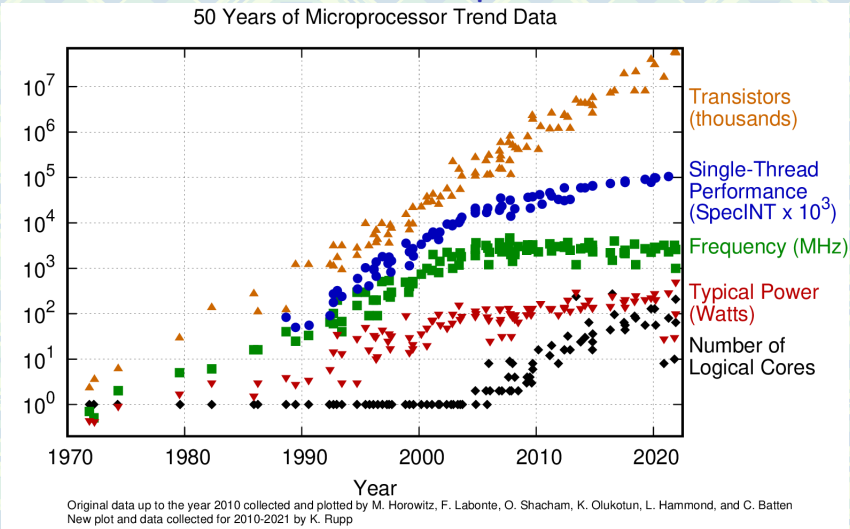
Construction of Locality–Aware Algorithms to Optimize Performance of Stencil Codes on Heterogeneous Hardware

Vadim Levchenko and Anastasia Perepelkina
Keldysh Institute of Applied Mathematics, Moscow, Russia

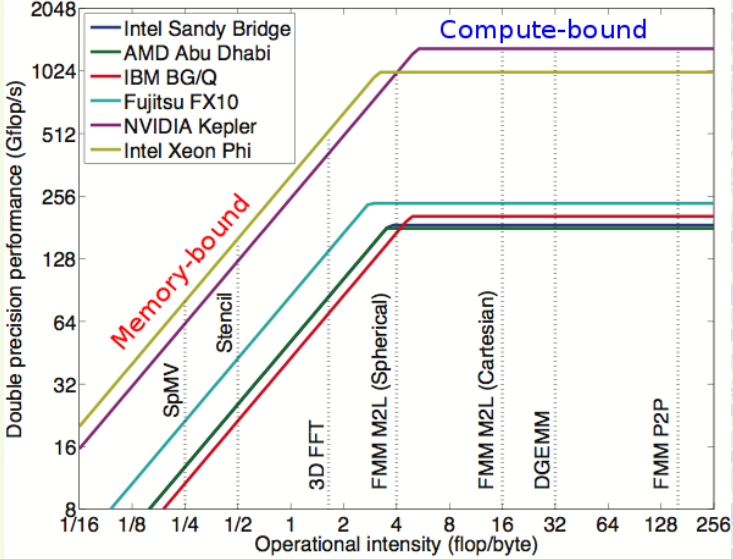
Russian Supercomputing Days 2023
September 25 — 26 :: Moscow

Key HPC problems: Parallelization as the source of performance



Perepelkina, A., Levchenko, V. (2020). *Synchronous and Asynchronous Parallelism in the LRnLA Algorithms*. In: Sokolinsky, L., Zymbler, M. (eds) Parallel Computational Technologies. PCT 2020. Communications in Computer and Information Science, vol 1263. Springer, Cham. DOI

Key HPC problems: Machine balance and memory wall problem



Application performance limitation

$$\Pi = \frac{\mathcal{O}(\text{Operations completed [FLOP]})}{T(\text{time[seconds]})}$$

$$\Pi \leq \min\left[\Pi_{\text{CPU}}, \Theta_{\text{RAM}}\mathcal{I}\right]$$

$$\mathcal{I} >?< \Pi_{\text{CPU}}/\Theta_{\text{RAM}}$$

Application operational intensity

$$\mathcal{I} = \frac{\mathcal{O}(\text{Operations completed [FLOP]})}{\mathcal{S}(\text{Data exchange with memory [B]})}$$

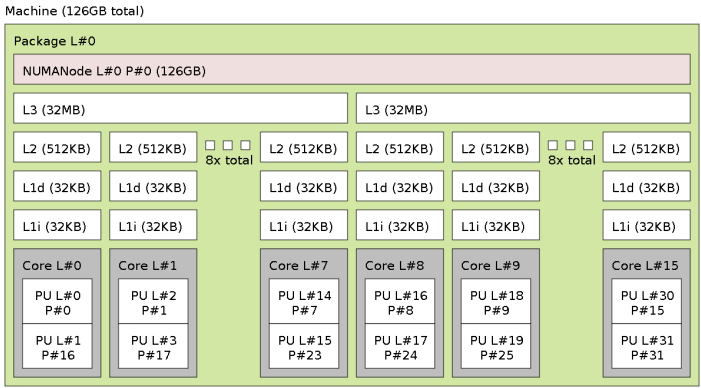
Machine (computer) balance ratio:

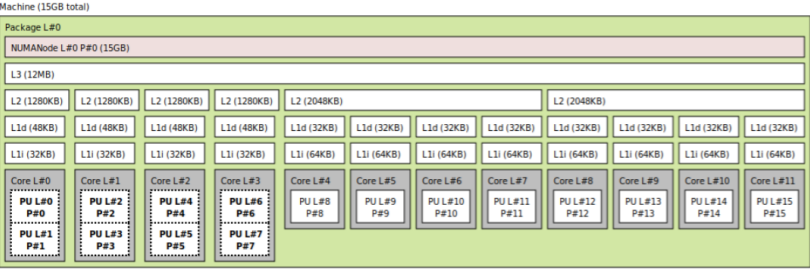
$$\frac{\Pi_{\text{CPU}}(\text{performance [FLOP/sec]})}{\Theta_{\text{RAM}}(\text{memory bandwith (B/sec)})}$$

S. Williams, A. Waterman, and D. Patterson. Roofline: an insightful visual performance model for multicore architectures. Commun. ACM, 52:65–76, 2009.

Key HPC problems: Locality wall

	AMD Ryzen		
	R9 3900	R9 5950X	R7 5800X3D
cores	12	16	8
base freq.	3.1GHz	3.4GHz	3.4GHz
boost freq.	4.3GHz	4.9GHz	4.5GHz
peak F32	1.4Tflops	2Tflops	1Tflops
L1 size	12x32KB	16x32KB	8x32KB
L1 BW	3TB/sec	3.8TB/sec	1.9TB/sec
L2 size	12x512KB	16x512KB	8x512KB
L2 BW	1.5TB/sec	2TB/sec	1TB/sec
L3 size	4x16MB	2x32MB	96MB
L3 BW	1.3TB/sec	1TB/sec	500GB/sec
RAM type		DDR4-3200	
RAM size	64GB	128GB	64GB
RAM BW	50GB/sec	50GB/sec	50GB/sec
flop/byte	28	40	20



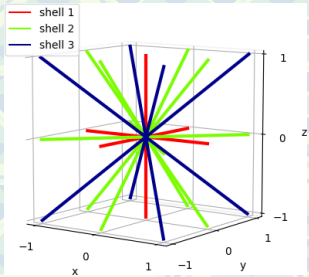


	Intel Core-i	
	i7-7800X	i5-1240P
cores	6	4P+8E
base freq.	3.5GHz	1.7, 1.2GHz
boost freq.	4GHz	4.4, 3.3GHz
peak F32	0.7Tflops	0.8Tflops
L1 size	6x32KB	4x48+8x64KB
L1 BW	2TB/sec	2TB/sec
L2 size	6x1MB	4x1.25MB+2x2MB
L2 BW	750GB/sec	1TB/sec
L3 size	8.25MB	12MB
L3 BW	500GB/sec	350GB/sec
RAM type	DDR4-2400	LPDDR5-4800
RAM size	128GB	16GB
RAM BW	75GB/sec	75GB/sec
flop/byte	9	11

Basics of the Lattice–Boltzmann Method

A set of discrete speeds $\vec{c}_i$, $i = 1, 2, .., Q$
Discrete Distribution Functions f_i

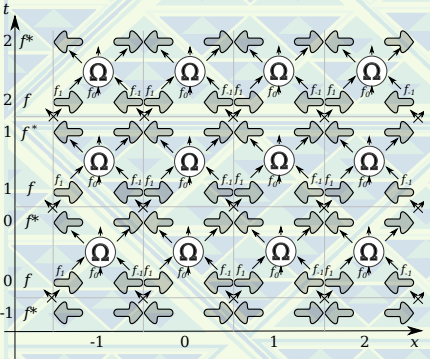
D3Q7-D3Q27
streaming stencil



BGK collision operator Ω: $f_i^*(\vec{x}, t) = \Omega(f_1(\vec{x}, t), f_2(\vec{x}, t), ...f_Q(\vec{x}, t))$.

streaming step: $f_i(\vec{x} + \vec{c}_i, t + 1) = f_i^*(\vec{x}, t); \quad i = 1, ..Q;$

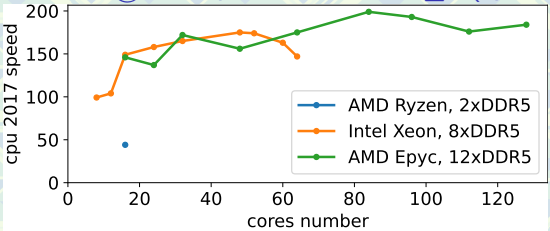
Krüger, T., Kusumaatmaja, H., Kuzmin, A., Shardt, O., Silva, G., Viggen, E.M.
The lattice Boltzmann method. Springer 10(978-3), 4–15 (2017)

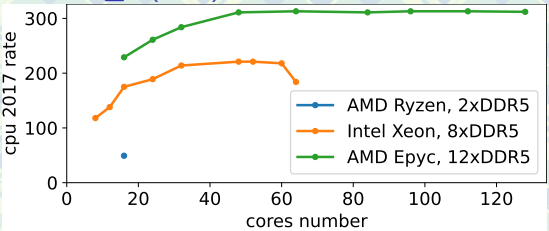


code	Q7	Q13	Q19	Q9	Q15	Q21	Q27
shells	0,1	0,2	0,1,2	0,3	0,1,3	0,2,3	0,1,2,3
flop/Lu	78	150	198	126	174	246	294
$\mathcal{O}$, 2fma/Lu	14	246	330	206	290	422	506
flop/Pu	11	12	10	14	12	12	11
2fma/Pu	16	19	17	23	19	20	19
$\mathcal{I}_{SW}$, F32 flop/byte	1.4	1.4	1.3	1.8	1.5	1.5	1.4

Current state of the locality wall problem for LBM on multicore CPU, examples

SPEC CPU®2017fp base: 619.lbm_s (speed), 519.lbm_r (rate)

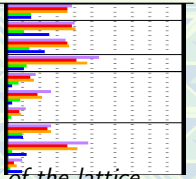




Modern optimized LBM implementation on OpenCL, FluidX3D

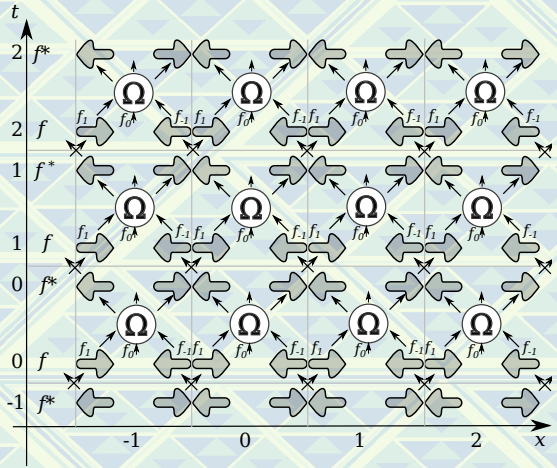
Device	Data sheet: FP64 TFLOPs/s	Data sheet: FP32 TFLOPs/s	Data sheet: memory GB	Data sheet: bandwidth GB/s	Measured memory bandwidth GB/s: coalesced read	coalesced write	misaligned read	misaligned write	LBM performance (D3Q19 SRT), MLUPs/s: FP64/FP64	FP64/FP32	FP32/FP32	FP32/FP16S	FP32/FP16C
Intel Core i9-10980XE	1.81	3.23	128	94	60	26	65	30	110	192	193	152	152
Intel Core i5-9600	0.30	0.60	16	43	35	14	39	16	52	94	99	51	113
Intel Core i7-8700K	0.36	0.71	16	51	38	26	51	18	54	105	108	77	122
Intel Xeon Phi 7210	2.66	5.32	192	102	35	62	65	61	167	241	275	136	124
4x Intel Xeon E5-4620 v4	1.34	2.69	512	273	104	52	126	54	151	239	266	241	139
2x Intel Xeon E5-2630 v4	0.70	1.41	64	137	65	66	88	86	111	152	169	133	74
2x Intel Xeon E5-2623 v4	0.33	0.67	64	137	25	24	43	26	52	69	77	61	34
2x Intel Xeon E5-2680 v3	0.96	1.92	64	137	110	43	102	50	111	194	211	146	156
Intel Core i7-4770	0.22	0.44	16	26	24	15	26	12	37	53	61	13	34
Intel Core i7-4720HQ	0.17	0.33	16	26	22	9	22	11	8	9	11	11	28

memory efficiency

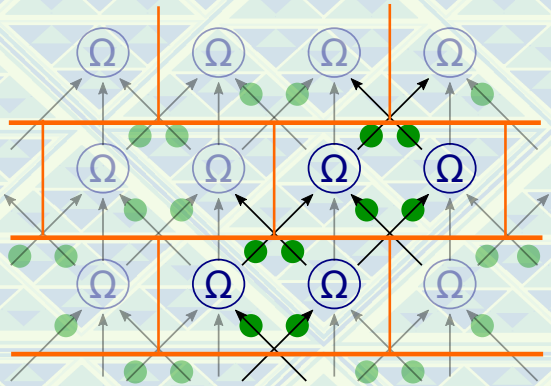


M.Lehmann, M.J.Krause, G.Amati, M.Sega, J.Harting, and S.Gekle, *Accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit, and customized 16-bit number formats*. Phys. Rev. E 106, 015308 (2022), DOI

Compact streaming update for LBM



compact cell's data size: 2..7 KB (D3Q7..27)

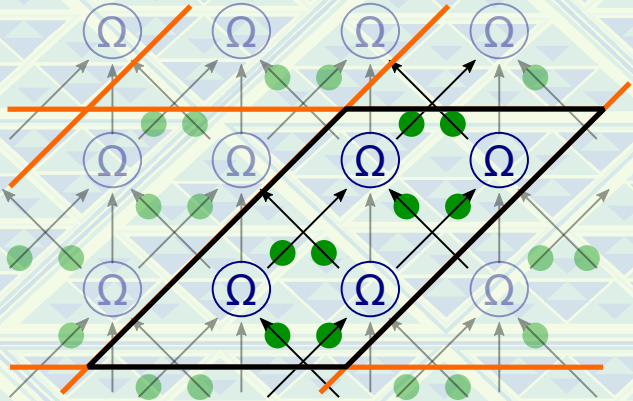


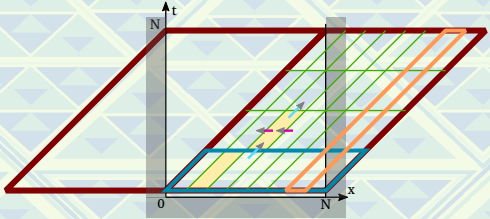
$\mathcal{I}_{\mathrm{SW}} = \mathcal{O}/2\mathcal{S}$

A.Perepelkina, V.Levchenko. *LRnLA Algorithm ConeFold with Non-local Vectorization for LBM Implementation*, (2019). In: V.Voevodin, S.Sobolev (eds) Supercomputing. RuSCDays 2018. CCIS, vol 965. Springer, Cham. DOI

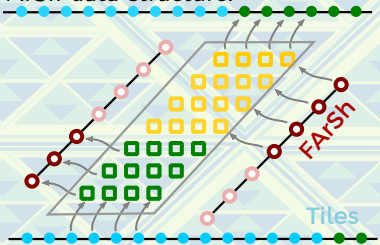
A.Perepelkina, V.Levchenko, A.Zakirov. *New Compact Streaming in LBM with ConeFold LRnLA Algorithms*, (2020). In: V.Voevodin, S.Sobolev (eds) Supercomputing. RuSCDays 2020. CCIS, vol 1331. Springer, Cham. DOI

Compact LRnLA cells and their composition to ConeTorre LRnLA algorithm





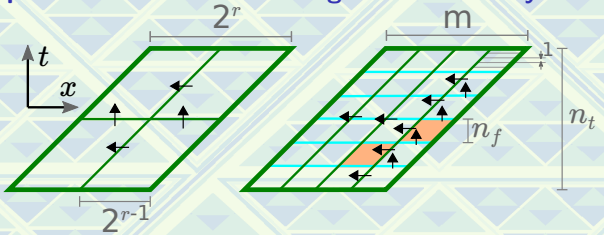
FArSh data structure:



A.Zakirov, A.Perepelkina, V.Levchenko, S.Khilkov, *Streaming techniques: revealing the natural concurrency of the lattice Boltzmann method*. J Supercomput 77, 11911–11929 (2021). DOI

Perepelkina, A., Levchenko, V.D. (2021). *Functionally Arranged Data for Algorithms with Space-Time Wavefront*. In: Sokolinsky, L., Zymbler, M. (eds), PCT 2021. CCIS, vol 1437. Springer, Cham. DOI

Optimal ConeTorre-based algorithm for many-core CPU



$$\mathcal{I}_{\mathsf{CT}} \equiv \frac{\mathcal{O}_{\mathsf{CT}}}{2\mathcal{S}_{\mathsf{CT}}} = \frac{2n_t(2m)^d\mathcal{O}}{2\left((2m)^d + 2n_t\Gamma^-_{(2m)^d}\right)\mathcal{S}}.$$

$$\Gamma^-_{(2m)^d} = (2m)^d - (2m-1)^d$$

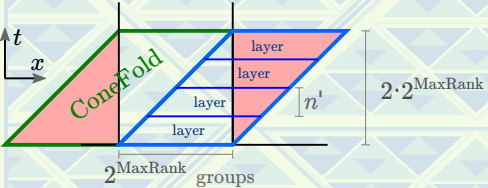
CT form	conditions	$\mathcal{I}/\mathcal{I}_{\mathsf{SW}}$	localization
low and wide layer	$dn_t \ll m$	$2n_t$	DRAM
ConeFold	$m = n_t \equiv n_{\mathsf{opt}}$	$2n_{\mathsf{opt}}/(d+1)$	on-chip casche
narrow and long	$n_t \gg m \gg 1$	$2m/d$	L1 (base) + L2 (FArSh)
1-group base	$m = 1$	$2^d/(2^d-1)$	

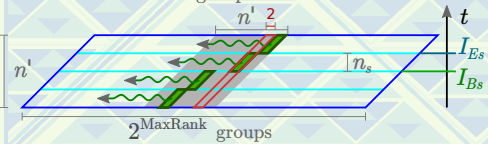
$$n_{\mathsf{opt}}: \quad ((2n_{\mathsf{opt}})^3 + 2n_{\mathsf{opt}}\Gamma^-_{(2n_{\mathsf{opt}})^3})\mathcal{S} \le \mathcal{M}_{\mathsf{L2+L3}} \approx 70\mathsf{MB}$$

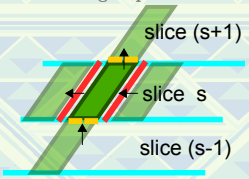
$$n_{\mathsf{opt}} \approx 11 \div 16 \to \frac{2n}{d+1} \approx 5 \div 8 \to \mathcal{I} \approx (5 \div 8)\cdot\mathcal{I}_{\mathsf{SW}}$$

Levchenko, V., Perepelkina, A. (2023). *An Efficient LRnLA Algorithm and Data Structure for Manycore and Multicore Computers with Hierarchical Cache*. In: Sokolinsky, L., Zymbler, M. (eds) Parallel Computational Technologies. PCT 2023. Communications in Computer and Information Science, vol 1868. Springer, Cham. DOI

FArShFold Properties: Localization Limits







(1) $\Pi_{\text{layer}}(n_{\text{opt}}) \le \min\left[\Pi_{\text{CPU}}, \Theta_{\text{RAM}}\mathcal{I}_{\text{layer}}\right]$

$\approx \min\left[\Pi_{\text{CPU}}, \Theta_{\text{RAM}} 2 n_{\text{opt}}\mathcal{I}_{\text{SW}}\right] = \Pi_{\text{CPU}},$

(7) $\Pi_{\text{CF},n'} \le \min\left[\Pi_{\text{layer}}, \Theta_{\text{RAM}}\dfrac{n'}{d+1}\mathcal{I}_{\text{SW}}\right],$

(8) $\Pi_{\text{FArShFold}} \le \min\left[\Pi_{\text{CF},n_{\text{opt}}}, \Theta_{\text{L3}}\dfrac{n_{\text{opt}}}{d+N_C/2}\mathcal{I}_{\text{SW}}\right],$

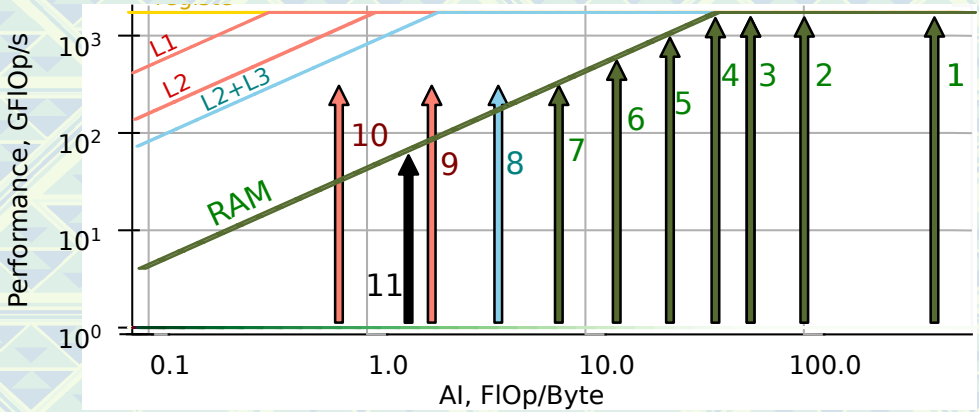
$\Pi_{\text{CT},1} \le \min\Big[\Pi_{\text{FArShFold}},$

(9–10) $\min\left(\Theta_{\text{L3}}\dfrac{n_{\text{L2}}}{d}, \Theta_{\text{L2}}\dfrac{n_{\text{L1}}}{d}, \Theta_{\text{L1}}\dfrac{2^d}{2^d-1}\right)\mathcal{I}_{\text{SW}}\Big].$

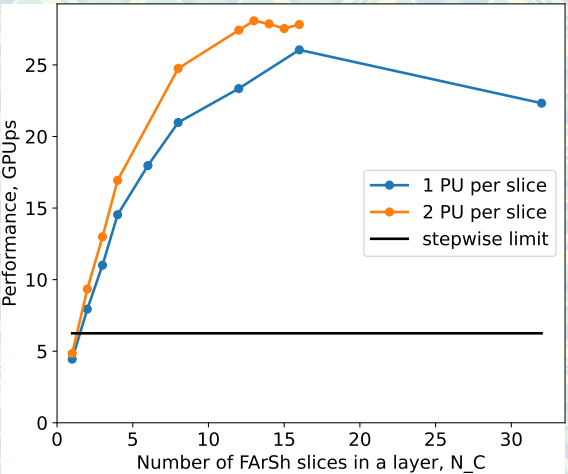
$2n_s\Gamma_{n_{\text{L2}}} \le \mathcal{M}_{\text{L2}}/\mathcal{S}, \qquad 2n_s\Gamma_{n_{\text{L1}}} \le \mathcal{M}_{\text{L1}}/\mathcal{S}$

(11) $\Pi_{\text{SW}} \le \min\left[\Pi_{\text{CPU}}, \Theta_{\text{RAM}}\mathcal{I}_{\text{SW}}\right]$

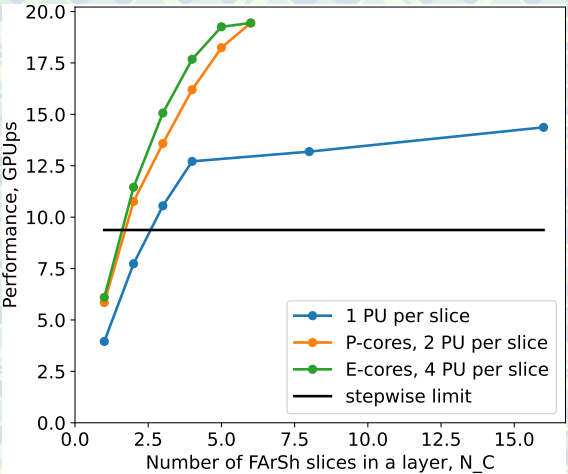
RoofLine model for the FArShFold algorithm for AMD Ryzen R9 5950



Performance benchmark for one full ConeFold update



AMD Ryzen R9 5950 (Zen3)
achieved performance 1.48 GLups



Intel Core-i i5-1240P(Adl)
achieved performance 1.02 GLups

Conclusion

- For the lattice Boltzmann method, we have constructed the FArShFold algorithm;
- GPU-level performance is obtained on CPU;
- FArSHFold is a temporal blocking solution from the family of LRnLA algorithms and uses a special FArSh data array for data exchange;
- With FArSh, data are distributed between cores and localized in L2 cache;
- We recommend to use the algorithm for memory-bound stencil codes implementations on many-core CPU.