ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CINECA

European
Processor
Initiative

Accelerating Sparse Linear Solvers in OpenFOAM using RISC-V Vector Extensions

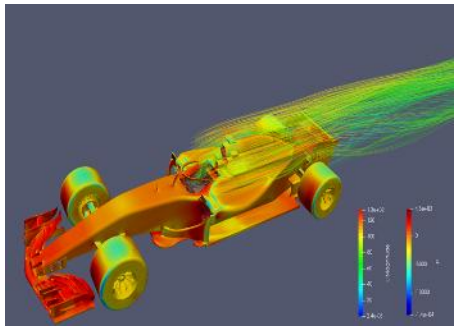
Gabriele Ceccolini[1], Federico Ficarelli[2], Filippo Barbari[2], Simone Bnà[2], Andrea Bartolini[1]

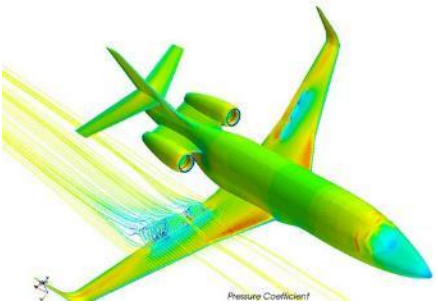
[1]*University of Bologna, Italy*, [2]*CINECA, Italy*

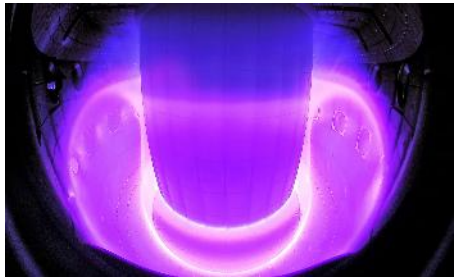
gabriele.ceccolini@studio.unibo.it

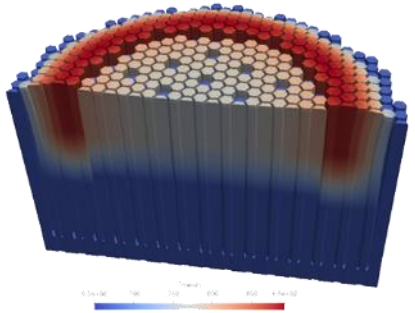
CFD economy and OpenFOAM framework

OpenFOAM









OpenFOAM® HPC Challenge (OHC-1)]

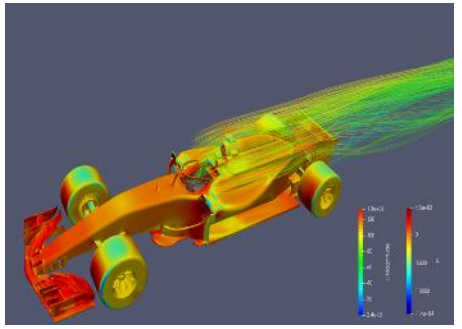
CINECA HPC Annual Report 2024/2025

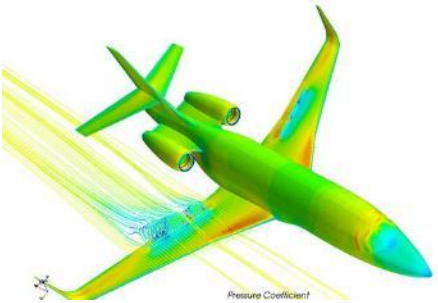
ENGYS, "2023 CFD Technology Trends Survey Results," ENGYS Blog, 2023

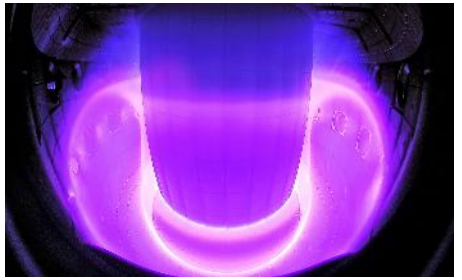
CFD economy and OpenFOAM framework

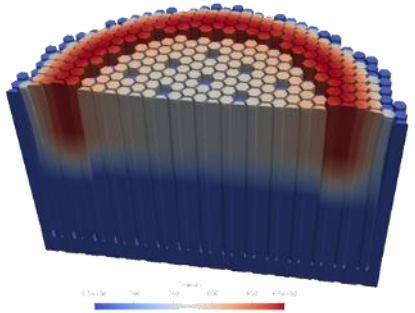
- Computational Fluid Dynamics (CFD) represents a major fraction of HPC usage (e.g., 8% of total core-hours at CINECA).

OpenFOAM









OpenFOAM® HPC Challenge (OHC-1)]

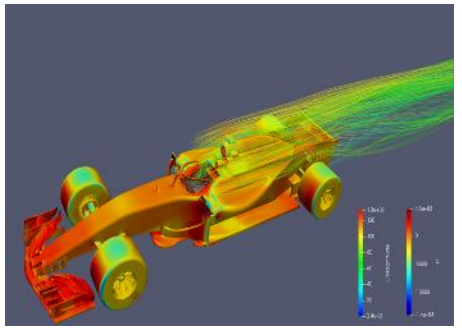
CINECA HPC Annual Report 2024/2025

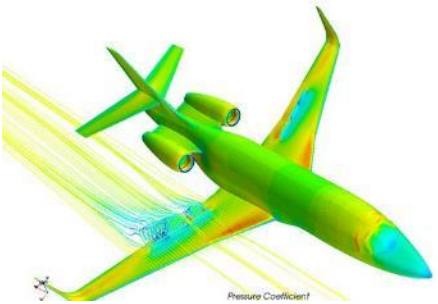
ENGYS, "2023 CFD Technology Trends Survey Results," ENGYS Blog, 2023

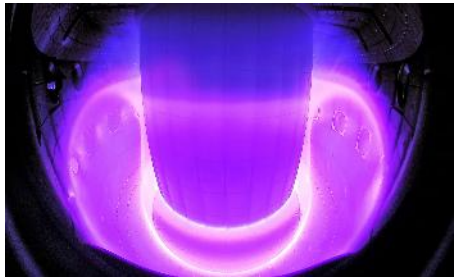
CFD economy and OpenFOAM framework

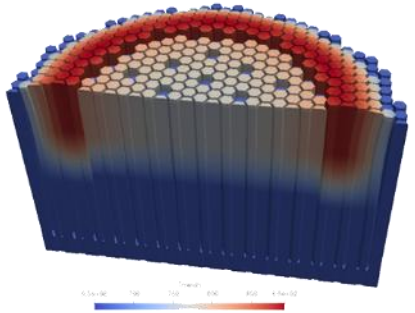
- Computational Fluid Dynamics (CFD) represents a major fraction of HPC usage (e.g., 8% of total core-hours at CINECA).
- Commercial CFD codes require per-core licenses, becoming extremely expensive for large-scale HPC.

OpenFOAM









OpenFOAM® HPC Challenge (OHC-1)]

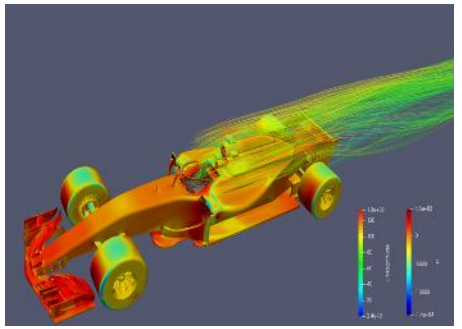
CINECA HPC Annual Report 2024/2025

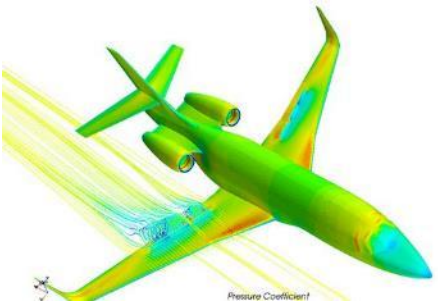
ENGYS, "2023 CFD Technology Trends Survey Results," ENGYS Blog, 2023

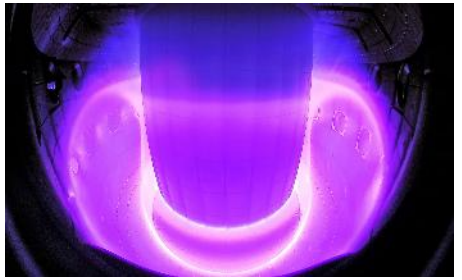
CFD economy and OpenFOAM framework

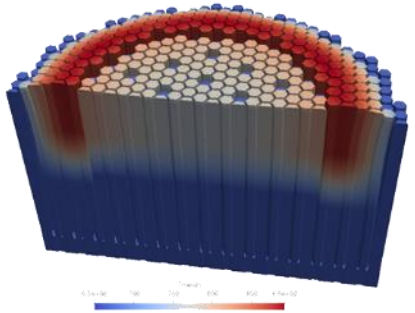
- Computational Fluid Dynamics (CFD) represents a major fraction of HPC usage (e.g., 8% of total core-hours at CINECA).
- Commercial CFD codes require per-core licenses, becoming extremely expensive for large-scale HPC.
- OpenFOAM is the leading open-source alternative, currently used by 50% of CFD professionals.

OpenFOAM









OpenFOAM® HPC Challenge (OHC-1)]

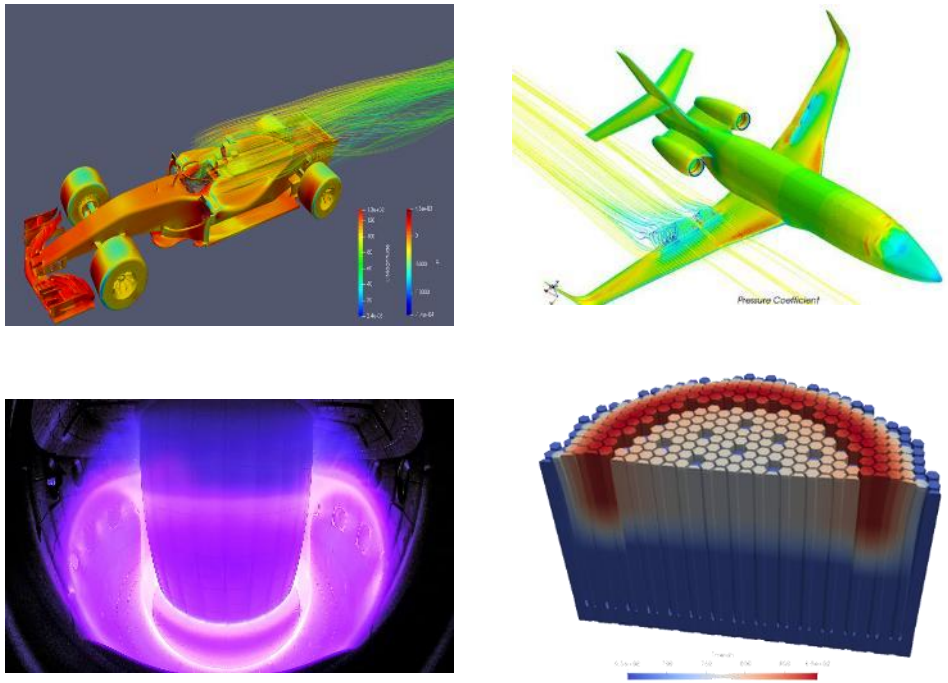
CINECA HPC Annual Report 2024/2025

ENGYS, "2023 CFD Technology Trends Survey Results," ENGYS Blog, 2023

CFD economy and OpenFOAM framework

- Computational Fluid Dynamics (CFD) represents a major fraction of HPC usage (e.g., 8% of total core-hours at CINECA).
- Commercial CFD codes require per-core licenses, becoming extremely expensive for large-scale HPC.
- OpenFOAM is the leading open-source alternative, currently used by 50% of CFD professionals.
- The energy consumed in 2024 by the CFD OpenFOAM jobs in the CPU partition of Leonardo was approximately 182MWh.

OpenFOAM



OpenFOAM® HPC Challenge (OHC-1)]

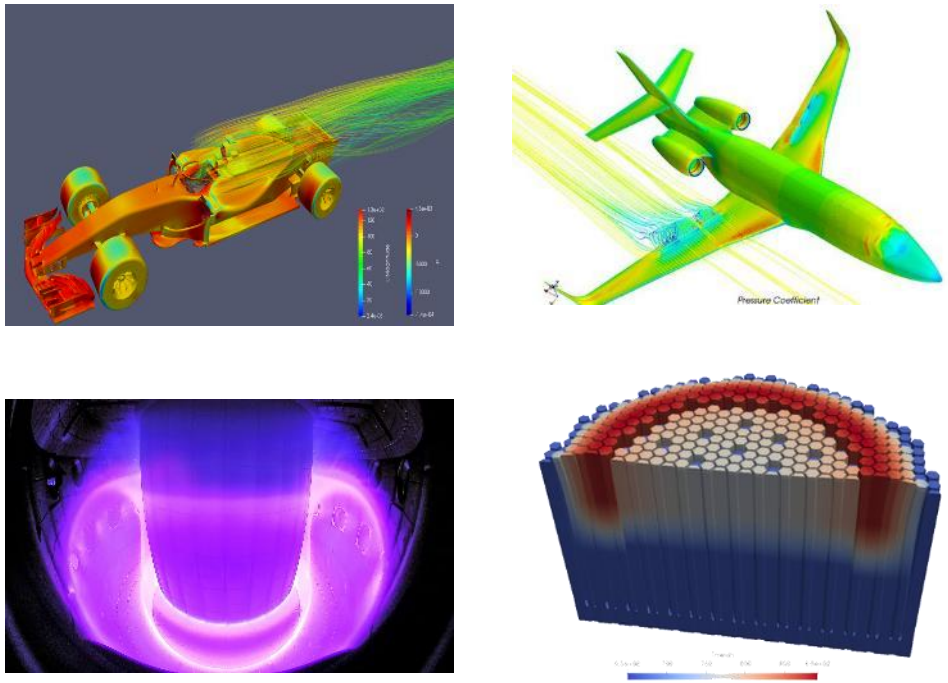
CINECA HPC Annual Report 2024/2025

ENGYS, "2023 CFD Technology Trends Survey Results," ENGYS Blog, 2023

CFD economy and OpenFOAM framework

- Computational Fluid Dynamics (CFD) represents a major fraction of HPC usage (e.g., 8% of total core-hours at CINECA).
- Commercial CFD codes require per-core licenses, becoming extremely expensive for large-scale HPC.
- OpenFOAM is the leading open-source alternative, currently used by 50% of CFD professionals.
- The energy consumed in 2024 by the CFD OpenFOAM jobs in the CPU partition of Leonardo was approximately 182MWh.
- OpenFOAM scales efficiently across thousands of cores (e.g., 32 768 CPU cores at OpenFOAM® HPC Challenge).

OpenFOAM



OpenFOAM® HPC Challenge (OHC-1)]

CINECA HPC Annual Report 2024/2025

ENGYS, "2023 CFD Technology Trends Survey Results," ENGYS Blog, 2023

OpenFOAM current state limitations

OpenFOAM current state limitations

- Based on sparse linear algebra; strongly memory bound.

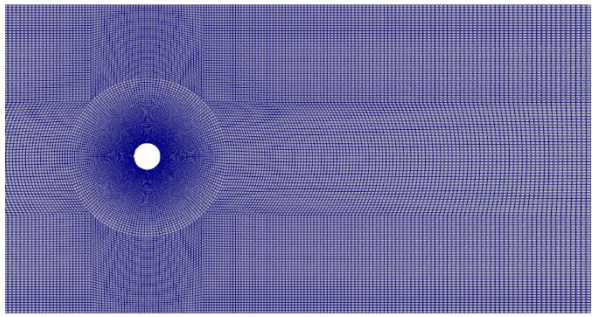
OpenFOAM current state limitations

- Based on sparse linear algebra; strongly memory bound.
- Current Parallelism Status:
  - Limited to domain decomposition and multicore execution using MPI

OpenFOAM current state limitations

- Based on sparse linear algebra; strongly memory bound.
- Current Parallelism Status:
  - Limited to domain decomposition and multicore execution using MPI

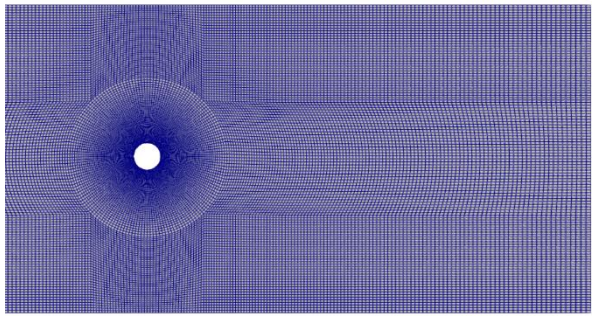
```
$ blockMesh
```



OpenFOAM current state limitations

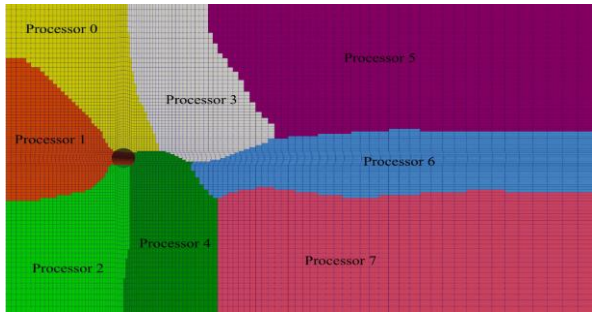
- Based on sparse linear algebra; strongly memory bound.
- Current Parallelism Status:
  - Limited to domain decomposition and multicore execution using MPI

```
$ blockMesh
```



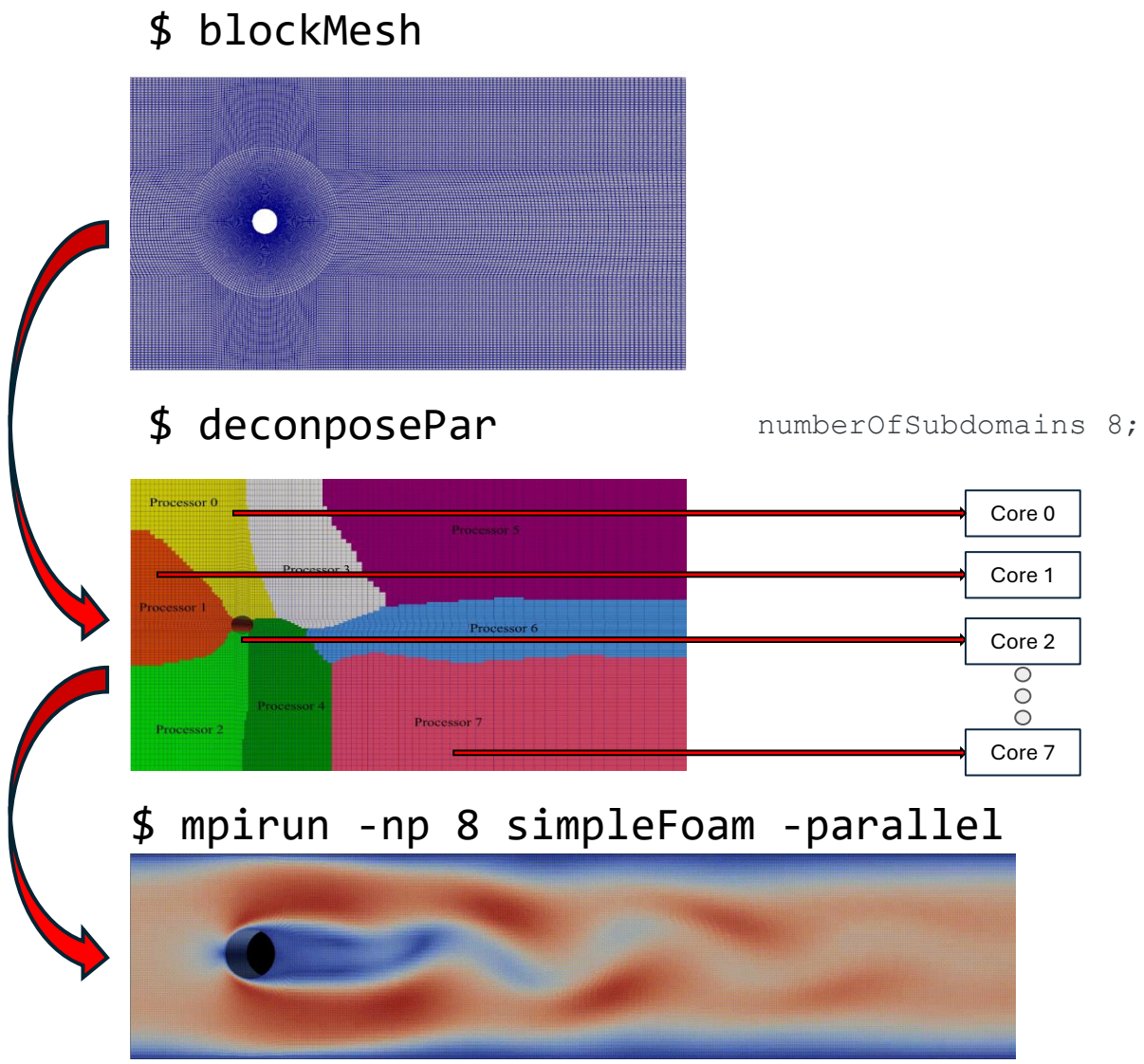
```
$ deconposePar
```

```
numberOfSubdomains 8;
```



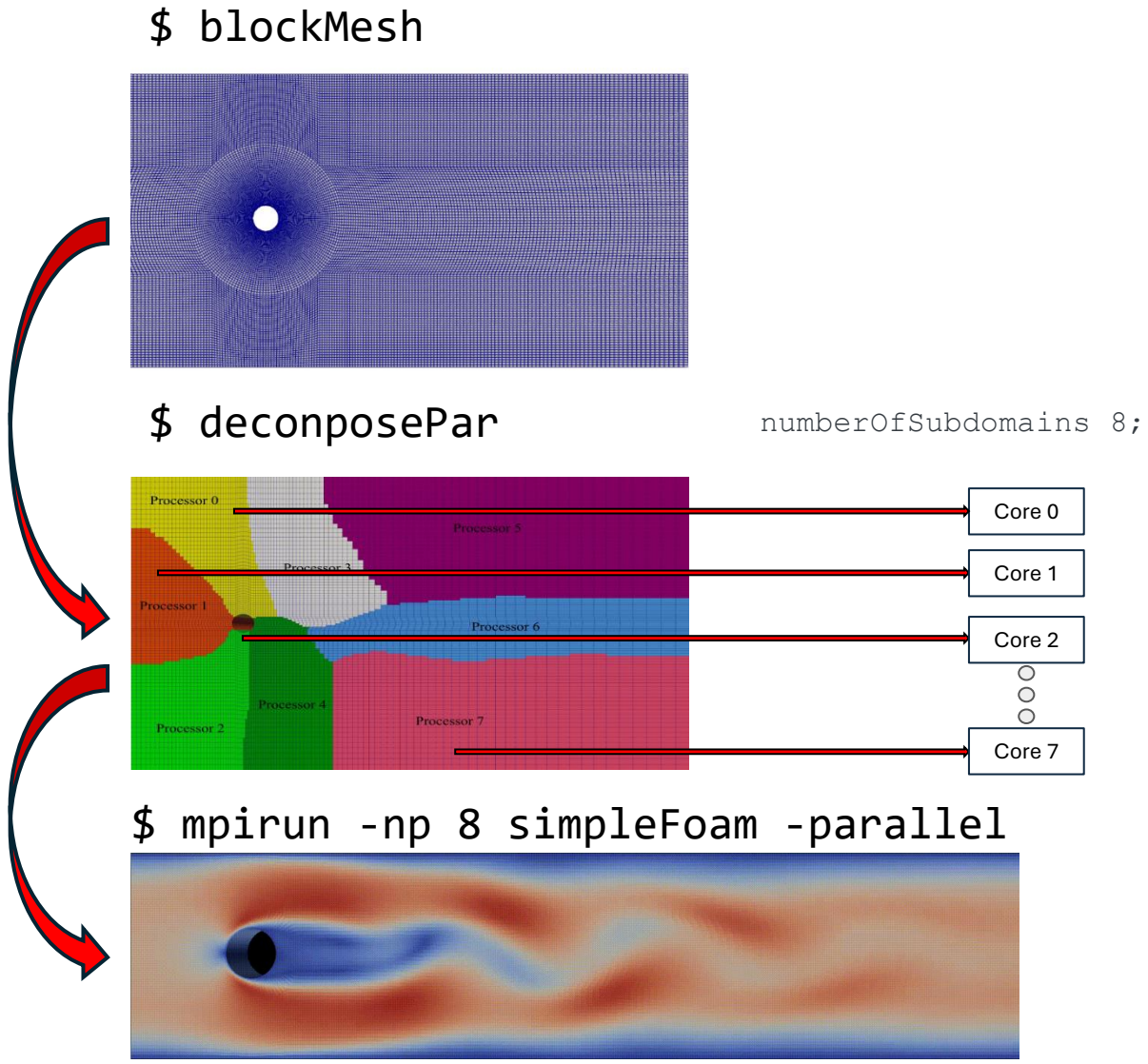
OpenFOAM current state limitations

- Based on sparse linear algebra; strongly memory bound.
- Current Parallelism Status:
  - Limited to domain decomposition and multicore execution using MPI



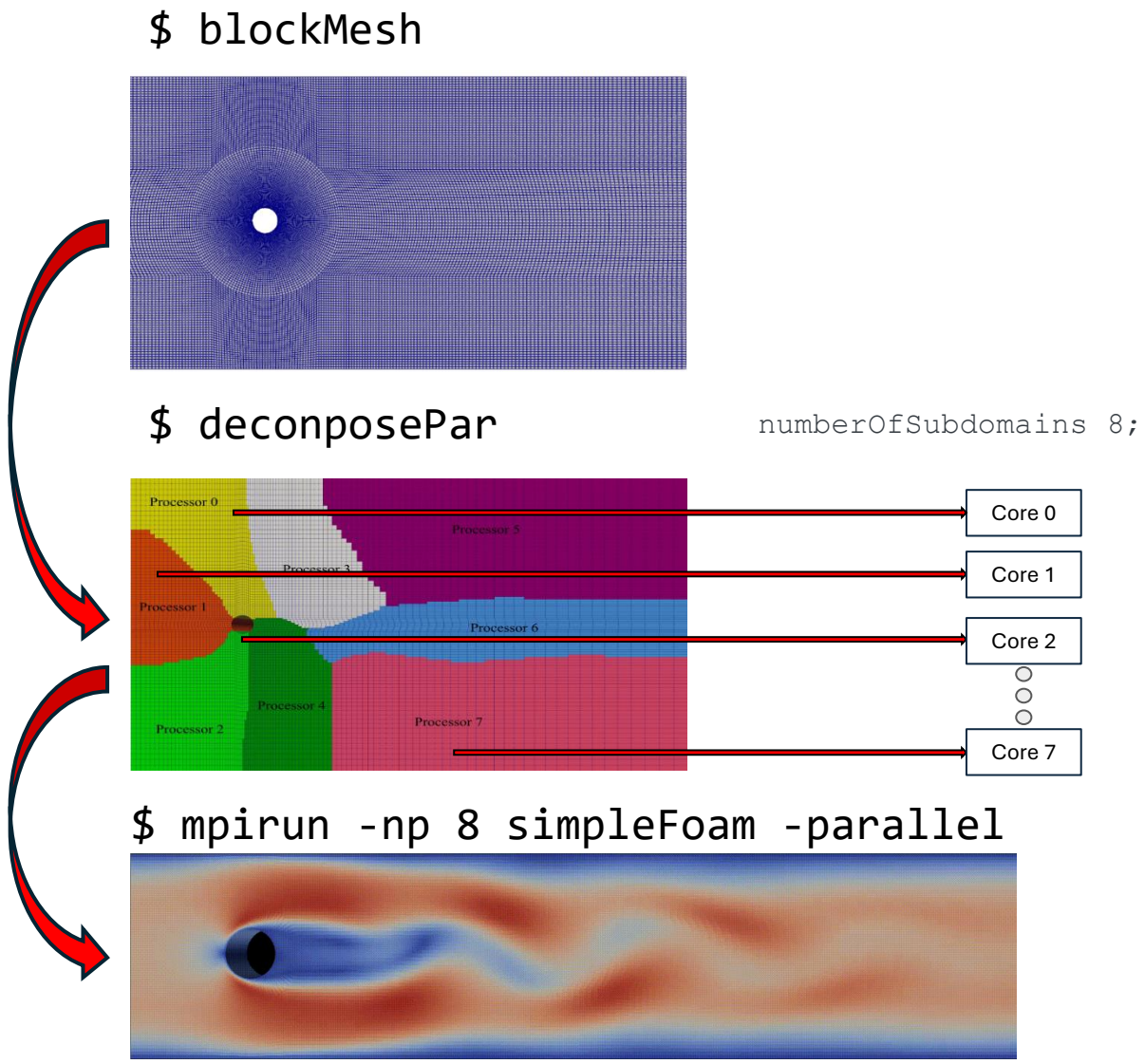
OpenFOAM current state limitations

- Based on sparse linear algebra; strongly memory bound.
- Current Parallelism Status:
  - Limited to domain decomposition and multicore execution using MPI
  - Poor vectorization (only compiler auto vectorization limited by internal sparse matrix format).



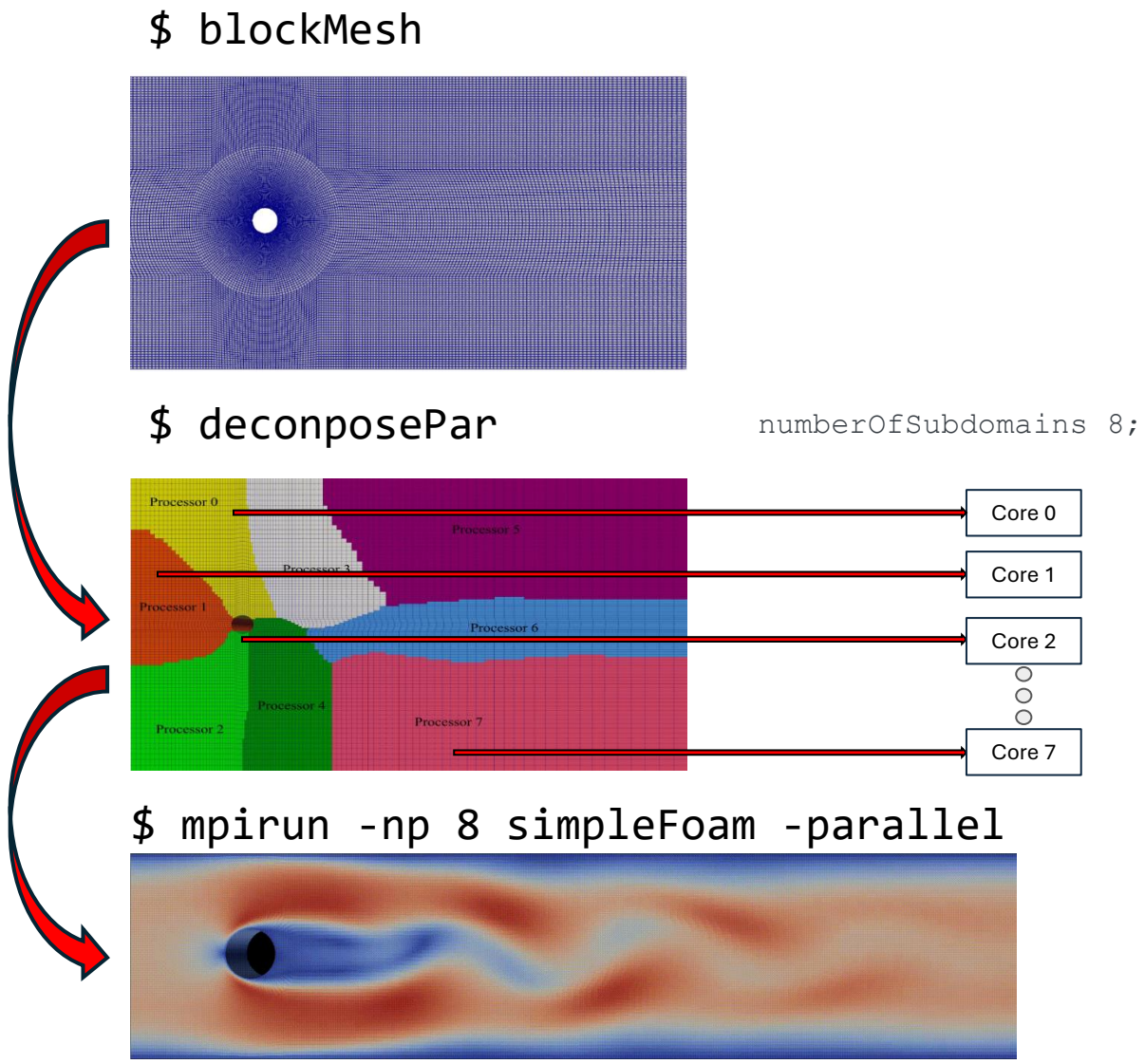
OpenFOAM current state limitations

- Based on sparse linear algebra; strongly memory bound.
- Current Parallelism Status:
  - Limited to domain decomposition and multicore execution using MPI
  - Poor vectorization (only compiler auto vectorization limited by internal sparse matrix format).
  - GPU support is limited (e.g. SPUMA) and not integrated with the upstream version



OpenFOAM current state limitations

- Based on sparse linear algebra; strongly memory bound.
- Current Parallelism Status:
  - Limited to domain decomposition and multicore execution using MPI
  - Poor vectorization (only compiler auto vectorization limited by internal sparse matrix format).
  - GPU support is limited (e.g. SPUMA) and not integrated with the upstream version
  - **Virtually, no use of data-level parallelism!**



Linear solvers and SpMV

Linear solvers and SpMV

- CFD mathematics is based on the discretization of fluid dynamics PDEs (e.g. Navier Stokes)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

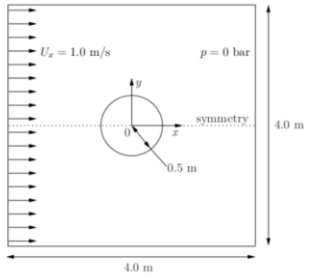


Figure 2.15: Geometry of flow round a cylinder

Linear solvers and SpMV

- CFD mathematics is based on the discretization of fluid dynamics PDEs (e.g. Navier Stokes)
- The spatial domain is discretized into a mesh of control volumes to derive one linear equation per cell.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

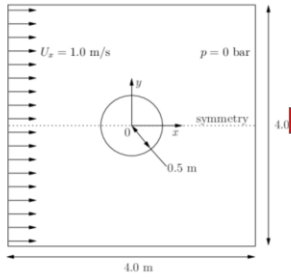


Figure 2.15: Geometry of flow round a cylinder

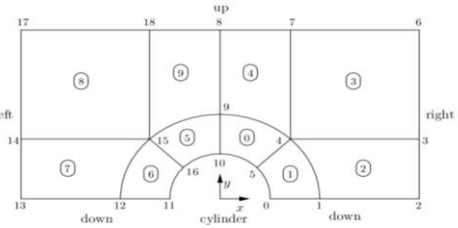


Figure 2.16: Blocks in cylinder geometry

Linear solvers and SpMV

- CFD mathematics is based on the discretization of fluid dynamics PDEs (e.g. Navier Stokes)
- The spatial domain is discretized into a mesh of control volumes to derive one linear equation per cell.
- This approach collapses the entire problem into **solving a large sparse linear system**.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

$$Ax = b$$

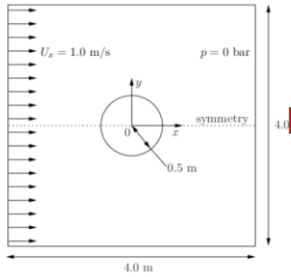


Figure 2.15: Geometry of flow round a cylinder

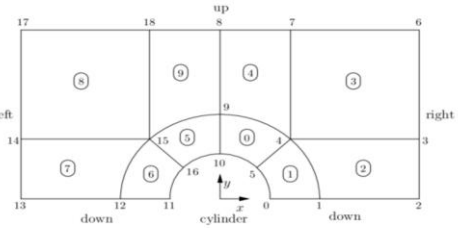
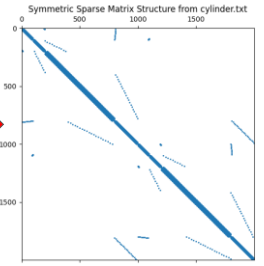


Figure 2.16: Blocks in cylinder geometry

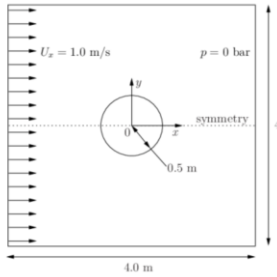


Linear solvers and SpMV

- CFD mathematics is based on the discretization of fluid dynamics PDEs (e.g. Navier Stokes)
- The spatial domain is discretized into a mesh of control volumes to derive one linear equation per cell.
- This approach collapses the entire problem into **solving a large sparse linear system.**
- We use linear iterative solvers computationally based on **sparse matrix vector product (SpMV)**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

$$Ax = b$$



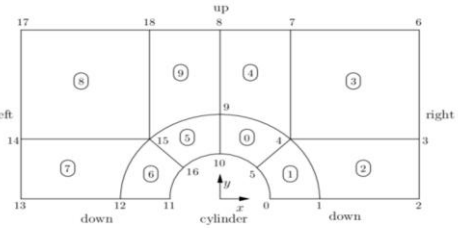


Figure 2.16: Blocks in cylinder geometry

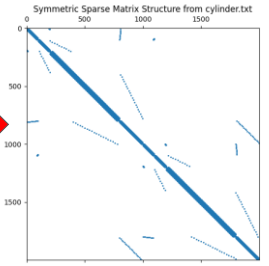


Figure 2.15: Geometry of flow round a cylinder

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
$\alpha_k := \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
$\beta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
$k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result

Linear solvers and SpMV

- CFD mathematics is based on the discretization of fluid dynamics PDEs (e.g. Navier Stokes)
- The spatial domain is discretized into a mesh of control volumes to derive one linear equation per cell.
- This approach collapses the entire problem into **solving a large sparse linear system.**
- We use linear iterative solvers computationally based on **sparse matrix vector product (SpMV)**
- **SpMV is one of the most computationally intensive kernel of a CFD code: good candidate for optimization**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \longrightarrow Ax = b$$

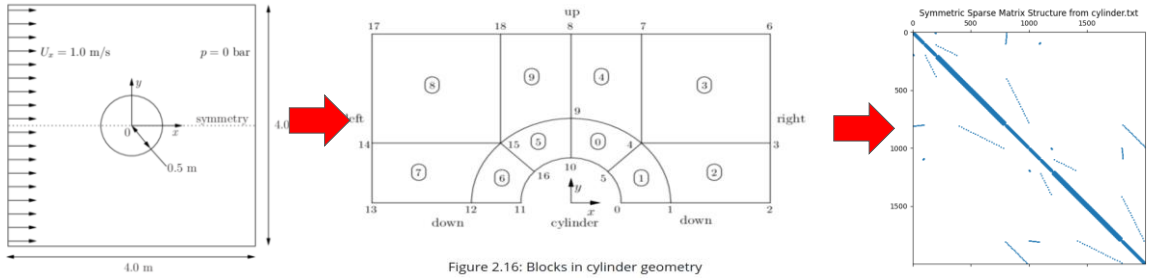
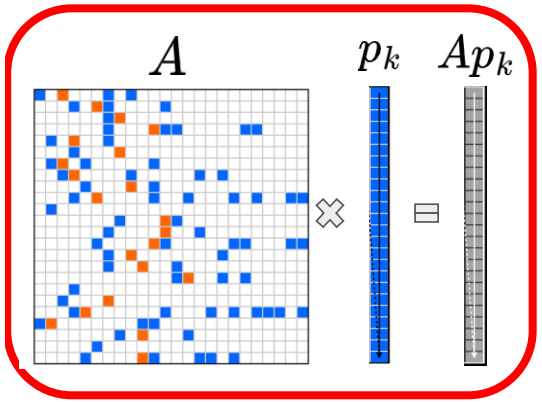


Figure 2.15: Geometry of flow round a cylinder

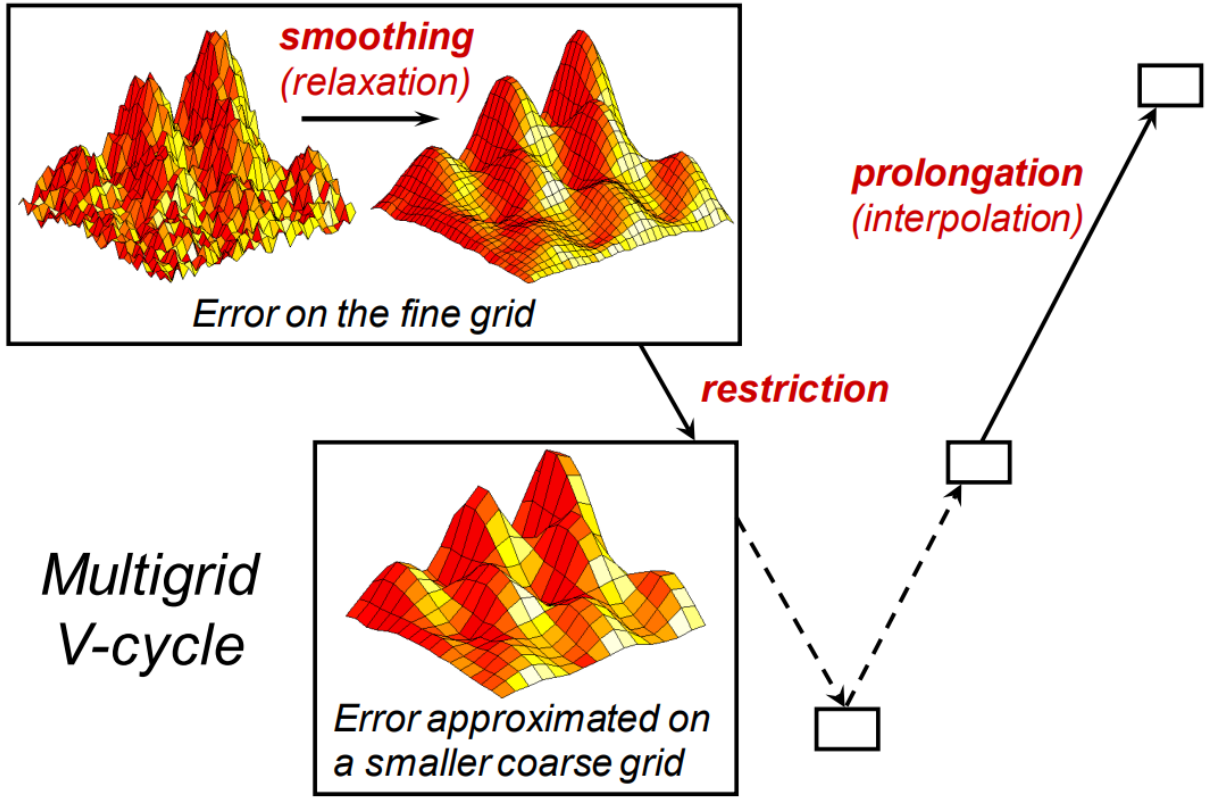
Figure 2.16: Blocks in cylinder geometry

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
$\alpha_k := \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
$\beta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
$k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result



Multigrid methods and Smoothers

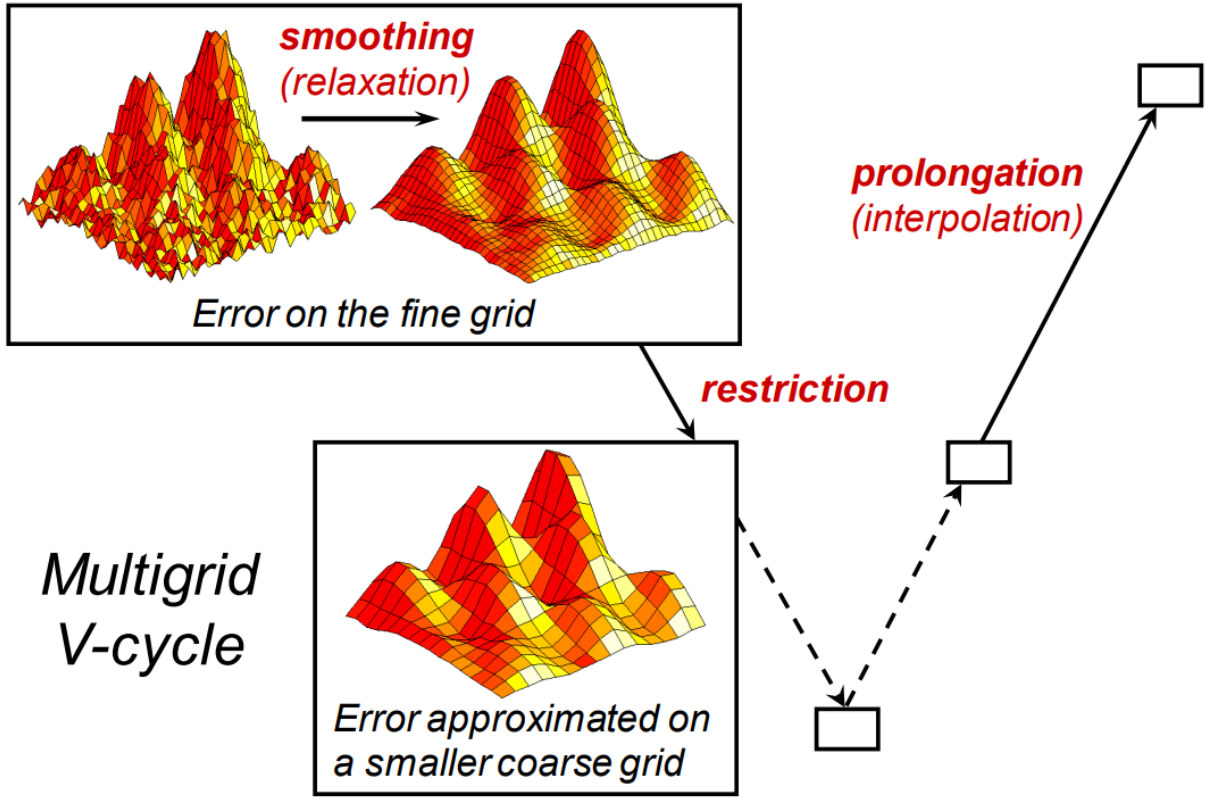
- Standard linear solvers effectively dampen efficiently high-frequency errors but stall on low-frequency errors across the mesh.



U. M. Yang, "*Parallel Multigrid Tutorial*", 20th Copper Mountain Conference on Multigrid Methods, 2021.

Multigrid methods and Smoothers

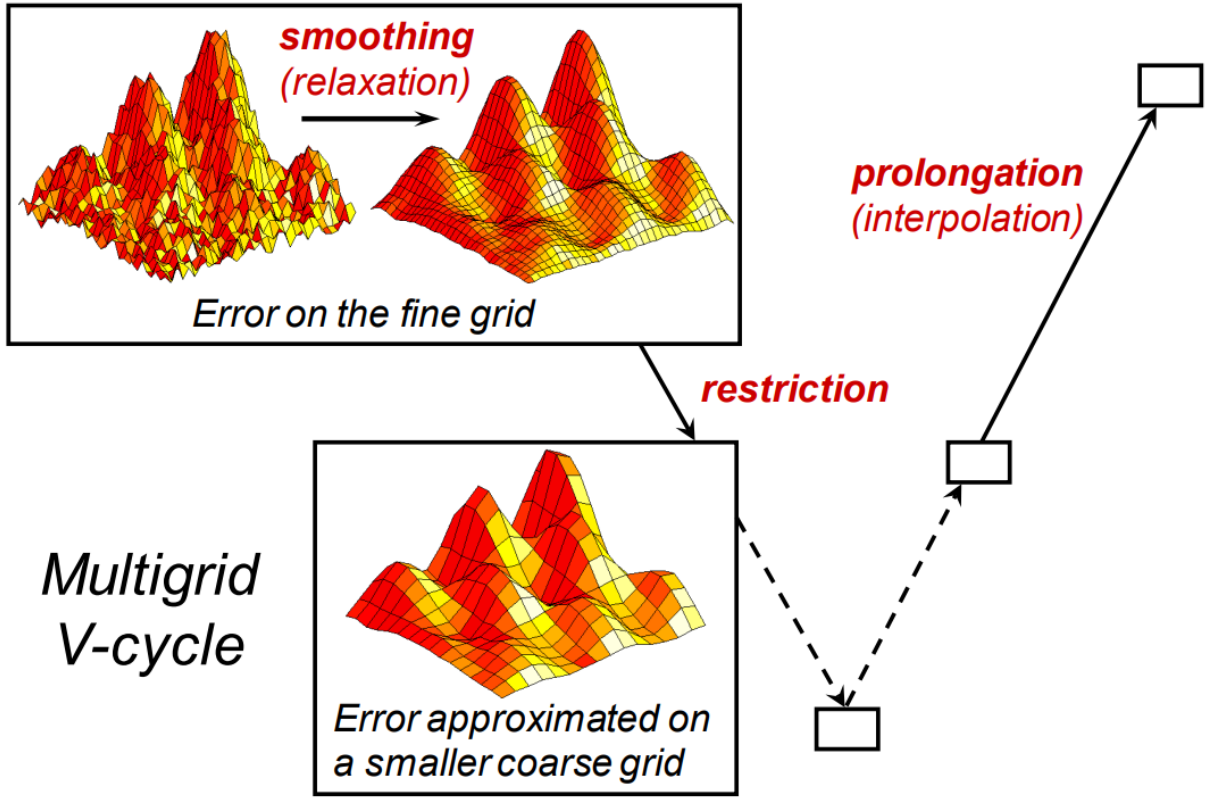
- Standard linear solvers effectively dampen efficiently high-frequency errors but stall on low-frequency errors across the mesh.
- To solve this, modern CFD solvers use the **Multigrid Method**, projecting the problem onto progressively coarser meshes.



U. M. Yang, "*Parallel Multigrid Tutorial*", 20th Copper Mountain Conference on Multigrid Methods, 2021.

Multigrid methods and Smoothers

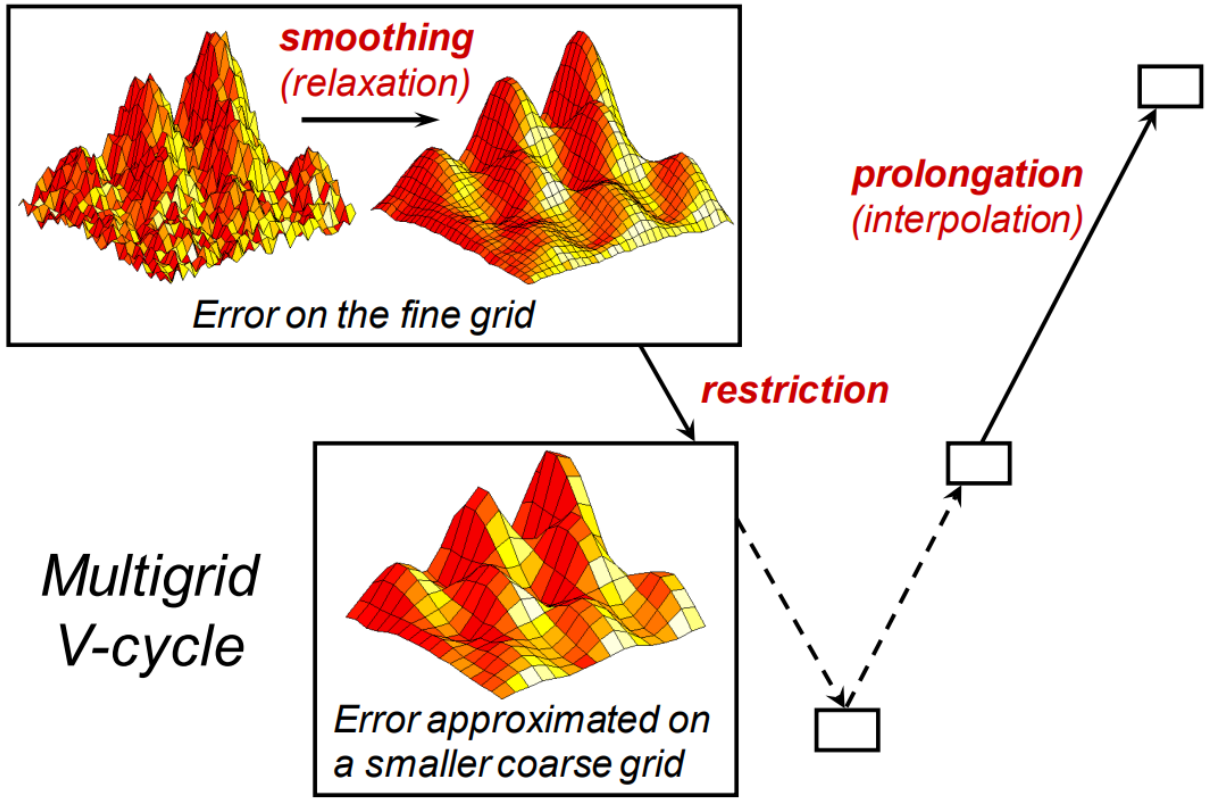
- Standard linear solvers effectively dampen efficiently high-frequency errors but stall on low-frequency errors across the mesh.
- To solve this, modern CFD solvers use the **Multigrid Method**, projecting the problem onto progressively coarser meshes.
- At each grid level, a linear solver acts as a "**smoother**," rapidly damp out high-frequency errors specific to that scale.



U. M. Yang, "*Parallel Multigrid Tutorial*", 20th Copper Mountain Conference on Multigrid Methods, 2021.

Multigrid methods and Smoothers

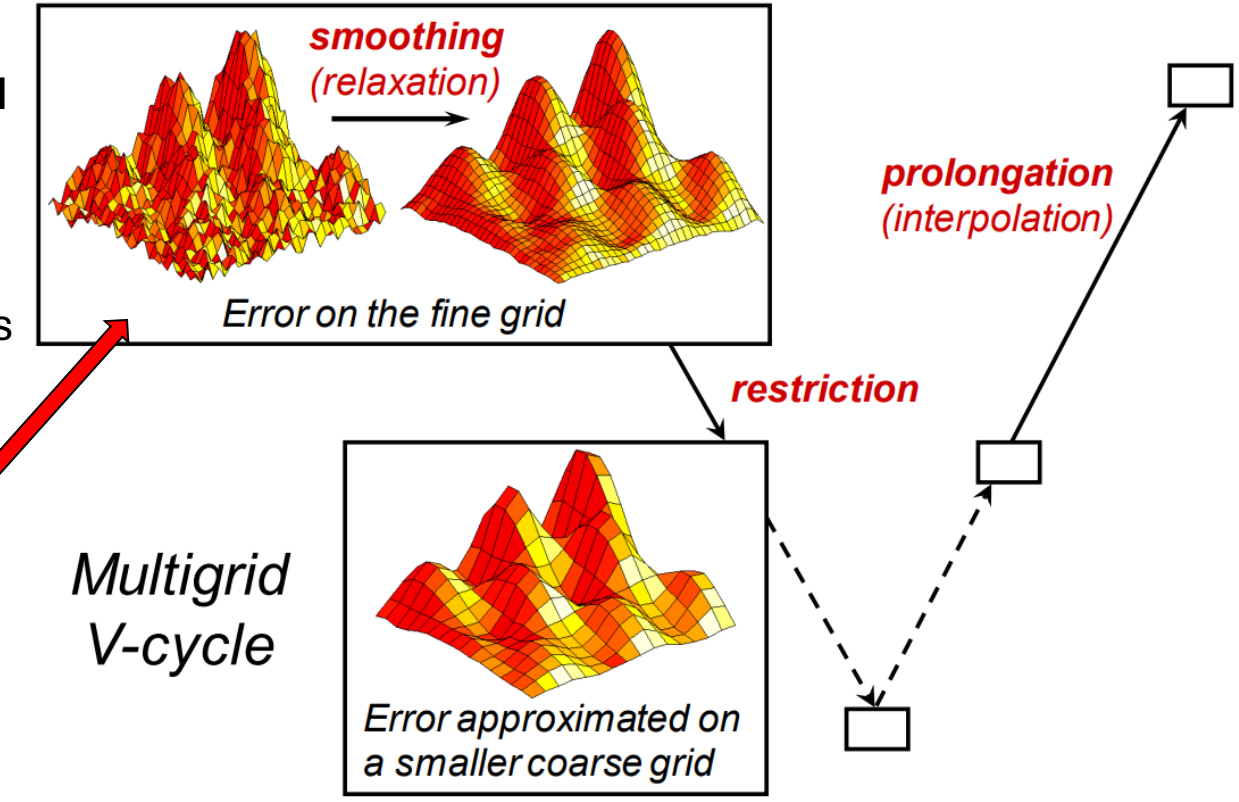
- Standard linear solvers effectively dampen efficiently high-frequency errors but stall on low-frequency errors across the mesh.
- To solve this, modern CFD solvers use the **Multigrid Method**, projecting the problem onto progressively coarser meshes.
- At each grid level, a linear solver acts as a "**smoother**," rapidly damp out high-frequency errors specific to that scale.
- As the grid coarsens, large low-frequency errors appear as high-frequency ones, allowing the smoother to finally damp out them.



U. M. Yang, "*Parallel Multigrid Tutorial*", 20th Copper Mountain Conference on Multigrid Methods, 2021.

Multigrid methods and Smoothers

- Standard linear solvers effectively dampen efficiently high-frequency errors but stall on low-frequency errors across the mesh.
- To solve this, modern CFD solvers use the **Multigrid Method**, projecting the problem onto progressively coarser meshes.
- At each grid level, a linear solver acts as a "**smoother**," rapidly damp out high-frequency errors specific to that scale.
- As the grid coarsens, large low-frequency errors appear as high-frequency ones, allowing the smoother to finally damp out them.
- **In our test case, this smoothing operation accounts for ~50% of the total compute time, making it the most intensive phase and our primary optimization target.**

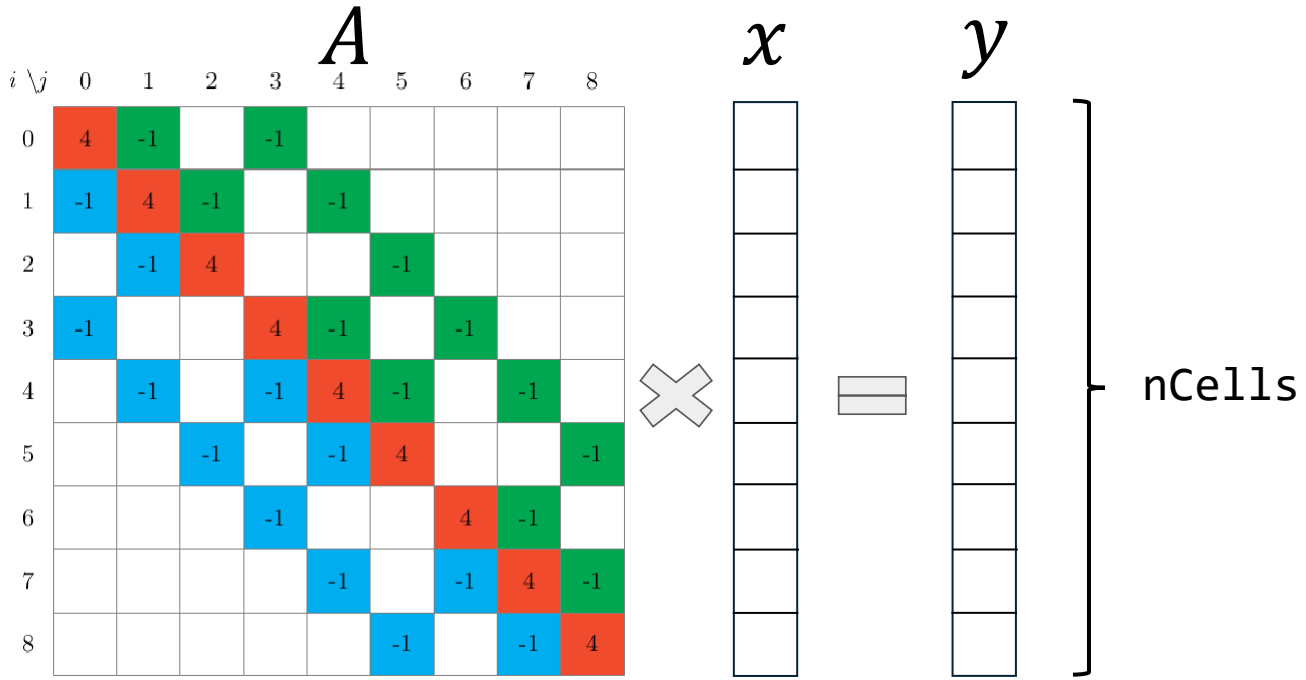


U. M. Yang, "*Parallel Multigrid Tutorial*", 20th Copper Mountain Conference on Multigrid Methods, 2021.

OpenFOAM native sparse matrix format

OpenFOAM native sparse matrix format

- LDU-COO is the internal sparse matrix format in OpenFOAM



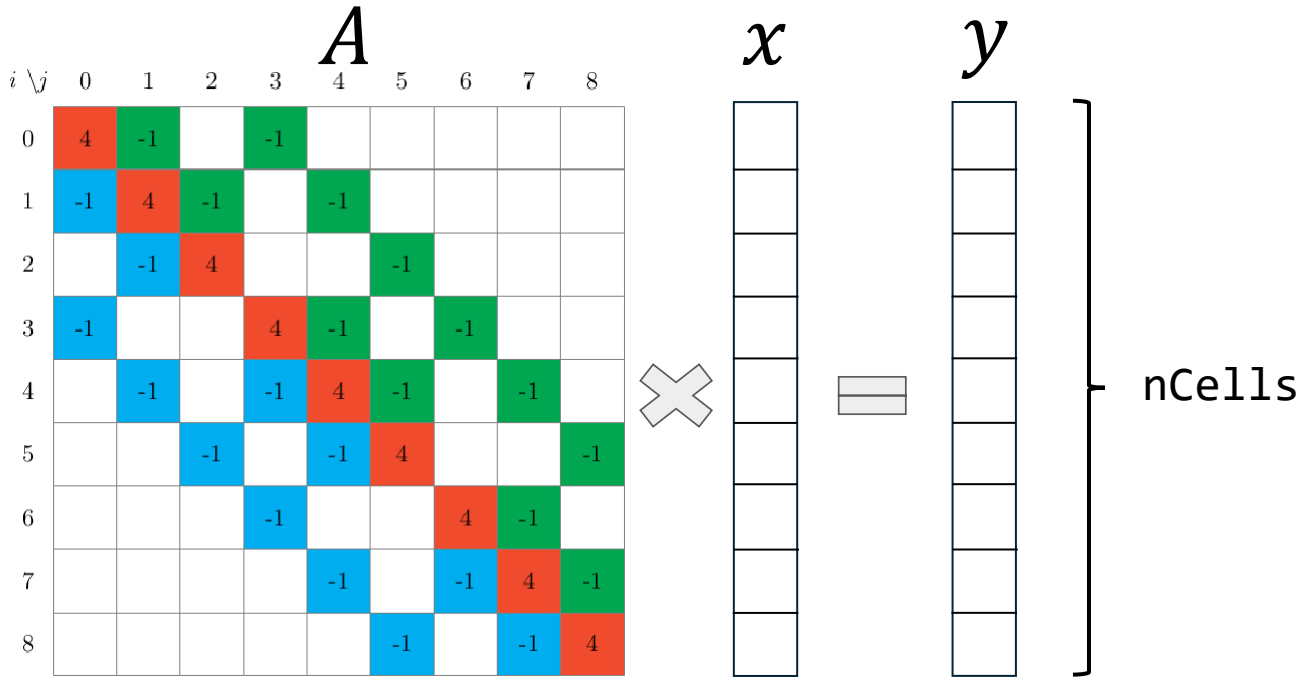

```
upper      = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
diag       = [ 4  4  4  4  4  4  4  4  4]
lower      = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]

upperAddr = [ 1  3  2  4  5  4  6  5  7  8  7  8]
lowerAddr = [ 0  0  1  1  2  3  3  4  4  5  6  7]
```

nFaces

Bnà, S., Spisso, I., Olesen, M., & Rossi, G. (2020). PETSc4FOAM: a library to plug-in PETSc into the OpenFOAM framework.

OpenFOAM native sparse matrix format

- LDU-COO is the internal sparse matrix format in OpenFOAM
- Hostile to vectorization
  - Indirect access on x and y




```
// 1. Diagonal contribute
for (size_t i = 0; i < nCells; i++) {
    y[i] = diag[i] * x[i];
}

// 2. Off-Diagonal contribute
for (size_t f = 0; f < nFaces; f++) {
    y[lowerAddr[f]] += upper[f] * x[upperAddr[f]];
    y[upperAddr[f]] += lower[f] * x[lowerAddr[f]];
}
```

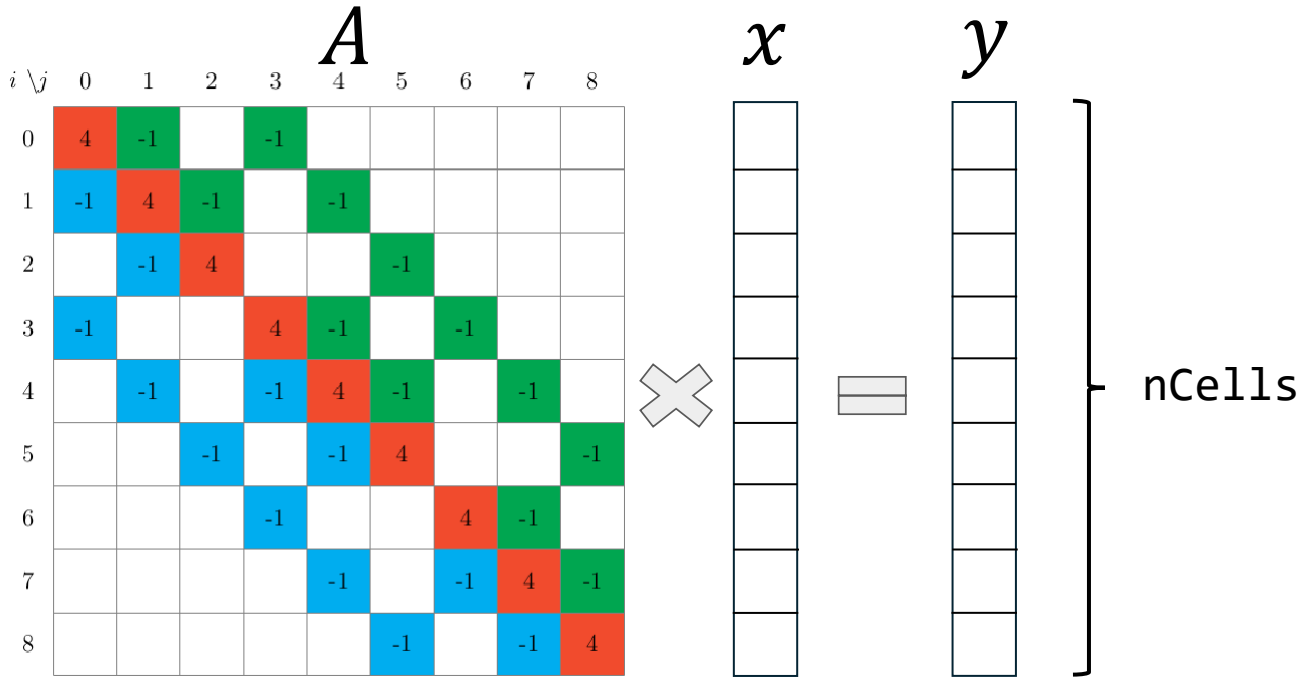
```
upper     = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
diag      = [ 4  4  4  4  4  4  4  4  4]
lower     = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]

upperAddr = [ 1  3  2  4  5  4  6  5  7  8  7  8]
lowerAddr = [ 0  0  1  1  2  3  3  4  4  5  6  7]
```

nFaces

Bnà, S., Spisso, I., Olesen, M., & Rossi, G. (2020). PETSc4FOAM: a library to plug-in PETSc into the OpenFOAM framework.

OpenFOAM native sparse matrix format

- LDU-COO is the internal sparse matrix format in OpenFOAM
- Hostile to vectorization
  - Indirect access on x and y
  - Race conditions on y update




```
// 1. Diagonal contribute
for (size_t i = 0; i < nCells; i++) {
    y[i] = diag[i] * x[i];
}

// 2. Off-Diagonal contribute
for (size_t f = 0; f < nFaces; f++) {
    y[lowerAddr[f]] += upper[f] * x[upperAddr[f]];
    y[upperAddr[f]] += lower[f] * x[lowerAddr[f]];
}
```

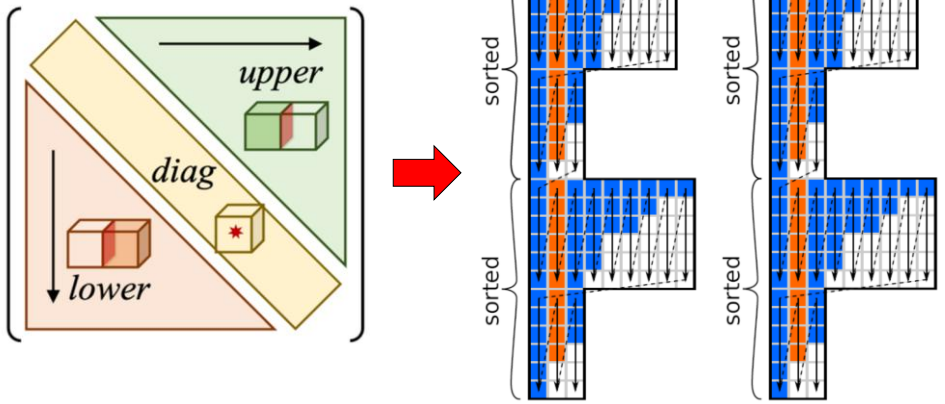
```
upper     = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
diag      = [ 4  4  4  4  4  4  4  4  4]
lower     = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]

upperAddr = [ 1  3  2  4  5  4  6  5  7  8  7  8]
lowerAddr = [ 0  0  1  1  2  3  3  4  4  5  6  7]
```

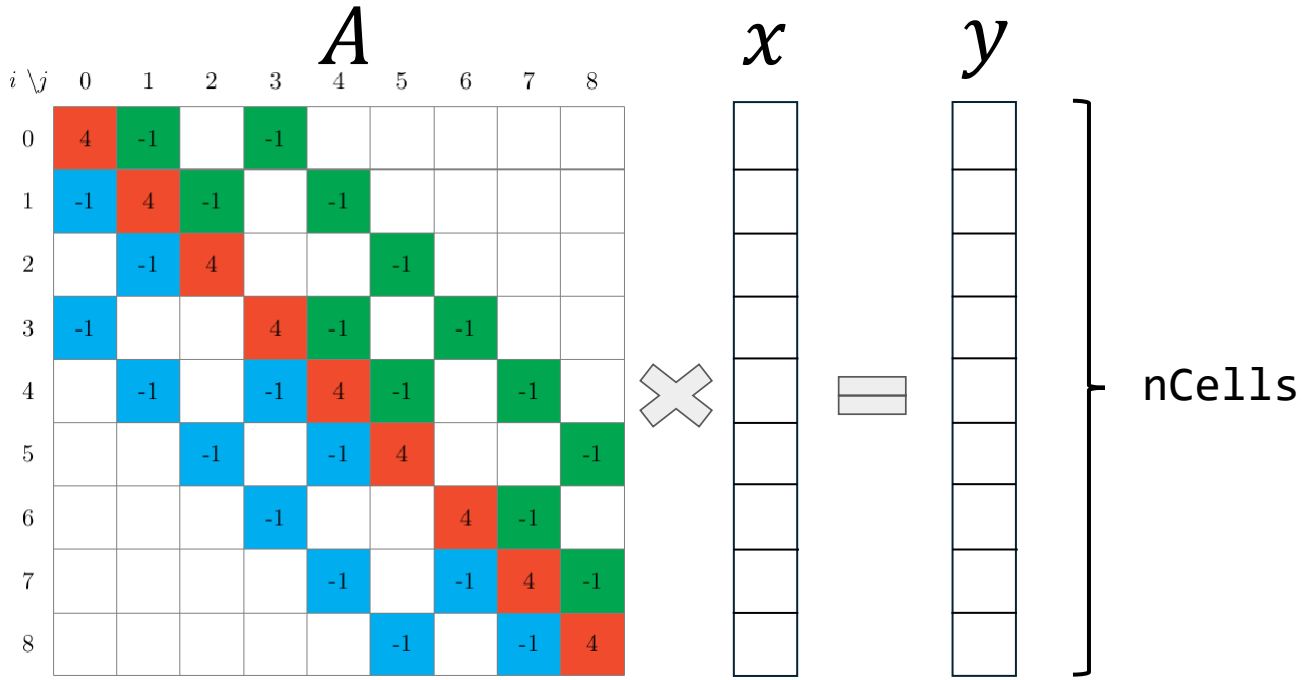
nFaces

Bnà, S., Spisso, I., Olesen, M., & Rossi, G. (2020). PETSc4FOAM: a library to plug-in PETSc into the OpenFOAM framework.

OpenFOAM native sparse matrix format

- LDU-COO is the internal sparse matrix format in OpenFOAM
- Hostile to vectorization
  - Indirect access on x and y
  - Race conditions on y update
- **Solution -> runtime matrix format conversion**






```
upper      = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
diag       = [ 4  4  4  4  4  4  4  4  4]
lower      = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]

upperAddr = [ 1  3  2  4  5  4  6  5  7  8  7  8]
lowerAddr = [ 0  0  1  1  2  3  3  4  4  5  6  7]
```

nFaces

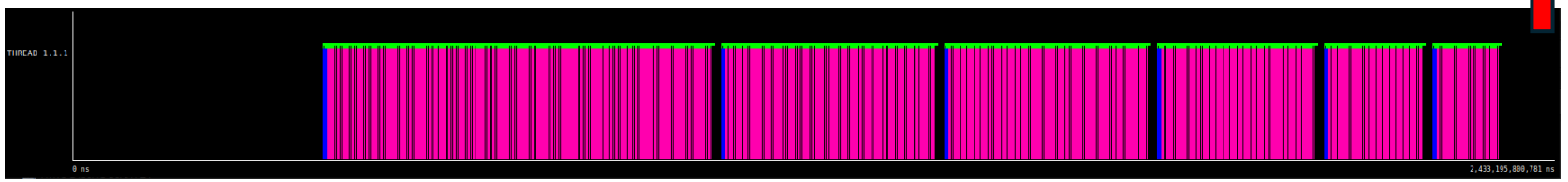
Bnà, S., Spisso, I., Olesen, M., & Rossi, G. (2020). PETSc4FOAM: a library to plug-in PETSc into the OpenFOAM framework.

Application profiling

V. Pillet et al. (1995) PARAVER: A tool to visualize and analyze parallel code http://tools.bsc.es/paraver

Application profiling

Full CFD simulation timeline profiled with Paraver



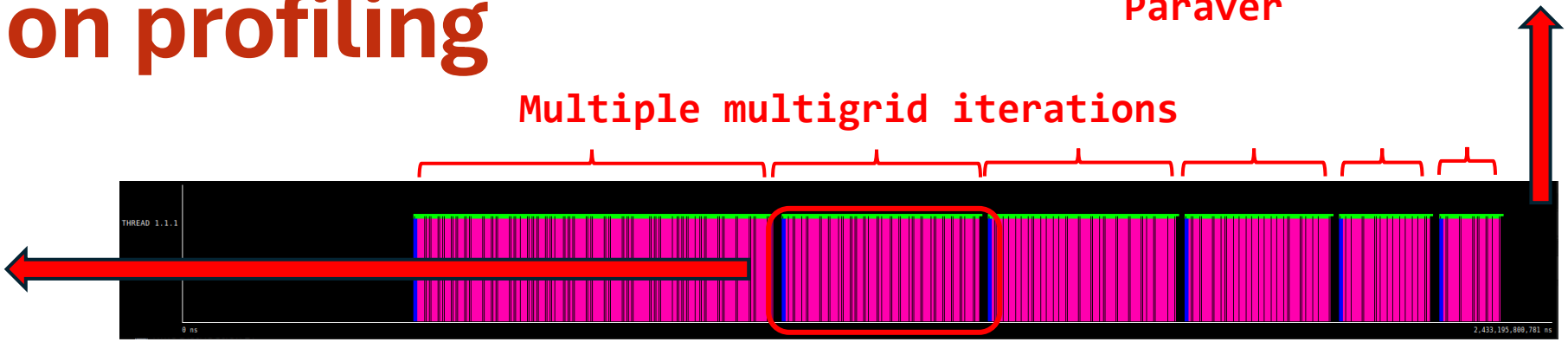
V. Pillet et al. (1995) PARAVER: A tool to visualize and analyze parallel code http://tools.bsc.es/paraver

Application profiling

Full CFD simulation timeline profiled with Paraver

Multiple multigrid iterations

Multiple multigrid method iterations are called during a CFD simulation



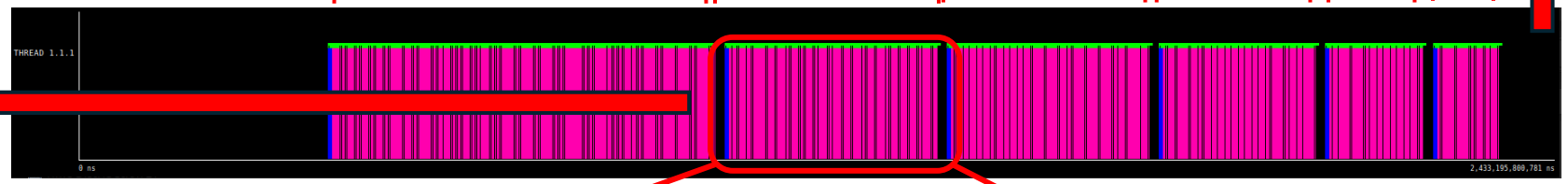
V. Pillet et al. (1995) PARAVER: A tool to visualize and analyze parallel code http://tools.bsc.es/paraver

Application profiling

Full CFD simulation timeline profiled with Paraver

Multiple multigrid iterations

Multiple multigrid method iterations are called during a CFD simulation



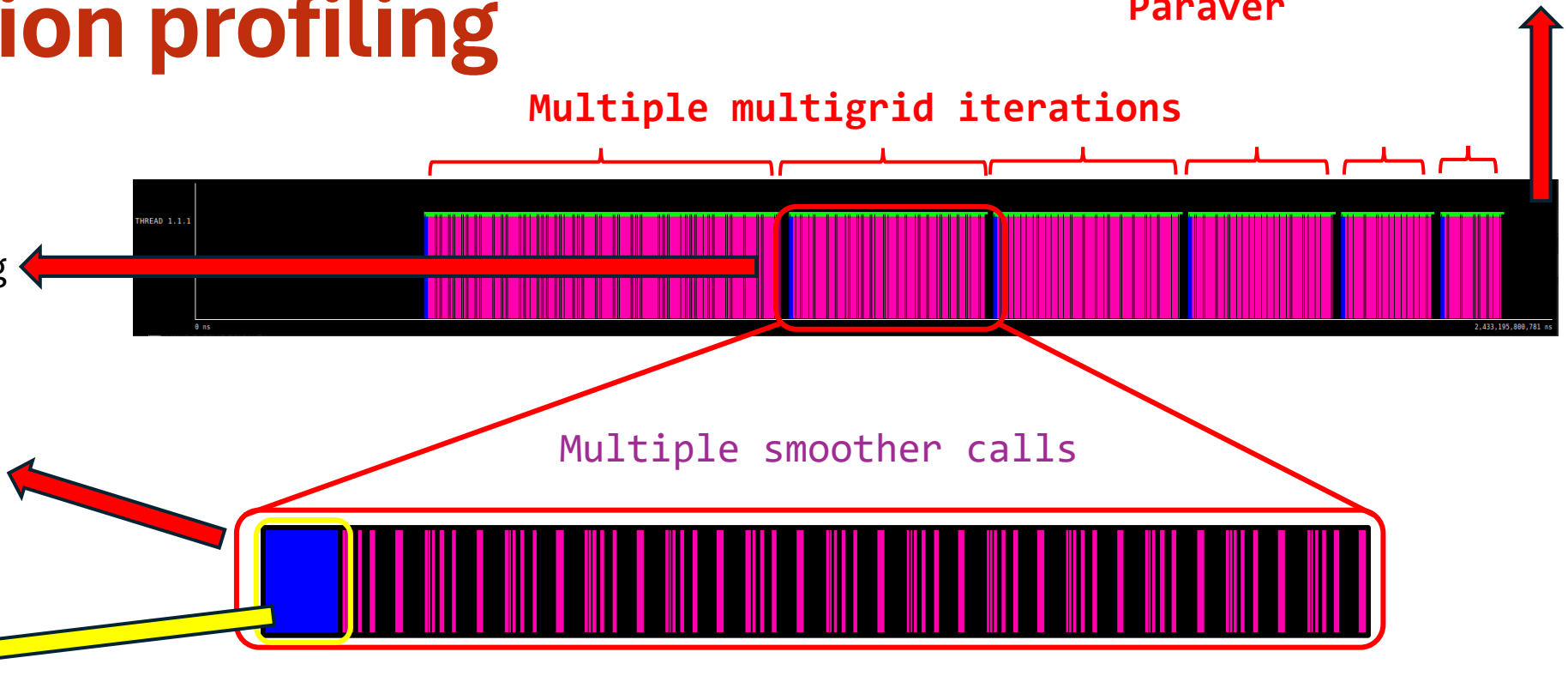
During a single multigrid iteration, the smoother is called tens to hundreds of times

Multiple smoother calls

V. Pillet et al. (1995) PARAVER: A tool to visualize and analyze parallel code http://tools.bsc.es/paraver

Application profiling

Full CFD simulation timeline profiled with Paraver



Multiple multigrid method iterations are called during a CFD simulation

During a single multigrid iteration, the smoother is called tens to hundreds of times

Format conversion overhead

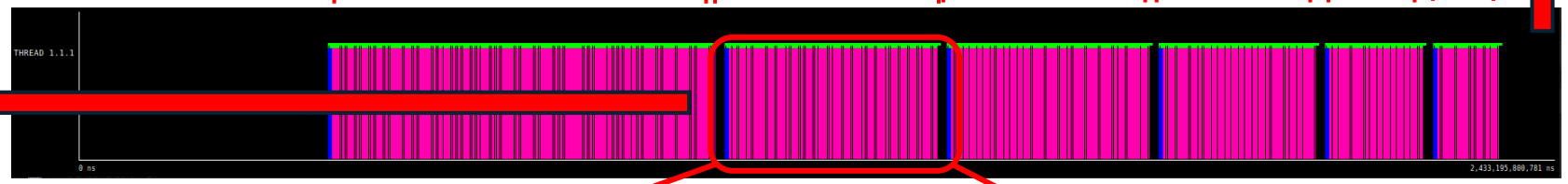
V. Pillet et al. (1995) PARAVER: A tool to visualize and analyze parallel code http://tools.bsc.es/paraver

Application profiling

Full CFD simulation timeline profiled with Paraver

Multiple multigrid iterations

Multiple multigrid method iterations are called during a CFD simulation



During a single multigrid iteration, the smoother is called tens to hundreds of times

Format conversion overhead

Multiple smoother calls

$(\boldsymbol{D}^{-1})_{ii} = \frac{1}{A_{ii}}$

$\psi^{(k+1)} = \psi^{(k)} + \omega \boldsymbol{D}^{-1}(\boldsymbol{b} - \boldsymbol{A}\psi^{(k)})$

Smoother

SpMV SpMV

$\boldsymbol{A}\psi^{(k)}$

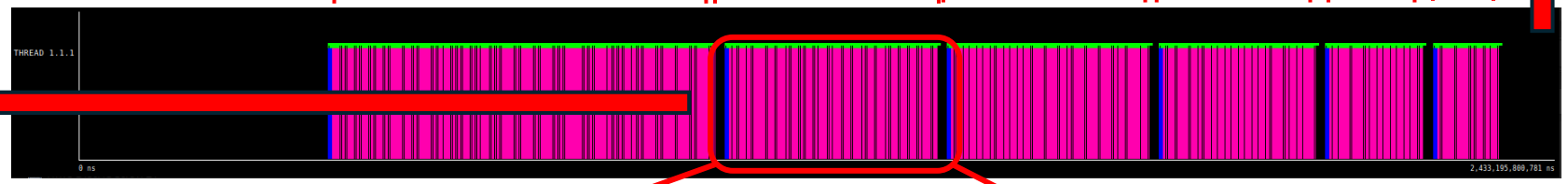
V. Pillet et al. (1995) PARAVER: A tool to visualize and analyze parallel code http://tools.bsc.es/paraver

Application profiling

Full CFD simulation timeline profiled with Paraver

Multiple multigrid iterations

Multiple multigrid method iterations are called during a CFD simulation



During a single multigrid iteration, the smoother is called tens to hundreds of times

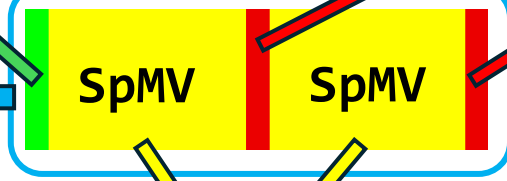
Format conversion overhead

Multiple smoother calls

$(\boldsymbol{D}^{-1})_{ii} = \frac{1}{A_{ii}}$

$\psi^{(k+1)} = \psi^{(k)} + \omega \boldsymbol{D}^{-1}(\boldsymbol{b} - \boldsymbol{A}\psi^{(k)})$

Smoother

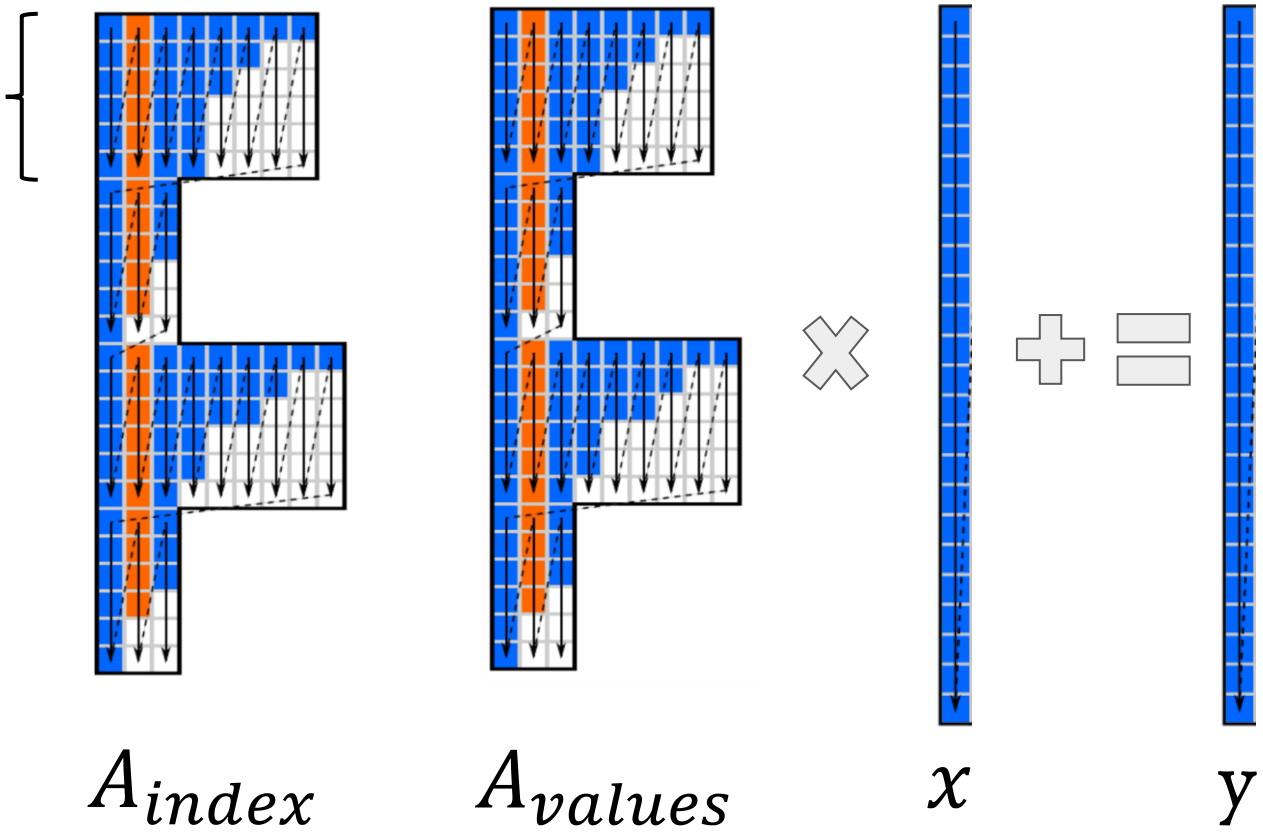


$\boldsymbol{A}\psi^{(k)}$

The smoother represents around 50% of the total application runtime, and the computation is dominated by SpMV

V. Pillet et al. (1995) PARAVER: A tool to visualize and analyze parallel code http://tools.bsc.es/paraver

SpMV SELL-C-σ vectorization

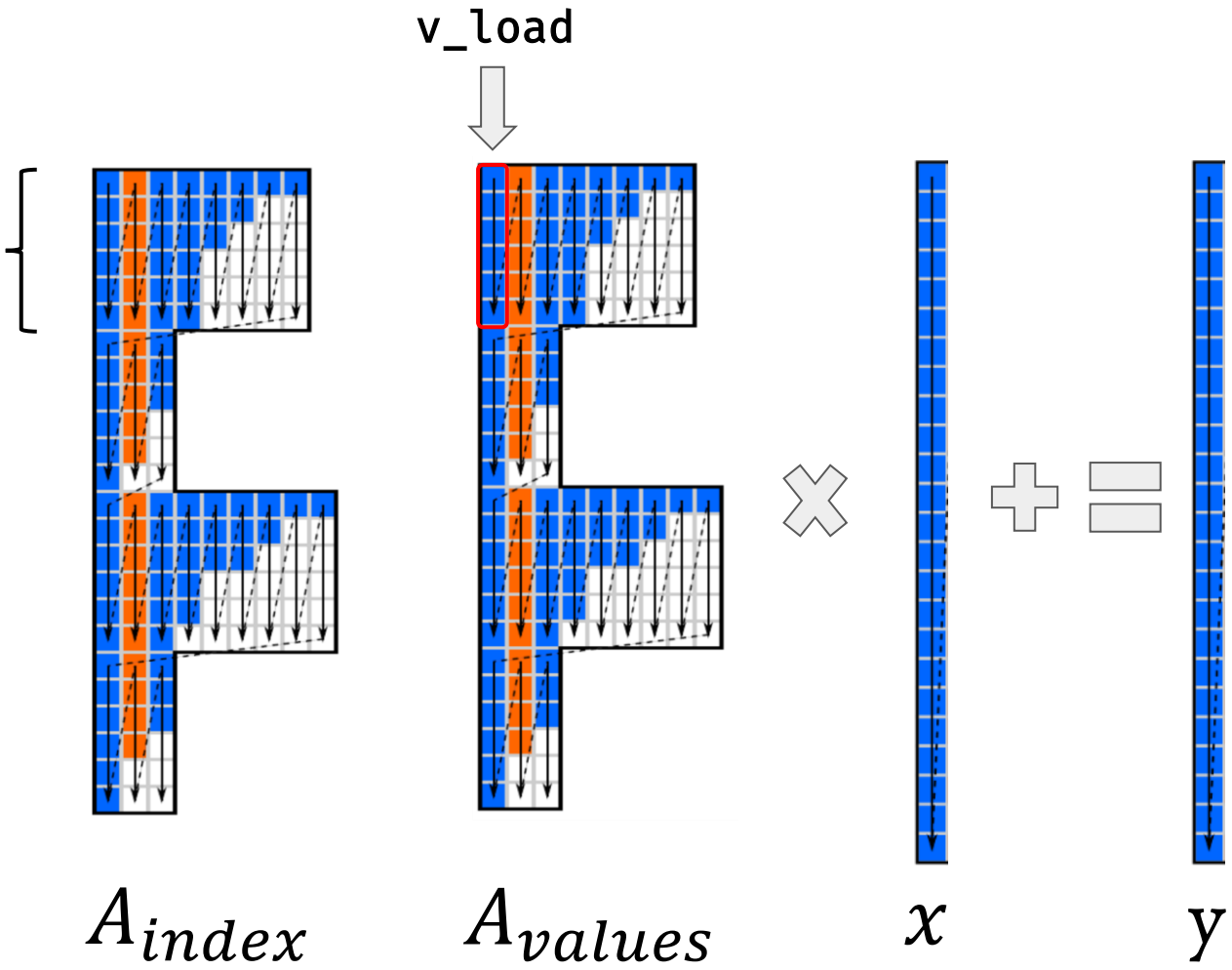


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization

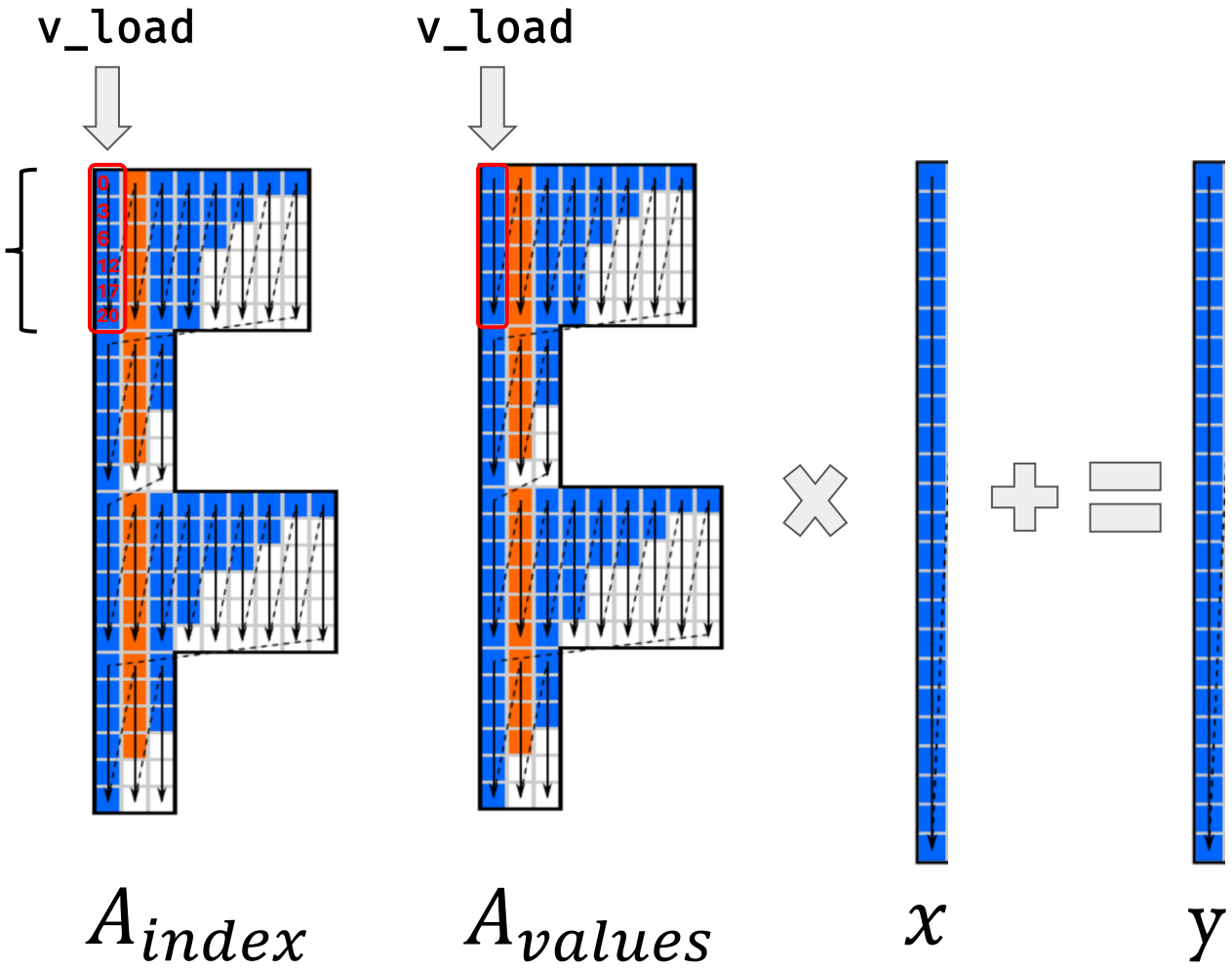


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization

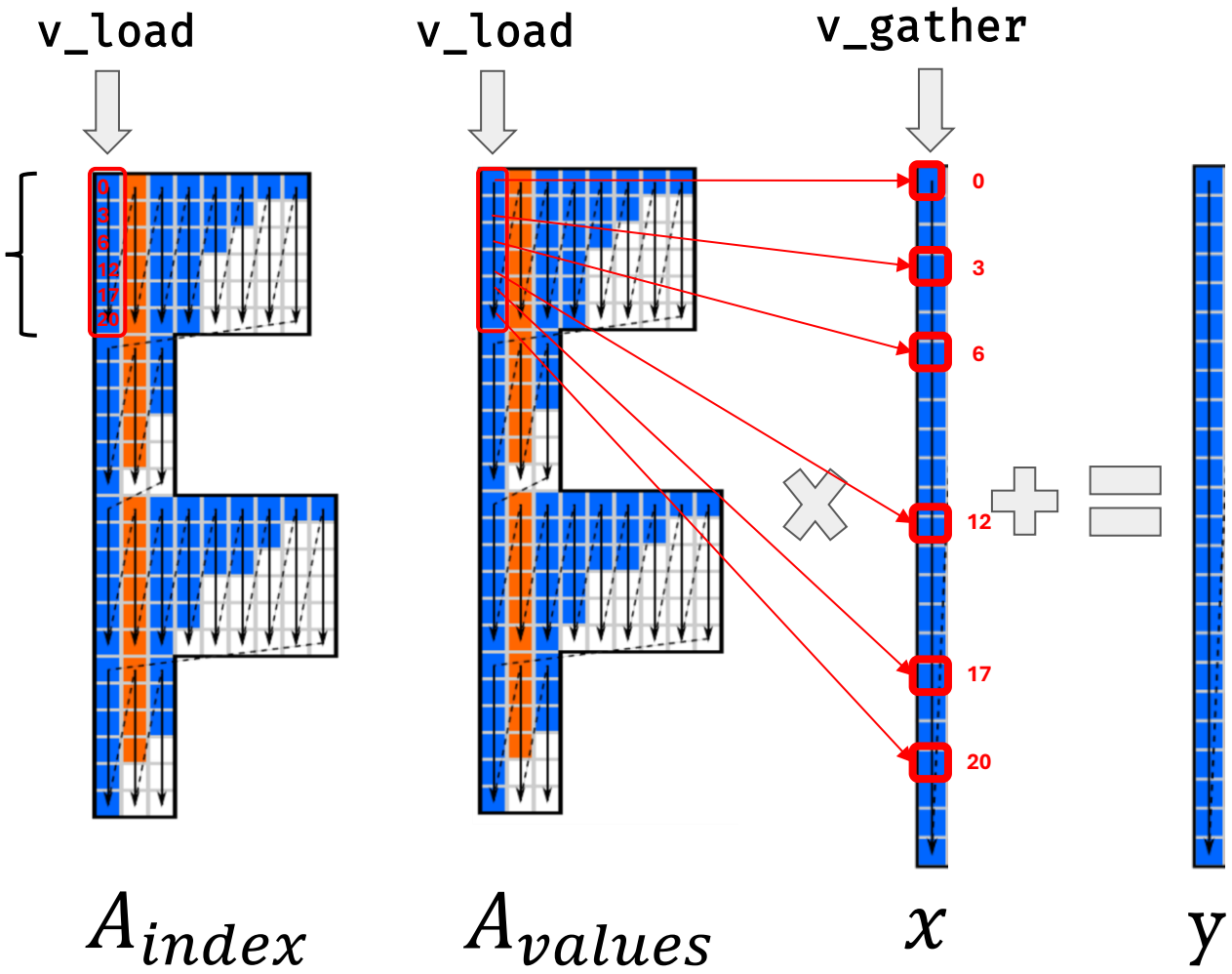


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization

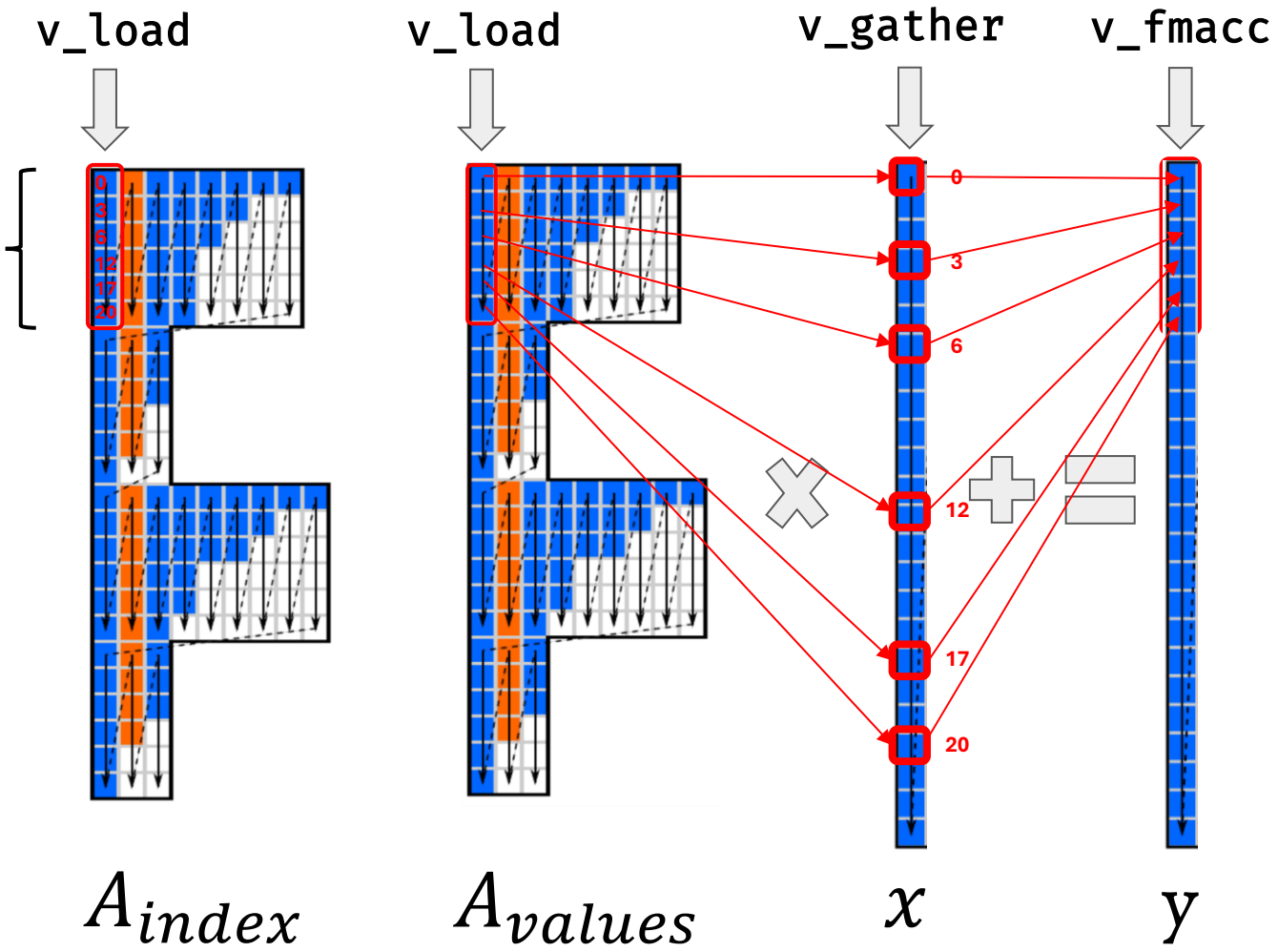


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization

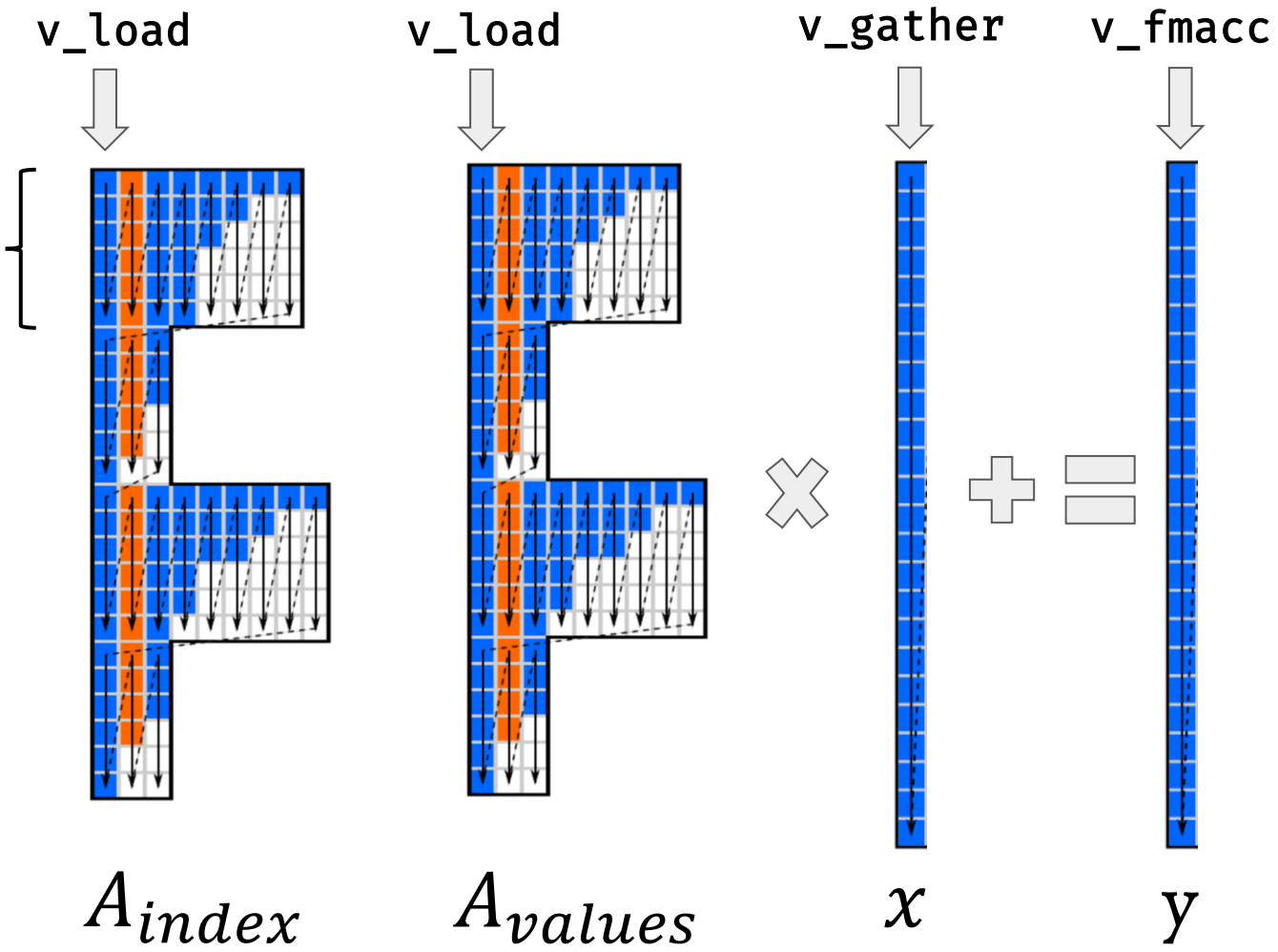


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization

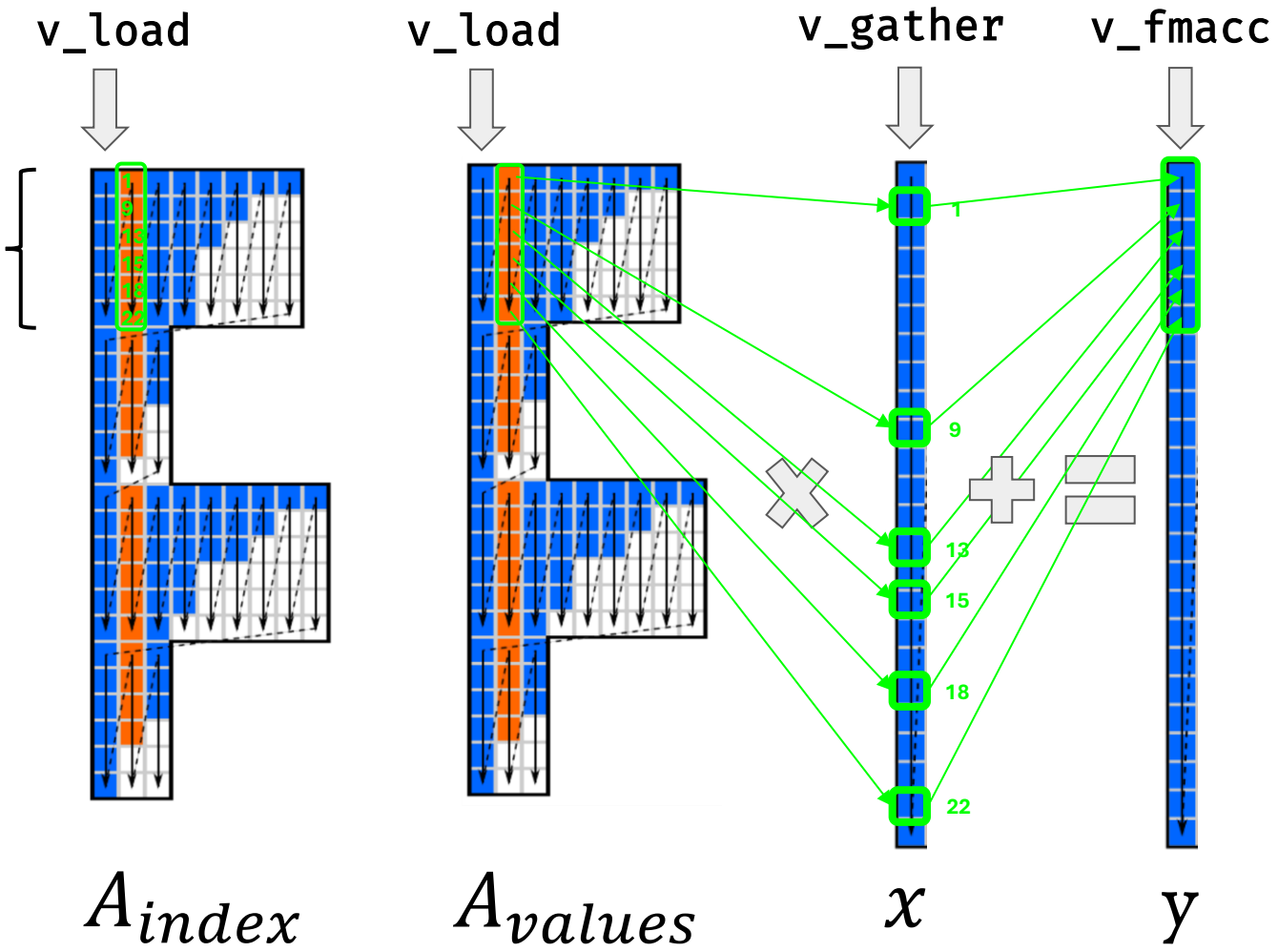


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization

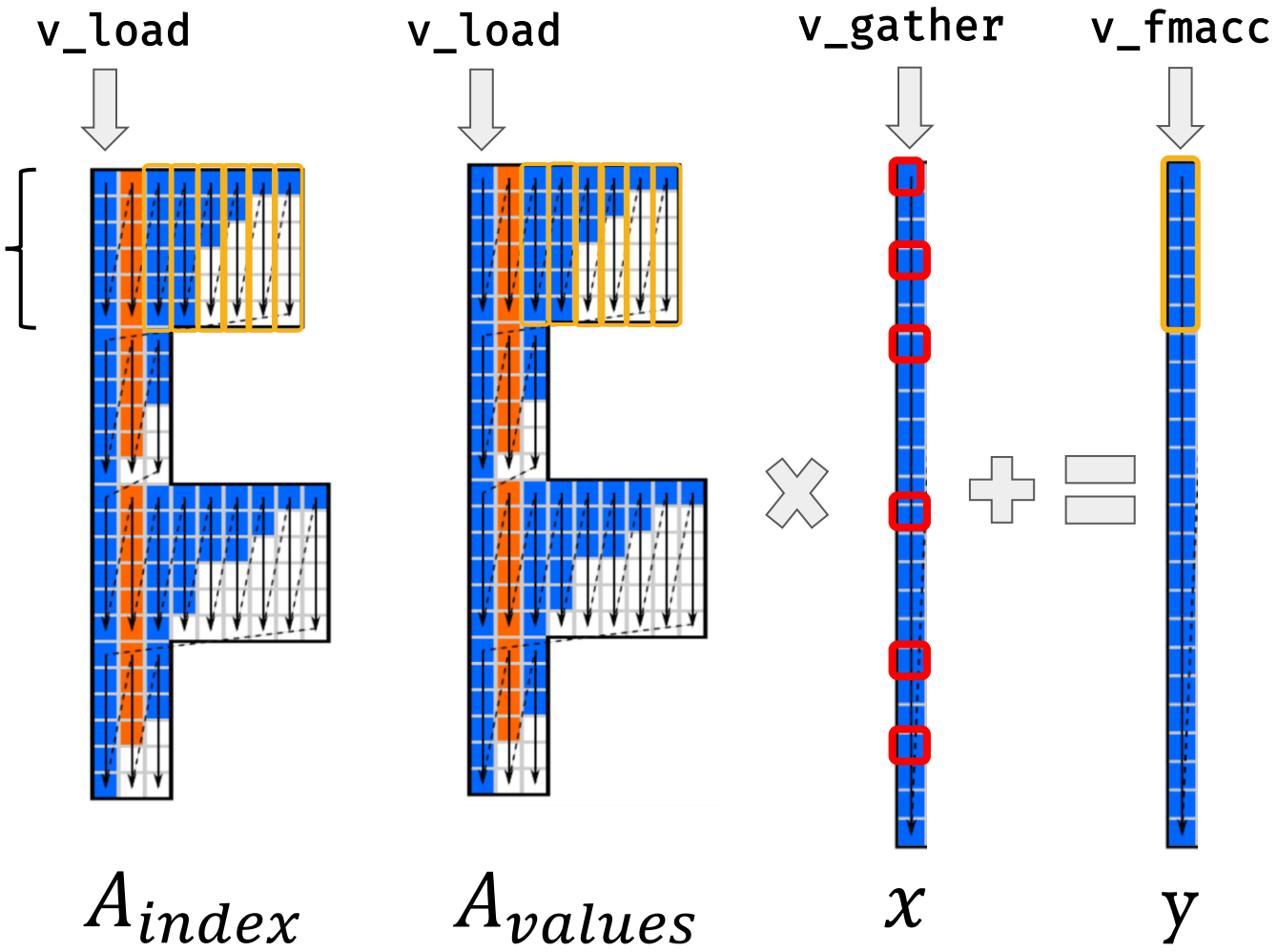


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization

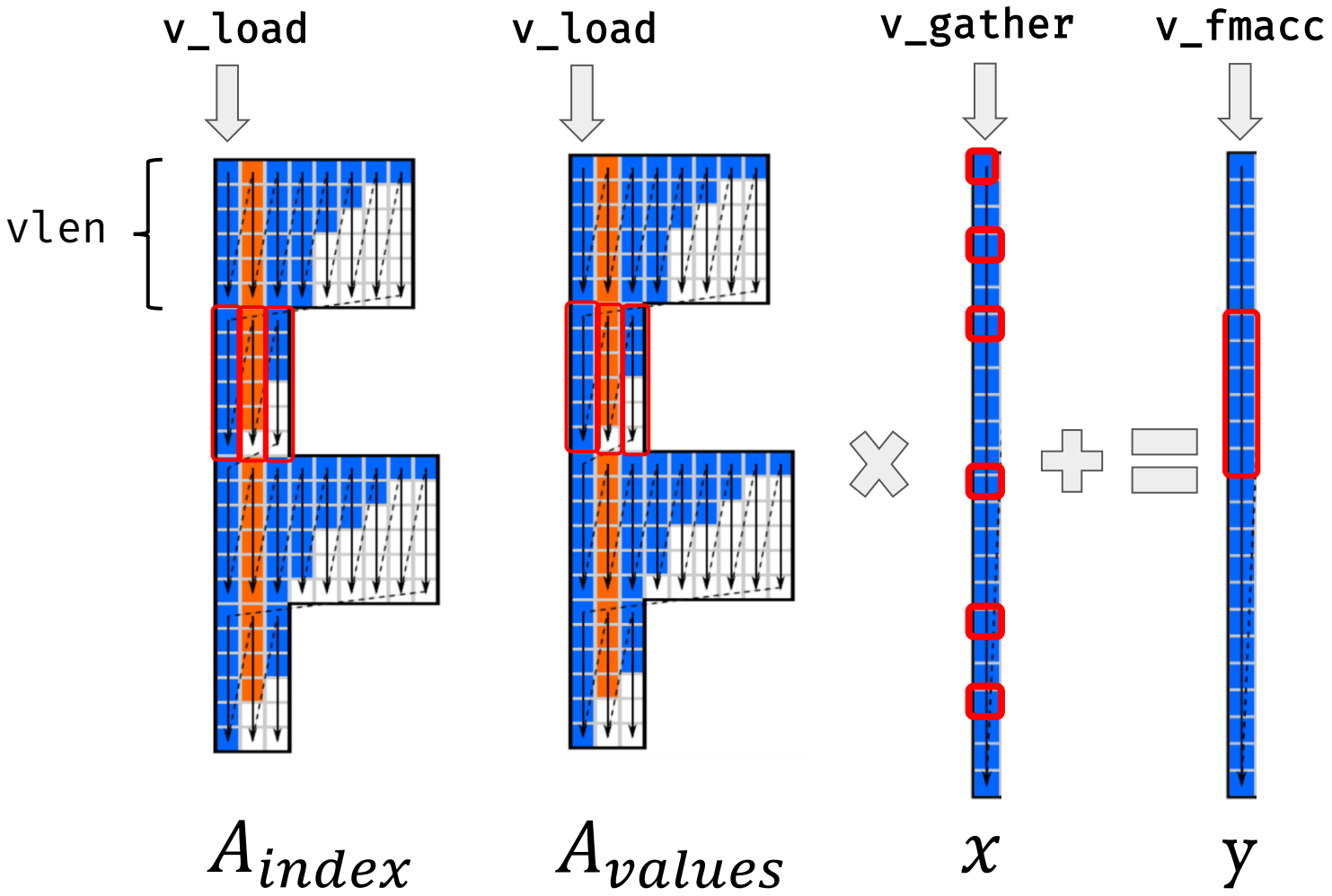


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization

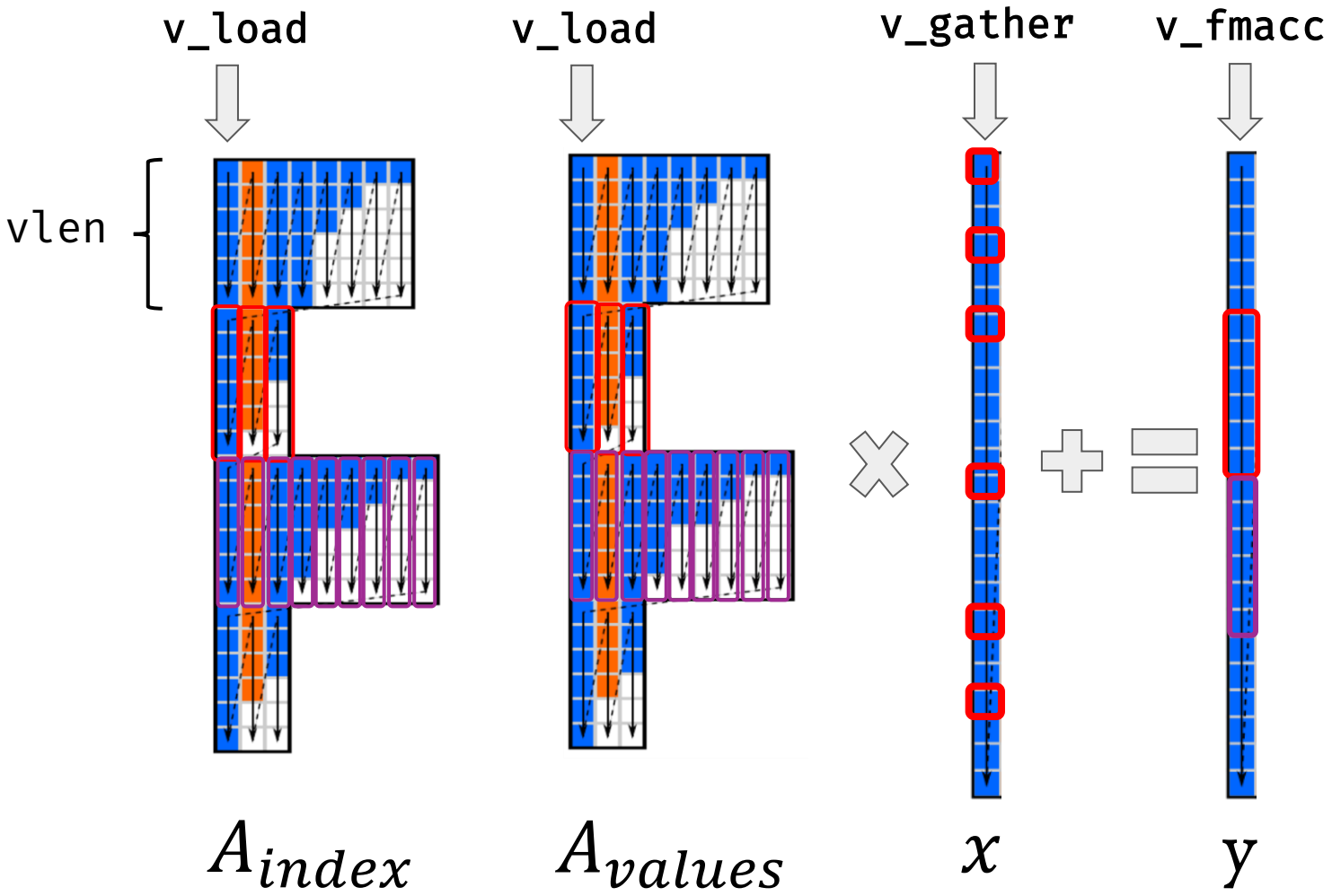


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization

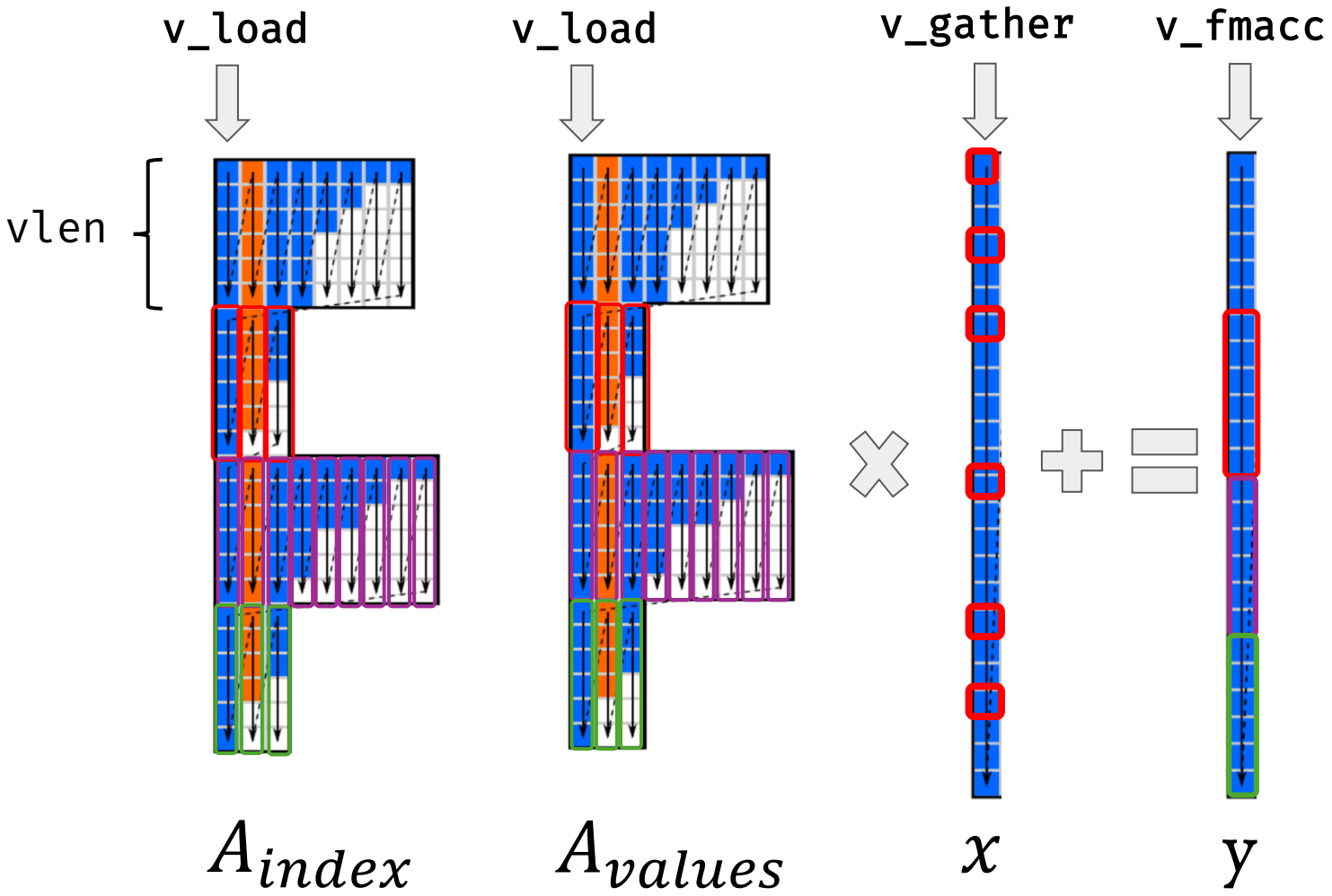


- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

SpMV SELL-C-σ vectorization



- A_{index}, A_{values} are stored by slice and within the slide by column
- Contiguous and predicable vector loads A_{values} and A_{index}
- No race conditions on y updates
- Especially in long vectors architectures (e.g. EPAC with `vlen=256`) we get huge benefits by hiding memory latency and better use of bandwidth during vector loads and stores [*]

M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM J. Sci. Comput.*, 2014.

[*] Pablo Vizcaino et al. (2023) Short reasons for long vectors in HPC CPUs: a study based on RISC-V https://doi.org/10.1145/3624062.3624231

RISC-V Vector target architectures

Sophon - SG2044

(XuanTie C920v2 core) RVV 1.0

- 64-core, general purpose server
- 12 stage out-of-order multiple issues superscalar pipeline design
- **128-bit vector registers (2 DP elements)**
- 64 kB L1, 2 MB L2, 64 MB L3 data cache
- Target: Commercial general-purpose servers
- **Short-vector architecture**

F. Mantovani et al. (2026) EPAC: The Last Dance
https://doi.org/10.1145/3801488.3806242

RISC-V Vector target architectures

Sophon - SG2044

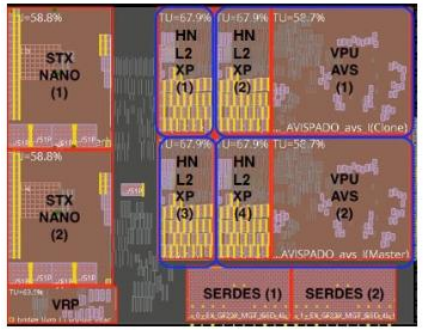
(XuanTie C920v2 core) RVV 1.0

- 64-core, general purpose server
- 12 stage out-of-order multiple issues superscalar pipeline design
- **128-bit vector registers (2 DP elements)**
- 64 kB L1, 2 MB L2, 64 MB L3 data cache
- Target: Commercial general-purpose servers
- **Short-vector architecture**

EPI – EPAC 1.5

(VEC tile) RVV 0.7.1

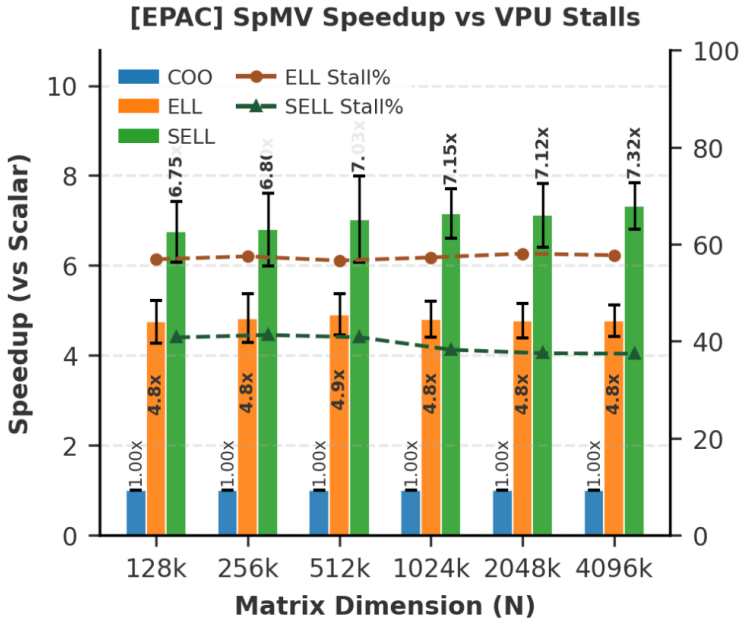
- Prototypal vector accelerator
- Scalar in-order RISC-V Avispado core + Vector processing unit (VPU)
- **16384-bit vector registers (256 DP elements)**
- 32 kB L1 (scalar core), 256 kB L2 data cache
- Target: Traditional HPC workloads, FP64
- **Long-vector architecture**

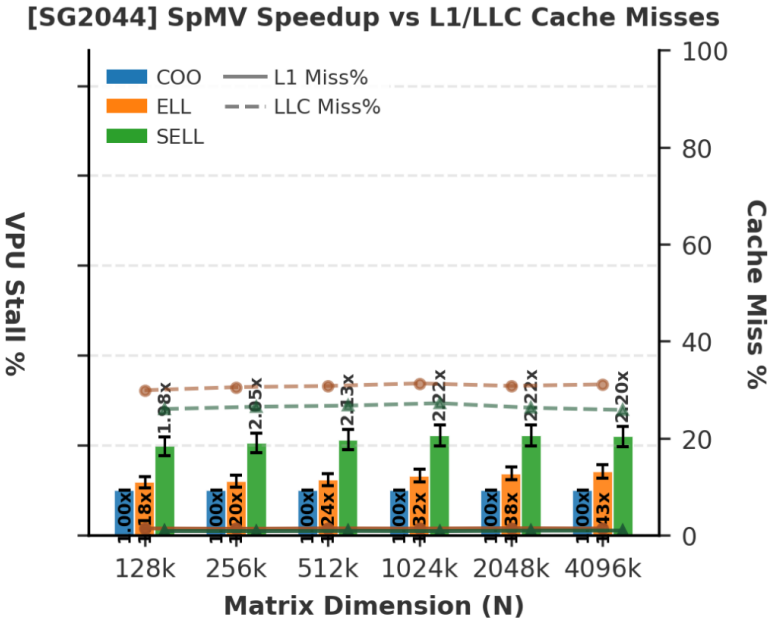


F. Mantovani et al. (2026) EPAC: The Last Dance
https://doi.org/10.1145/3801488.3806242

SpMV vectorization speedup

EPI – EPAC 1.5 (Long-vector)
vlmax=256



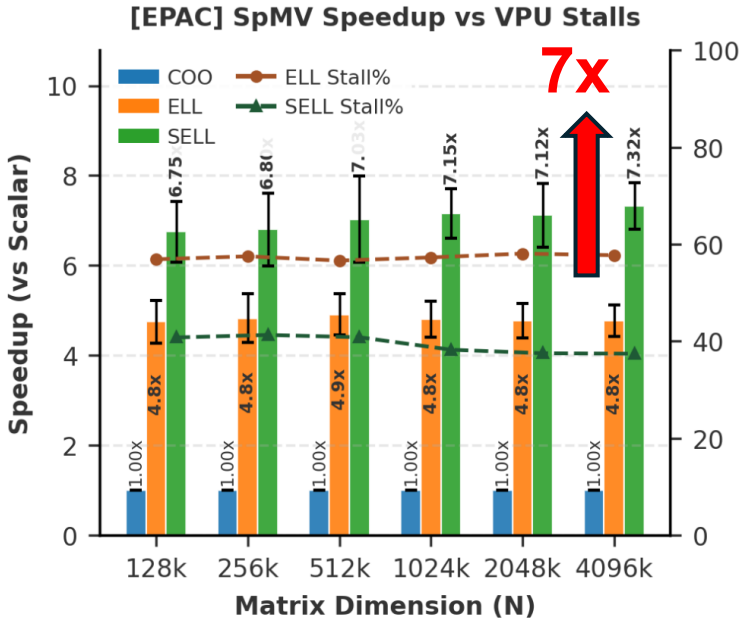


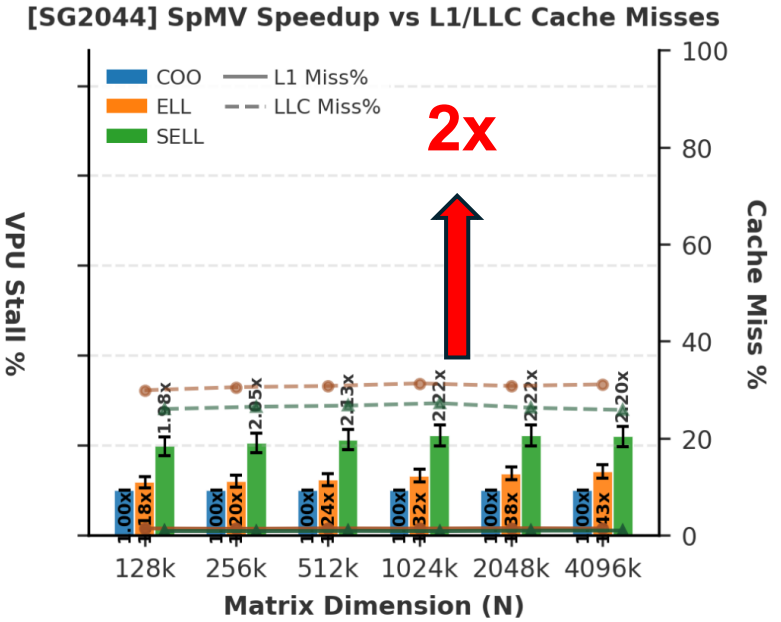
Sophon SG2044 (Short-vector)
vlmax=2

SpMV vectorization speedup

- 7x speedup on EPAC (long vector) and 2x on SG2044 (short vector)

EPI – EPAC 1.5 (Long-vector)
vlmax=256



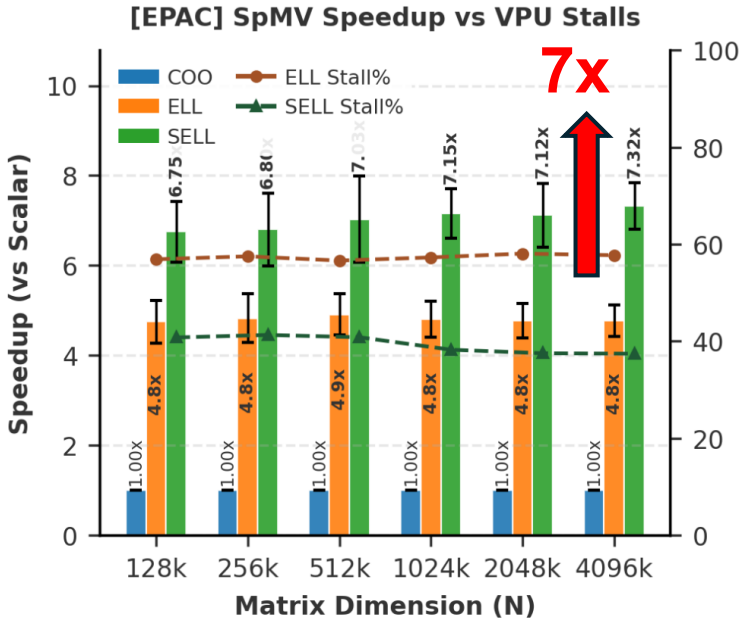


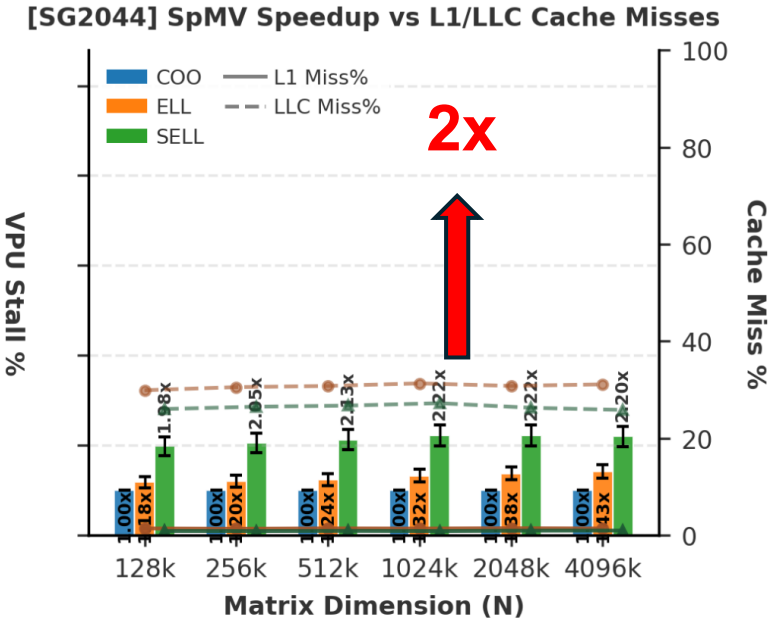
Sophon SG2044 (Short-vector)
vlmax=2

SpMV vectorization speedup

- 7x speedup on EPAC (long vector) and 2x on SG2044 (short vector)
- The long-vector approach proves to be very efficient for SpMV in SELL format.

EPI – EPAC 1.5 (Long-vector)
vlmax=256



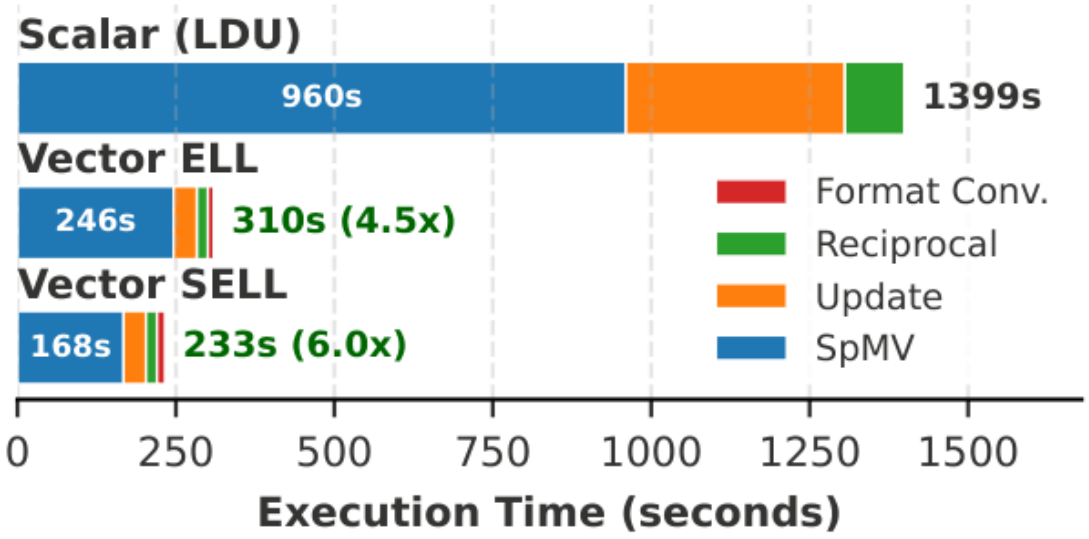


Sophon SG2044 (Short-vector)
vlmax=2

Smoother speedup

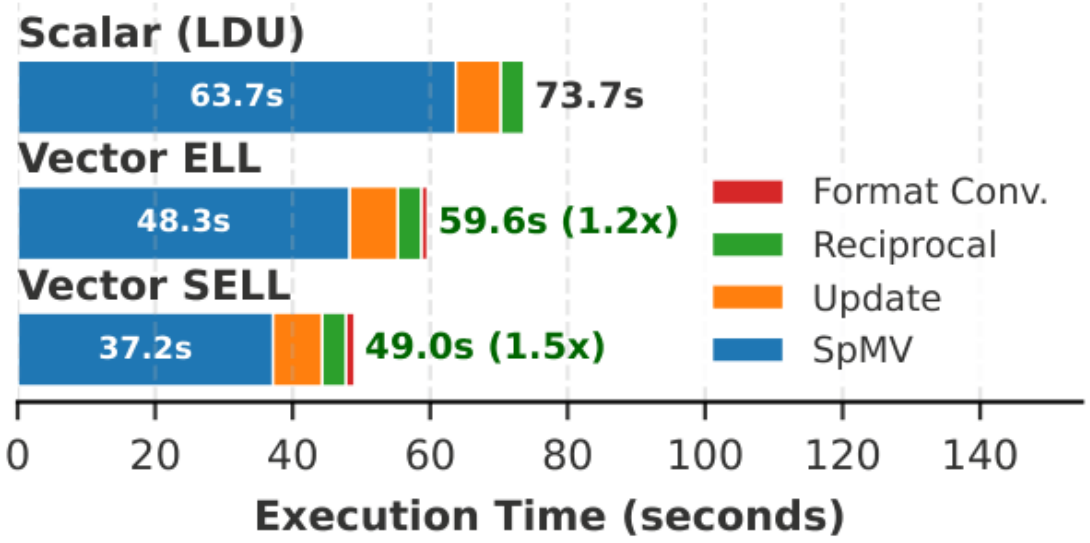
EPI – EPAC 1.5 (Long-vector)

vlmax=256

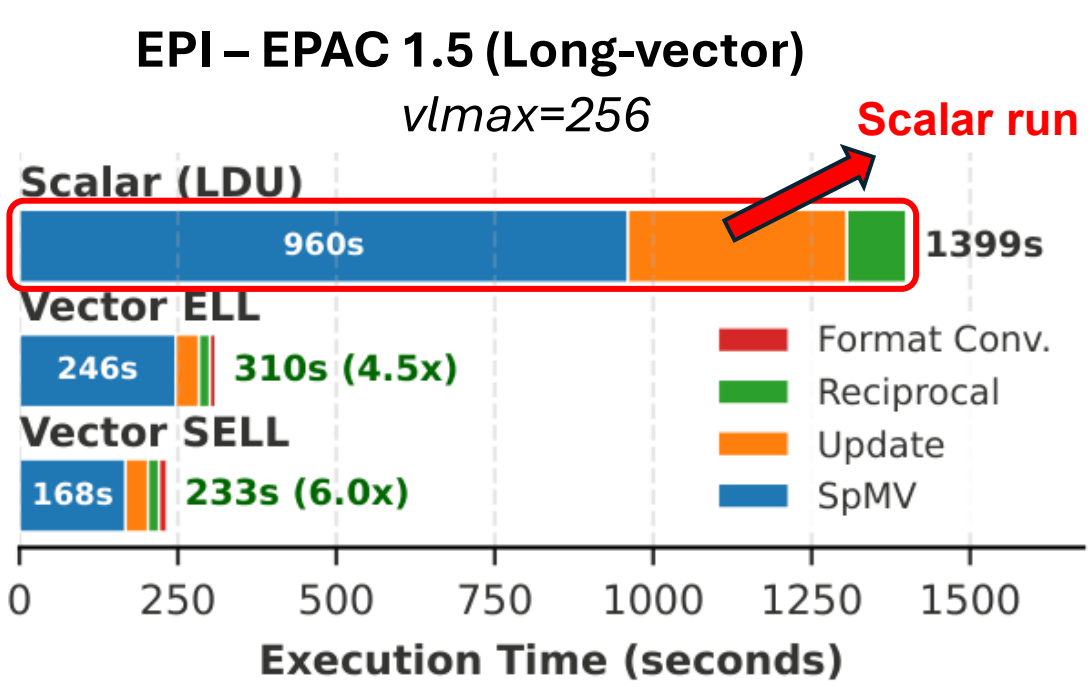


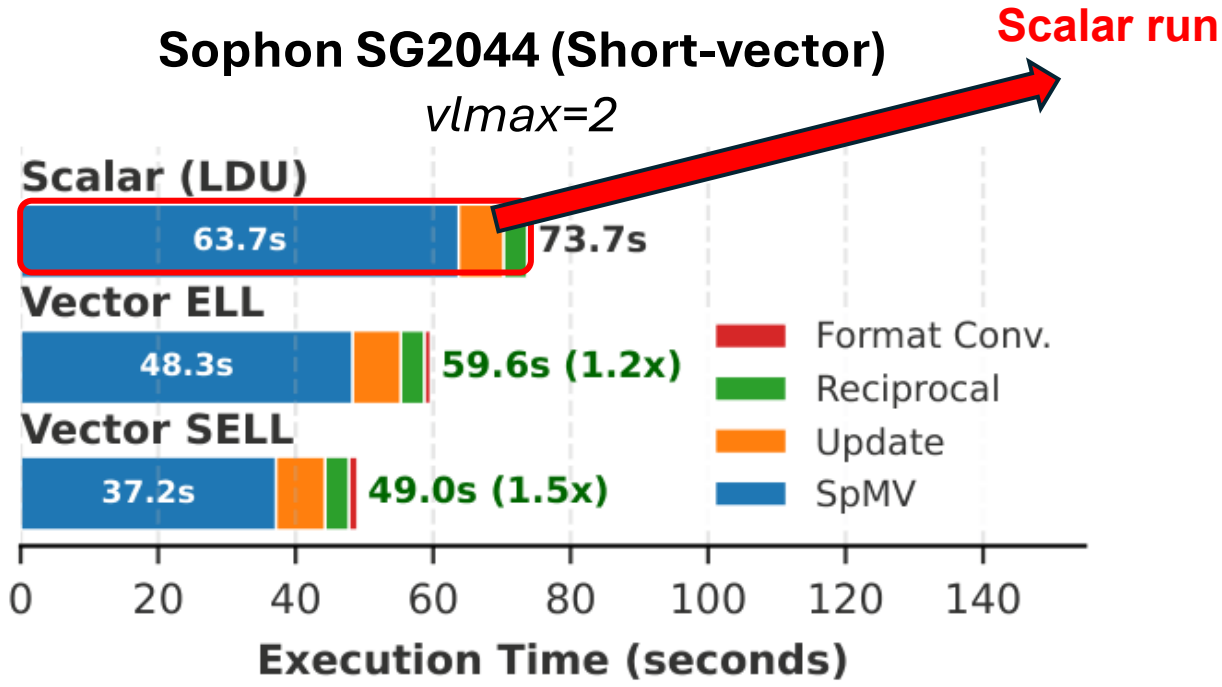
Sophon SG2044 (Short-vector)

vlmax=2



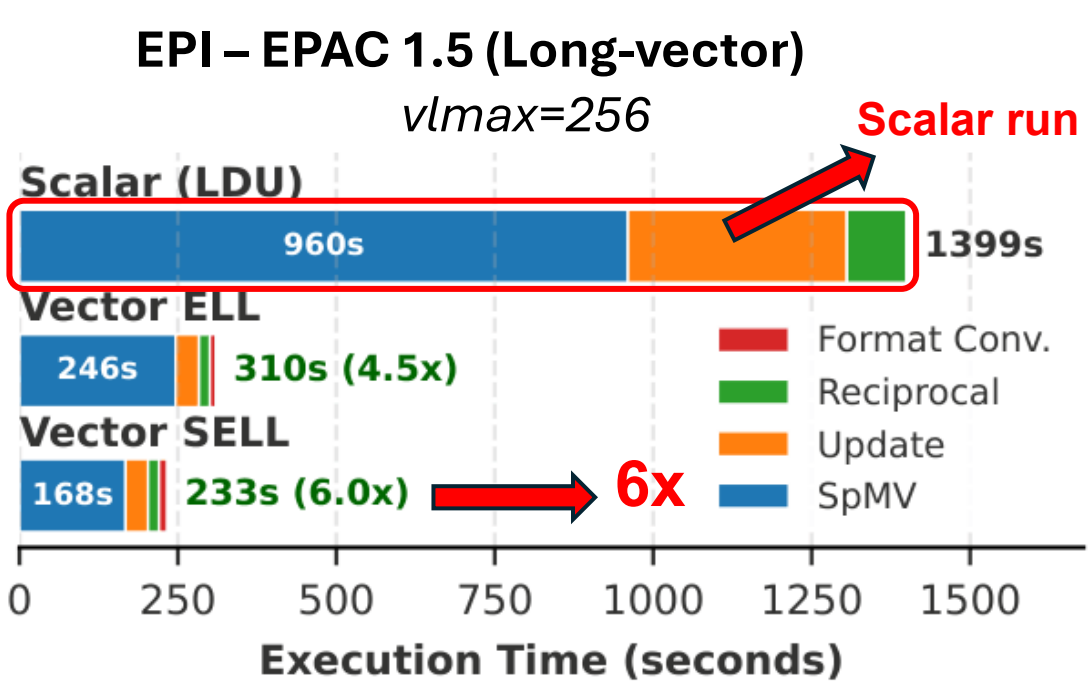
Smoother speedup

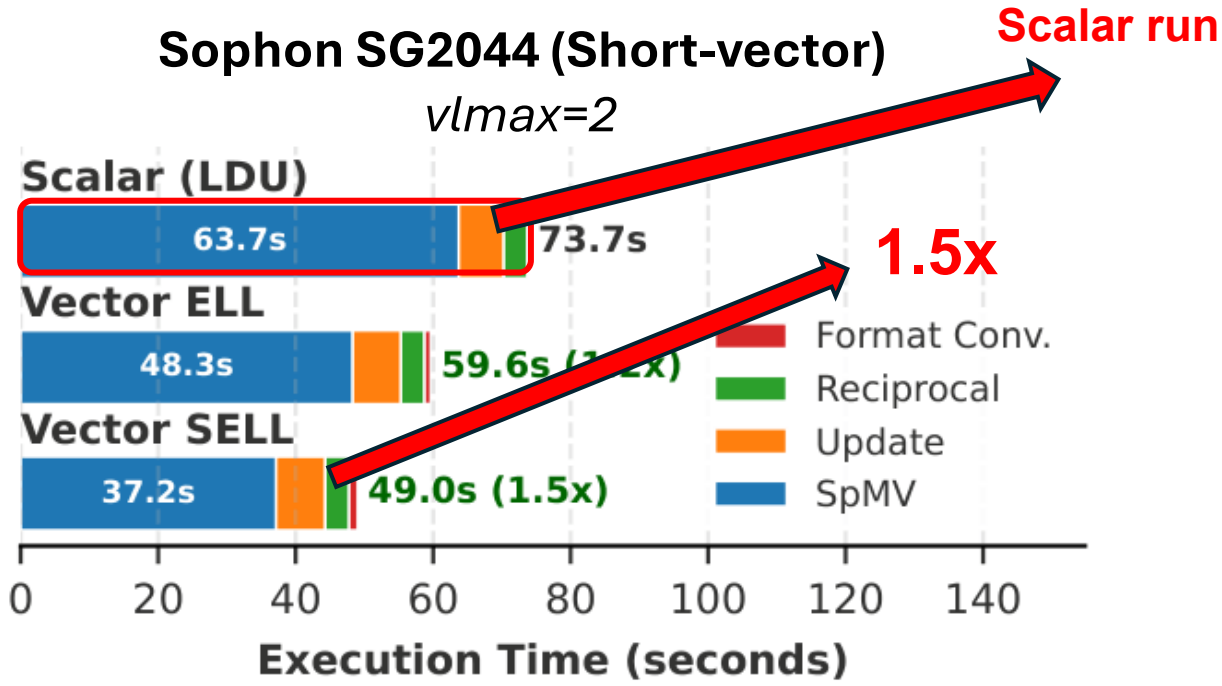




Smoother speedup

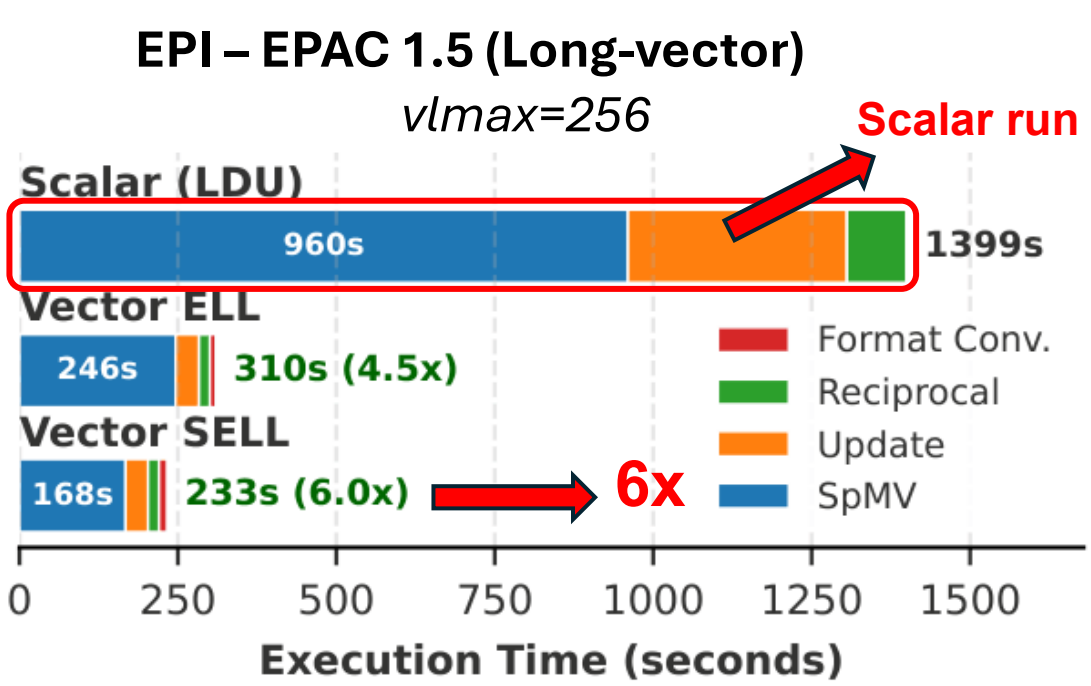
- 6x speedup on EPAC (long vector) and 1.5x on SG2044 (short vector)
- Marix format conversion is negligible (only 1-2% of total time)

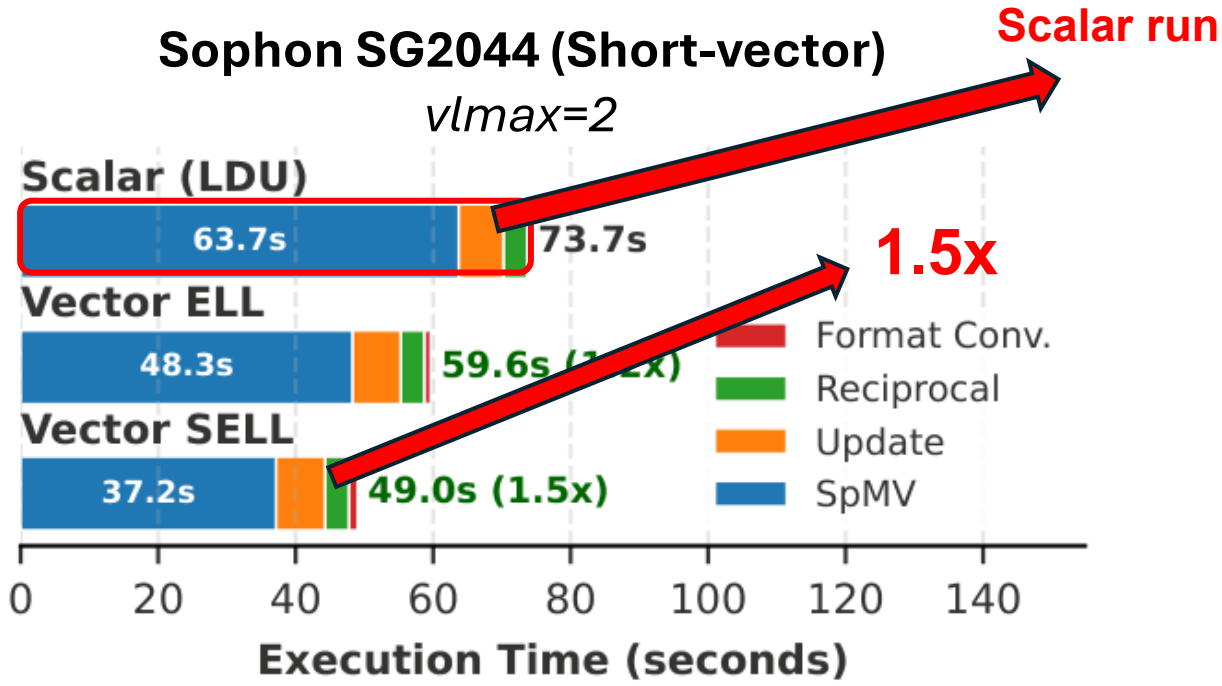




Smoother speedup

- 6x speedup on EPAC (long vector) and 1.5x on SG2044 (short vector)
- Marix format conversion is negligible (only 1-2% of total time)
- The long-vector approach proves to be more efficient for both sparse kernels (SpMV) and dense kernels.





Conclusions and future works

E. Venieri et al. (2026) Monte Cimone v3: Where RISC-V Stands in High-Performance Computing https://arxiv.org/abs/2605.22831

Conclusions and future works

Conclusions

- Non-invasive OpenFOAM optimization on vector architectures has been proved using run-time format conversion.

E. Venieri et al. (2026) Monte Cimone v3: Where RISC-V Stands in High-Performance Computing https://arxiv.org/abs/2605.22831

Conclusions and future works

Conclusions

- Non-invasive OpenFOAM optimization on vector architectures has been proved using run-time format conversion.
- The effectiveness of long vector architectures on memory-bound kernels and CFD solvers has been proven.

E. Venieri et al. (2026) Monte Cimone v3: Where RISC-V Stands in High-Performance Computing https://arxiv.org/abs/2605.22831

Conclusions and future works

Conclusions

- Non-invasive OpenFOAM optimization on vector architectures has been proved using run-time format conversion.
- The effectiveness of long vector architectures on memory-bound kernels and CFD solvers has been proven.

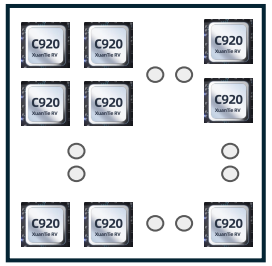
Next steps

Strong scaling study

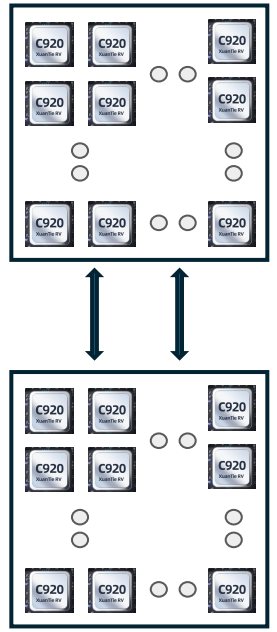
Single-core **Multi-core** **Multi-node**

XuanTie C920v2

Sophon SG2044 (64 core)



Monte Cimone v3 RISC-V Cluster (2x SG2044 nodes)



E. Venieri et al. (2026) Monte Cimone v3: Where RISC-V Stands in High-Performance Computing https://arxiv.org/abs/2605.22831

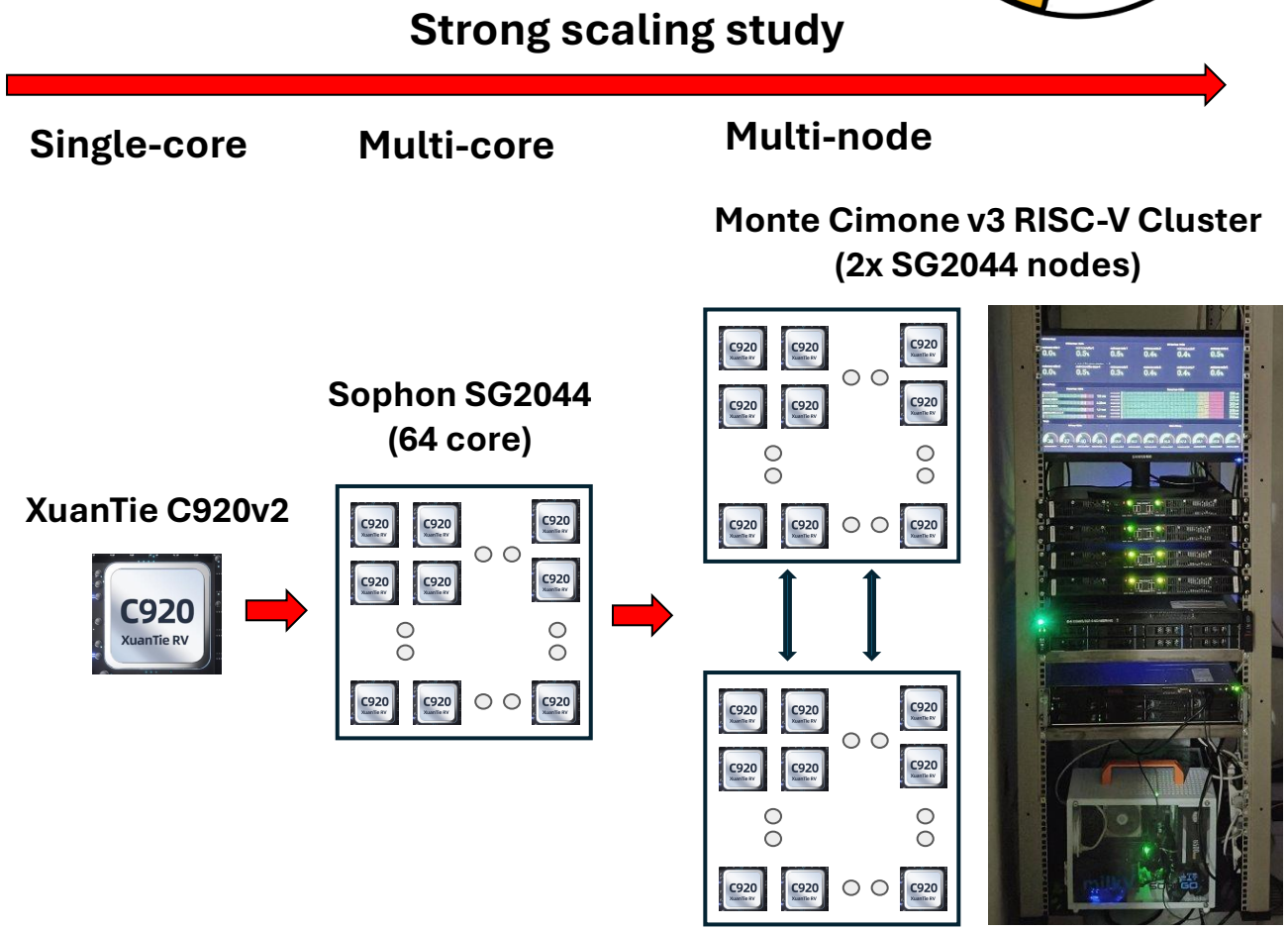
Conclusions and future works

Conclusions

- Non-invasive OpenFOAM optimization on vector architectures has been proved using run-time format conversion.
- The effectiveness of long vector architectures on memory-bound kernels and CFD solvers has been proven.

Next steps

- Complete the vectorization of other components of the multigrid solver



E. Venieri et al. (2026) Monte Cimone v3: Where RISC-V Stands in High-Performance Computing https://arxiv.org/abs/2605.22831

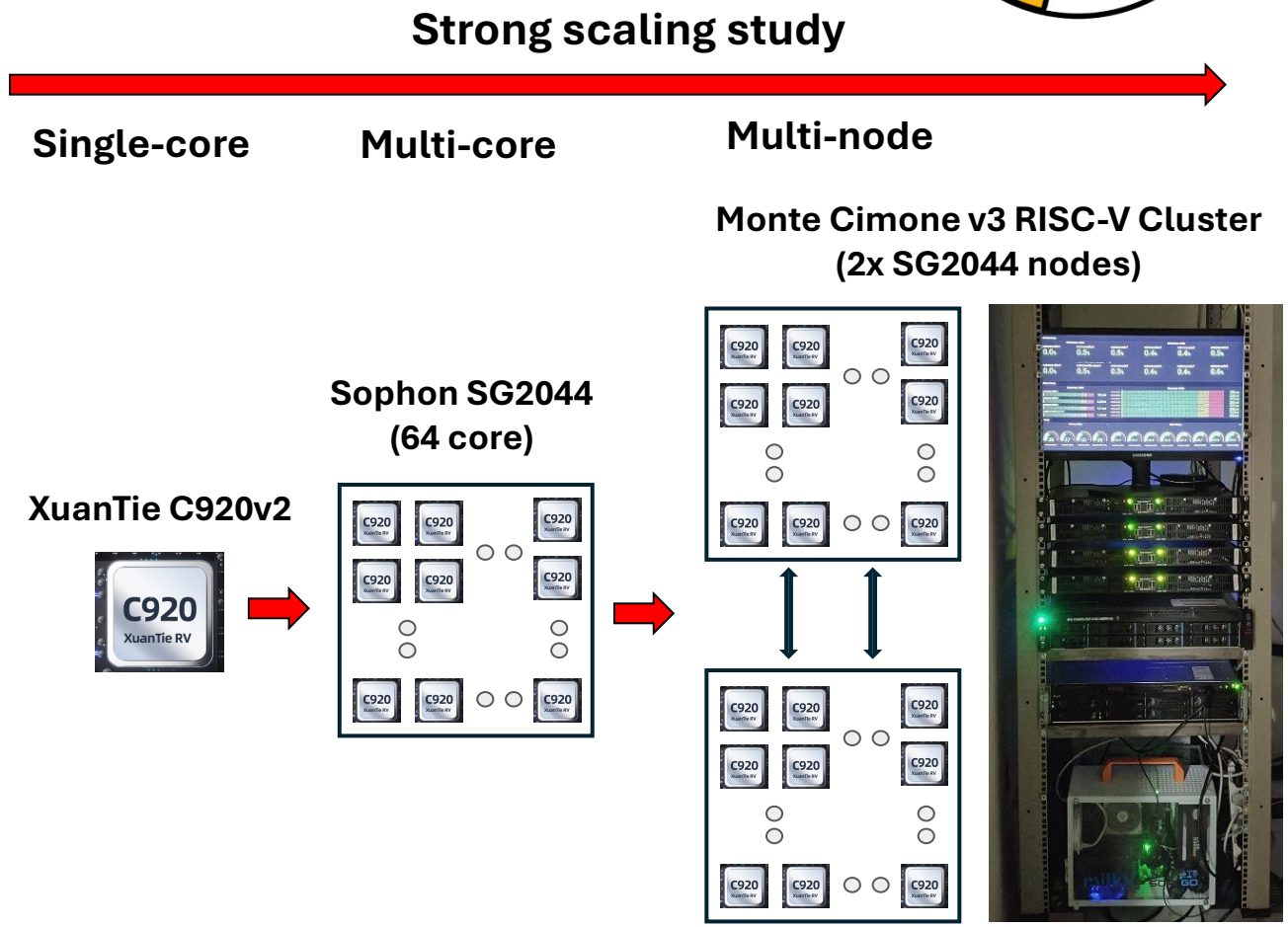
Conclusions and future works

Conclusions

- Non-invasive OpenFOAM optimization on vector architectures has been proved using run-time format conversion.
- The effectiveness of long vector architectures on memory-bound kernels and CFD solvers has been proven.

Next steps

- Complete the vectorization of other components of the multigrid solver
- Strong scaling scalability study using Monte Cimone HPC cluster



E. Venieri et al. (2026) Monte Cimone v3: Where RISC-V Stands in High-Performance Computing https://arxiv.org/abs/2605.22831