SedFoam Analysis

(POP2_AR_082)

Kingshuk Haldar, HLRS

EU H2020 Centre of Excellence (CoE) 1 December 2018 – 30 November 2021

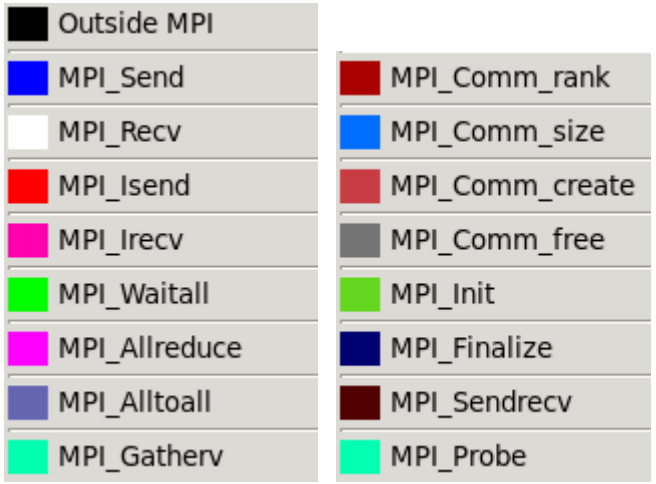
Grant Agreement No 824080

Background

- Name of the code: SedFoam
- Scientific area: Engineering
- Contact: Cyrille Bonamy <cyrille.bonamy@univ-grenoble-alpes.fr>
- Programming language and model: C++ and MPI
- Platform: Occigen cluster (Brodwell/ Haswell)
- POP metrics

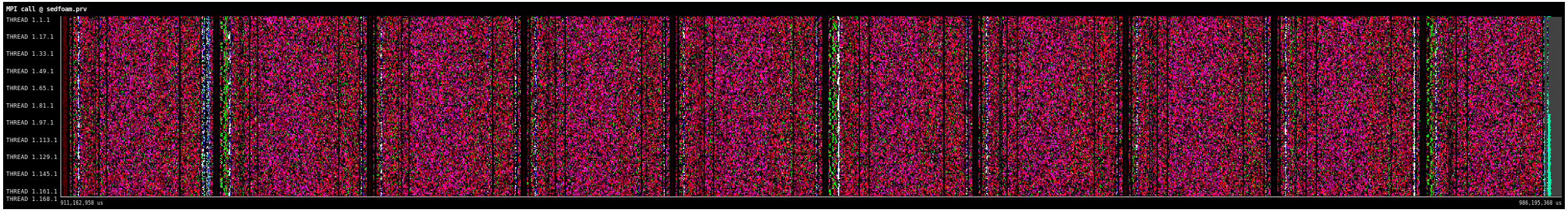
Application Structure

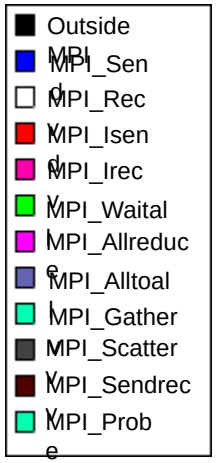
- Timeline view – full



Application Structure

- Timeline view – “Region 3”

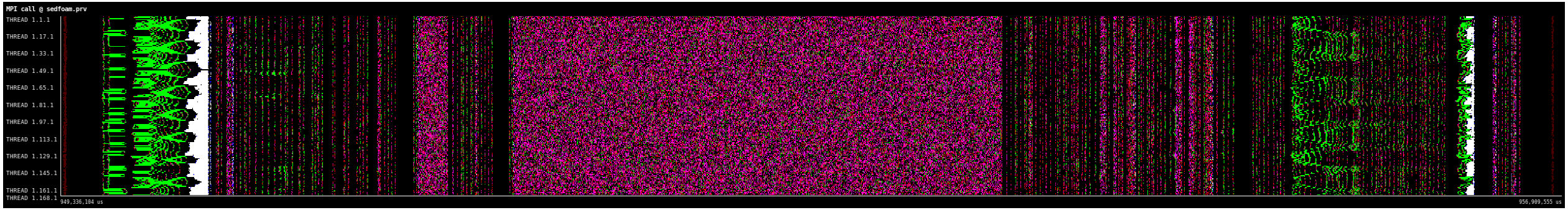


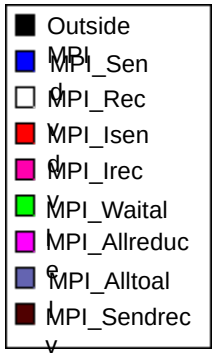


- 10 distinct similar pattern can be seen which are the calculation iterations

Application Structure

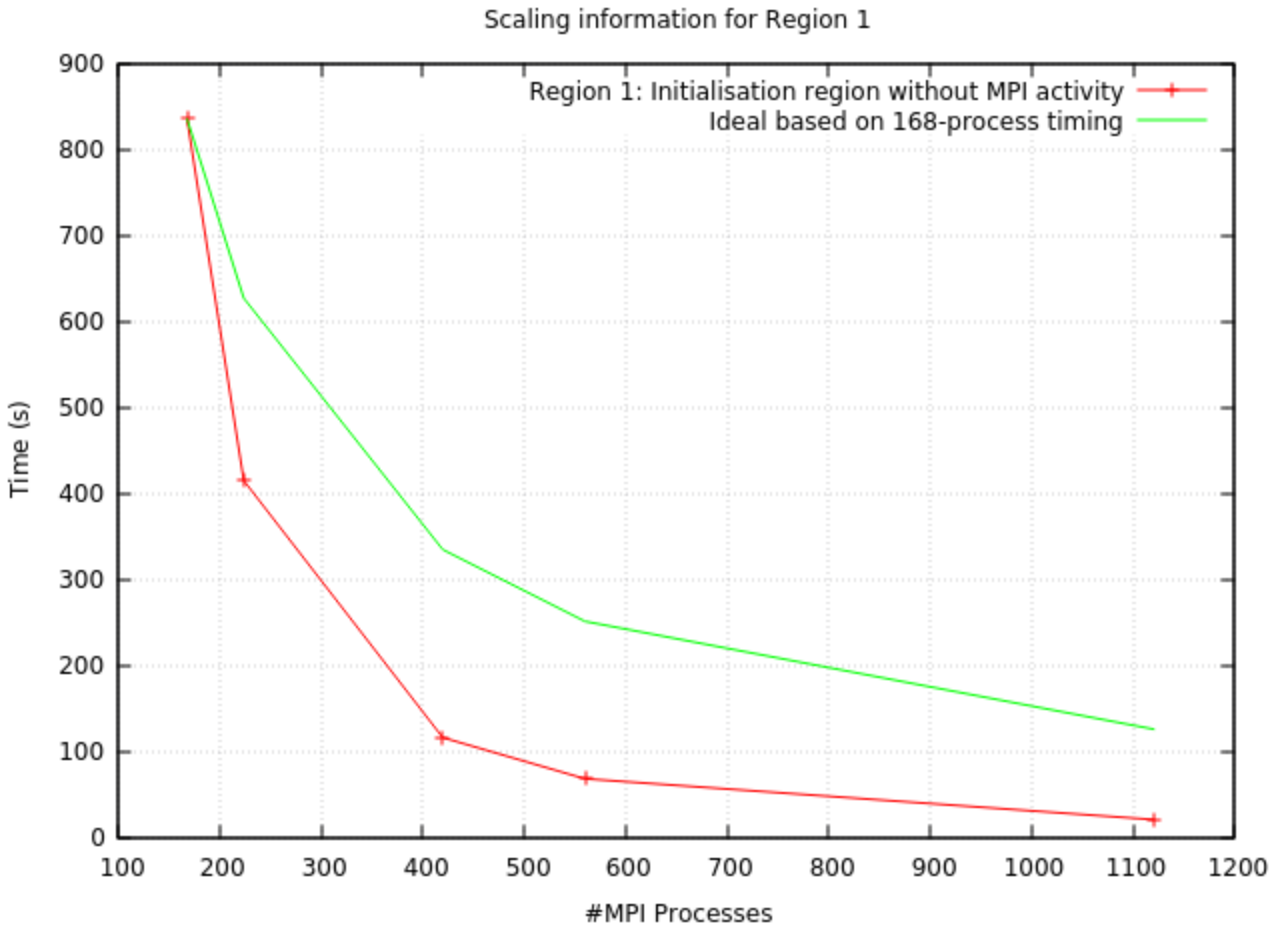
- Timeline view – Zoomed in to 1 iteration





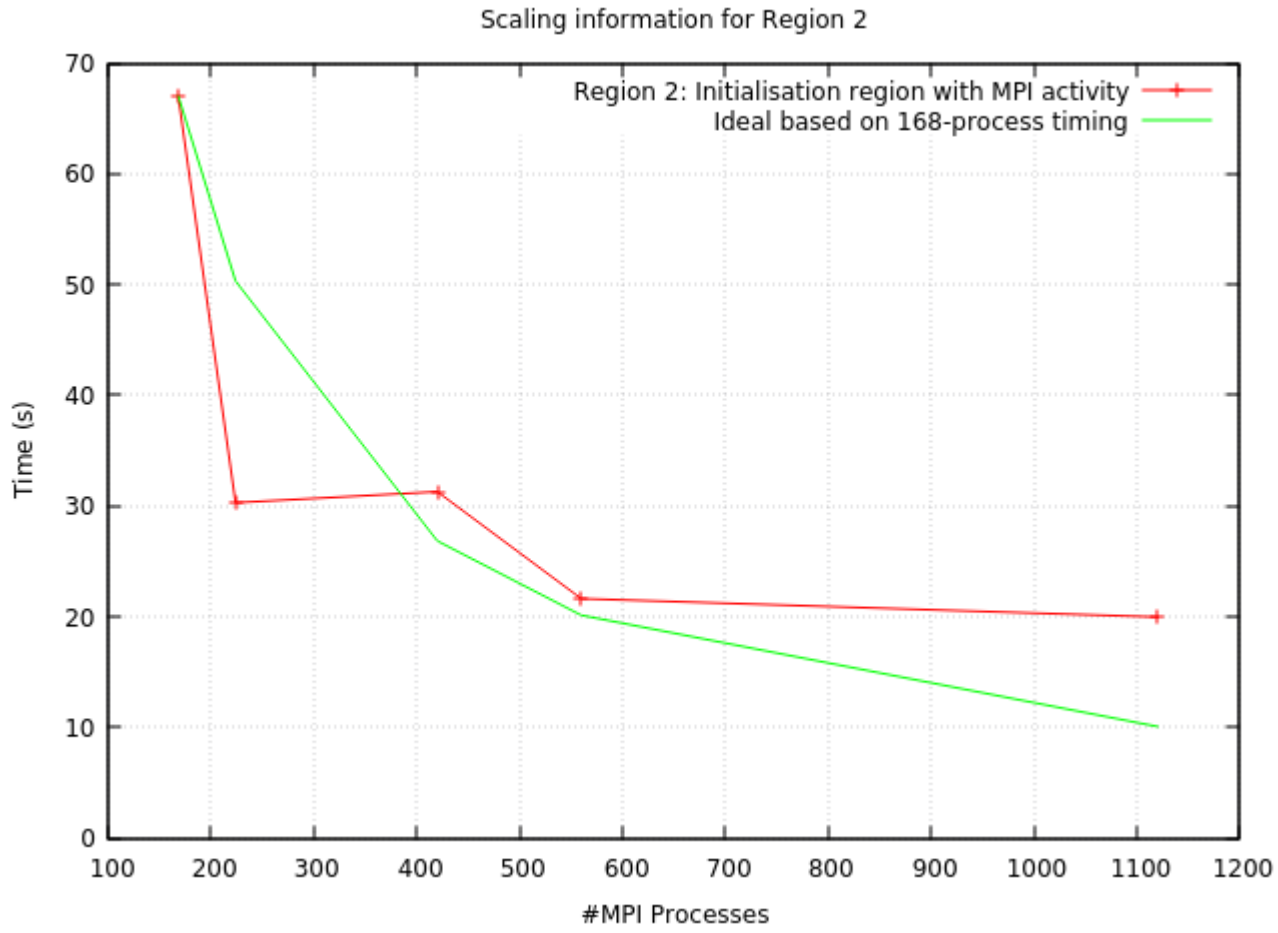
Scalability

- Scaling data for different regions (Region 1)



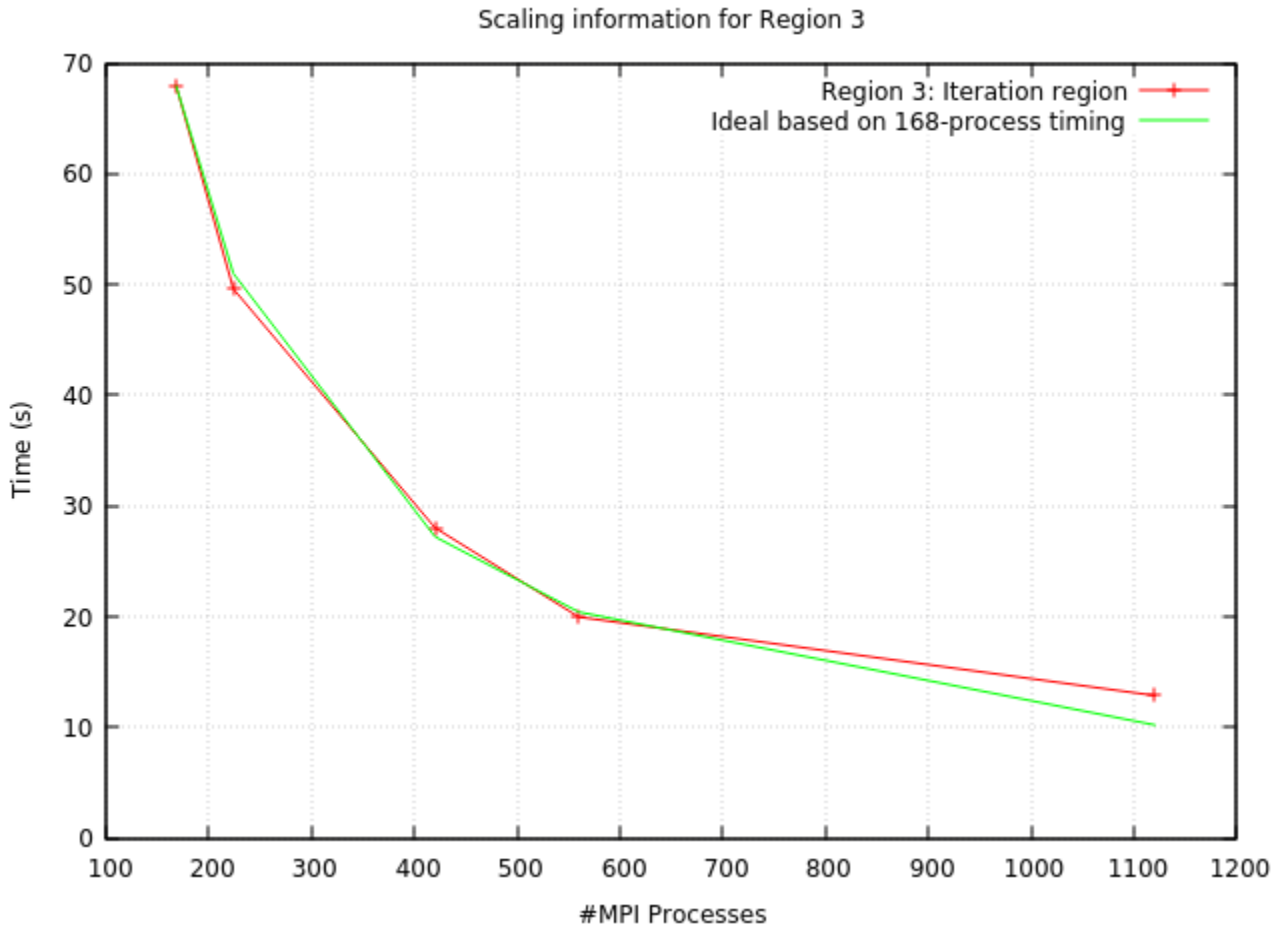
Scalability

- Scaling data for different regions (Region 2)



Scalability

- Scaling data for different regions (Region 3)



Scalability – Key Points

- Scaling of iterations region closely follows ideal-time till 560 processes.
- For 1120 processes, it becomes less than ideal.
- If production runs consist of less iteration, “Region-1” dominates indicating an opportunity to optimise serial performance for that region.

POP Metrics

- Analysis is not done on whole timeline.
- A “Focus of Analysis” (FoA) is chosen from the timeline, such that
  - It consists of one or many calculation iteration which occurs many times in a run.
  - Does not include initialisation and finalisation steps which occur only once in a run.

POP Metrics

- 9 iterations were chosen initially from the timeline.
- Due to large size of trace-data, isolating that region was a problem and later a single iteration were chosen.

Time information

- For FoA

	168	224	420	560	1120
Runtime (s)	7.665	5.393	2.815	2.066	1.541
Speedup	1.00	1.42	2.72	3.71	4.97
Simulated ideal runtime (s)	7.315	5.064	2.482	1.716	1.337
Average non-MPI duration (s)	6.539	4.490	2.034	1.419	0.6707
Maximum non-MPI duration (s)	6.982	4.851	2.277	1.601	0.7707

- Ideal runtime: predicted simulation runtime for and ideal network (unlimited bandwidth, 0 latency)

Time information

- Speedup is superlinear till 560 process (224: 107%, 420: 109%, 560: 111%, 1120: 75%).
- Investigate H/W counters to determine superlinear scaling.

Hardware information

- For FoA

	168	224	420	560	1120
Useful instructions (total, x1e12)	2.590	2.545	2.640	2.709	2.892
Instruction scalability	100%	102%	98.1%	95.6%	89.6%
Useful cycles (total, x1e12)	2.819	2.565	2.184	2.033	1.912
Average IPC	0.92	0.99	1.2	1.3	1.5
IPC Scalability	100%	108%	132%	145%	165%
Average Frequency (GHz)	2.57	2.55	2.56	2.56	2.55
Frequency scalability	100%	99.4%	99.6%	99.7%	99.2%

Hardware information

- Superlinear IPC scaling found. IPC still seems to be increasing at 1120 processes.
- This indicates fitting the problem in cache nicely with increasing processes as amount of calculation per process decreases.
- More experiments around 1120 processes is suggested to determine peak IPC.
- Frequency remains constant as expected.
- Amount of instructions scales sublinearly as more instructions are needed to prepare for communication.

POP Metrics

- FoA: POP parallel metrics

	168	224	420	560	1120
Parallel efficiency	85.3%	83.3%	72.3%	68.7%	43.5%
Load balance	93.7%	92.6%	89.3%	88.6%	87.0%
Communication efficiency	91.1%	89.9%	80.9%	77.5%	50.0%
Serialization efficiency	95.4%	95.8%	91.7%	93.3%	84.6%
Transfer efficiency	95.4%	93.9%	88.2%	83.1%	59.1%
Computation scalability	100%	109%	129%	138%	146%
Global efficiency	85.3%	90.9%	92.9%	95%	63.6%

POP Metrics – Key Points

- POP parallel metrics indicate serious problem with communication efficiency.
- Further analysis shows actual transfer contributes to this more than serialisation effect (waiting for others).
- Investigate communication further to determine reason.
- Load-balance is just acceptable and would be another place to look at afterwards.

Communications

- FoA: Average percent of time spent in different MPI calls

	Outside MPI	MPI_Send	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Allreduce	MPI_Alltoll	MPI_Sendrecv
168	85.31	0.01	1.11	0.65	0.52	8.77	3.44	0.11	0.09
224	83.26	0.01	1.12	0.91	0.69	9.39	4.09	0.06	0.47
420	72.26	0.02	2.85	1.43	1.18	13.35	8.69	0.11	0.11
560	68.70	0.03	3.65	1.83	1.49	16.52	7.24	0.22	0.34
1120	43.55	0.04	4.45	2.15	1.79	8.74	38.71	0.21	0.36

Communications

- FoA: Average number of calls of different MPI routines

	Outside MPI	MPI_Send	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Allreduce	MPI_Alltoll	MPI_Sendrecv
168	10238	89	891.11	3688	3688	849	1820	3	10
224	9667	90	90	3486	3486	807	1694	3	10
420	9440	90	90	3377	3377	805	1688	3	10
560	9574	90	90	3416	3416	819	1730	3	10
1120	9578	90	90	3407	3407	824	1745	3	10

Communications

- FoA: Percent of time spent for different message sizes

MPI_Allreduce	8 byte	16 byte
168	96.13	3.87
224	92.01	7.99
420	96.02	3.98
560	94.49	5.51
1120	98.24	1.76

MPI_Send	1 byte	4 byte	8 byte	24 byte
168	9.83	7.09	76.39	10.23
224	9.91	7.01	75.78	10.80
420	9.71	7.18	76.51	10.20
560	8.45	7.06	77.73	10.29
1120	8.93	7.17	77.22	10.26

Communications

- Significant time is spent on messages of size 8 bytes or less.
- Non-insignificant time is also spent on messages of 1 byte.
- MPI_Waitall() and MPI_Allreduce() both take very different amount of times across MPI processes. This indicates unpredictable collective/ p2p performance. This also results in serialisation.
- Interestingly, number of those calls per processdo not increase.
- This makes latency of the network the main bottleneck and explains the bad transfer efficiency.

I/O

- I/O is not performed during calculation iteration and is not analysed.

Summary

- Physical transfer of MPI messages is the main bottleneck of the program.
- Suggested packing of messages to reduce number of messages and increase its sizes if possible.
- Suggested experiments to determine strong scaling limit.
- Afterwards, dealing with load-balance is also recommended.

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.