POP assessments with BSC tools

Judit Gimenez (judit@bsc.es)

EU H2020 Centre of Excellence (CoE)

1 December 2018 – 30 November 2021

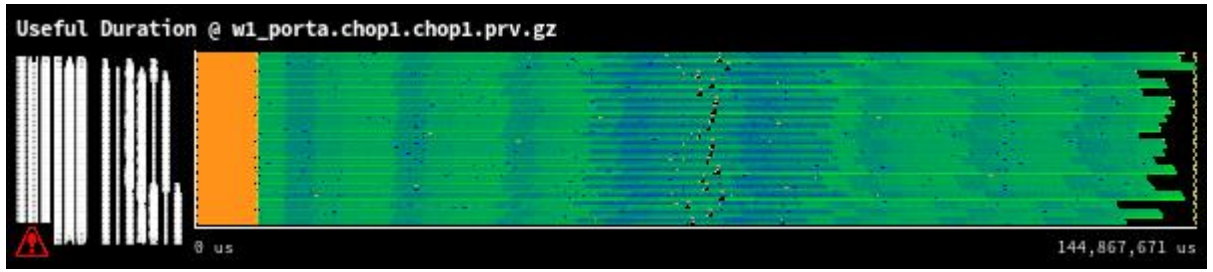
Grant Agreement No 824080

Insights on a MPI study

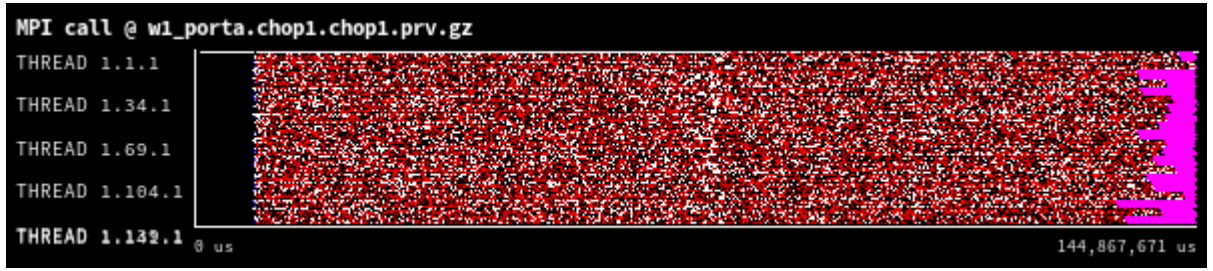
MPI study: background & FoA

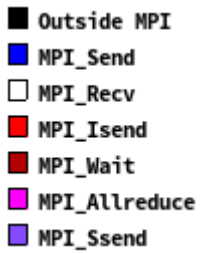
- Name of the code: PORTA
- Programming: C; MPI (+POSIX threads)
- Scale (#cores): 141, 281, 561, 1121, 2241 (weak scaling)
- Platform: BSC MareNostrum4 (node=2 Intel Xeon Platinum 8160 24C @2.1 GHz)

Duration of the computing regions



MPI calls

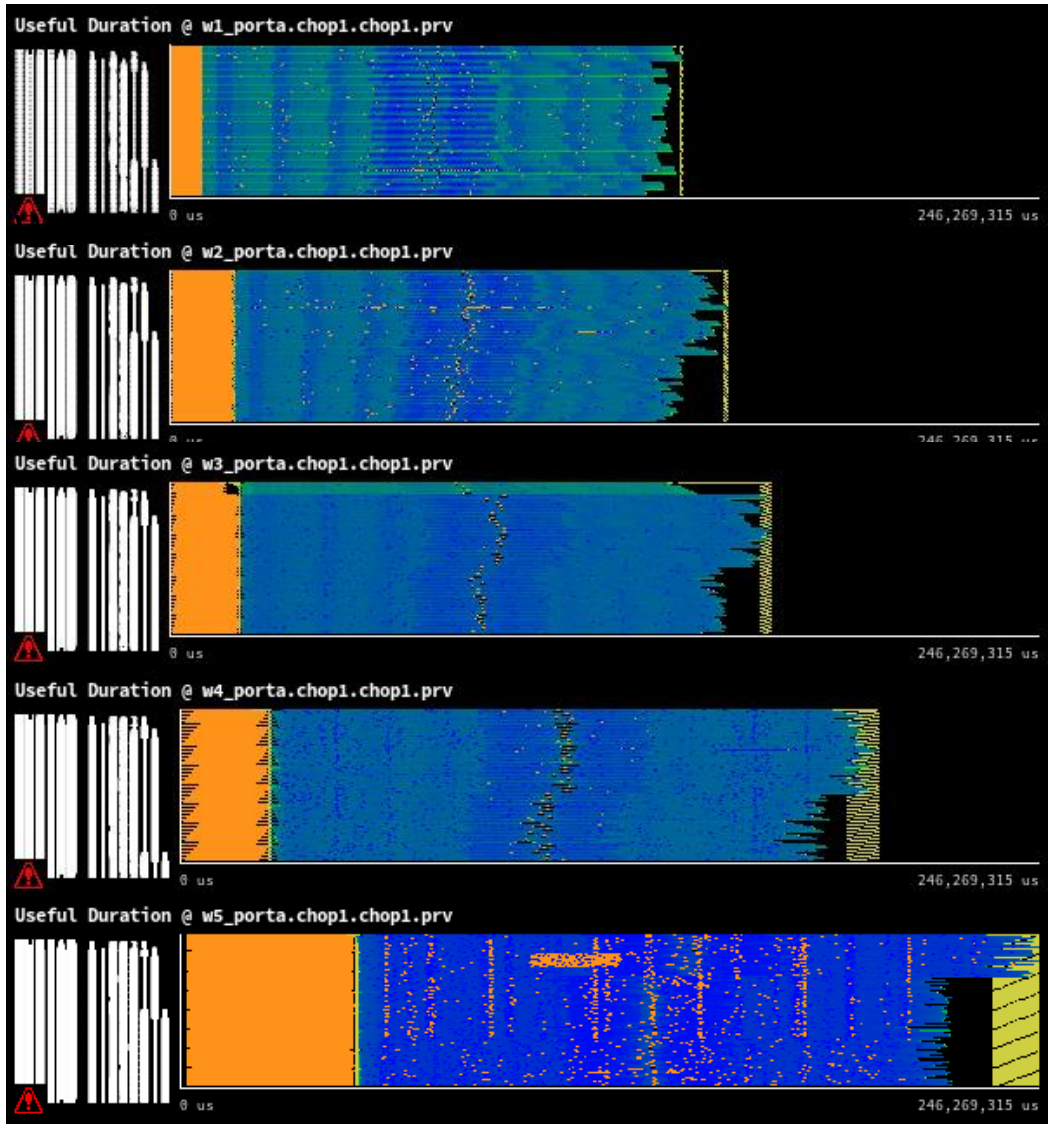


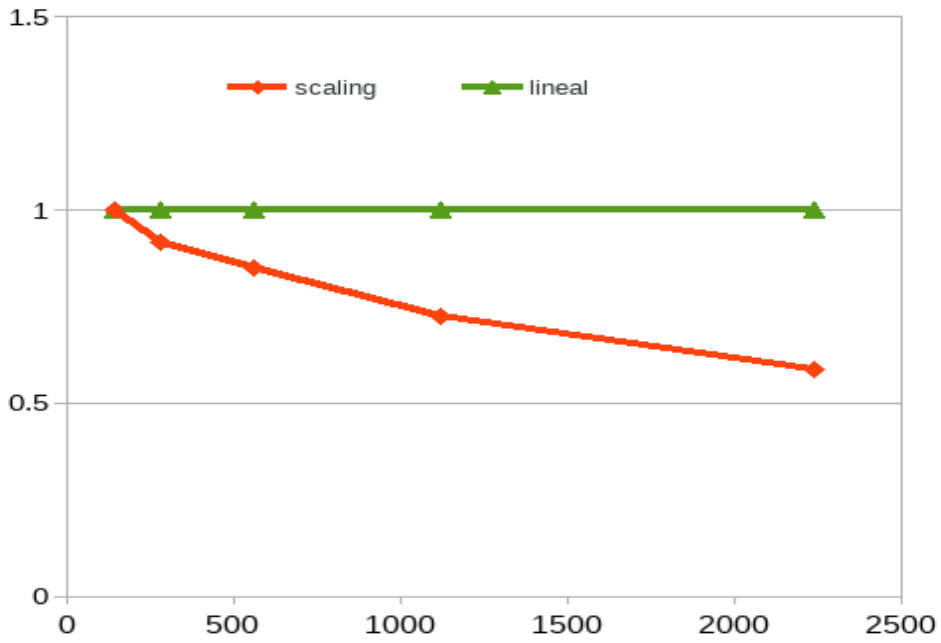


With the small input (W1) the granularity of the main computations are in the range of 70-90 milliseconds (large computation 8 sec.)

Communications using both synchronous and asynchronous calls and ranks are globally synchronized with an MPI_Allreduce

Code scaling



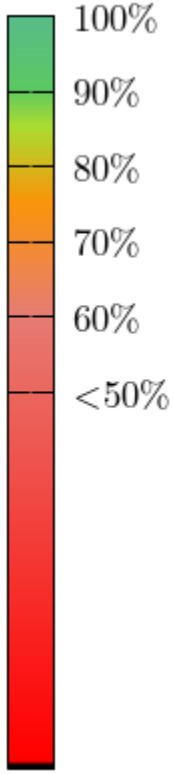


All the timelines (right) have the scale of W5

- Up to W3 the scaling is good.
- W4 and W5 show an **increase of the unbalance** and a bad scaling of the large computing phase.
- **Large computation (orange regions) have severe scaling problems**

Efficiency model analysis

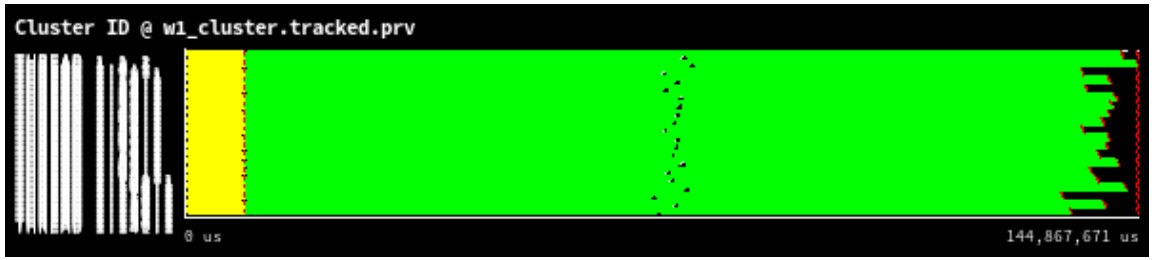
	141	281	561	1121	2241
Global efficiency	91.42	83.74	77.62	66.02	53.60
-- Parallel efficiency	91.42	88.35	85.46	81.02	72.48
-- Load balance	92.47	90.34	89.81	91.28	86.62
-- Communication efficiency	98.87	97.79	95.16	88.76	83.67
-- Serialization efficiency	99.23	98.04	95.44	89.40	84.54
-- Transfer efficiency	99.64	99.75	99.70	99.29	98.97
-- Computation scalability	100.00	94.78	90.82	81.48	73.96
-- IPC scalability	100.00	102.30	100.99	100.22	103.10
-- Instruction scalability	100.00	92.64	88.15	81.27	71.76
-- Frequency scalability	100.00	100.02	102.03	100.04	99.96

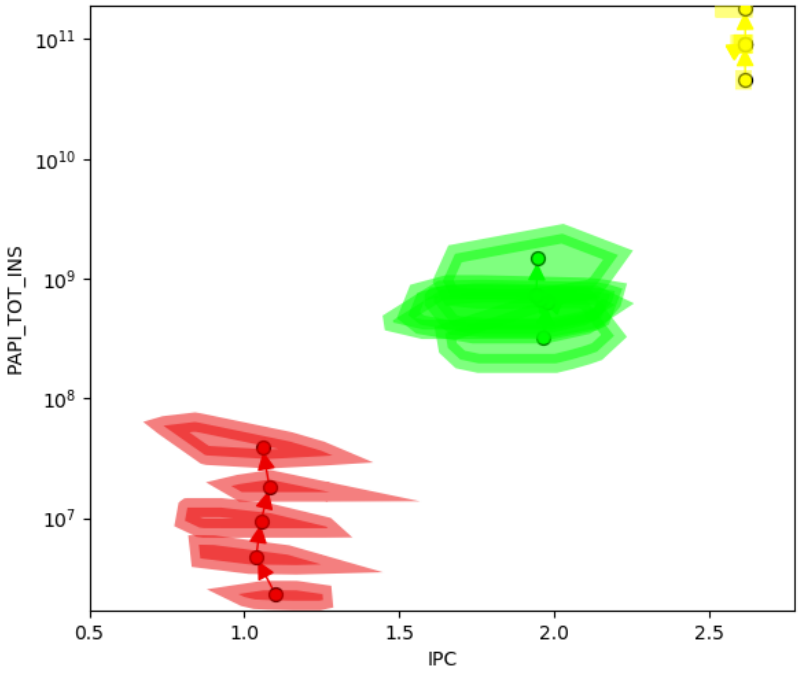


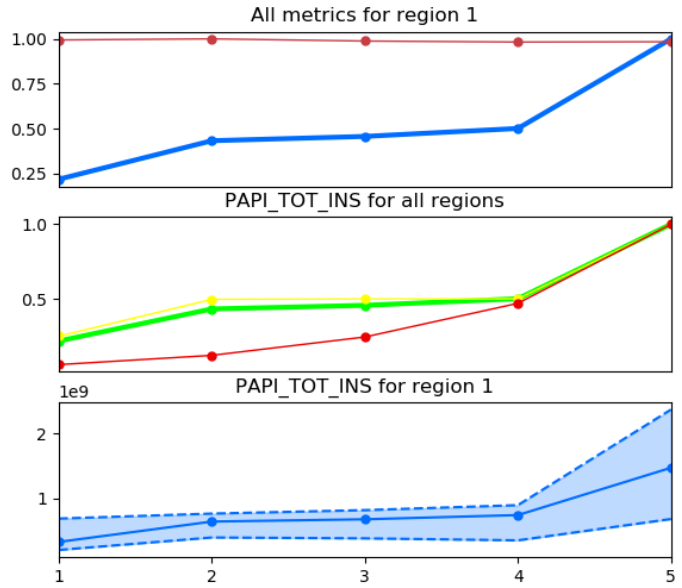
- The **scaling** of the code is limited by:
  - An **increase in the total number of instructions** (code replication).
  - **Serialization and dependencies**.
- The main degradation for W1 is due to **global load balance** despite the efficiency is still very good.

- Efficiencies lower than 80% indicate space for improvement. Lower than 60% there is a clear need for improvement.
- The average IPC is 2.03 that is a good value for MN4 where frequently is limited to 1.2-1.5

Computations scaling





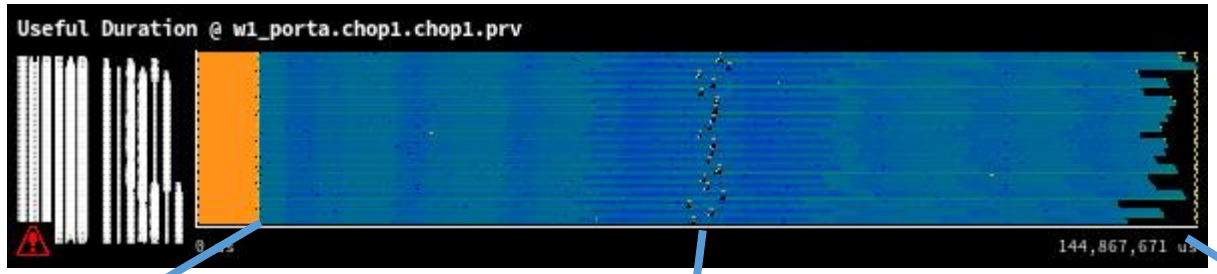


- Tracking the evolution of the clusters we can see that:
  - Clusters 1 and 2 only increase the number of instructions (per instance) for the configurations that increase number of frequencies per rank.
  - The number of grid points increases the number of invocations of the clusters
  - The variability on instructions for cluster 1 increases drastically in W5.

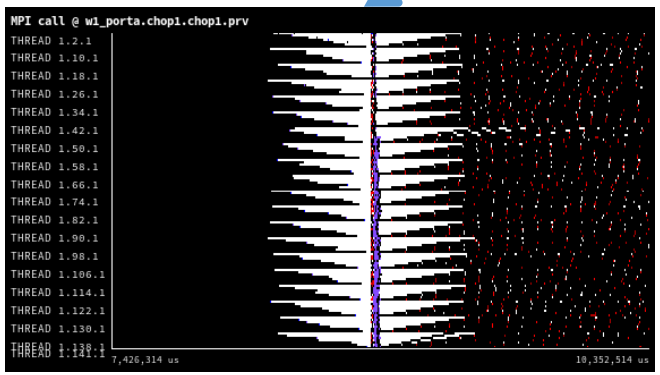
MPI waiting time

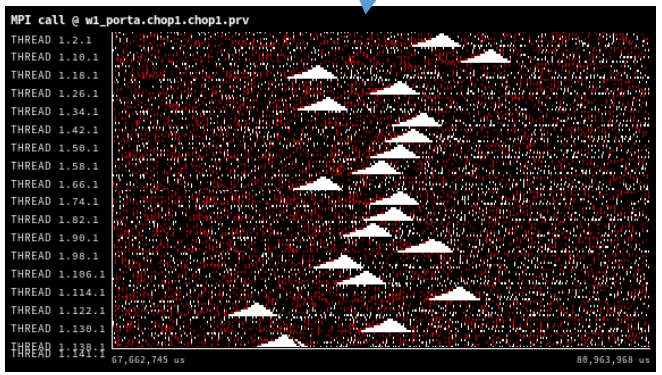
- Most of the MPI waiting time for MPI_Recv and MPI_Allreduce is concentrated in 3 regions

W1



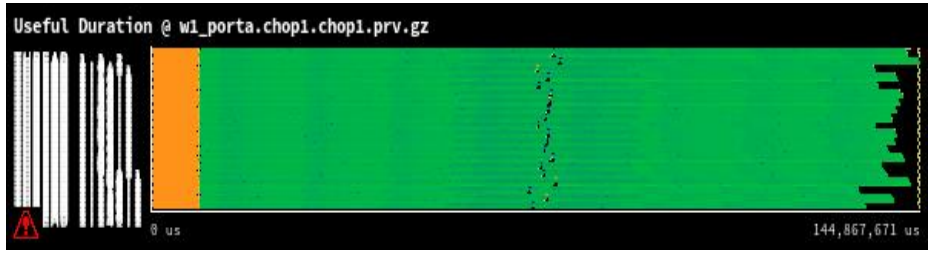
Part of it is reported as global **unbalance**, but also as **serialization** (compensated unbalances in MPI_Recv)



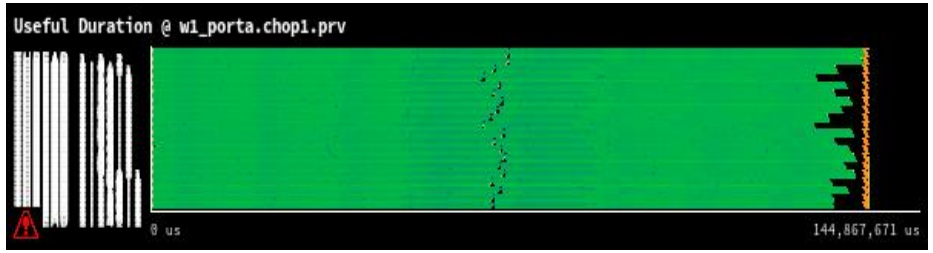


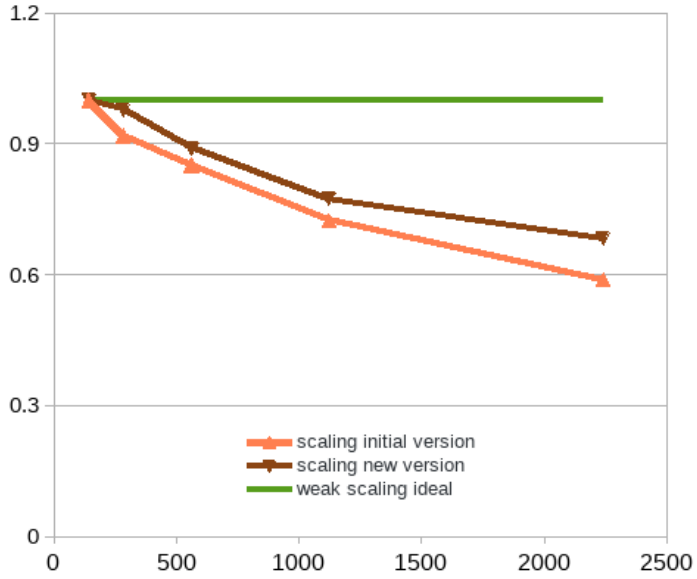
Improved version (few months later)

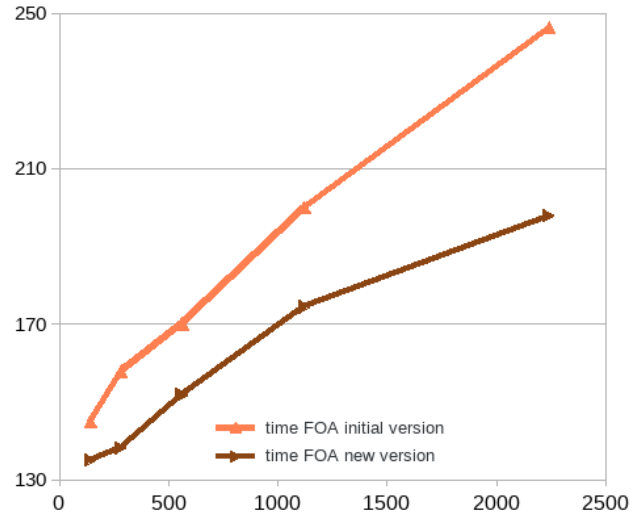
FOA first Audit



FOA Follow on Audit







The improved code reduces 7% the iteration (smallest scale).

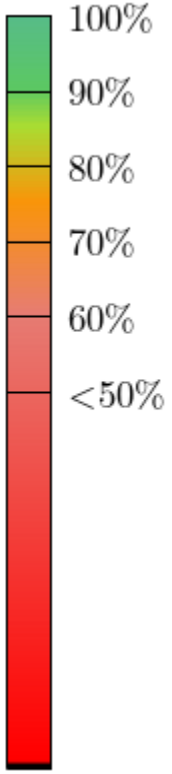
- Refactoring large computation
- Reduce code replication

Comparing the scaling efficiency and elapsed time with respect to the previous version

- The scaling (left) has been improved in average an 8% that goes up to 16% with W5
- The iteration time (right) has been reduced in average a 12% (close to 20% with W5)

Efficiency model analysis

	141	281	561	1121	2241
Global efficiency	92.04	89.81	81.69	71.01	62.61
-- Parallel efficiency	92.04	89.23	86.07	83.35	73.38
-- Load balance	92.84	91.03	89.91	90.19	82.45
-- Communication efficiency	99.14	98.03	95.73	92.42	88.99
-- Computation scalability	100.00	100.64	94.91	85.19	85.33
-- IPC scalability	100.00	100.46	100.19	98.68	98.85
-- Instruction scalability	100.00	100.18	94.70	86.30	86.39
-- Frequency scalability	100.00	100.01	100.03	100.03	99.92



- The scaling of the code has been improved significantly by **reducing the code replication** (W5: instruction scaling from 71.76 to 86.39)
- The parallel efficiency reports very similar values (W5: 72.48 vs. 73.38)
  - **Comm. eff. improves** (83.67 → 88.99)
  - **LB eff degrades** (86.62 → 82.45)

- Efficiencies lower than 80% indicate space for improvement. Lower than 60% there is a clear need for improvement.
- The average IPC is 1.96 (previously 2.03)

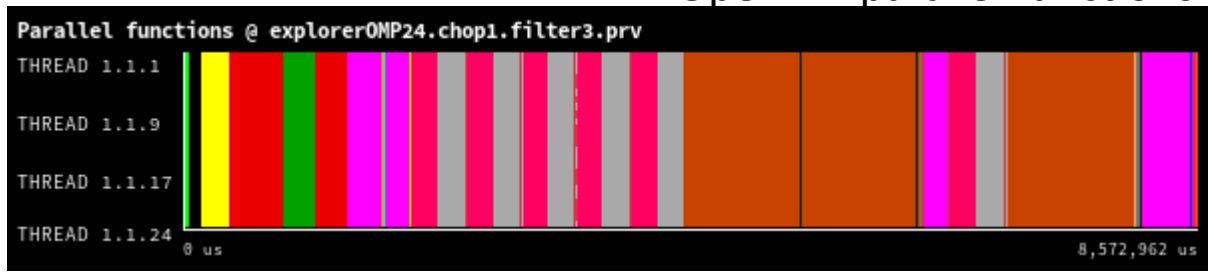
Insights on an OpenMP study

OpenMP study: background and FoA

- Name of the code: Explorer
- Programming: C++; MPI, OpenMP
- Scale (#cores): 1 MPI rank scaling OpenMP (4 , 24, 48)
- Platform: BSC MareNostrum4 (node=2 Intel Xeon Platinum 8160 24C @2.1 GHz)

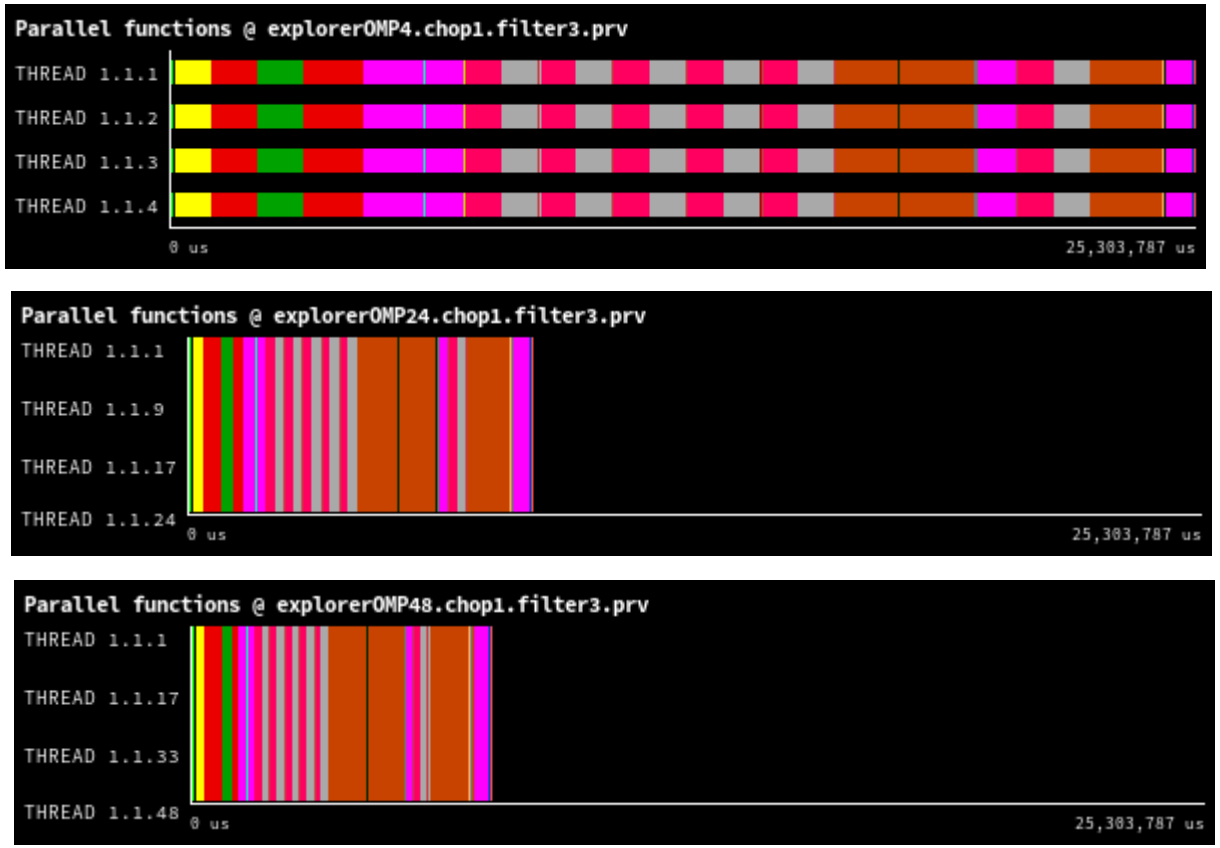
User events

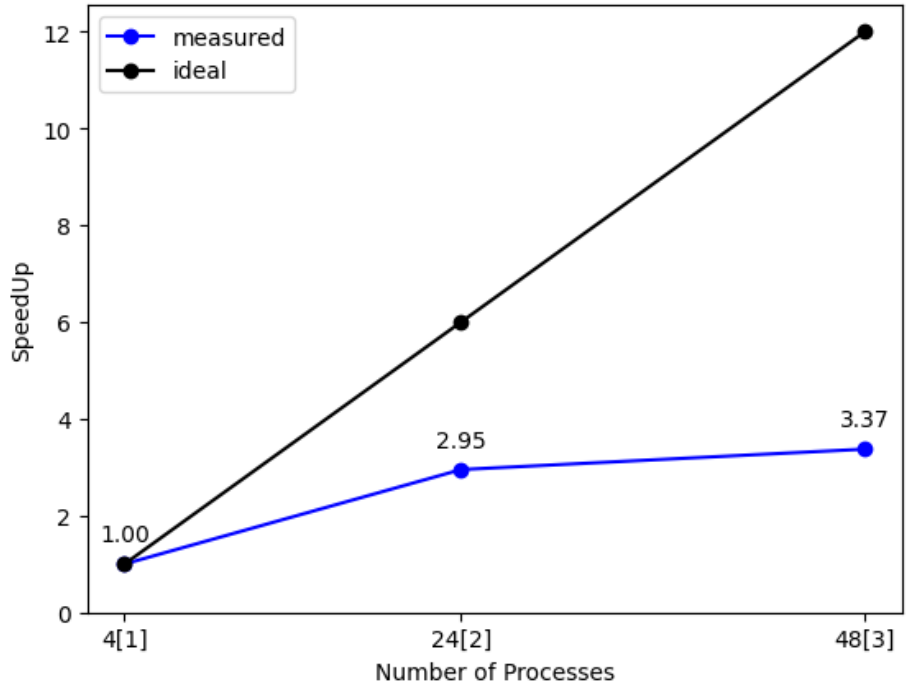
OpenMP parallel functions



The OpenMP parallel function reflects the application structure with the phases defined by the user events

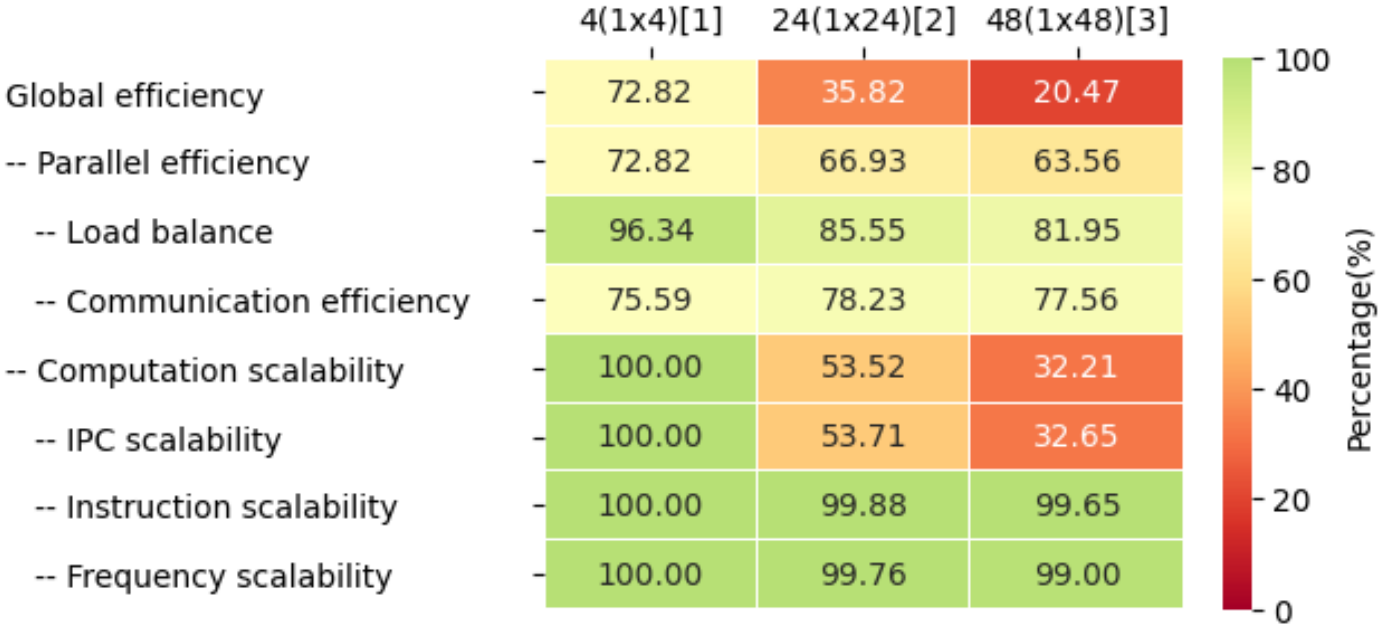
Code scaling





The OpenMP scaling is poor. With 24 processes the efficiency is 49% (2.95x with 6x more resources). Doubling to 48 there is a very small time reduction (efficiency is only 28% of an ideal scaling).

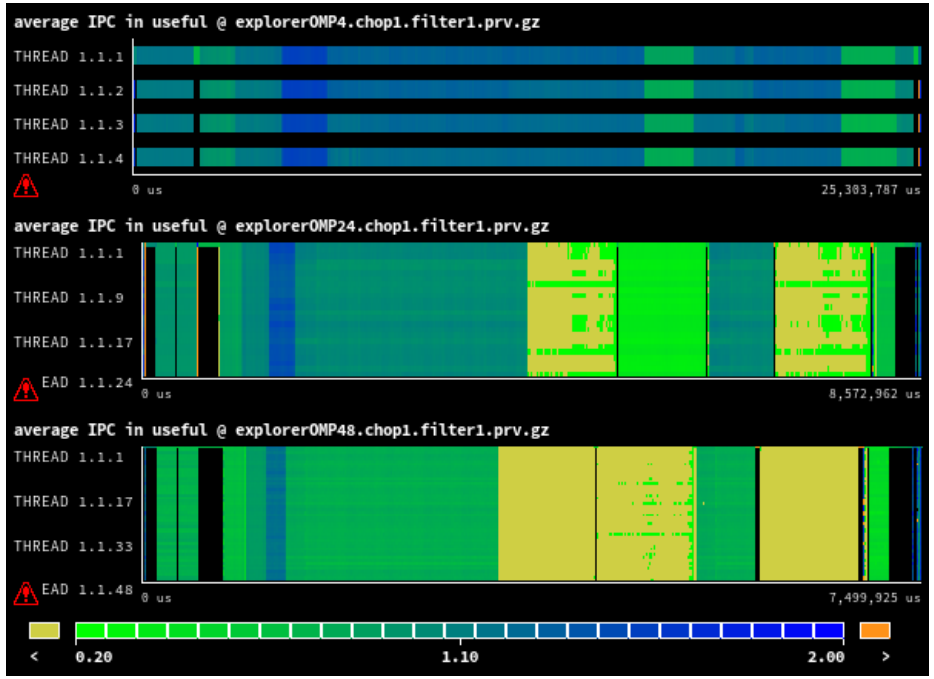
Efficiency model analysis



- The **scaling** of the code is limited mainly by:
  - **A drastic reduction of the IPC**
  - An **increase of the load unbalance**
- The main degradation with 4 threads is due to **time inside the OpenMP runtime** (Comm. Eff.)

- Efficiencies lower than 80% indicate space for improvement. Lower than 60% there is a clear need for improvement.
- The average IPC is 1.14 with 4 threads and it significantly decreases. That seems to indicate the problem can be related with the accesses to the shared memory and/or with the frequent calls to the OpenMP runtime.

IPC degradation analysis



(left) All timelines are in the same color scale. The timelines show that **the IPC reduction is on the computation of almost all the phases**.

(right) Only the pink and red parallel functions maintain an acceptable IPC. The brown one has a very low IPC for all the runs but also suffers the biggest reduction with the scale.

Average

	Interact..:compute	Interact..:compute	Interact..:compute	CachedPa..uteCache	CachedPa..uteCache	Interact..:compute
Total	5.29	4.96	4.91	4.47	3.75	3.50
Average	1.32	1.24	1.23	1.12	0.94	0.88
Maximum	1.36	1.27	1.25	1.13	0.95	0.90
Minimum	1.29	1.22	1.21	1.10	0.93	0.86
StDev	0.03	0.02	0.02	0.01	0.01	0.01
Avg/Max	0.97	0.98	0.98	0.99	0.99	0.98

Average

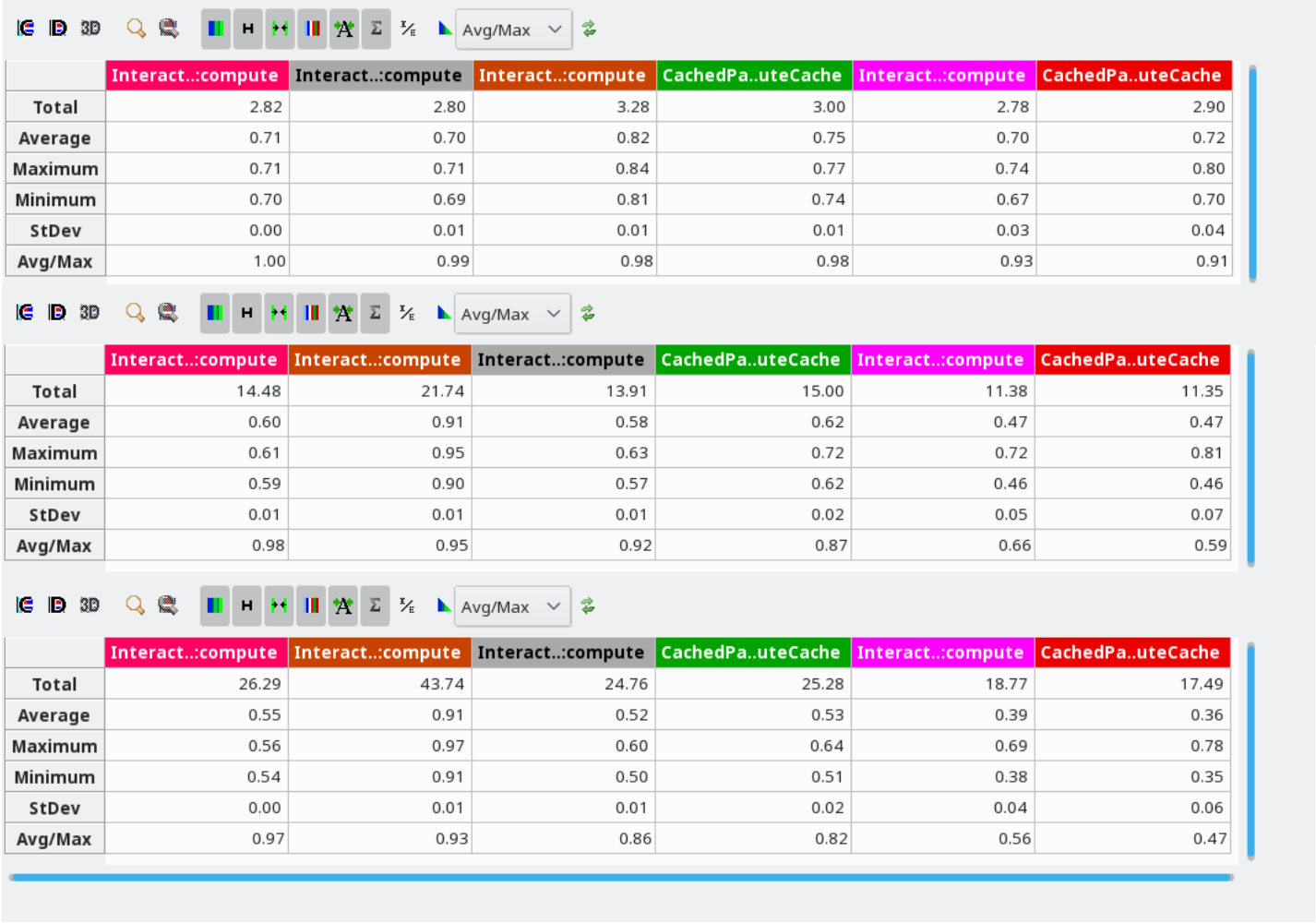
	Interact..:compute	CachedPa..uteCache	Interact..:compute	Interact..:compute	CachedPa..uteCache	Interact..:compute
Total	26.31	23.66	22.89	22.86	19.85	5.47
Average	1.10	0.99	0.95	0.95	0.83	0.23
Maximum	1.15	1.01	1.00	1.01	0.84	0.29
Minimum	1.05	0.85	0.90	0.90	0.80	0.21
StDev	0.03	0.03	0.03	0.03	0.01	0.02
Avg/Max	0.95	0.98	0.95	0.94	0.98	0.78

CachedPa..uteCache

Average

	Interact..:compute	CachedPa..uteCache	Interact..:compute	Interact..:compute	CachedPa..uteCache	Interact..:compute
Total	37.94	36.74	31.66	31.56	28.14	5.80
Average	0.79	0.77	0.66	0.66	0.59	0.12
Maximum	0.90	0.82	0.70	0.82	0.62	0.21
Minimum	0.74	0.71	0.59	0.58	0.56	0.11
StDev	0.03	0.02	0.02	0.03	0.02	0.01
Avg/Max	0.88	0.93	0.95	0.80	0.95	0.56

Parallel functions computations



	Interact..:compute	Interact..:compute	Interact..:compute	CachedPa..uteCache	Interact..:compute	CachedPa..uteCache
Total	2.82	2.80	3.28	3.00	2.78	2.90
Average	0.71	0.70	0.82	0.75	0.70	0.72
Maximum	0.71	0.71	0.84	0.77	0.74	0.80
Minimum	0.70	0.69	0.81	0.74	0.67	0.70
StDev	0.00	0.01	0.01	0.01	0.03	0.04
Avg/Max	1.00	0.99	0.98	0.98	0.93	0.91

	Interact..:compute	Interact..:compute	Interact..:compute	CachedPa..uteCache	Interact..:compute	CachedPa..uteCache
Total	14.48	21.74	13.91	15.00	11.38	11.35
Average	0.60	0.91	0.58	0.62	0.47	0.47
Maximum	0.61	0.95	0.63	0.72	0.72	0.81
Minimum	0.59	0.90	0.57	0.62	0.46	0.46
StDev	0.01	0.01	0.01	0.02	0.05	0.07
Avg/Max	0.98	0.95	0.92	0.87	0.66	0.59

	Interact..:compute	Interact..:compute	Interact..:compute	CachedPa..uteCache	Interact..:compute	CachedPa..uteCache
Total	26.29	43.74	24.76	25.28	18.77	17.49
Average	0.55	0.91	0.52	0.53	0.39	0.36
Maximum	0.56	0.97	0.60	0.64	0.69	0.78
Minimum	0.54	0.91	0.50	0.51	0.38	0.35
StDev	0.00	0.01	0.01	0.02	0.04	0.06
Avg/Max	0.97	0.93	0.86	0.82	0.56	0.47

The average metric is pointing that the application has **a very fine granularity**.

The average row reflects the %time outside the OpenMP runtime

→Values lower than 0.7 indicate a high OpenMP overhead.

→With 24 and 48 threads all functions have a high OpenMP overhead (the brown one has a high value because computations do not scale).

The Avg/Max reflects the load balance

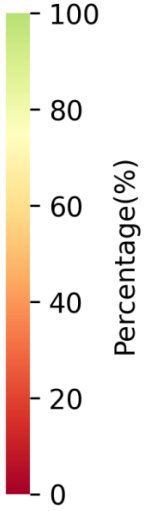
→ Values lower than 0.8 indicate the balance need to be improved.

→ The unbalance with 24 and 48 is concentrated the pink and red parallel functions.

Efficiency model – OpenMP sched.

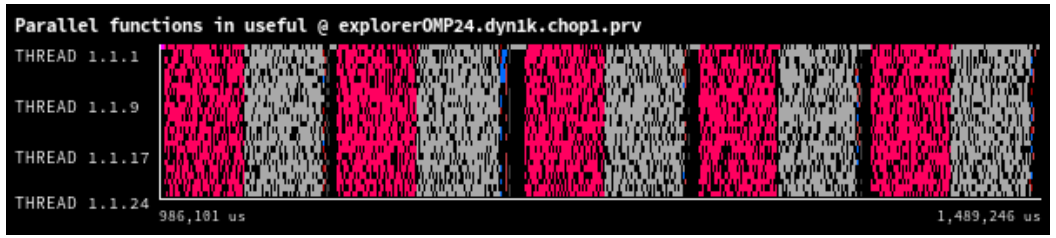
- The application was using the default scheduling (dynamic with chunk size = 1)

	dynamic 100	dynamic 1000	guided 1000	static
Global efficiency	79.26	81.50	80.02	77.19
-- Parallel efficiency	79.26	78.06	77.28	74.06
-- Load balance	79.73	78.74	78.59	78.88
-- Communication efficiency	99.41	99.14	98.34	93.89
-- Computation scalability	100.00	104.41	103.54	104.23
-- IPC scalability	100.00	104.18	103.18	103.98
-- Instruction scalability	100.00	100.32	100.41	100.31
-- Frequency scalability	100.00	99.90	99.94	99.93

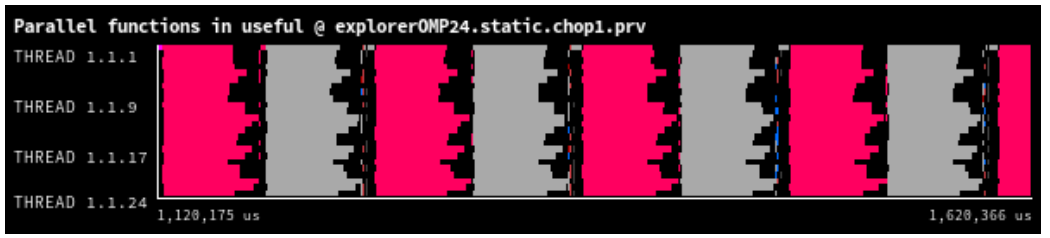


- All configurations (24 threads) are very similar and improve the default (parallel efficiency = 66.9%)
- The average IPC is in the range 0.69 - 0.72. Being still low, but an improvement w.r.t. the default scheduling (0.61)

dynamic, 1000



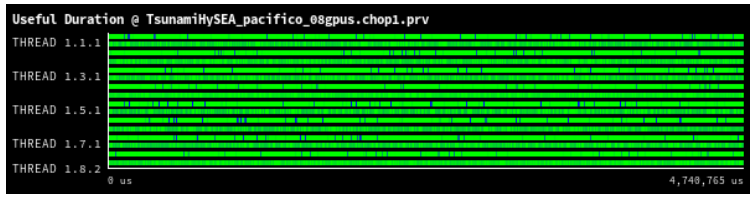
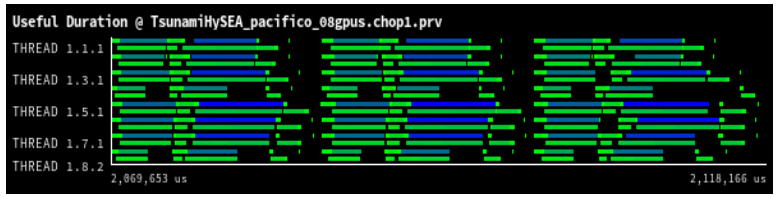
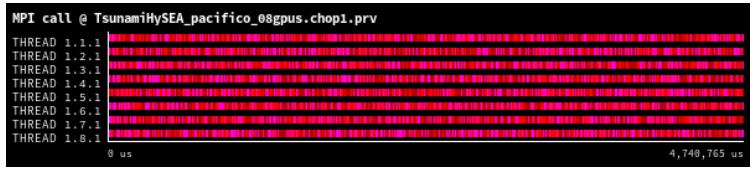
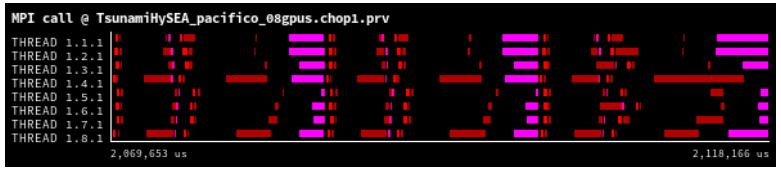
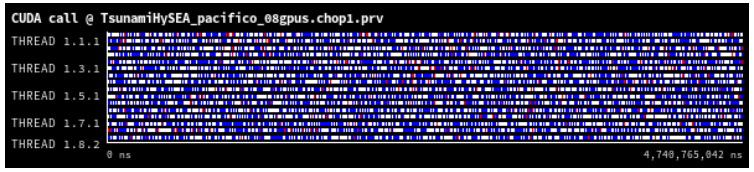
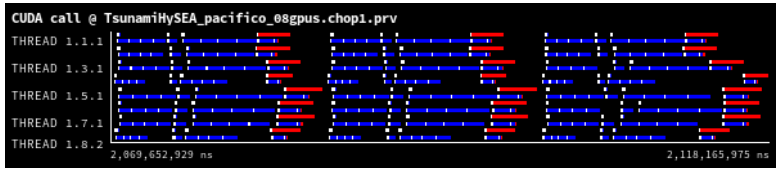
static



Insights on a MPI + CUDA study

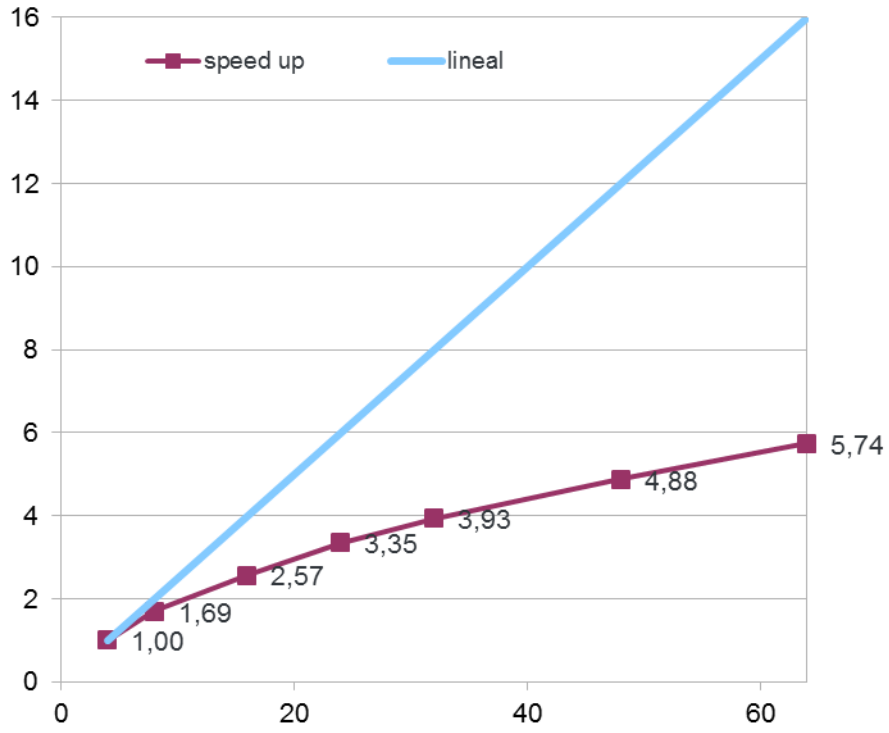
MPI + CUDA: background and FoA

- Name of the code: Tsunami-HySEA
- Programming: C++; MPI, CUDA
- Scale: from 1 MPI rank + 1 GPU to 64 MPI ranks + 64 GPUs
- Platform: BSC CTE Power (first 2 studies) and Leonardo (last study)

	300 iterations	Zoom in few iterations
Computations		
MPI calls		
CUDA runtime		

Scalability

- Time scalability of the FOA with respect to linear scaling



The execution with 1 MPI + 1 GPU was discarded for the analysis because a significant different behaviour of the MPI task.

With 16 MPI + 16 CUDA the speed-up is 64% of lineal scaling.

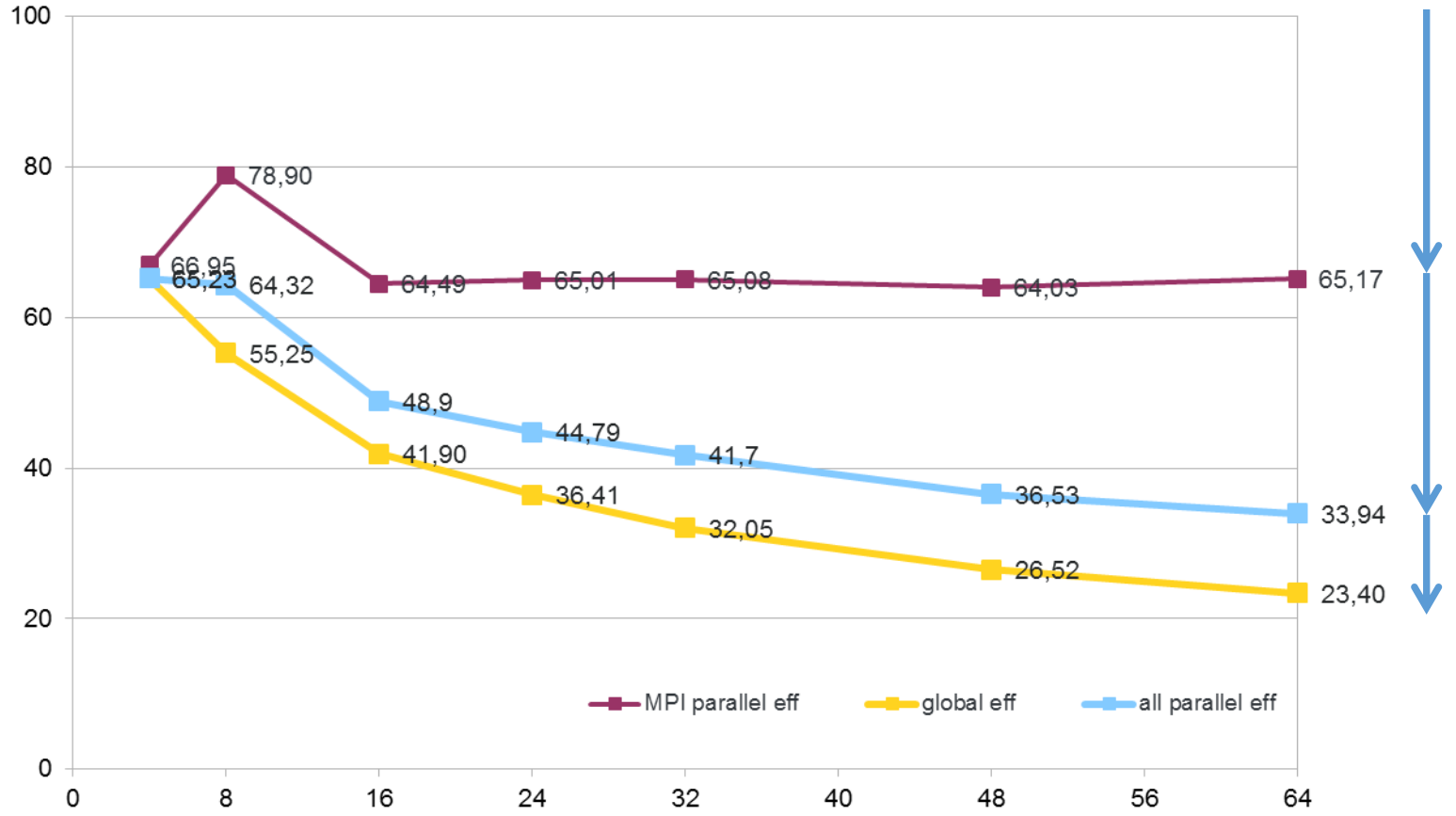
With 64 MPI + 64 CUDA the speed-up goes down to 35%.

Efficiency model

	4	8	16	24	32	48	64
Global efficiency	65.23	55.25	41.90	36.41	32.05	26.52	23.40
Parallel efficiency	65.23	64.32	48.9	44.79	41.7	36.53	33.94
Load Balance	72.57	77.79	67.41	67.86	68.15	64.92	64.67
Communication eff.	89.88	82.68	72.54	66.01	61.19	56.27	52.47
Computation scalability	100.00	85.89	85.69	81.30	76.87	72.60	68.94

- Mainly **communications** but also **computations** limit the code scalability
- The reference run with 4 MPI + 4 GPUs already reports poor efficiency
- **Load balance is poor** in all the configurations **but there is limited degradation** with the scale
- No counters available at GPUs to compute IPC and Instructions scaling factors

Efficiency model



MPI component

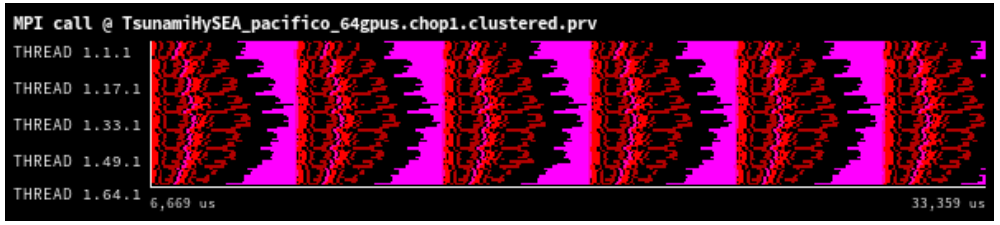
CUDA component

Computations scaling

- Big impact from MPI and CUDA components.
- MPI flat behaviour except with 8 ranks

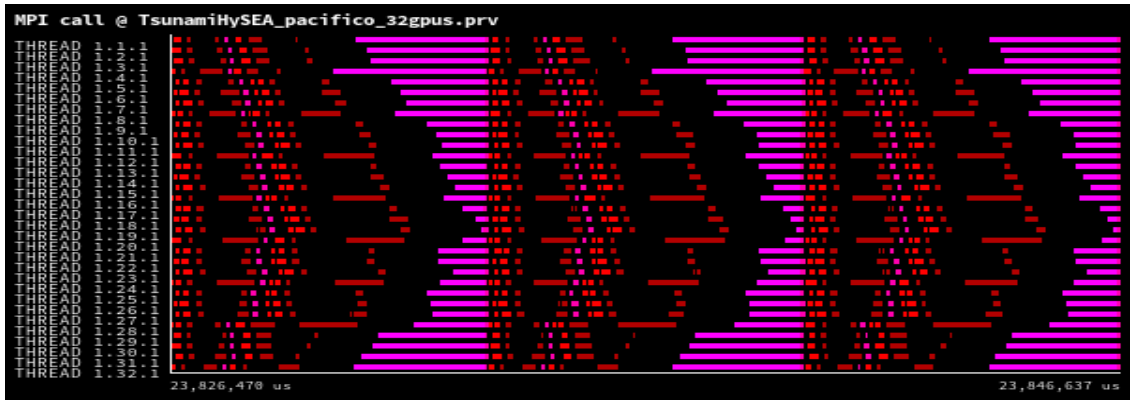
Load balance and data transfer

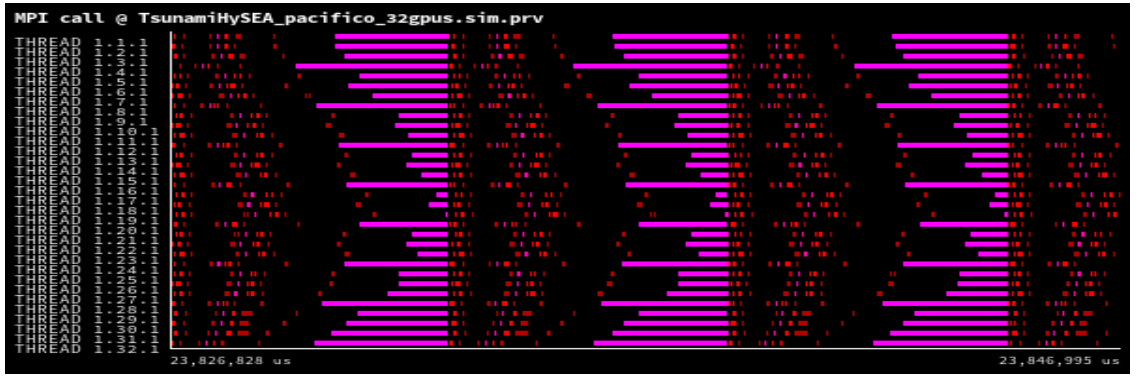
- Scaling of one iteration (MPI processes only)



Parallel efficiency 65%, **Load Balance 76%** , Comm. 85%

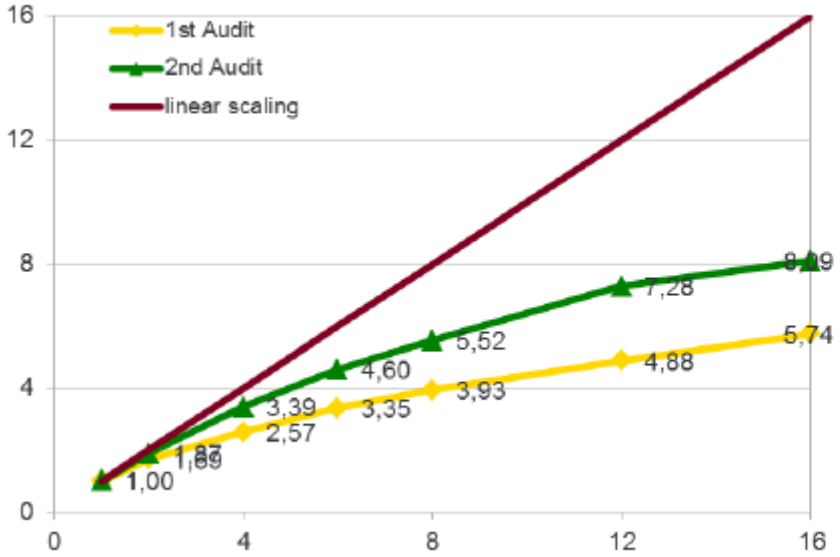
With instantaneous communications most of the point to point calls disappear and the allreduce increases a bit → **data transferred in point to point is limited by network resources**



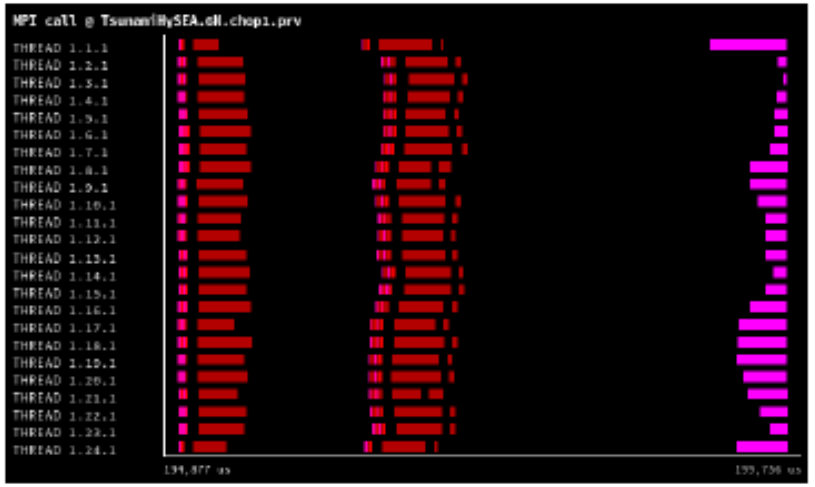


Serialization 99%, **Transfer 84%**

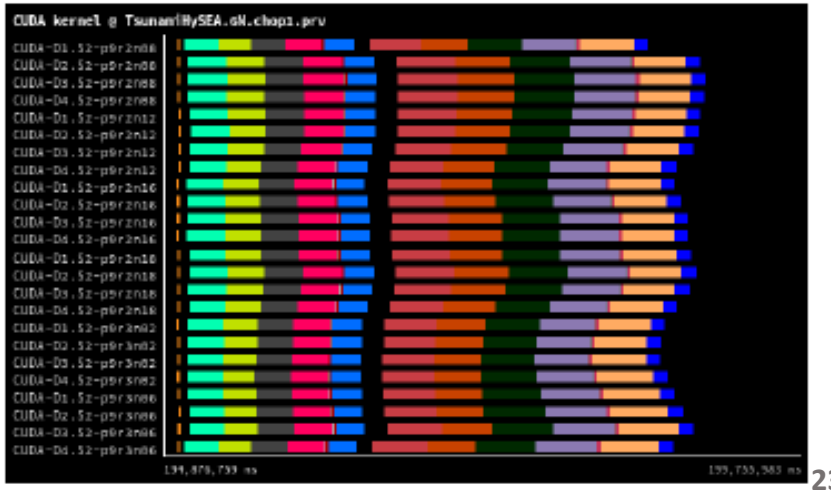
Code scaling (v1 vs. v2)



MPI calls



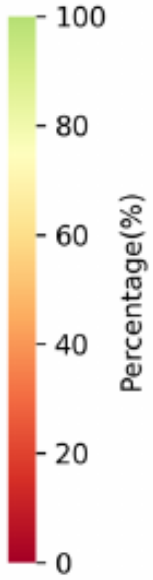
CUDA kernel



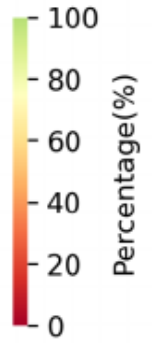
- The scaling is improved an average of 30% (40% for the runs of at least 4 nodes).
- Main improvements (previous audit suggestions):
  - Overlap MPI point to point calls with the kernels execution.
  - Improve balance between MPI ranks

Efficiency model (all resources vs gpus)

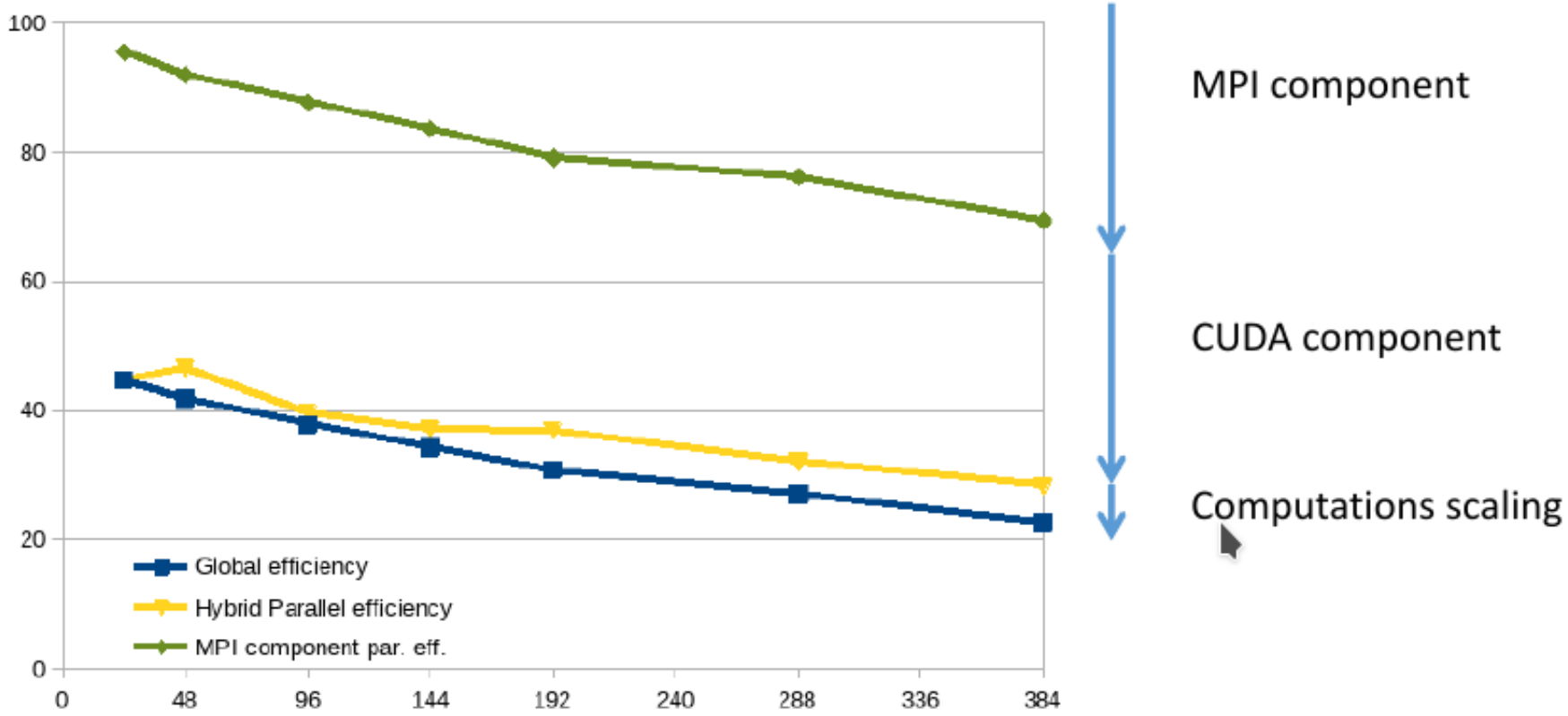
	24	48	96	144	192	288	384
Global efficiency	44.72	41.83	37.86	34.28	30.85	27.14	22.62
-- Computation scalability	100.00	89.85	95.37	92.46	83.47	84.57	79.80
-- Hybrid Parallel efficiency	44.72	46.56	39.70	37.07	36.96	32.10	28.35
-- Load balance	48.68	52.89	48.39	48.73	52.89	49.97	50.84
-- MPI contribution LB eff.	96.41	95.78	95.23	95.08	94.34	94.22	89.63
-- CUDA contribution LB eff.	50.50	55.22	50.82	51.25	56.06	53.04	56.72
-- Communication efficiency	91.86	88.02	82.03	76.07	69.88	64.23	55.76
-- MPI contribution comm. eff.	99.09	96.12	92.13	88.05	83.93	80.92	77.56
-- CUDA contribution comm. eff.	92.71	91.58	89.03	86.39	83.26	79.38	71.89



	24	48	96	144	192	288	384
GPUs only Global efficiency	89.07	83.30	75.40	68.27	61.45	54.06	45.05
-- GPUs only Computation scalability	100.00	98.77	96.56	94.37	92.22	88.15	84.21
-- GPUs only Parallel efficiency	89.07	84.34	78.09	72.34	66.63	61.33	53.50
-- GPUs only Load Balance	96.96	95.83	95.19	95.09	95.34	95.48	95.94
-- GPUs only Communication eff.	91.86	88.02	82.03	76.07	69.88	64.23	55.76



Programing model contribution

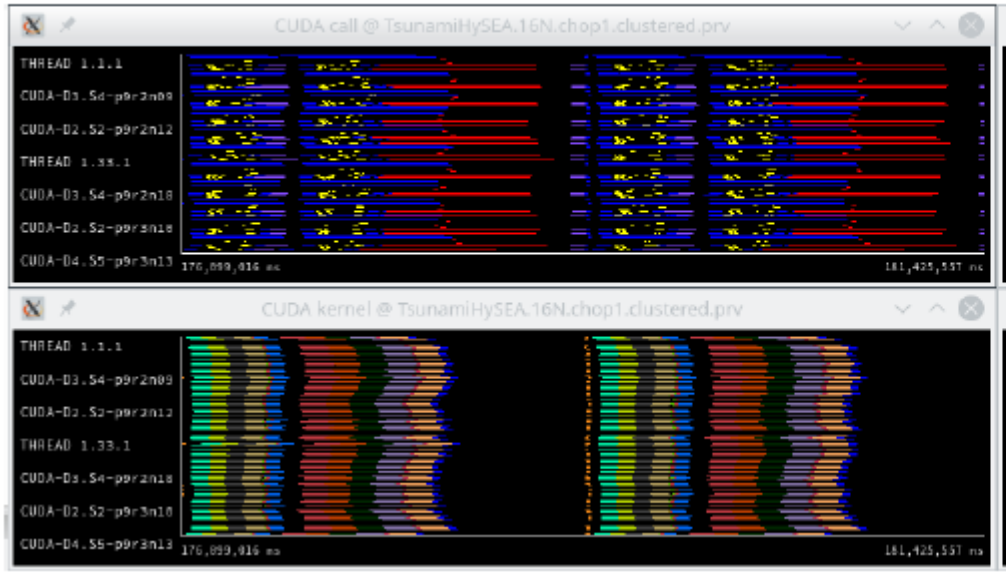


- **Bigger impact from CUDA component** with a similar contribution in all the scales
- MPI contribution increases with the scale, but still lower than CUDA

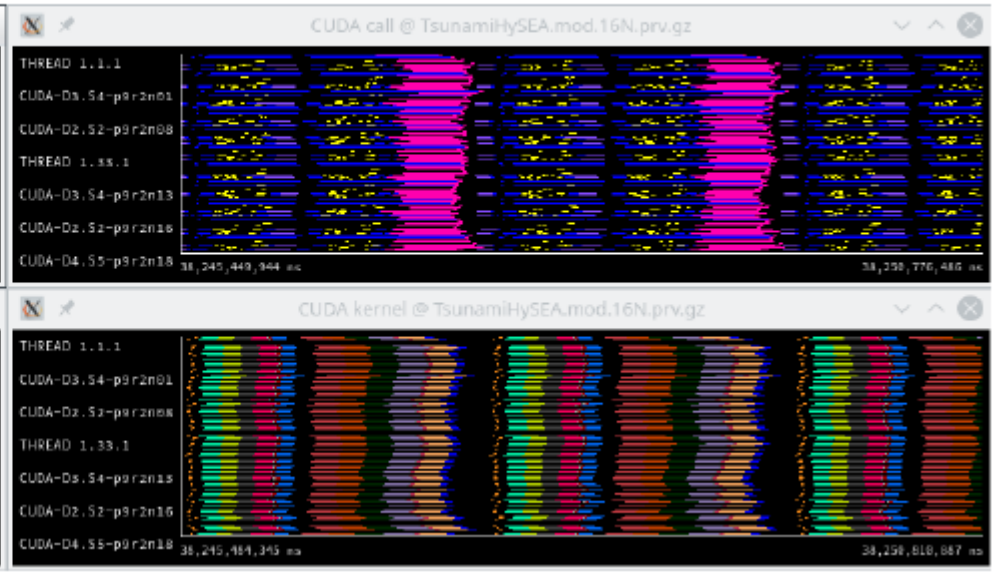
Quick validation of a bottleneck

- Discussing the analysis with the code developers allowed to identify a **performance problem with a CUDA reduction phase**.

Audited version

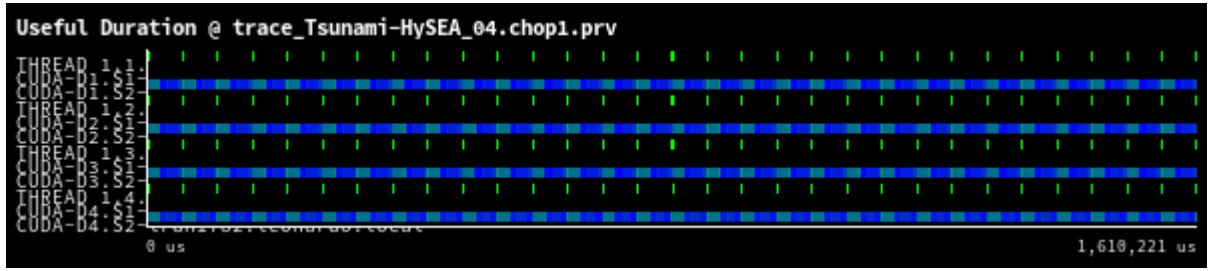


Modified version

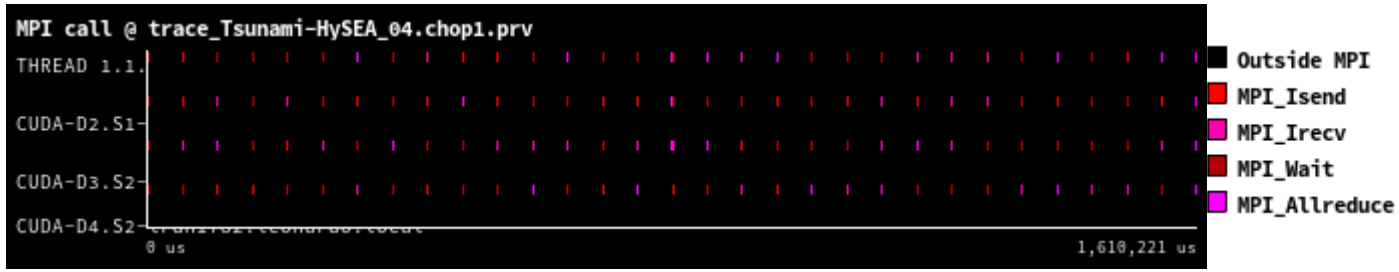


FOA of 2023 version

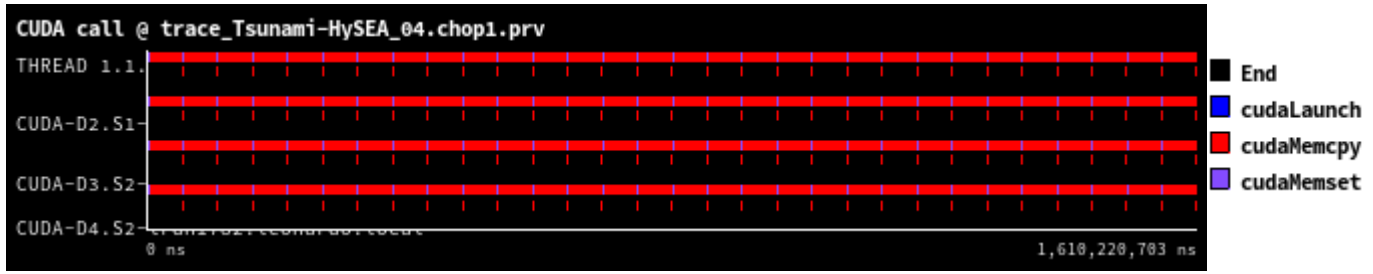
Computations



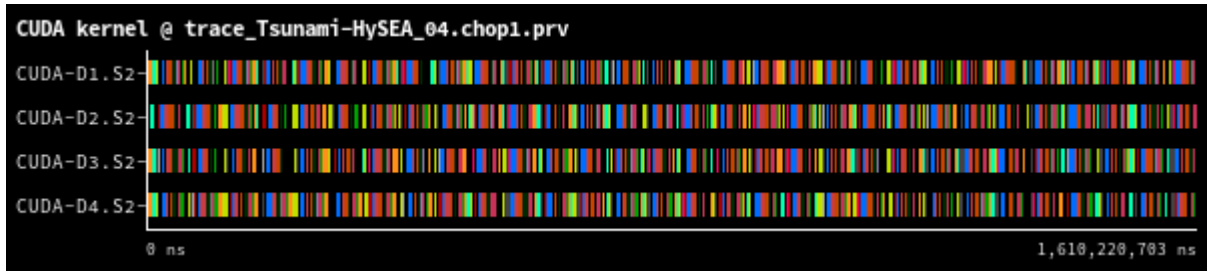
MPI calls



CUDA runtime



CUDA kernel

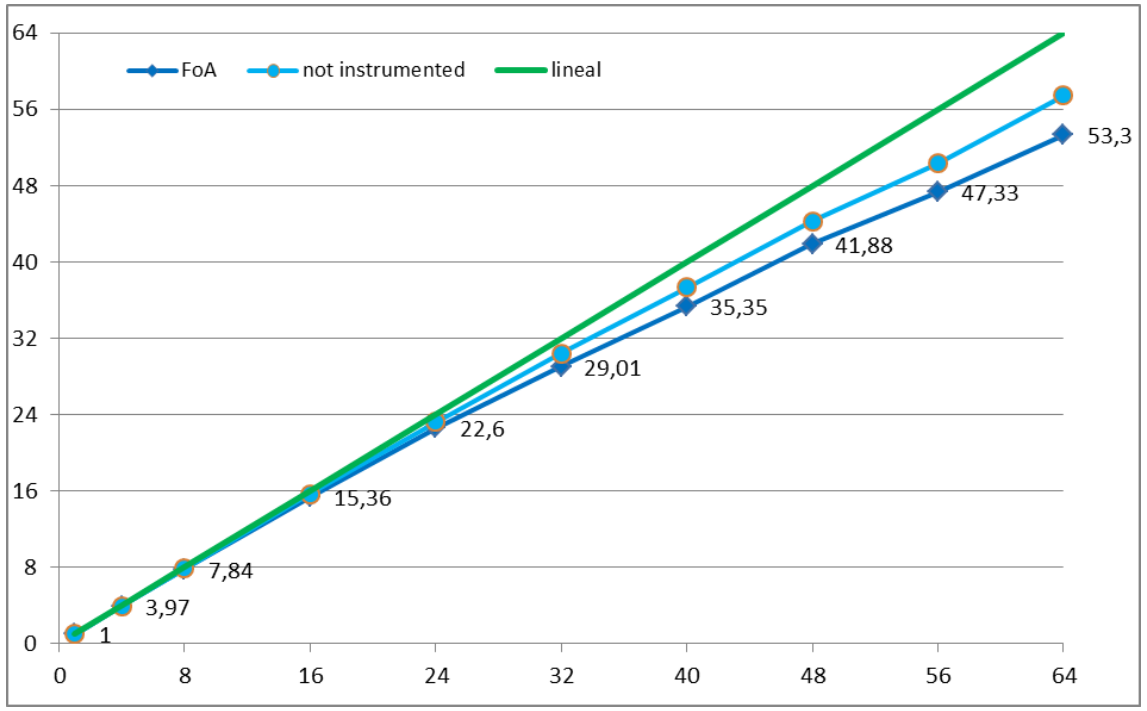


Focusing on 30 iterations and 4 GPUs we can already see:

- the weight of MPI is quite small (less than 1%)
- CPUs spend most of the time in the cudaMemcpy call (over 99%)
- GPUs are executing kernels most of the time (94%)

Scalability

- Time scalability of the FOA with respect to linear scaling

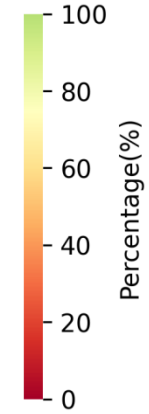


The application reports a good scaling up to 64 GPUs. In fact without instrumentation the user reports a little bit better scaling that is included in the plot and goes to 57.4 with 64 GPUs.
With 16 GPUs the speed-up is 96% of lineal scaling. With 32GPUs is 91%. With the largest scale of 64 GPUs the efficiency is 83%.

As reference of the code improvement, in the previous POP assessment in Nov'20 (for a different input case and configuration) the scaling efficiency with 64 GPUs was just over 50%.

Efficiency model

	1	4	8	16	24	32	40	48	56	64
-- Global Parallel efficiency	47.43	47.19	46.80	46.27	45.73	44.48	43.61	43.30	42.34	42.04
-- Global Load balance	50.00	49.91	49.97	49.91	49.92	49.83	49.59	49.64	49.59	49.40
-- GPUs Global efficiency	94.85	94.08	92.94	91.03	89.33	85.99	83.83	82.77	80.17	79.00
-- GPUs Parallel efficiency	94.85	94.33	93.53	92.39	91.23	88.67	86.87	86.18	84.19	83.52
-- GPUs Load Balance	100.00	99.79	99.86	99.65	99.59	99.33	98.77	98.80	98.62	98.15
-- GPUs Communication efficiency	94.85	94.53	93.66	92.71	91.61	89.27	87.95	87.23	85.37	85.09
-- GPUs Computation scalability	100.00	99.73	99.37	98.53	97.92	96.98	96.50	96.05	95.22	94.59



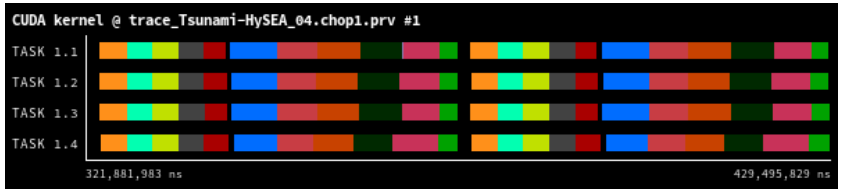
- Based on the low contribution of the CPUs, the efficiency analysis is focused on the GPUs including the global parallel efficiency and load balance as reference.
- Considering all the resources allocated, the efficiencies are around 40-50% because the work is concentrated in the GPUs (reported as unbalance). The small degradation in the parallel efficiency is also observed in the GPUs metrics.
- The GPUs global efficiency reports a degradation with the scale that is related mainly with the communication efficiency but also with the computation scalability. The load balance between GPUs is ok for all the scales.
- No counters available at GPUs to compute IPC and Instructions scaling factors.

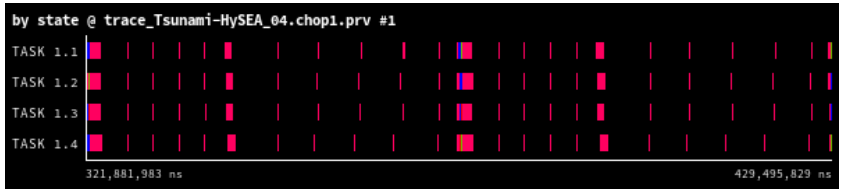
GPU idling time vs. CPU activity

- Comparing 4 GPUs (up) and 64 GPUs (down)

Total

	Memory transfer	Configuring accelerator	Wait/WaitAll	Group Communication	Running	Immediate Send	Immediate Receive
Total	20.8905 %	0.6158 %	0.5416 %	0.4034 %	0.1522 %	0.0483 %	0.0122 %
Average	5.2226 %	0.1540 %	0.1354 %	0.1008 %	0.0380 %	0.0121 %	0.0031 %
Maximum	5.7037 %	0.1547 %	0.1422 %	0.1608 %	0.0414 %	0.0154 %	0.0037 %
Minimum	4.9679 %	0.1532 %	0.1284 %	0.0182 %	0.0345 %	0.0089 %	0.0024 %
StDev	0.2860 %	0.0007 %	0.0066 %	0.0561 %	0.0024 %	0.0032 %	0.0006 %
Avg/Max	0.9156	0.9952	0.9520	0.6271	0.9189	0.7828	0.8175

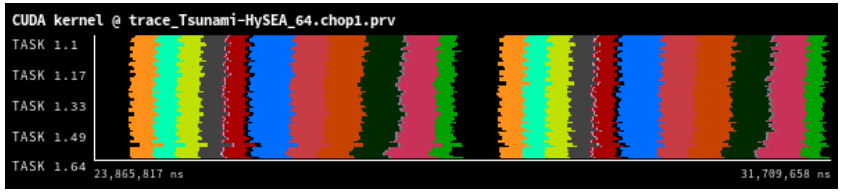


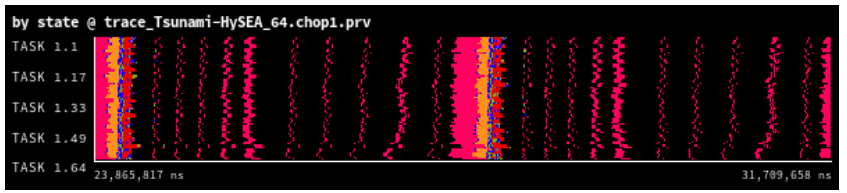


Average

	Memory transfer	Group Communication	Wait/WaitAll	Configuring accelerator	Running	Immediate Send	Immediate Receive
Total	566.3759 %	264.3879 %	136.0887 %	55.1044 %	17.5300 %	9.5718 %	4.4942 %
Average	8.8496 %	4.1311 %	2.1264 %	0.8610 %	0.2739 %	0.1496 %	0.0702 %
Maximum	9.3694 %	5.1927 %	3.2838 %	1.8026 %	0.4432 %	0.2971 %	0.1390 %
Minimum	8.6255 %	2.0512 %	0.9941 %	0.5623 %	0.1805 %	0.0775 %	0.0299 %
StDev	0.1311 %	0.6532 %	0.7086 %	0.2559 %	0.0676 %	0.0519 %	0.0269 %
Avg/Max	0.9445	0.7956	0.6475	0.4777	0.6180	0.5034	0.5050

Memory transfer



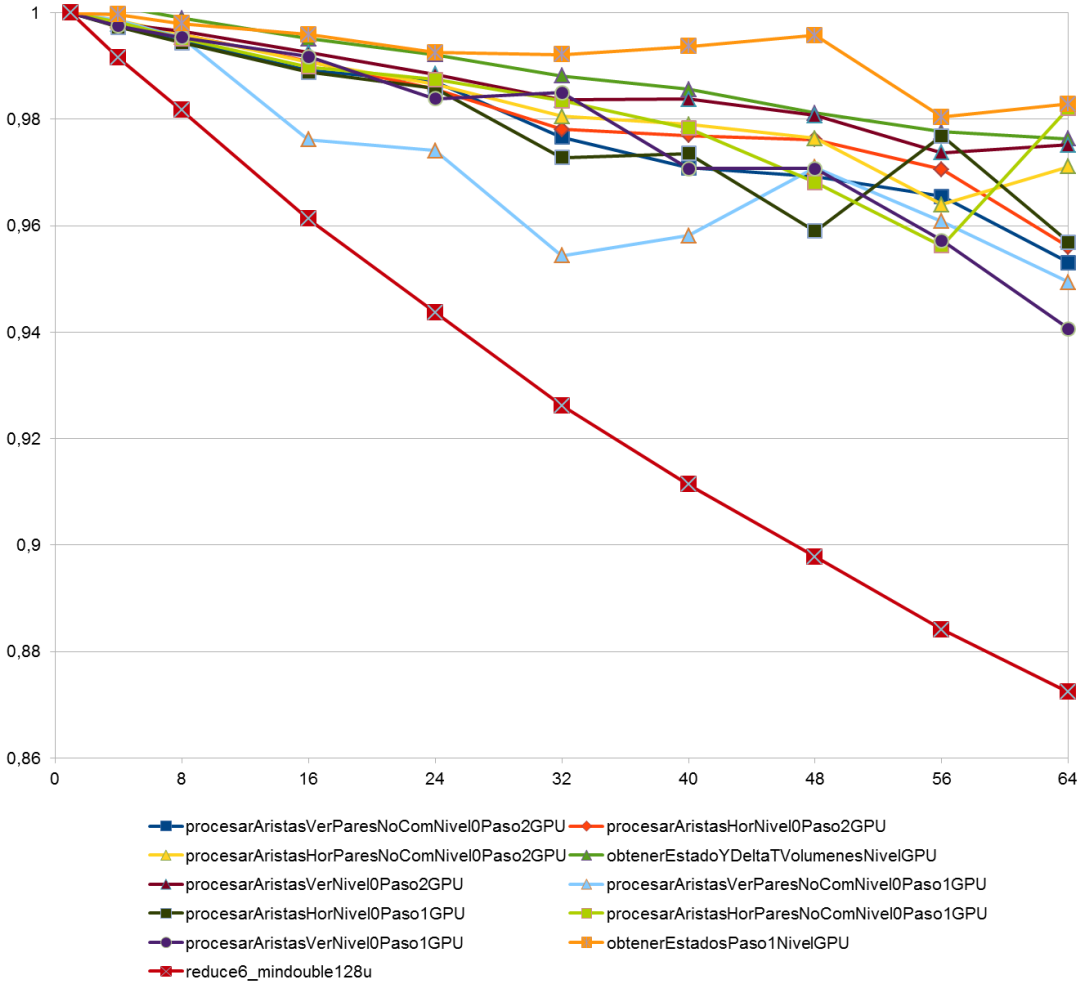


When scaling the biggest increases are related with cudaMemcpy and MPI_Allreduce.

The MPI_Allreduce has a better balance with 64 MPI ranks → increase due to date transfer

CUDA Kernels – scaling efficiency

- Top 11 kernels (>=4% execution time with 64 GPUs)

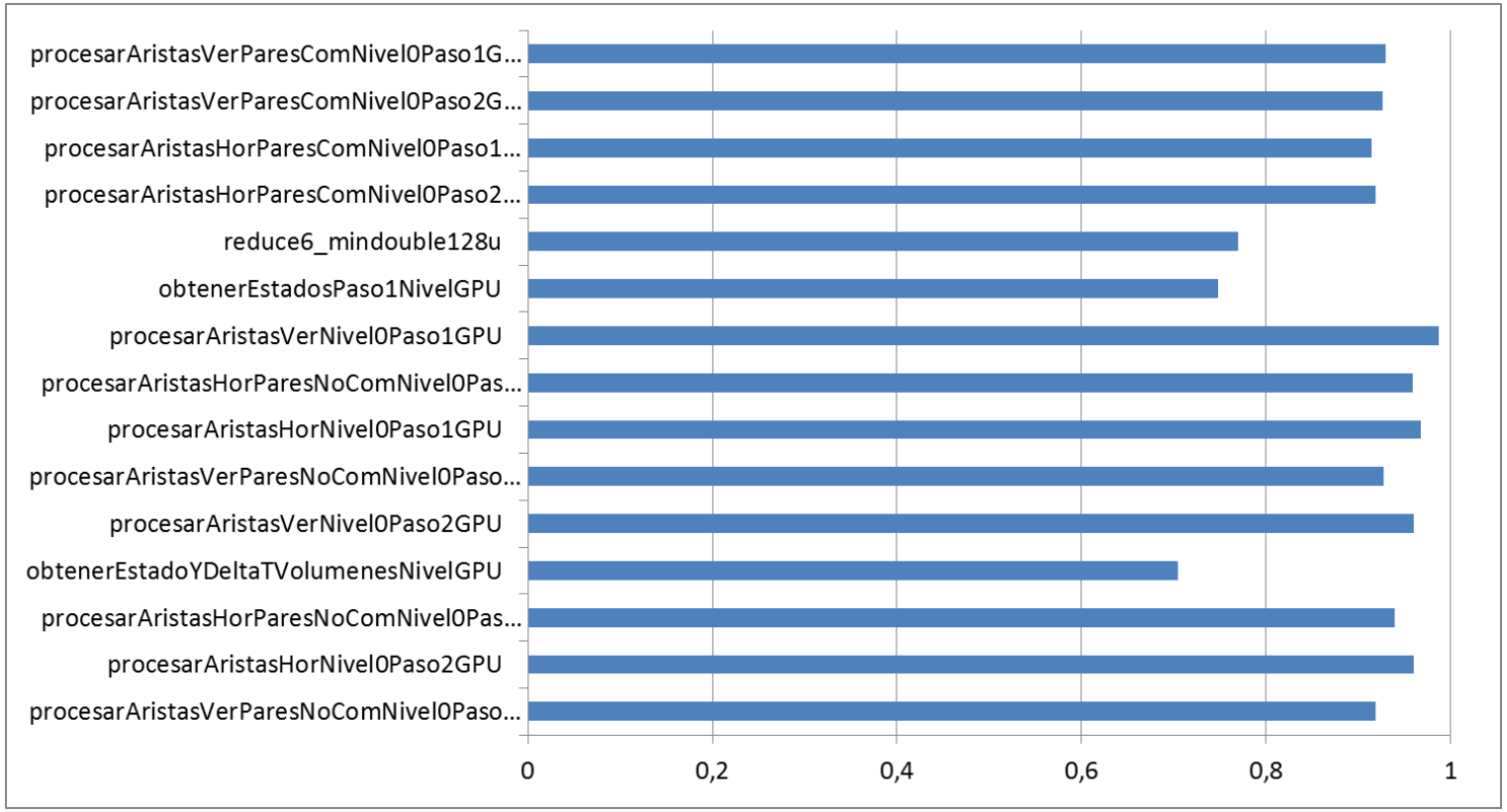


Eliminating the 4 kernels mentioned in the previous slide we can see that 10 kernels have a very good scaling efficiency between 0.94 and 0.98

Only the last kernel in the list that represents a 4.89% has a bigger degradation but still is 0.87 with 64 GPUs.

CUDA Kernels – load balance

- 64 GPUs



A load balance bigger than 0.8 can be considered good. There are 3 kernels with a load balance lower than 0.8.

The lowest load balance is 0.7 for *obtenerEstadoYDeltaTVolumenesNivelGPU* that represents 9% of the execution time. The other 2 kernels with low value represent less than 5% each.

Performance Optimisation and Productivity 3

A Centre of Excellence in HPC

Contact:

- https://www.pop-coe.eu
- pop@bsc.es
- @POP_HPC
- 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.