Enabling lighter and faster simulations with repeated matrix blocks

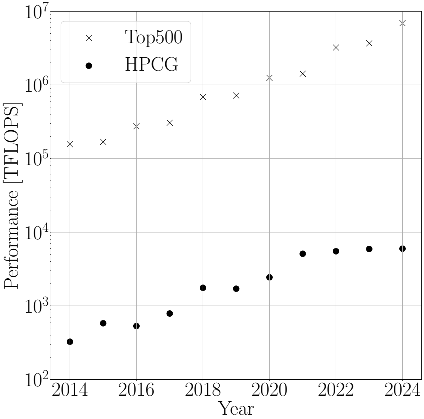
J. Plana-Riu[1], F.X. Trias[1], G. Colomer[1], À. Alsalti-Baldellou[2], X. Álvarez-Farré[3], A. Oliva[1]

[1]Heat and Mass Transfer Technological Centre
Technical University of Catalonia
[2]Department of Civil, Environmental and Architectural Engineering (ICEA)
University of Padova
[3] High-Performance Computing and Visualization Team
SURF

35th Parallel Computational Fluid Dynamics (ParCFD) International Conference
Bonn, Germany
Sept. 3rd, 2024

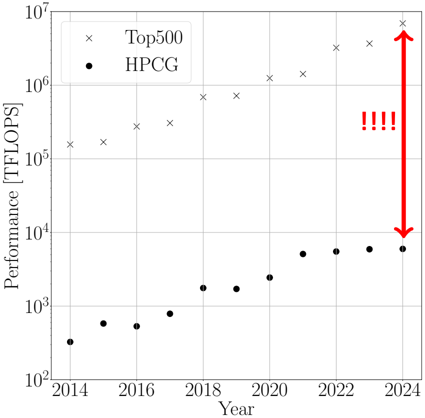
Introduction

One big problem...



Introduction

One big problem...

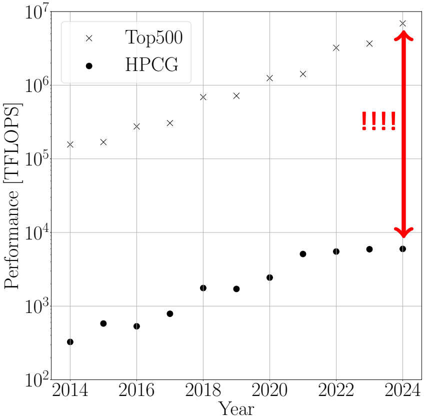


Existing resources...

- Computational power from current top HPC systems is in the exaflop range...
- Sparse algebra, however...
  - has a low arithmetic intensity
  - is limited by memory bandwidth
- HPCG is the benchmark for us.

Introduction

One big problem...



Existing resources...

- Computational power from current top HPC systems is in the exaflop range...
- Sparse algebra, however...
  - has a low arithmetic intensity
  - is limited by memory bandwidth
- HPCG is the benchmark for us.

Possible solutions...

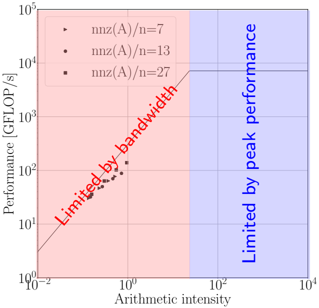
- Improving arithmetic intensity!

Introduction

Arithmetic intensity

Roofline model[1]

- Memory-bound
- Compute-bound



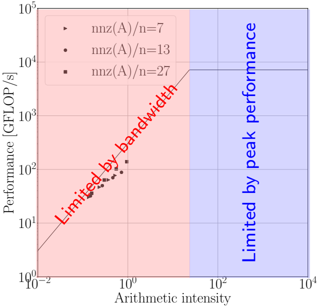
[1] S. Williams et al. "Roofline: an insightful visual performance for multicore architectures," *Commun. ACM* **52**, 2009

Introduction

Arithmetic intensity

Roofline model[1]

- Memory-bound
- Compute-bound



What is the arithmetic (or operational) intensity?

- Ratio between the number of operations and the amount of data that has to be handled (sent/received)

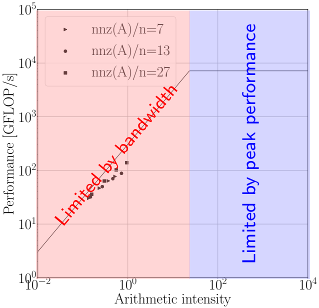
[1] S. Williams et al. "Roofline: an insightful visual performance for multicore architectures," *Commun. ACM* **52**, 2009

Introduction

Arithmetic intensity

Roofline model[1]

- Memory-bound
- Compute-bound



What is the arithmetic (or operational) intensity?

- Ratio between the number of operations and the amount of data that has to be handled (sent/received)

$$\begin{pmatrix} A & \\ & A \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \qquad A(u_1 \quad u_2)$$

2 SpMV 1 SpMM with 2 RHS

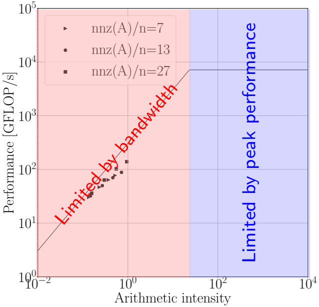
[1] S. Williams et al. "Roofline: an insightful visual performance for multicore architectures," *Commun. ACM* **52**, 2009

Introduction

Arithmetic intensity

Roofline model[1]

- Memory-bound
- Compute-bound



What is the arithmetic (or operational) intensity?

- Ratio between the number of operations and the amount of data that has to be handled (sent/received)

Oper.	# Mat.	# Vec. sent	# Vec. recv	# Ops
2x SpMV	2	2	2	2
1x 2-SpMM	1	2	2	2

Oper.	Equivalent arithmetic intensity**
2x SpMV	2/(2+2+2)=1/3
1x 2-SpMM	2/(1+2+2)=2/5

[1] S. Williams et al. "Roofline: an insightful visual performance for multicore architectures," *Commun. ACM* **52**, 2009

Repeated matrix blocks

- CFD simulations are full of sparse matrix-vector products (SpMV):
  - $u^{n+1} = u^{*,n+1} - G\psi^{n+1}$
  - $u_i^* = u^n + \Delta t \sum_{j=1}^{i-1} a_{ij}(Du_j - C(u_j)u_j)$
  - $\mathrm{r}_{k+1} = \mathrm{r}_k - \alpha_k A\mathrm{p}_k$
  - ...

Following the previous example...

- If some repeated matrrix block structures are present SpMV can be translated to SpMM:
  - Symmetries
  - Repeated geometry patterns
  - Ensemble averaging parallel-in-time

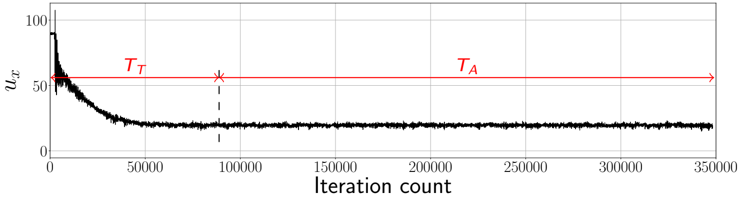
Repeated matrix blocks

- CFD simulations are full of sparse matrix-vector products (SpMV):
  - $u^{n+1} = u^{*,n+1} - G\psi^{n+1}$
  - $u_i^* = u^n + \Delta t \sum_{j=1}^{i-1} a_{ij}(Du_j - C(u_j)u_j)$
  - $\mathsf{r}_{k+1} = \mathsf{r}_k - \alpha_k \mathsf{A}\mathsf{p}_k$
  - ...

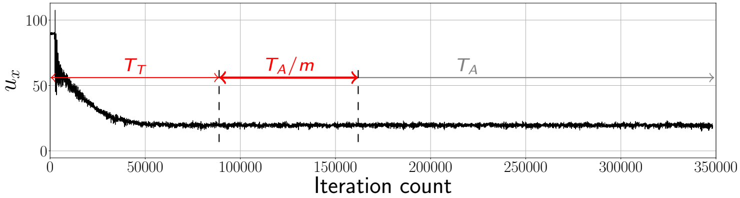
Following the previous example...

- If some repeated matrrix block structures are present SpMV can be translated to SpMM:
  - Symmetries
  - Repeated geometry patterns
  - **Ensemble averaging parallel-in-time**

Ensemble averaging parallel-in-time



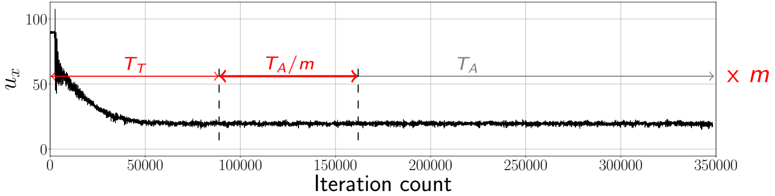
Ensemble averaging parallel-in-time



Ensemble average

$$U_x = \frac{1}{m}\sum_{i=1}^{m} < u_{x,i} > = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{T-T_T}\int_{T_T}^{T} u_{x,i}dt$$

Ensemble averaging parallel-in-time



Ensemble average

$$U_x = \frac{1}{m}\sum_{i=1}^{m} < u_{x,i} > = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{T - T_T}\int_{T_T}^{T} u_{x,i}\,dt$$

m simulations as a single one...

$$\mathbb{C} = I_m \otimes C$$

$$\mathbb{D} = I_m \otimes D$$

$$\mathbb{G} = I_m \otimes G$$

Ensemble averaging parallel-in-time

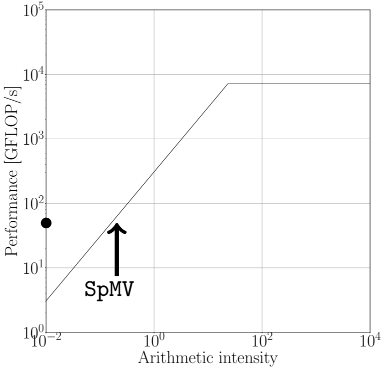
$$\frac{du}{dt} + C(u)u = -Gp + Du \rightarrow \frac{dU}{dt} + \mathbb{C}(U)U = -\mathbb{G}P + \mathbb{D}U$$

- Block structures appear in $\mathbb{C}, \mathbb{G}, \mathbb{D}$
- SpMV's can be translated to SpMM!
  - Increases the arithmetic intensity

Ensemble averaging parallel-in-time

$$\frac{du}{dt} + C(u)u = -Gp + Du \rightarrow \frac{dU}{dt} + \mathbb{C}(U)U = -\mathbb{G}P + \mathbb{D}U$$

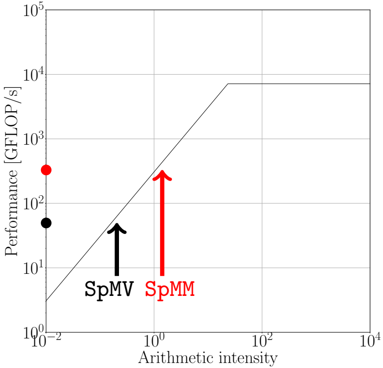
- Block structures appear in $\mathbb{C}, \mathbb{G}, \mathbb{D}$
- SpMV's can be translated to SpMM!
    - Increases the arithmetic intensity



Ensemble averaging parallel-in-time

$$\frac{du}{dt} + C(u)u = -Gp + Du \rightarrow \frac{dU}{dt} + \mathbb{C}(U)U = -\mathbb{G}P + \mathbb{D}U$$

- Block structures appear in $\mathbb{C}, \mathbb{G}, \mathbb{D}$
- SpMV's can be translated to SpMM!
  - Increases the arithmetic intensity
  - Increases performance $\rightarrow$ speed-up!



Ensemble averaging parallel-in-time

Generation of speed-up

- Speed-up in `SpMV` is guaranteed by increasing the AI
- How does this translate to the whole simulation and iteration?
  - T_T will be simulated m times...
  - Speed-up in T_A has to be big enough!

[2] B.I. Krasnopolsky, "An approach for accelerating incompressible turbulent flow simulations based on simultaneous modelling of multiple ensembles," *Comput. Phys. Commun.* **229**, 2018

Ensemble averaging parallel-in-time

Generation of speed-up

- Speed-up in SpMV is guaranteed by increasing the AI
- How does this translate to the whole simulation and iteration?
  - T_T will be simulated m times...
  - Speed-up in T_A has to be big enough!

Times ratio β

$$\beta = \frac{T_A}{T_T}$$

Estimation of simulation speed-up[2]

$$P_m = \frac{1+\beta}{m+\beta}\frac{5m}{5m-3\theta(m-1)}$$

- Speed-up of the iteration, $P_{m,ite}$
- Extension to the whole simulation

[2] B.I. Krasnopolsky, "An approach for accelerating incompressible turbulent flow simulations based on simultaneous modelling of multiple ensembles," *Comput. Phys. Commun.* **229**, 2018

Ensemble averaging parallel-in-time

Generation of speed-up

- Speed-up in SpMV is guaranteed by increasing the AI
- How does this translate to the whole simulation and iteration?
  - T_T will be simulated m times...
  - Speed-up in T_A has to be big enough!

Times ratio β

$$\beta = \frac{T_A}{T_T}$$

Estimation of simulation speed-up[2]

$$P_m = \frac{1+\beta}{m+\beta} \frac{5m}{5m - 3\theta(m-1)}$$

- Speed-up of the iteration, $P_{m,ite}$
- Extension to the whole simulation

[2] B.I. Krasnopolsky, "An approach for accelerating incompressible turbulent flow simulations based on simultaneous modelling of multiple ensembles," *Comput. Phys. Commun.* **229**, 2018

Methodology

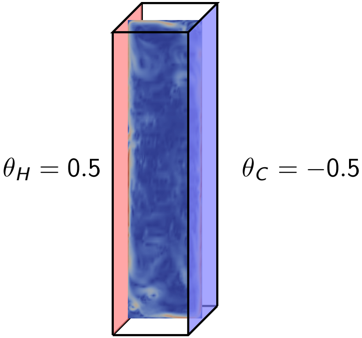
Case definition

Differentially heated cavity of aspect ratio 4

- Semi-discrete governing equations

$$M\boldsymbol{u}_s = 0_c,$$
$$\Omega \frac{d\boldsymbol{u}_c}{dt} + C(\boldsymbol{u}_s)\boldsymbol{u}_c - \frac{\mathrm{Pr}}{\mathrm{Ra}^{1/2}} D\boldsymbol{u}_c + \Omega G_c \boldsymbol{p_c} + \Omega \boldsymbol{f}_c = 0_c,$$
$$\Omega \frac{d\boldsymbol{\theta}_c}{dt} + C(\boldsymbol{u}_s)\boldsymbol{\theta}_c - \frac{1}{\mathrm{Ra}^{1/2}} D\boldsymbol{\theta}_c = 0_c$$

- 3rd-order Heun Runge-Kutta[3], SAT[4]
- Ra=10^{10}, Pr=0.71
- $\boldsymbol{f}_c = (0, \mathrm{Pr}\boldsymbol{\theta}, 0)$



[3] B. Sanderse and B. Koren, "Accuracy analysis of explicit Runge-Kutta methods applied to the incompressible Navier-Stokes equations," *J. Comput. Phys.* **231**, 2012

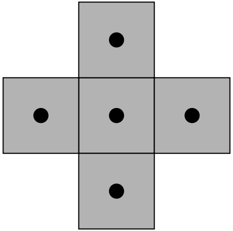
[4] J. Plana-Riu et al, "Cost-vs-accuracy analysis of self-adaptive time-integration methods," *10th THMT*, 2023

Numerical tests in CPU

Construction of test case

Characteristics

- Run in 1 MN5 GPP-HighMem partition node:
  - 2x Intel Xeon Platinum 8480+ 56C 2GHz
  - 1024GB of RAM memory
  - 2x54 OpenMP threads within the socket
  - 2 MPI processes (1x socket)
- Mesh: 220x880x220 (42.6M cells)
  - 400k cells per CPU
- Run in three different discretizations for the Laplacian operator: 7p,13p, and 27p
- Influence of Poisson solver iterations: (150, 350, 550)
- Run for 1, 2, 4, and 8 simultaneous flow states

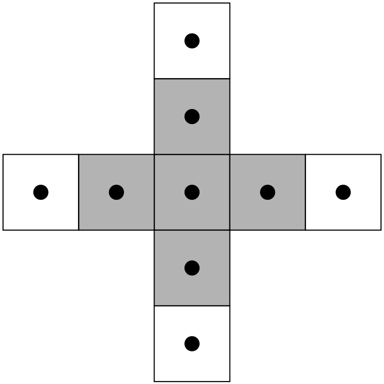


Numerical tests in CPU

Construction of test case

Characteristics

- Run in 1 MN5 GPP-HighMem partition node:
  - 2x Intel Xeon Platinum 8480+ 56C 2GHz
  - 1024GB of RAM memory
  - 2x54 OpenMP threads within the socket
  - 2 MPI processes (1x socket)
- Mesh: 220x880x220 (42.6M cells)
  - 400k cells per CPU
- Run in three different discretizations for the Laplacian operator: 7p,13p, and 27p
- Influence of Poisson solver iterations: (150, 350, 550)
- Run for 1, 2, 4, and 8 simultaneous flow states

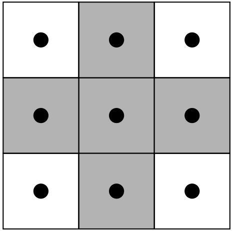


Numerical tests in CPU

Construction of test case

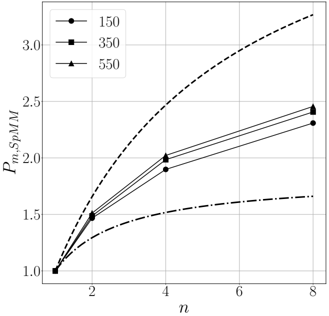
Characteristics

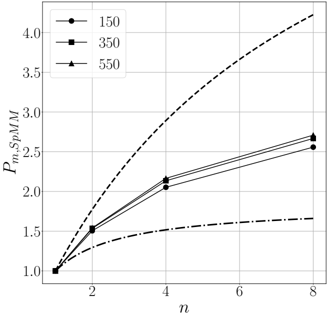
- Run in 1 MN5 GPP-HighMem partition node:
  - 2x Intel Xeon Platinum 8480+ 56C 2GHz
  - 1024GB of RAM memory
  - 2x54 OpenMP threads within the socket
  - 2 MPI processes (1x socket)
- Mesh: 220x880x220 (42.6M cells)
  - 400k cells per CPU
- Run in three different discretizations for the Laplacian operator: 7p,13p, and 27p
- Influence of Poisson solver iterations: (150, 350, 550)
- Run for 1, 2, 4, and 8 simultaneous flow states

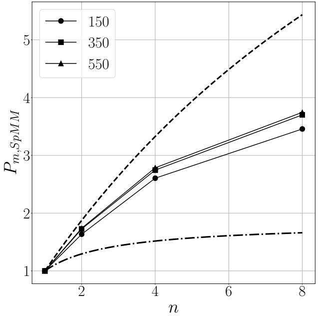


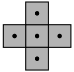
Numerical tests in CPU

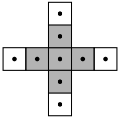
Results. Speed-up for SpMM

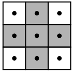






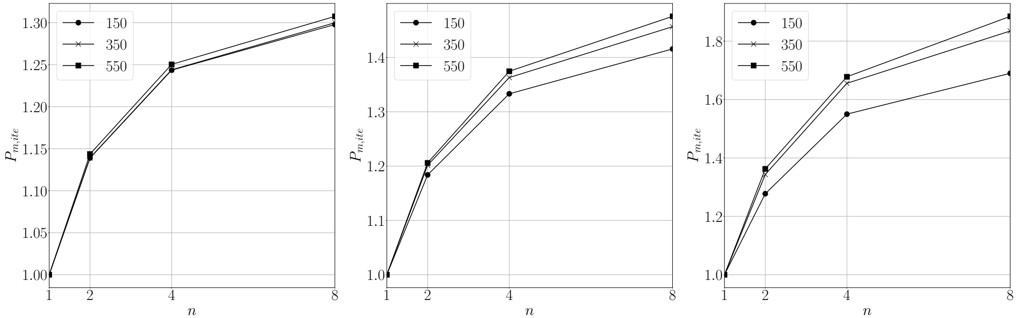


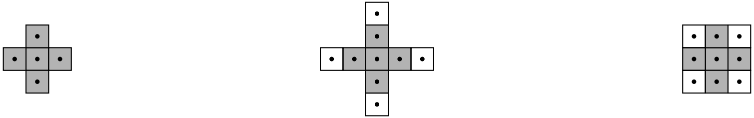




Numerical tests in CPU

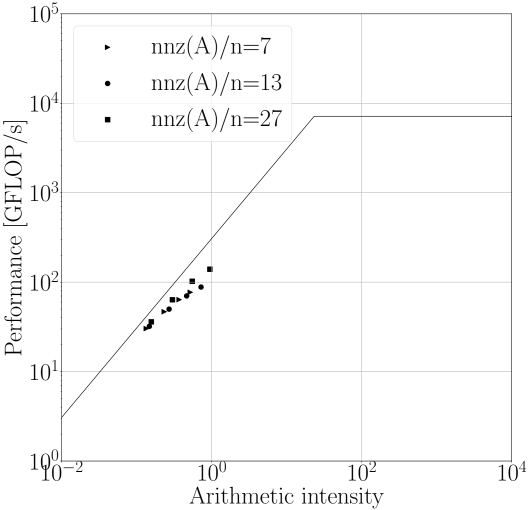
Results. Speed-up for iteration





Numerical tests in CPU

Results. Roofline analysis



System properties

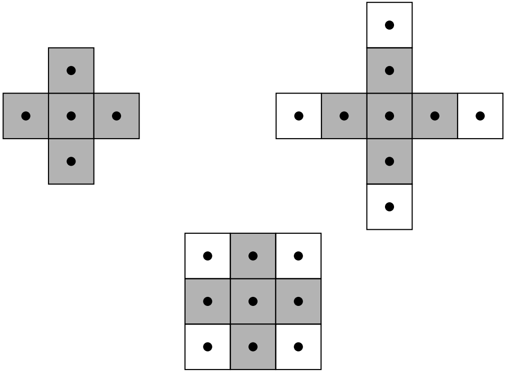
- 1x MareNostrum 5 GPP Node:
  - Peak performance: $\approx$ 7155.86 GFLOP/s
  - Memory bandwidth: 307.2 GB/s

Numerical tests in GPU

Construction of test case

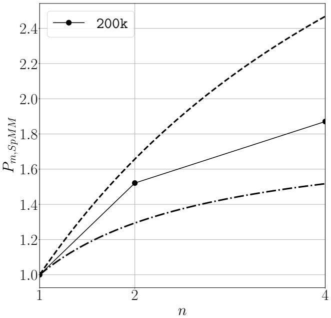
Characteristics

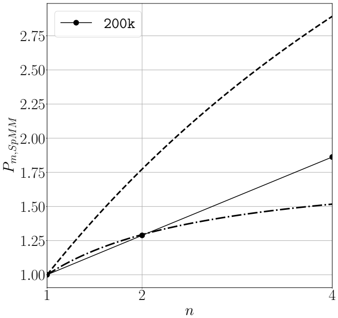
- Run in 1 JFF GPU node:
  - 1x Nvidia A100 80GB PCIe
  - 2x Intel Xeon Gold 6442Y 48C
  - 1024GB of RAM memory
  - 96 OpenMP threads for preprocessing
  - 1 MPI process for computing + OpenCL
- Mesh: 176x704x176 (21M cells)
- Run in three different discretizations for the Laplacian operator: 7p,13p, and 27p
- Run for 1, 2, 4 simultaneous flow states

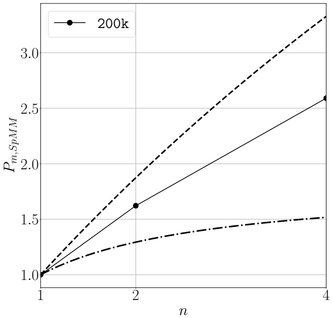


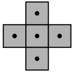
Numerical tests in GPU

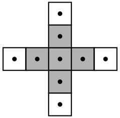
Results. Speed-up for SpMM

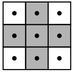






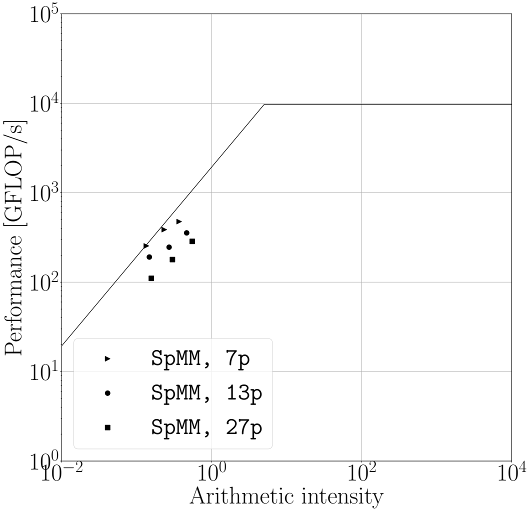






Numerical tests in GPU

Results. Roofline analysis



System properties

- 1x Nvidia A100 80GB PCIe:
  - Peak performance: $\approx$ 9700 GFLOP/s
  - Memory bandwidth: 1935 GB/s

Conclusions

Concluding remarks

- Method to exploit repeated block structures is presented, with PiT as an example
- Implementation in TFA+HPC^2 allows converting `SpMV` to `SpMM` without modifying call
- Ready to use in both CPU and GPU architectures
- Results with CPU and GPU as expected, within upper and lower bounds for all cases.
- As GPUs are loaded as much as possible, some latency issues reduces performance for higher nnz.