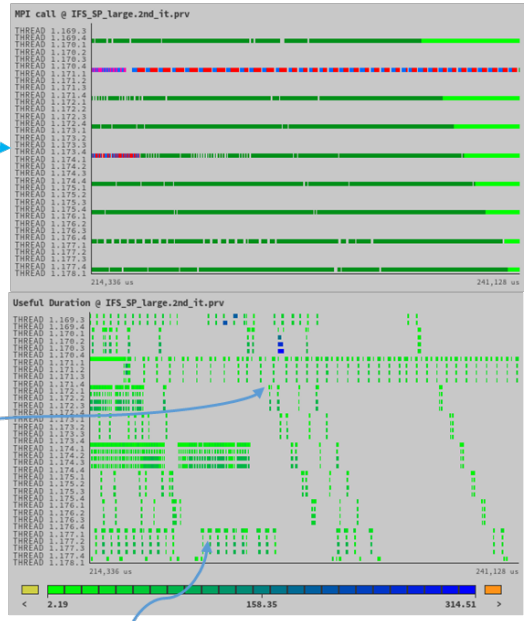
Imbalance

Transfer sensitive

Why does not disappear ?



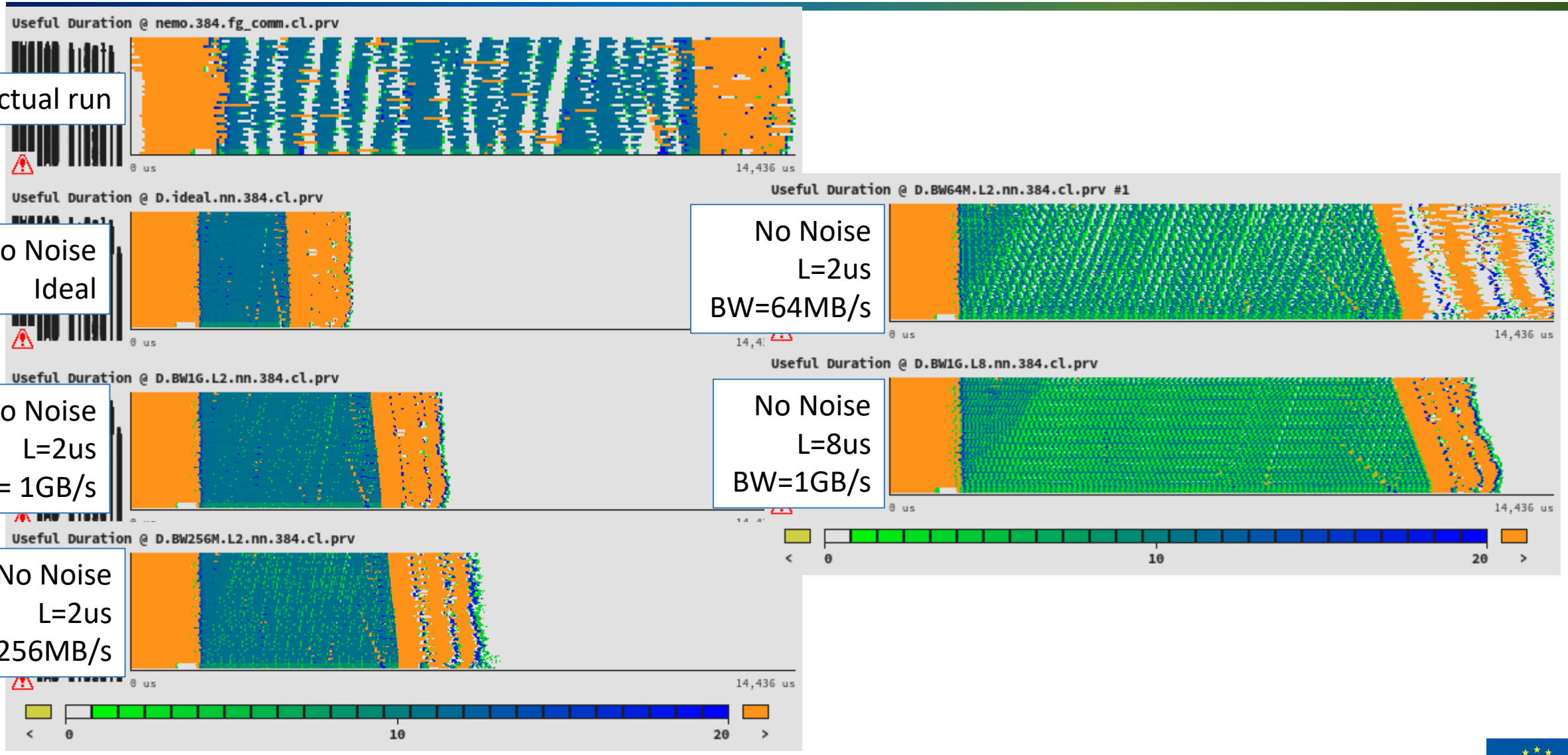
IFS-SP 420x4



Serialized unpacks

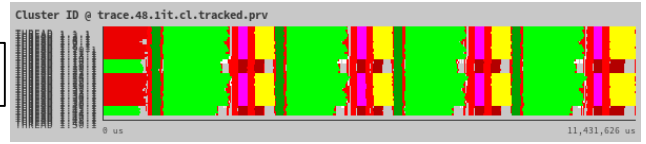
Very fine grain parallelization of individual unpacks

What ifs

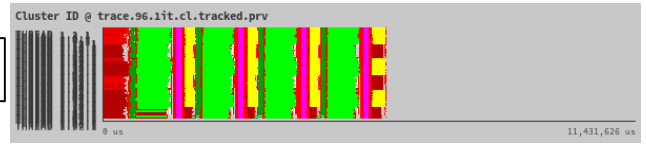

Actual run
Useful Duration @ nemo.384.fg_comm.cl.prv
0 us
14,436 us
No Noise
Ideal
Useful Duration @ D.ideal.nn.384.cl.prv
0 us
No Noise
L=2us
BW= 1GB/s
Useful Duration @ D.BW1G.L2.nn.384.cl.prv
No Noise
L=2us
BW=256MB/s
Useful Duration @ D.BW256M.L2.nn.384.cl.prv
0 us
14,436 us
< 0 10 20 >
No Noise
L=2us
BW=64MB/s
Useful Duration @ D.BW64M.L2.nn.384.cl.prv #1
0 us
14,436 us
No Noise
L=8us
BW=1GB/s
Useful Duration @ D.BW1G.L8.nn.384.cl.prv
0 us
14,436 us
< 0 10 20 >

MPI strong scaling

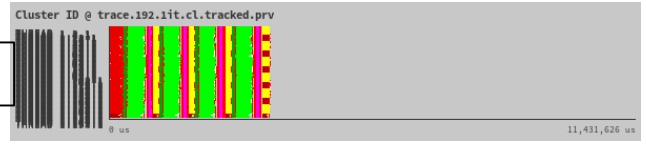
48



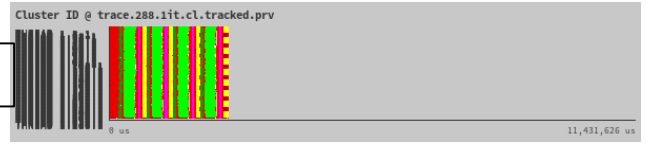
96



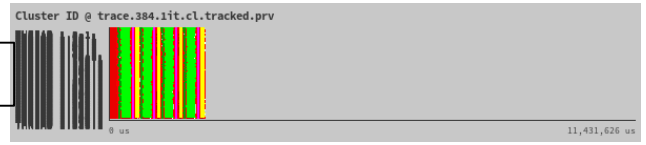
192

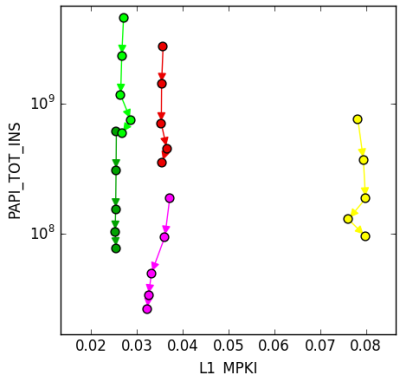


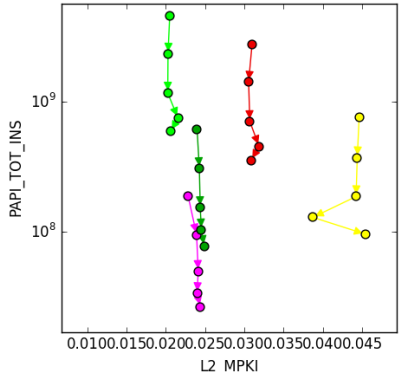
288

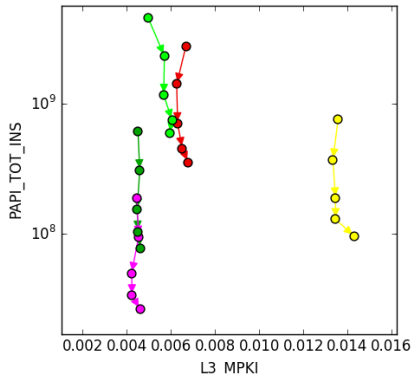


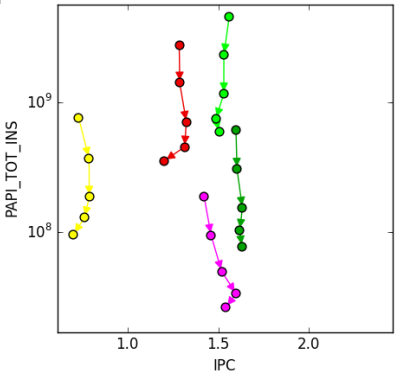
384

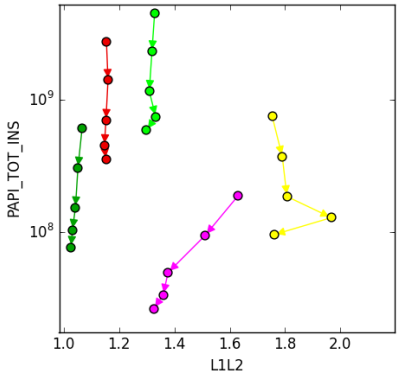


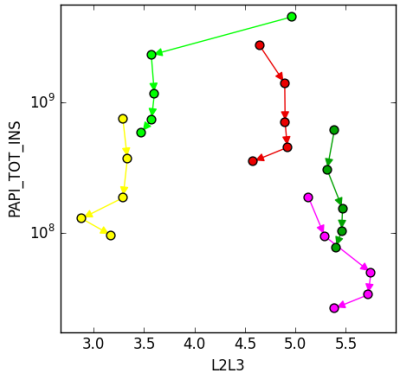












MPI+OMP strong scaling

48x1

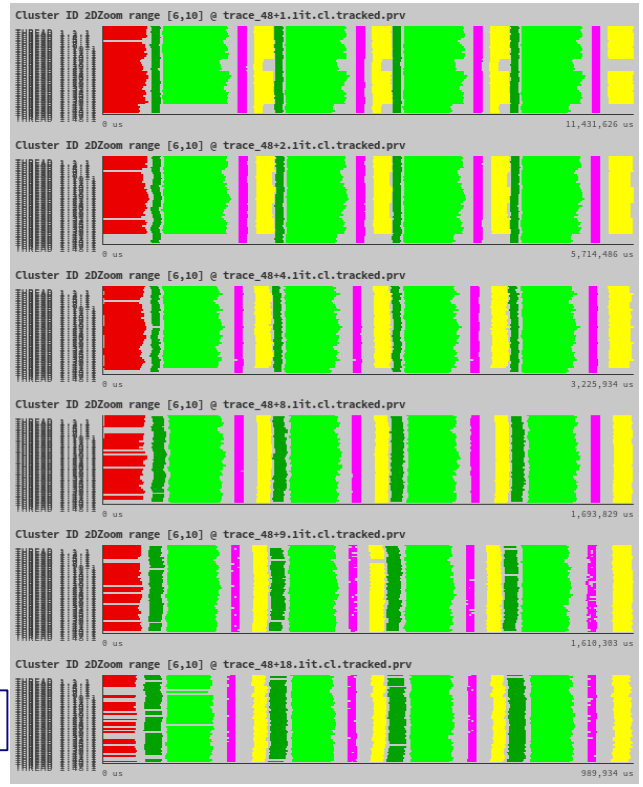
48x2

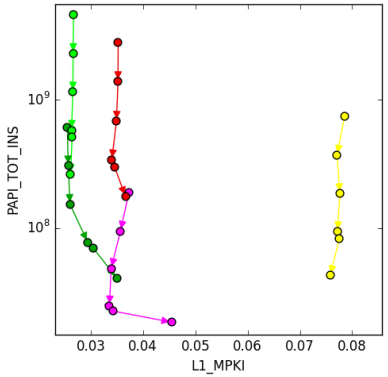
48x4

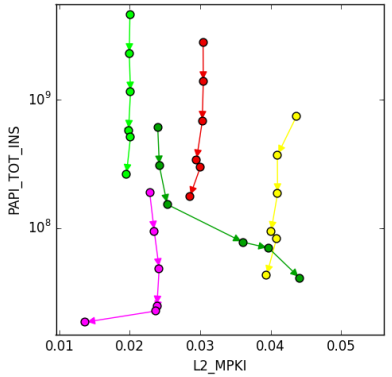
48x8

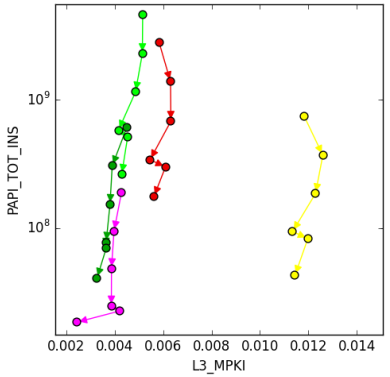
48x9

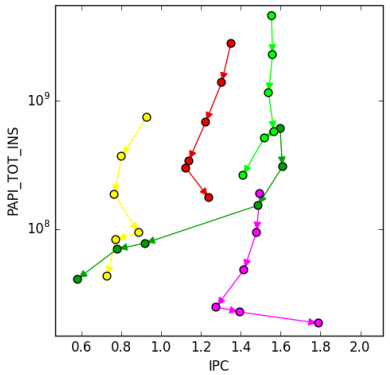
48x18

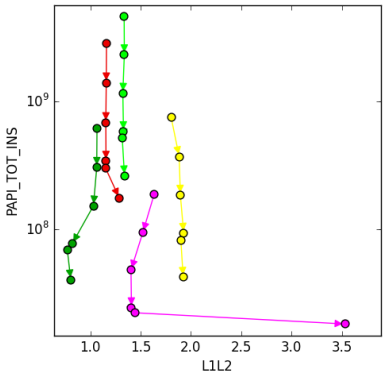


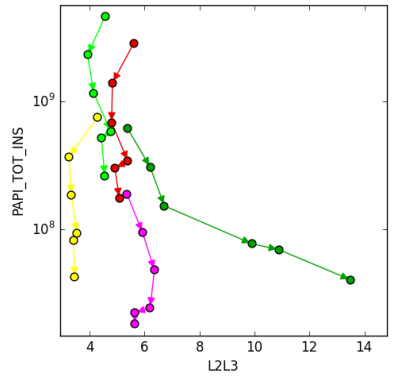




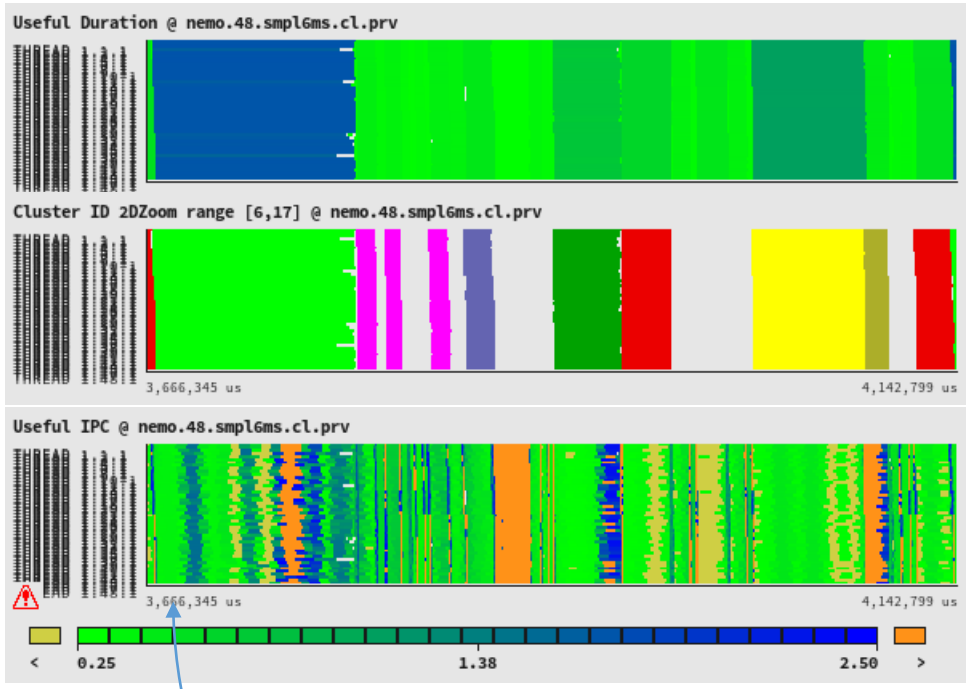


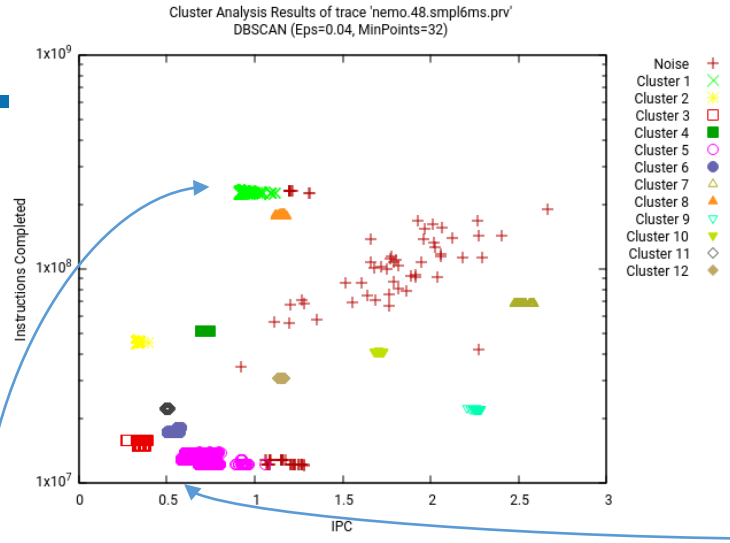




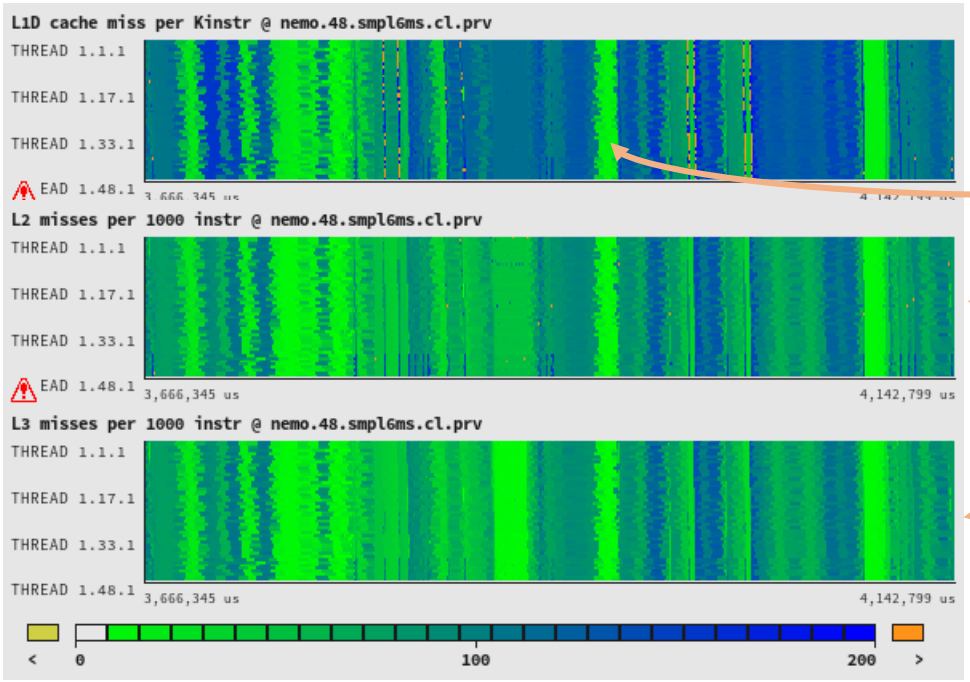


Sampled traces





Regions with poor IPC

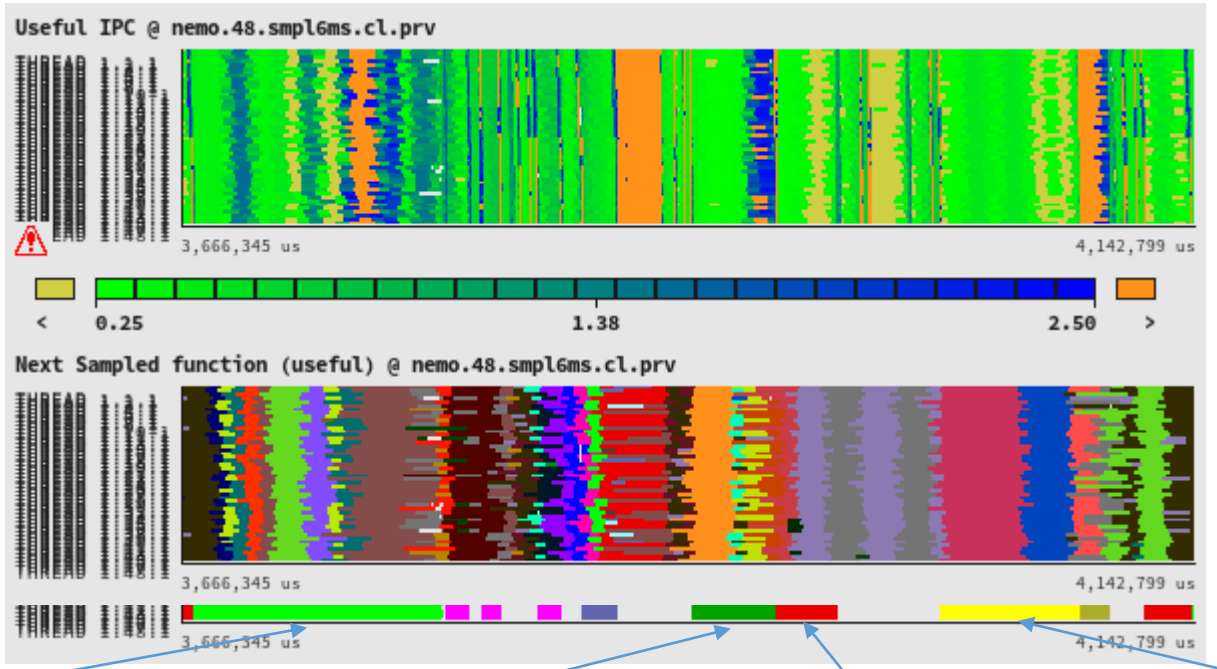


Cache miss ratios “explaining” IPC

Limited benefit of L3

Very poor IPC sub regions within region of moderate average IPC

Sampled traces



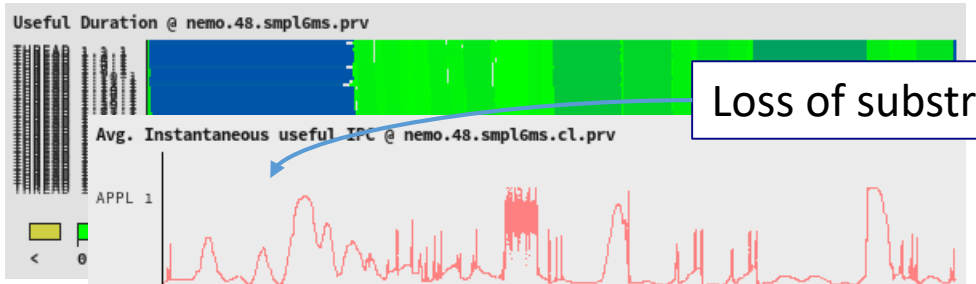
- End
- domvvl_m..nterpol_ [domvvl_mp_dom_vvl_interpol_]
- eosbn2_mp_bn2_
- zdftke_m..tke_tke_ [zdftke_mp_tke_tke_]
- zdftke_m..tke_avn_ [zdftke_mp_tke_avn_]
- zdfiwm_m..zdf_iwm_ [zdfiwm_mp_zdf_iwm_]

- End
- dynzdf_m..dyn_zdf_ [dynzdf_mp_dyn_zdf_]
- sshwzv_mp_wzv_
- traqsr_m..tra_qsr_ [traqsr_mp_tra_qsr_]
- traadv_m..tra_adv_ [traadv_mp_tra_adv_]

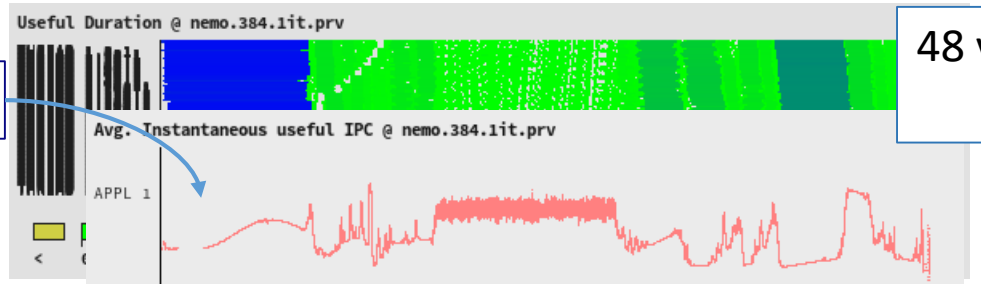
- End
- lbclnk_m.._3d_ptr_ [lbclnk_mp_mpp_lnk_3d_ptr_]
- ldftra_m..eiv_trp_ [ldftra_mp_ldf_eiv_trp_]
- traadv_f..adv_fct_ [traadv_fct_mp_tra_adv_fct_]
- traadv_f.._nonosc_ [traadv_fct_mp_nonosc_]

- End
- lbclnk_m.._3d_ptr_ [lbclnk_mp_mpp_lnk_3d_ptr_]
- traadv_f..adv_fct_ [traadv_fct_mp_tra_adv_fct_]
- traldf_i..ldf_iso_ [traldf_iso_mp_tra_ldf_iso_]
- trazdf_m..zdf_imp_ [trazdf_mp_tra_zdf_imp_]
- tra_nxt_vvl

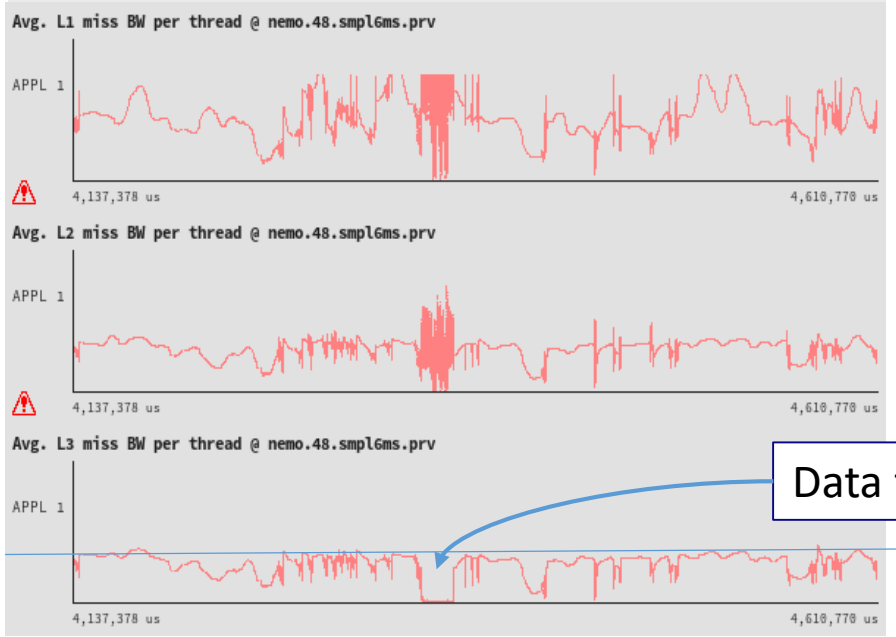
Aggregated time vaying behavior

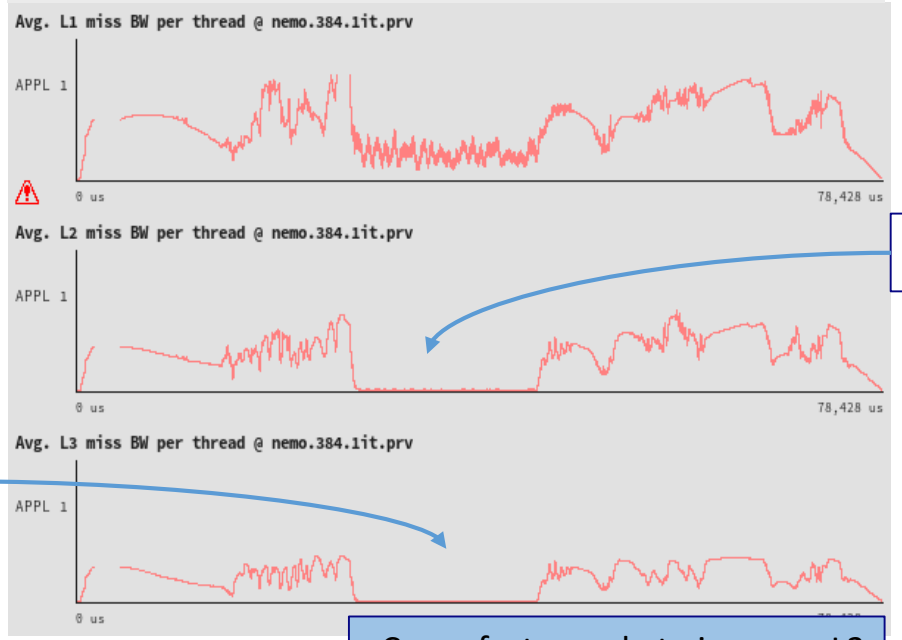


Loss of substructure?



48 vs 384 processes
IPC - BW





Data fits in L2!

Data fits in L3!

Node BW limit

Can refactor code to improve L3 use? Blocking?

Co-design: no need of L3 ?

POP and ES

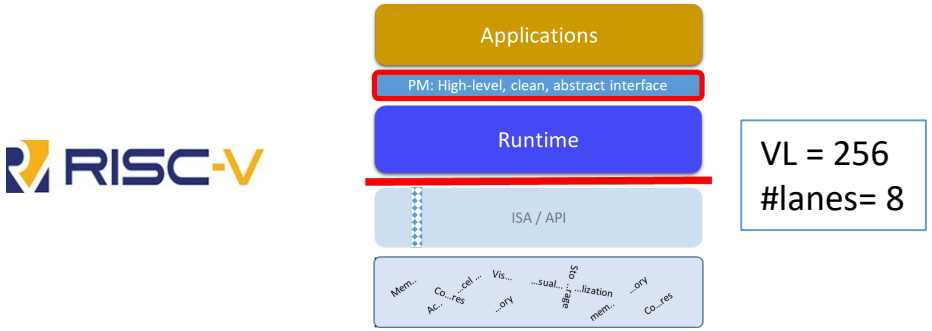
- Observations …
  - Granularities
  - Instruction scaling
  - IPCs and Memory bandwidth
  - L3 use
  - Pack-unpack
  - False sharing
- Recommendations: Asynchrony and overlap
  - Tasks
- … and co-design
  - RISC-V vector
  - OpenMP
    - Features: Free agents, precompiled task graphs
    - Libraries: DLB, TAMPI, TALP

RISC-V & Long vectors

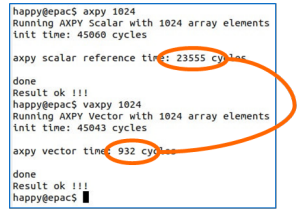
- Raise ISA semantic level
  - Vector instructions == tasks
  - “less words, more work”
  - The importance of ISA

- Parallelism
  - Decouple Front end – back end
    - Less pressure, throughput orientation
  - OoO execution

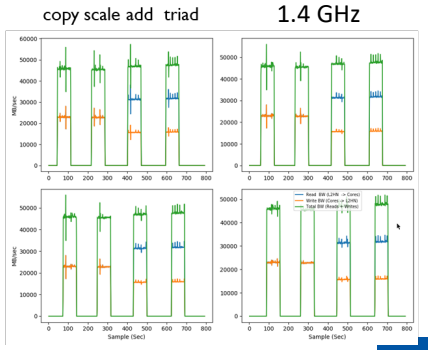
- Osmotic membrane
  - Convey access pattern semantics to the architecture.
  - Potential to optimize memory throughput.



VL = 256
#lanes= 8



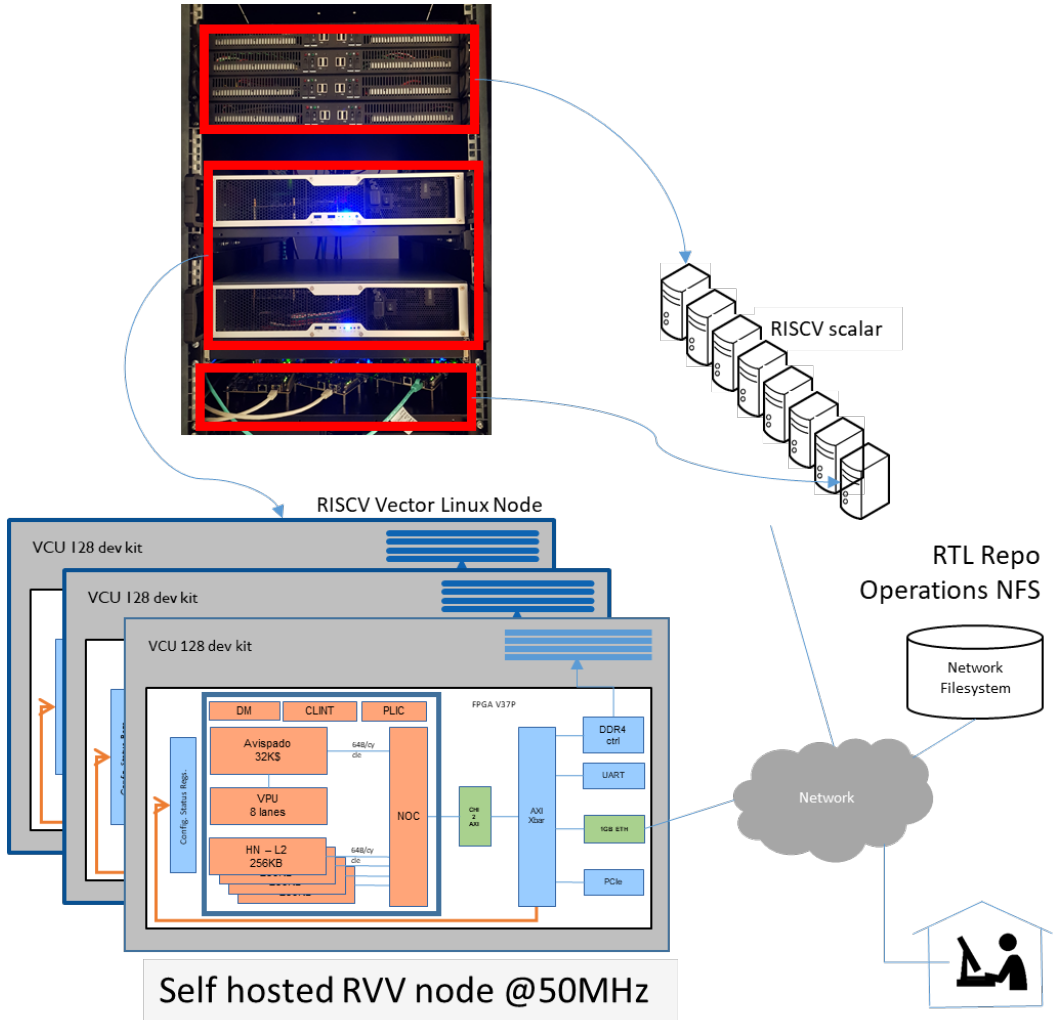
~25x
while only 8x FPUs
→ Long vectors !!
→ Memory Bandwidth



EPI SDV ecosystem

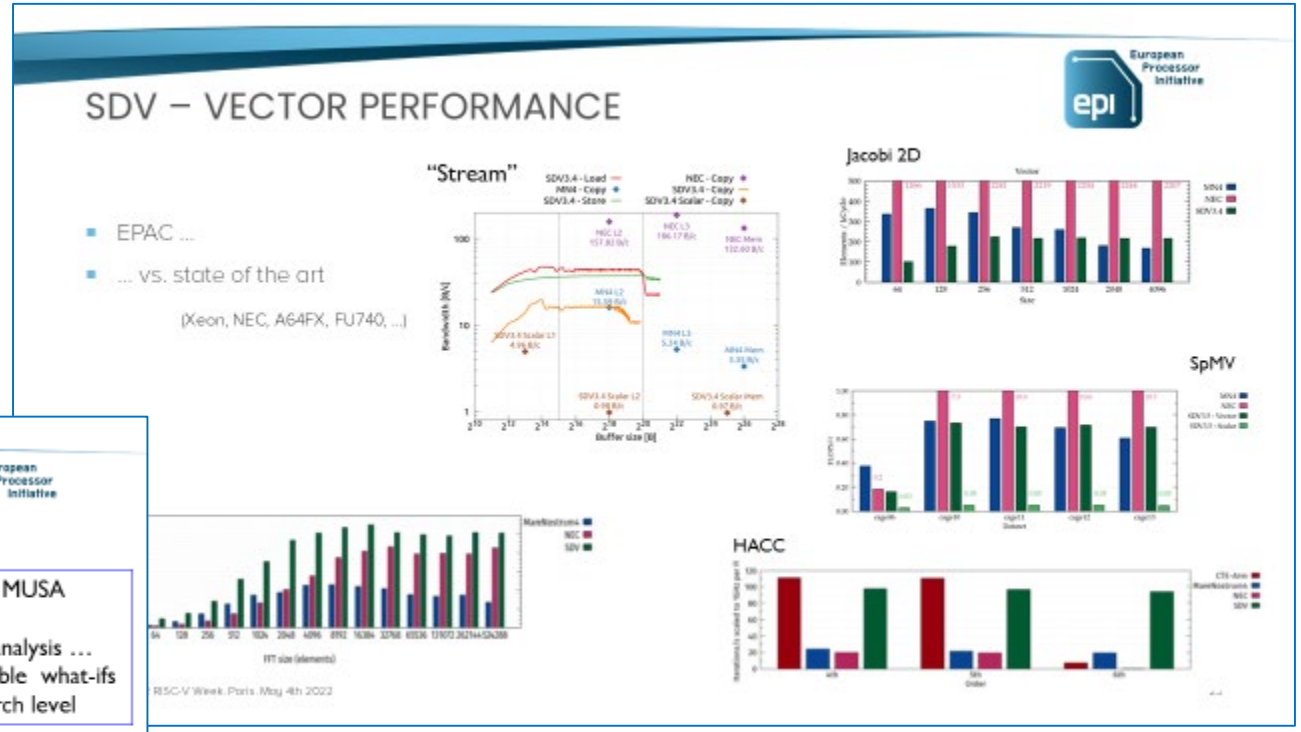
- RISC-V cluster
  - Commercially available RISC-V platforms
  - Porting and configuring HPC software stack and increase productivity (e.g., SLURM, MPI, OpenMP, BSC tools, SDV1.2)

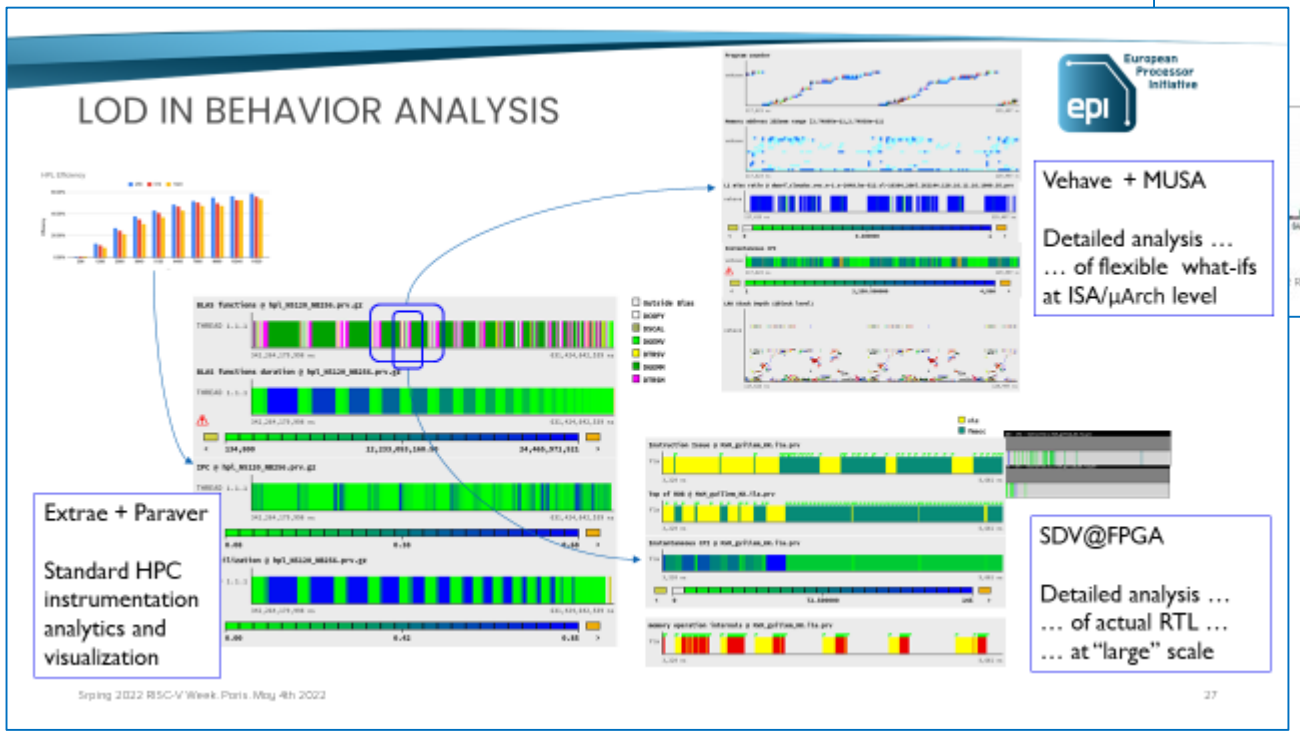
- SDV: RVV @ FPGA nodes
  - CI Infrastructure: Validation at “scale”
  - Software development and co-design steering
    - Test real “complex” codes @ real RTL
    - EPAC1.5 RTL improvement
    - Give to EPI partners and interested users easy access to the latest EPAC technology
    - Two step procedure



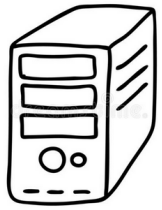
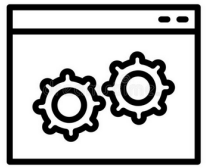
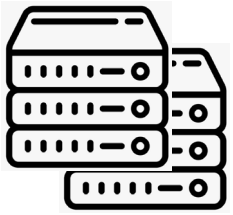
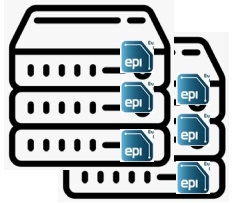
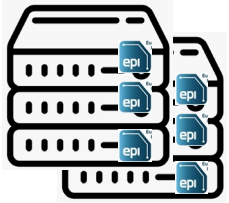
Contact : filippo.mantovani@bsc.es

EPI SDV ecosystem





EPI SDV roadmap

When	Today	June 2022	Dec. 2022	Q1 2023	Q1 2024
HW available	RISC-V scalar platform x 8 	Vehave: Simulator of vec. Instr. 	3 FPGA Single core + VEC acc. 	X FPGA single core + VEC acc. as a multinode 	X FPGA VEC acc. multicore 	EPAC chip 1.5 Multicore and multinode 	EPAC chip 2.0 Multicore and multinode 
What can we do with it?	- Validate code on RISC-V - Test compiler infrastructure	- Analyze vectorizing potential and optimum vector length	- Test VEC acc. architecture	- Test MPI applications on VEC acc.	- Test OpenMP applications on VEC acc.	- Test applications on EPAC 1.5	- Test applications on EPAC 2.0

Performance Optimisation and Productivity

A Centre of Excellence in HPC

Contact:

https://www.pop-coe.eu

mailto:pop@bsc.es

@POP_HPC

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676553 and 824080.