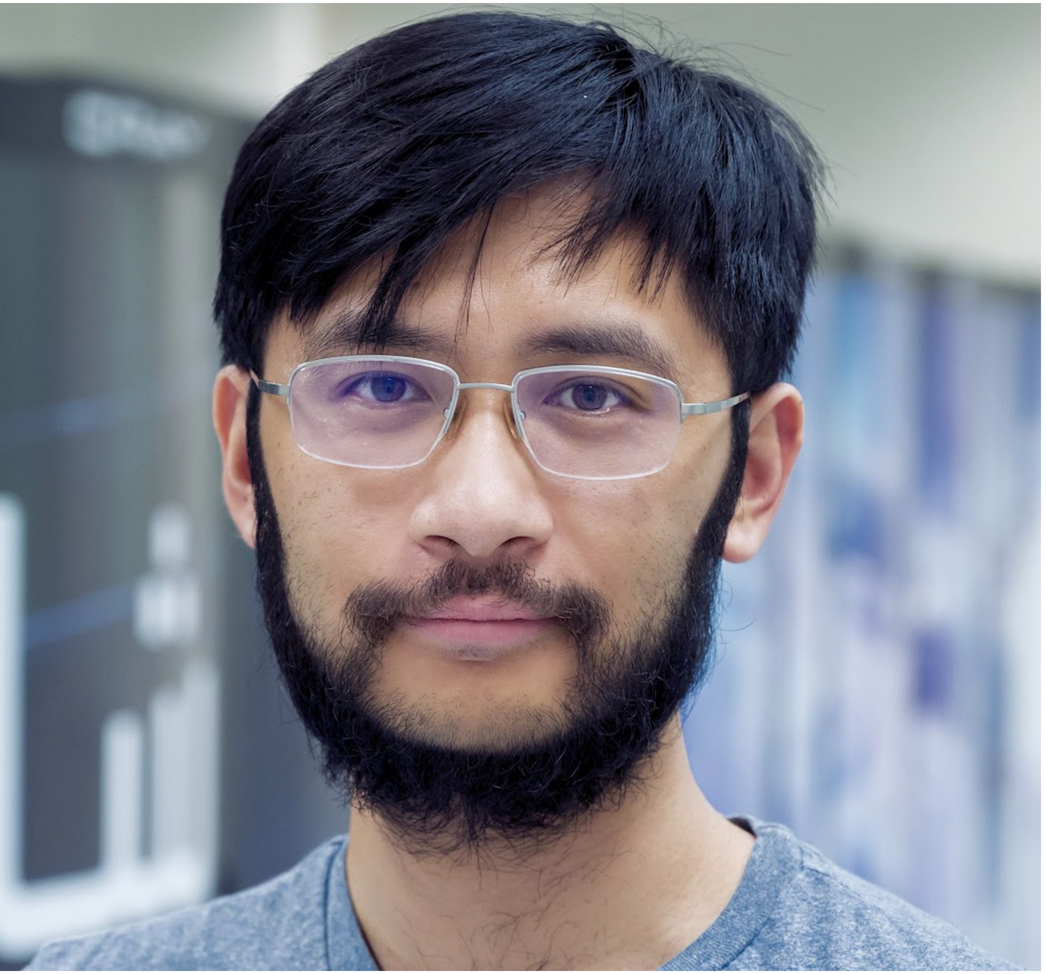
Exploiting Data Sparsity in Scientific Applications

Yuxi Hong

LBNL

April 4th, 2024



Yuxi Hong

Education

- Tsinghua University, Beijing
  - BSc in Electrical Engineering
  - MS in Electronics Engineering
- KAUST, KSA
  - PhD in Computer Science

  (Advisor: David E. Keyes)

Research Interest

- Numerical Linear Algebra
- Tile Low Rank Algorithm Design
- GPU Computing

Highlights

Applications

Adaptive Optics System for Ground-based Telescopes

Seismic Redatuming by Inversion

Algorithms

Stochastic Levenberg-Marquardt Method (SLM)
Discrete time Algebraic Riccati Equation (DARE)
Tile Low-Rank Matrix Vector Multiplication (TLR-MVM)

Batched TLR-MVM
Mixed Precision TLR-MVM

Softwares

SHIPS

DARE

TLR-MVM

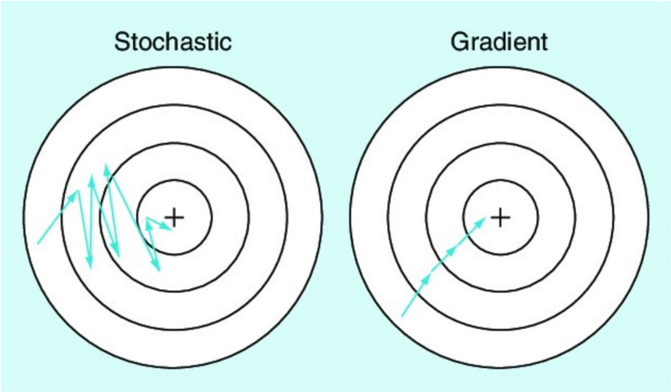
TLR-MDC

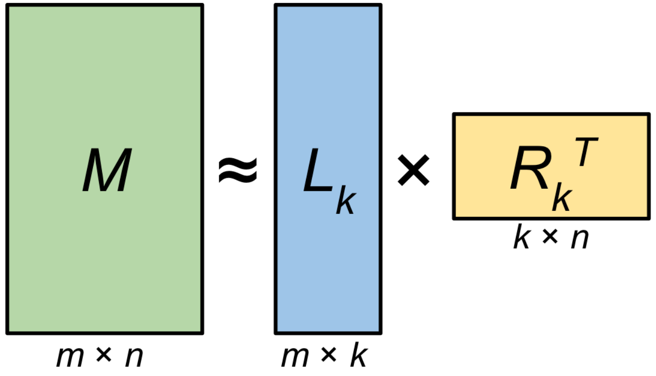
Architectures

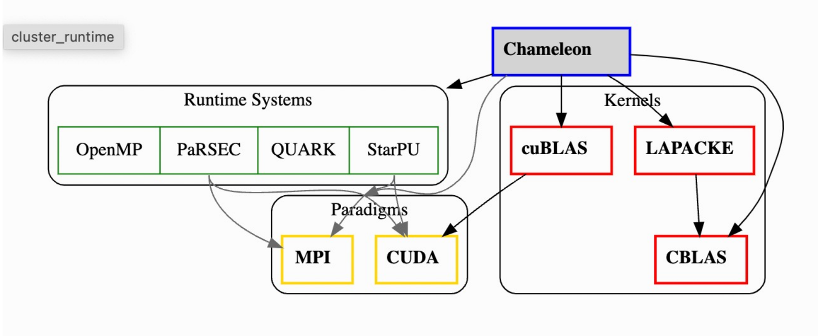
Content

- Introduction
- Adaptive Optics for Ground-based Telescopes
  - Stochastic Levenberg-Marquardt Method in Soft Real-Time Controller
  - Discrete Algebraic Riccati Equation in Soft Real-Time Controller
  - TLR-MVM in Hard Real-Time Controller
- Seismic Redatuming by Inversion Using Marchenko-based Methods
  - Batched TLR-MVM
  - Mixed-Precision Batched TLR-MVM on NVIDIA GPU

Algorithmic Solutions for HPC Applications







Stochastic Algorithms

- Approximating gradient / Hessian by using subsampling methods to reduce per iteration cost
- Randomness increases possibility to reach the global minimum

Algebraic Compression

- Exploiting the exponential decay of singular values inside data matrices
- Deploying SVD-like algorithms to select k largest singular values / vectors up to application required accuracy to carry on computation

Linear Algebra Runtime System

- Hiding data movement by asynchronous execution in run time system
- Mixed precision computation in runtime system

Emerging new hardware and features

HBM Technology

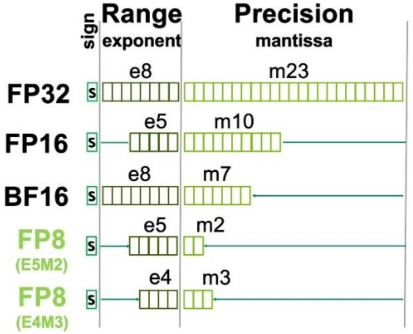
FUJITSU FX1000 ARM: 1 TB /s

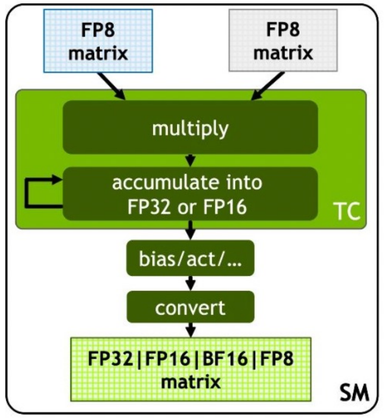
NEC Vector Engine: 1.2 TB/s

AMD Instinct MI 250: 3.2 TB/s

NVIDIA Hopper GPU: 3 TB/s

Low / Multi Precision

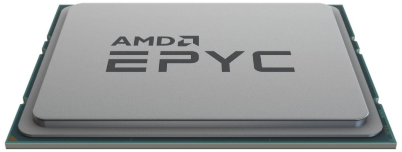




Huge Last Level Cache

AMD EPYC Processors

- Rome / Milan: up to 256 MB
- Milan-X: up to 768 MB
- LLC Bandwidth: 3.7 TB/s

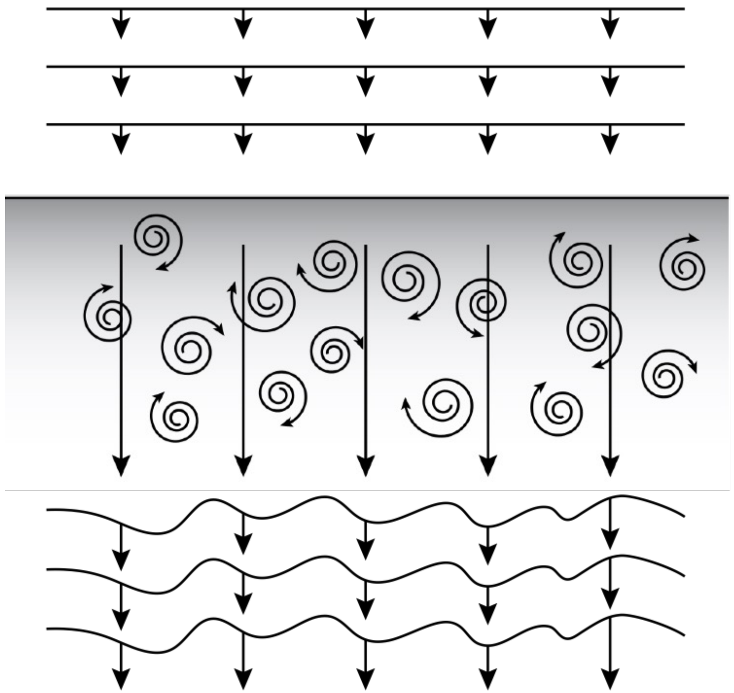


Static Random-Access Memory From AI accelerator

Content

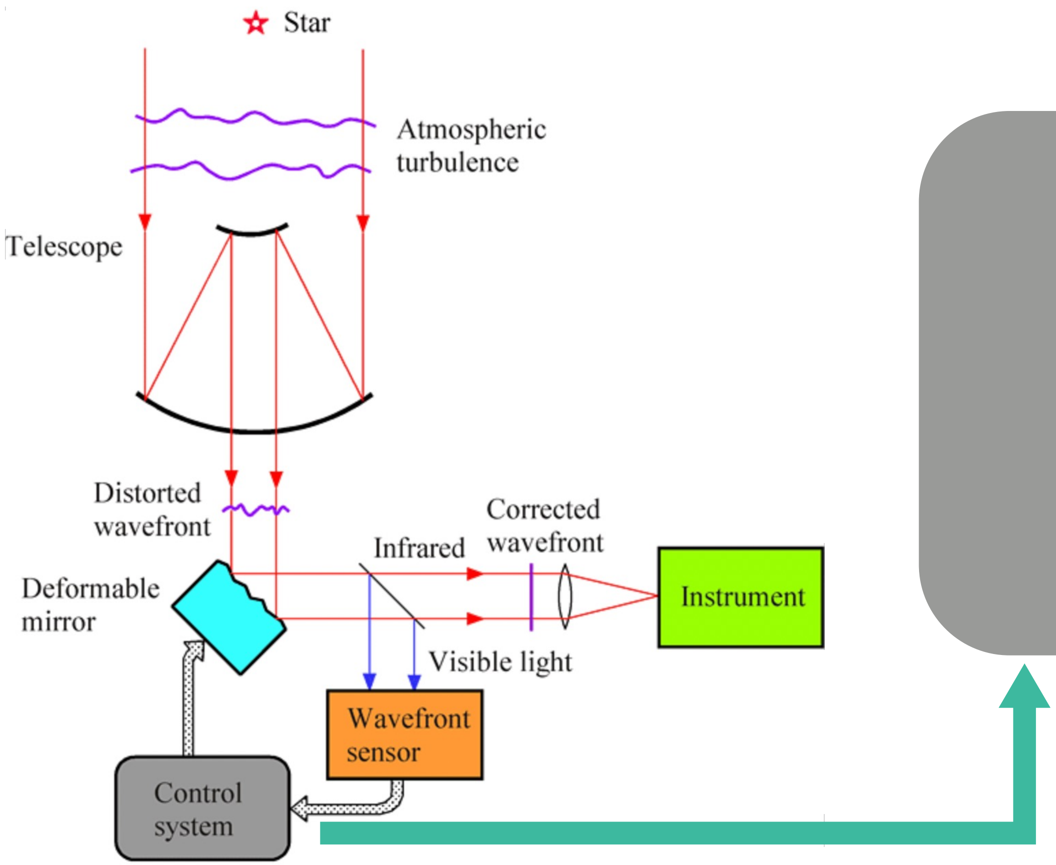
- Introduction
- Adaptive Optics for Ground-based Telescopes
  - Stochastic Levenberg-Marquardt Method in Soft Real-Time Controller
  - Discrete Algebraic Riccati Equation in Soft Real-Time Controller
  - TLR-MVM in Hard Real-Time Controller
- Seismic Redatuming by Inversion Using Marchenko-based Methods
  - Batched TLR-MVM
  - Mixed-Precision Batched TLR-MVM on NVIDIA GPU

The Atmosphere Turbulence

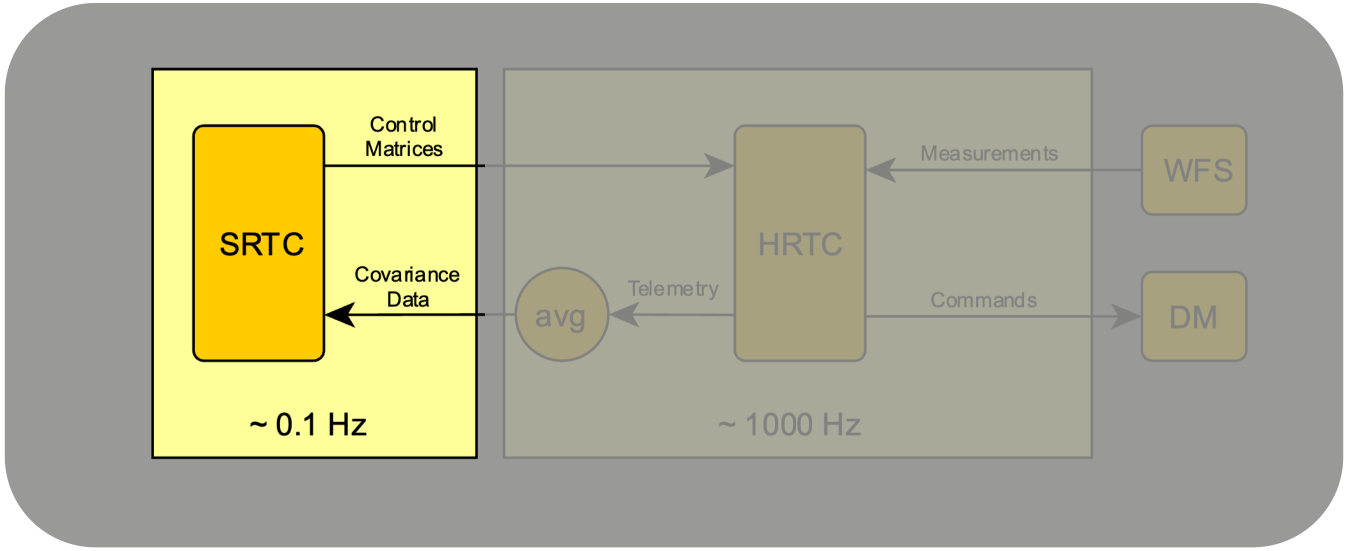


- Distorts the trajectory of light rays
- Reduces astronomy images quality

Adaptive Optics System: Soft Real-Time Controller and Hard Real-Time Controller



Control System

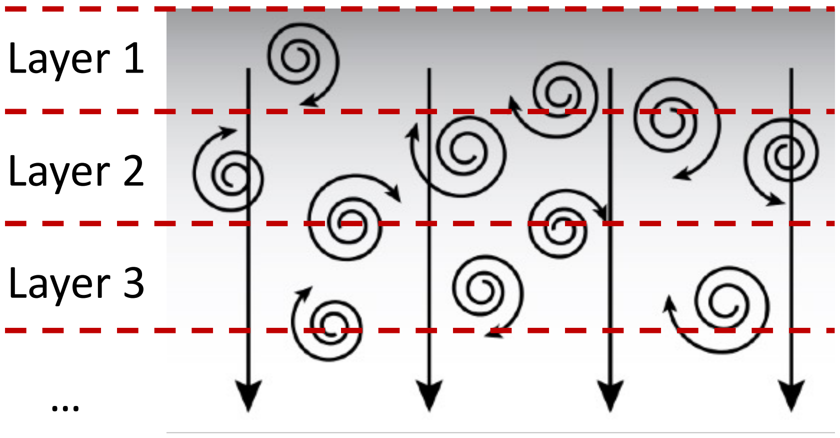


SRTC: Extract the atmosphere turbulence parameters

HRTC: Apply the correction

Closer Look at Soft Real-Time Controller

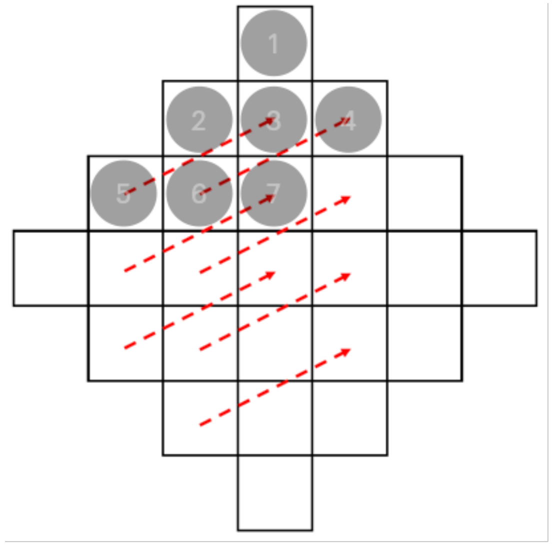
How to build model?



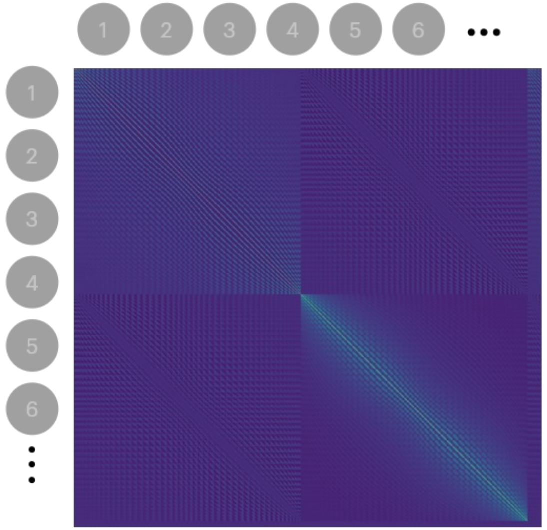
Atmosphere turbulence model

- We can split the turbulence eddies by several layers. Parameters of each layer is turbulence strength and bi-dimensional wind velocities.

Generation of covariance matrix



Simplified wavefront sensor array



Covariance matrix

- We can also get an expression of covariance matrix using WFS's physical locations and parameters.

Problem Definition and Stochastic Levenberg-Marquardt Method

$$F(x) = vec(C_{i,j}^{ana}(x) - C_{i,j}^{num})$$

# of elems ≈ 40k x 40k

10 layer analytical atmosphere model suggested by European Southearn Observatory:
- Trubulence strength,
- Bi-dimentional wind velocity

30 parameters in all

Measurement numerical data is collected accumulatively every 15s to 30s to get higher enough signal-noise ratio.

Algorithm 1 SLM algorithm with fixed regularization.

Initialization: : Choose initial x^0. Choose a constant μ. Generate a random index sequence $Rand_{seq}$. Choose a data fraction df to decide the samples size used per iteration. Record initial $\|\tilde{g}^0\|_\infty$. Choose stopping criteria ϵ.

1: **for** $k = 1, 2, ..., K$ **do**
2: Get random indices from $Rand_{seq}$
3: Calculate ${J^k}^\top J^k$ and $\tilde{g}^k$ using GPU
4: If $\|\tilde{g}^k\|_\infty \,/\, \|\tilde{g}^0\|_\infty \leq \epsilon$, exit algorithm
5: Solve linear system: $({J^k}^\top J^k + \mu I)\delta x = \tilde{g}^k$
6: Update x^{k+1}: $x^{k+1} = x^k - ({J^k}^\top J^k + \mu I)^{-1}\tilde{g}^k = x^k - \delta x$
7: **end for**

Approximated Hessian *Fixed Regularization 1e-6*

Problem definition

$$\min_{x \in R^d} f(x) \coloneqq \frac{1}{2}||F(x)||^2 = \frac{1}{2}\sum_{i=1}^{N} F_i(x)^2$$

$$J^k = S^k \nabla F(x^k) \qquad \tilde{g}^k = \nabla F(x^k) S^k F^k$$

Stochastic Jacobian vector *Stochastic gradient vector*

STOA: Levenberg-Marquardt Method

Our SLM leverages data sparsity of the matrix and use a sub-sampling method to solve the problem. It randomly selects items inside Covariance matrix to form the approximated gradient and Hessian.

Theory

Assumption 1. *($\mathcal{L}$-smoothness) The function f is $\mathcal{L}$ smooth if its gradient is $\mathcal{L}$-Lipschitz continuous, that is, for all $x, y \in \mathbb{R}^d$,*

$$f(x) \leq f(y) + \nabla f(y)^\top (x-y) + \frac{\mathcal{L}}{2}\|x-y\|^2.$$

Assumption 2. *(Boundness of stochastic gradient) The stochastic gradient is bounded by $G > 0$, that is,*

$$\mathbb{E}\left[\|\tilde{g}^k\|\right]^2 \leq G^2.$$

The latter inequality implies $\mathbb{E}\left[\|\tilde{g}^k\|\right] \leq G$ (using Jensen's inequality).

Assumption 3. *The function F Jacobian is bounded by $\kappa_J > 0$, that is,*

$$\mathbb{E}\left[\|\nabla F(x^k)\|\right] \leq \kappa_J.$$

The latter Assumption implies $\mathbb{E}\left[\|J^k\|\right] \leq \kappa_J$, independently from S^k.

★ **Theorem 1.** *Let Assumptions 1, 2 and 3 hold. Let $K > 0$, $\mu_0 > 0$ and $\mu = \mu_0\sqrt{K+1}$, then*

$$\frac{\sum_{k=0}^{K} \mathbb{E}\|\nabla f^k\|^2}{K+1} \leq \frac{C}{\sqrt{K+1}},$$

where

$$C := \mu_0 f^0 + \frac{2\kappa_J^2 G^2 + \mathcal{L}G^2}{2\mu_0}.$$

★ **Corollary 1.** *Let Assumptions 1, 2 and 3 hold. Let $K > 0$, $\mu_0 > 0$ and $\mu = \mu_0\sqrt{K+1}$, then*

$$\boxed{\min_{k\in\{0,\ldots,K\}} \mathbb{E}[\|\nabla f^k\|^2] \leq \mathcal{O}\left(\frac{1}{\sqrt{K+1}}\right).}$$

- The Minimum of the gradient norm square over iterations is of the order of $O(\frac{1}{\sqrt{K+1}})$, which is the classical complexity bound know for SGD and its variants.

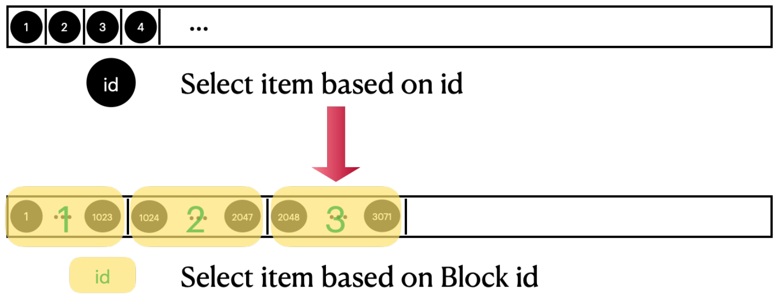
Implementation

- **Stochastic Hessian and Gradient Kernel Design (the most time consuming kernel in SLM)**

  We design stochastic HG kernel to compute approximate Hessian and gradient. Each CUDA thread is responsible for one sample in the optimization problem. We get numerical Jacobian using finite difference approximation.

- **Block Random Index**

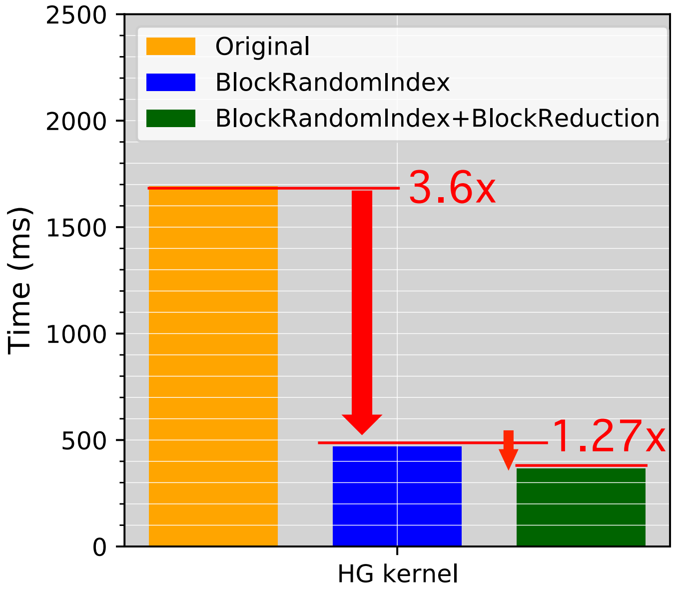
  If we select index randomly, we will have irregular memory access issue. We group the index together and select the Index by group id to have coalesced memory access pattern.



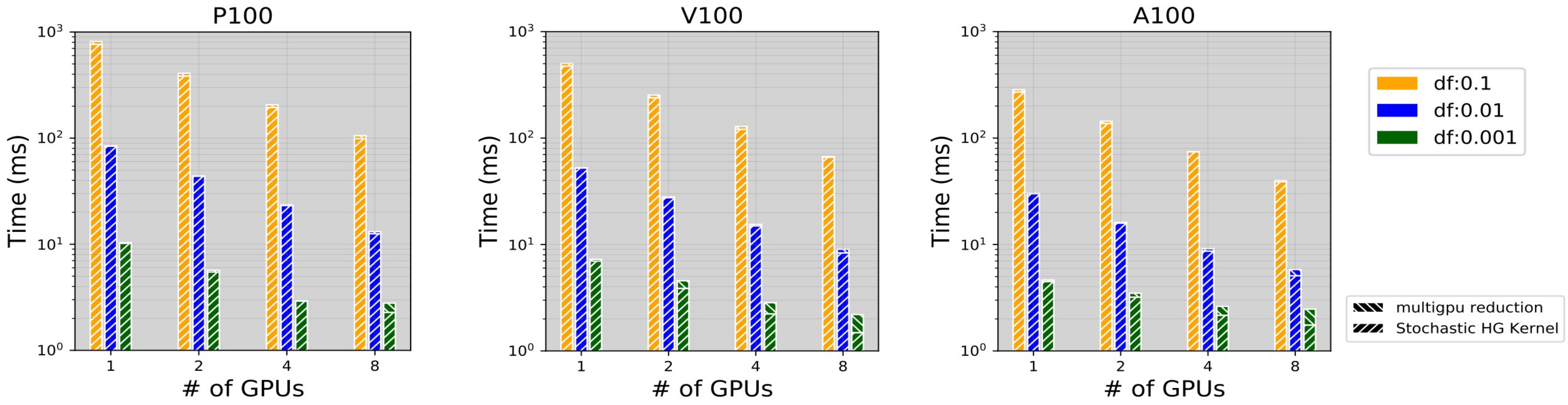
- **Reduction Optimization**

  We use NVIDIA cub library to perform block-level reduction. Then we use atomic operation for global reduction.

Performance



Performance Decomposition of Stochastic HG Kernel



Time breakdown of 1 iteration SLM method

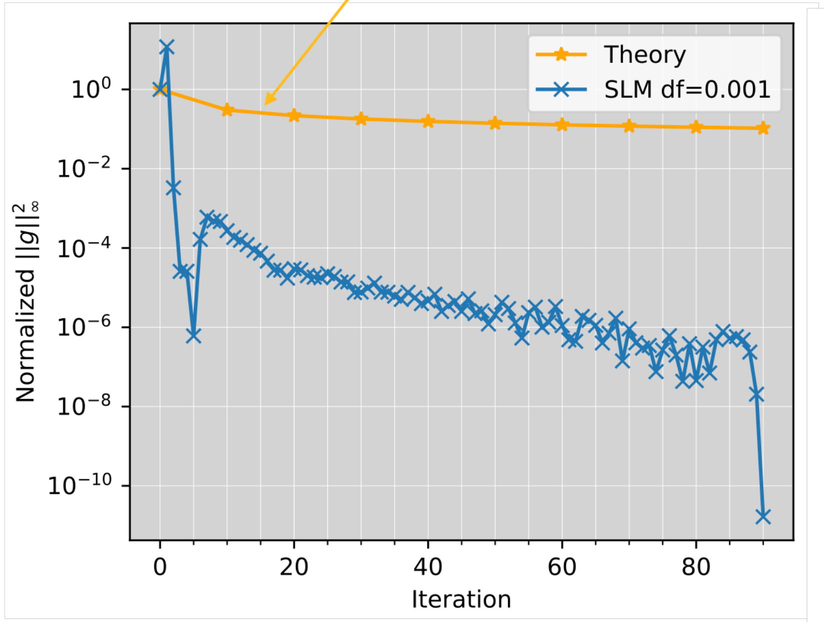
End-to-End Simulation of SLM with Different Data Fractions

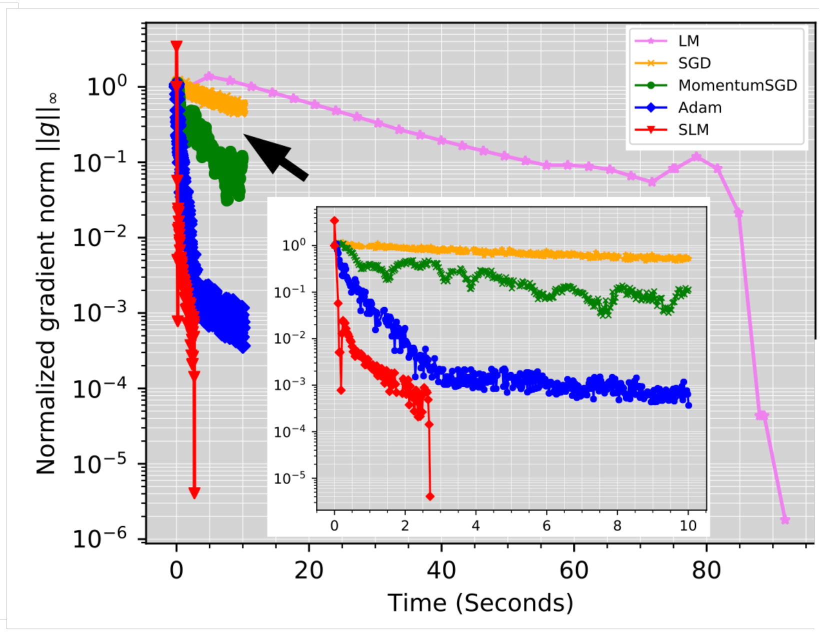
data fraction	Strehl Ratio
1e-1	25.55%
1e-2	25.57%
1e-3	25.56%
1e-4	22.8%
1e-5	15.8%

- Strehl Ratio (SR) is the most important criteria for AO system, 1% difference is significant.
- Optimal SR (using ground truth parameters) for this system is 25.57%.
- When we further decrease data fraction below 1e-3, it will damage final SR.
- 1e-3 is a data fraction level that balance efficiency and SR.

Numerical Results using a single NVIDIA A100 GPU

Theory convergence $O(\frac{1}{\sqrt{K+1}})$





df	SLM	Time (s)	SGD	Time (s)
1e-3	25.57%	2.67	14.96%	10.0
1e-2	25.57%	9.93	21.61%	10.0
1e-1	25.56%	5.93	22.43%	10.0

df	Adam	Time (s)	M-SGD	Time (s)
1e-3	25.55%	10.0	23.7%	10.0
1e-2	25.58%	10.0	23.96%	10.0
1e-1	25.51%	10.0	23.16%	10.0

Experiment Setting for different optimizer parameters (tuned for fastest convergence):
df = 1e-3, lr=0.5, $\mu = 1e^{-6}$,$\varepsilon = 1e^{-5}$
Time out for stochastic algorithms: 10 s.

- SLM converges fastest among all optimizers.
- LM is much slower than other optimizers.

- Optimal Strehl Ratio is 25.6%.
- Levenberg-Marquardt method is 25.6%.
- SLM and Adam reach optimal Strehl Ratio.

Remarks

➢ Accelerating the soft real time controller of AO system by

- Designing Stochastic Levenberg-Marquardt Method
- Presenting the theory of SLM Method with fixed regularization
- Implementing and optimizing new stochastic HG kernel
- Comparing the performance of SLM with other first-order methods and origin LM method

Content

- Introduction

- Adaptive Optics for Ground-based Telescopes
  - Stochastic Levenberg-Marquardt Method in Soft Real-Time Controller
  - Discrete Algebraic Riccati Equation in Soft Real-Time Controller
  - TLR-MVM in Hard Real-Time Controller

- Seismic Redatuming by Inversion Using Marchenko-based Methods
  - Batched TLR-MVM
  - Mixed-Precision Batched TLR-MVM on NVIDIA GPU

The Discrete time Algebraic Riccati Eequation (DARE)

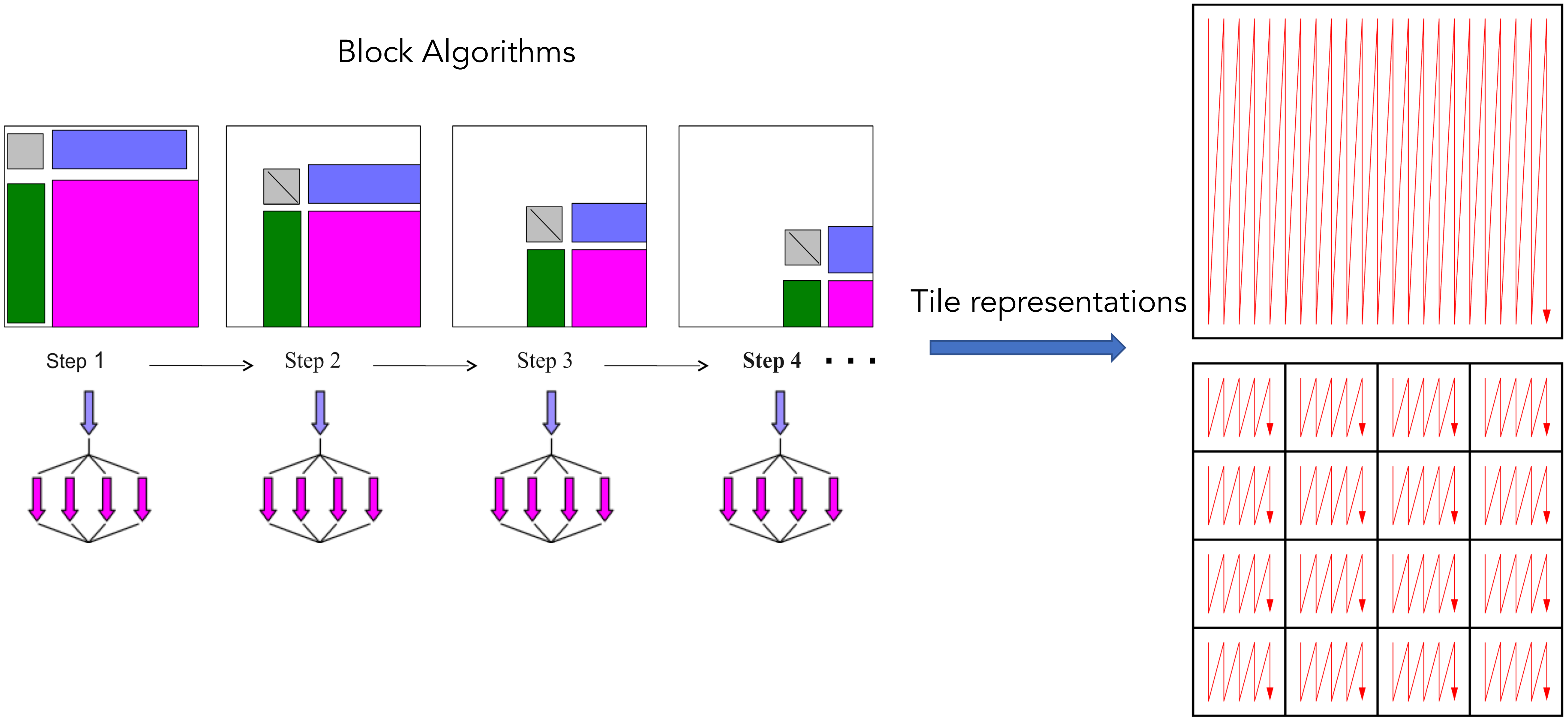
- Predictive control (LQG) for better closed-loop performance of SRTC (Strehl Ratio > 40% for large dimension):
  - Engenders (thought to be) prohibitive arithmetic complexity

$$P = A^T PA - (A^T PB)(R + B^T PB)^{-1}(B^T PA) + Q$$

$A : D \times D$
$B : D \times 18694$
$Q : D \times D$ (sym matrix)
$R : 18694 \times 18694$ (diagonal matrix)

  - D is spacial resolution, the larger the better. The largest D we investigate is 28210.
  - Defined as an iterative procedure converging to the solution
  - Rich matrix operations in Level-3 BLAS
  - Represents a good candidate for GPU hardware accelerations
  - Requires a large memory footprint

The Discrete time Algebraic Riccati Eequation (DARE)



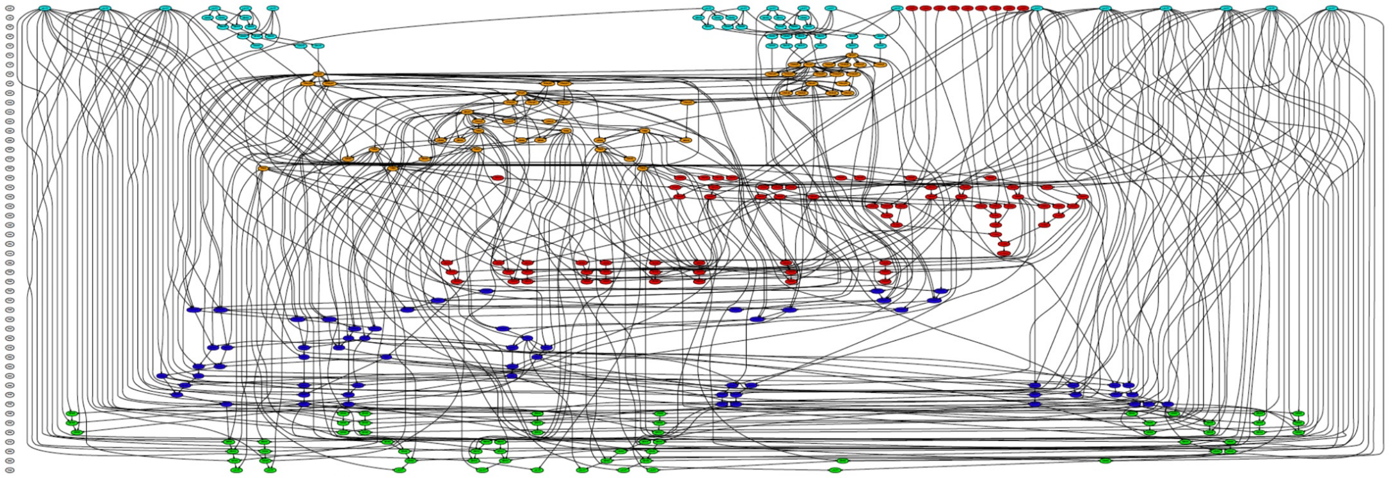
Impelmentation and Optimization Details

- Rely on Chameleon v1.1
- Powered by StarPU v1.3.9 dynamic runtime system
- Vendor software stack
  - NVIDIA CUDA v11.4
  - Intel MKL BLAS v2020
- Hardware settings
  - Host: dual-socket 28-core Intel IceLake
  - Devices: 4 NVIDIA A100 GPUs (80GB)
  - 1TB main memory

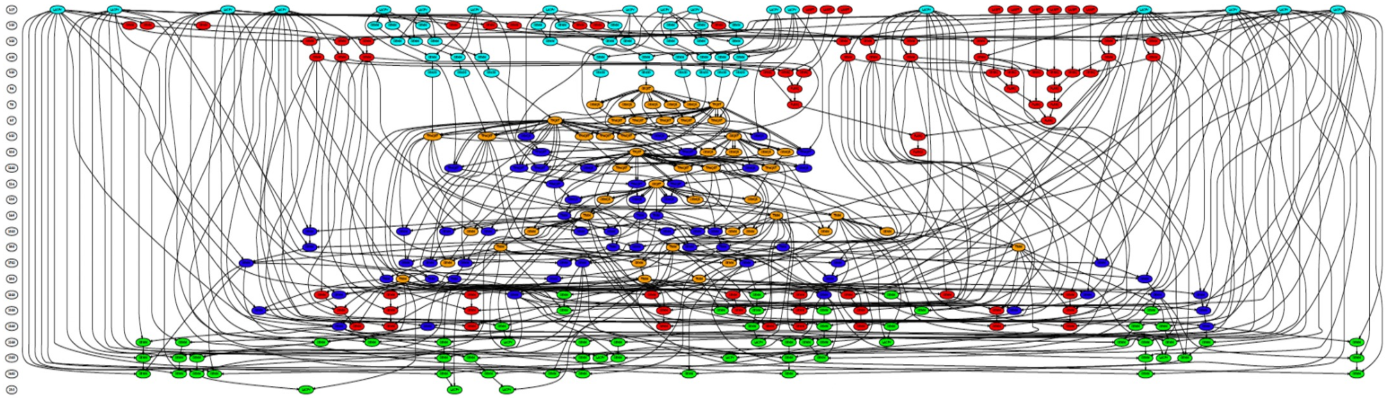
$$P = A^T PA - (A^T PB)\boxed{(R + B^T PB)^{-1}}(B^T PA) + Q$$

- **Asynchronous kernel executions**
  - Overlap data movement with useful computations
  - Increase hardware occupancy
  - Enhance data locality
- **Multithreaded BLAS on CPU**
  - GPU likes large tile sizes
  - CPU saturates with large tile sizes
  - Relieve CPU pressure by enabling multithreaded BLAS
  - Reduce the critical path
- **Structure-aware matrix computations**
  - Non-symmetric matrices
  - But diagonal dominant
  - Use of LU-based solver with no pivoting instead of QR-based solver

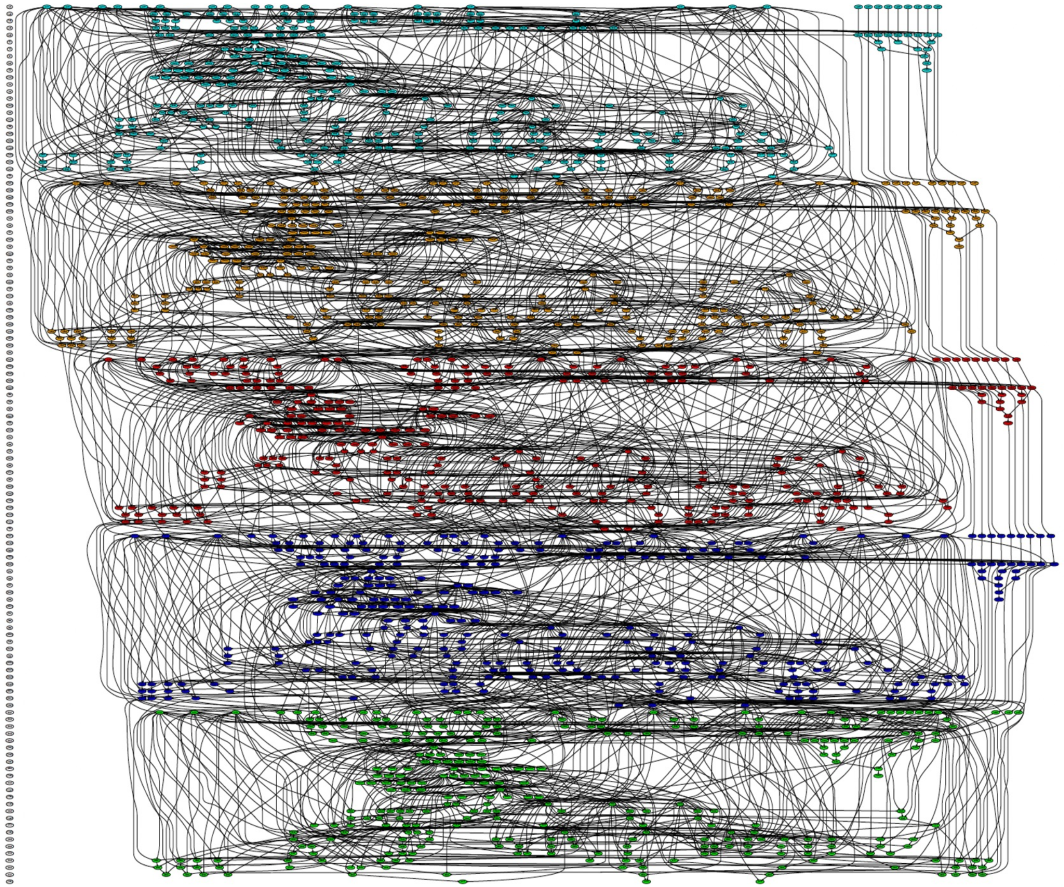
Computation DAG of DARE



Single Synchronized Iteration



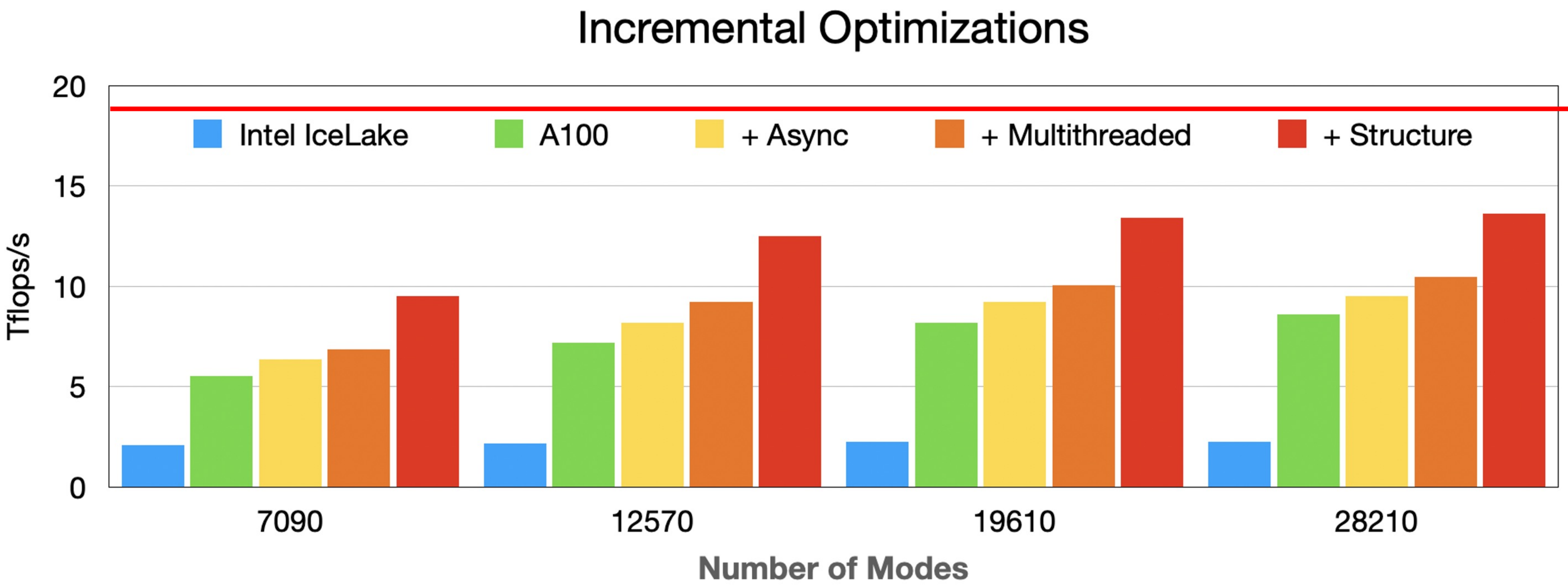
Single Asynchronized Iteration



Five Asynchronized Iterations

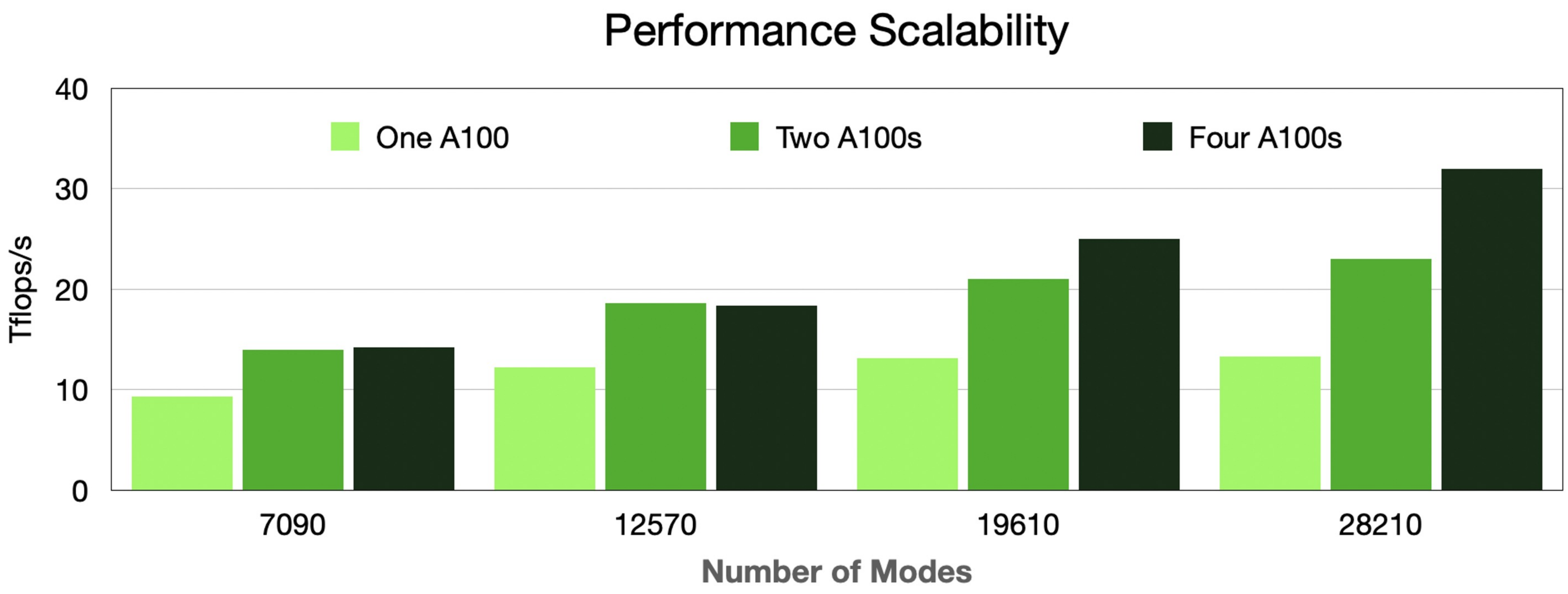
Performance Breakdown

Peak A100 FP64 (Tensor Core) FLOPS: 19.5 TFLOPs/s



- 6X against CPU
- Around 7min for the large MAVIS dimension

Scalability Results



- Decent scalability for large number of modes

Remarks

➢ Accelerating the soft real time controller of AO system by

- Designing high performance DARE implementation with several optimization strategies.

Content

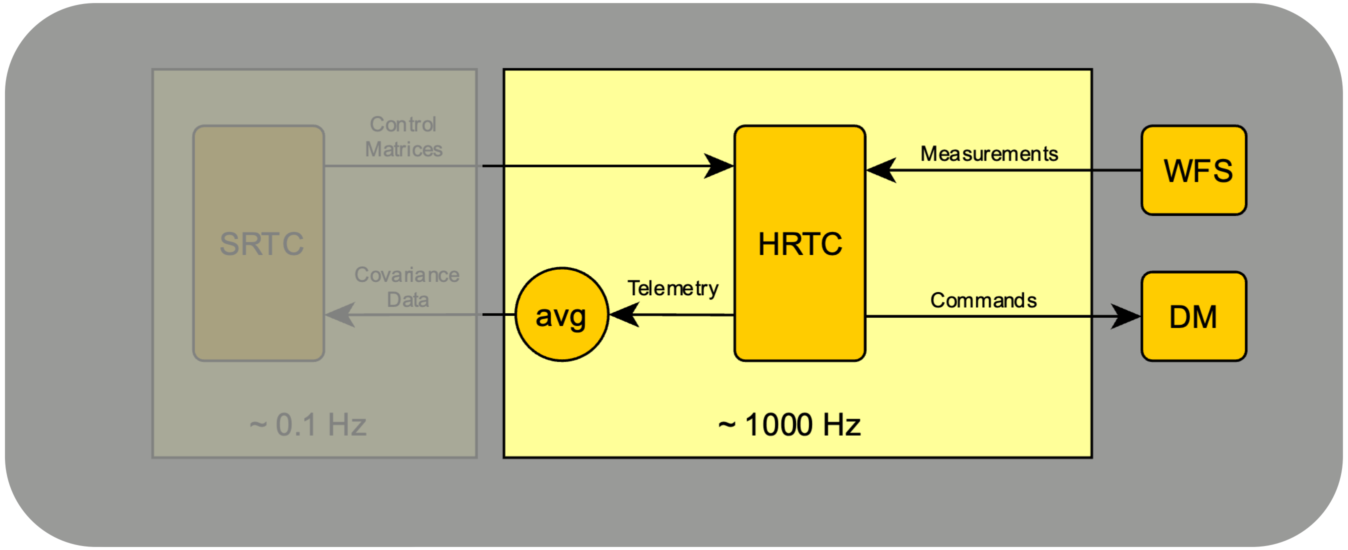
- Introduction

- Adaptive Optics for Ground-based Telescopes
  - Stochastic Levenberg-Marquardt Method in Soft Real-Time Controller
  - Discrete Algebraic Riccati Equation in Soft Real-Time Controller
  - TLR-MVM in Hard Real-Time Controller

- Seismic Redatuming by Inversion Using Marchenko-based Methods
  - Batched TLR-MVM
  - Mixed-Precision Batched TLR-MVM on NVIDIA GPU

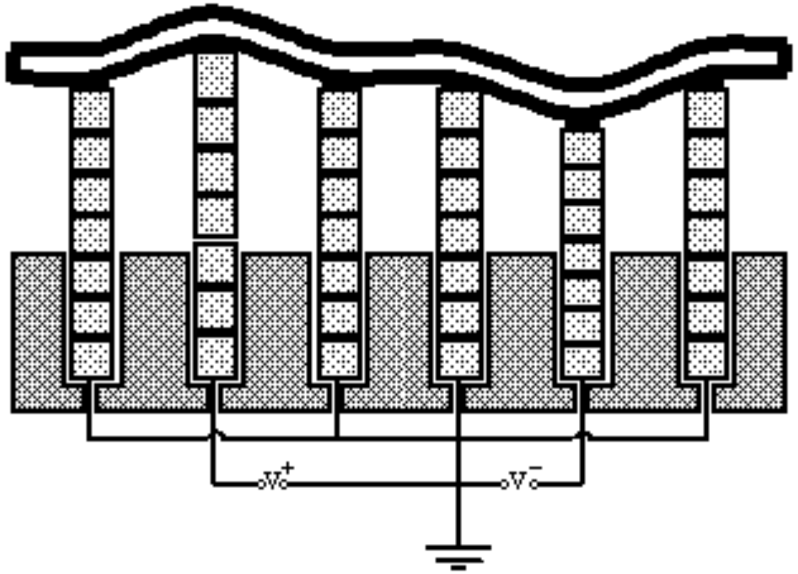
The Hard Real-Time Controller

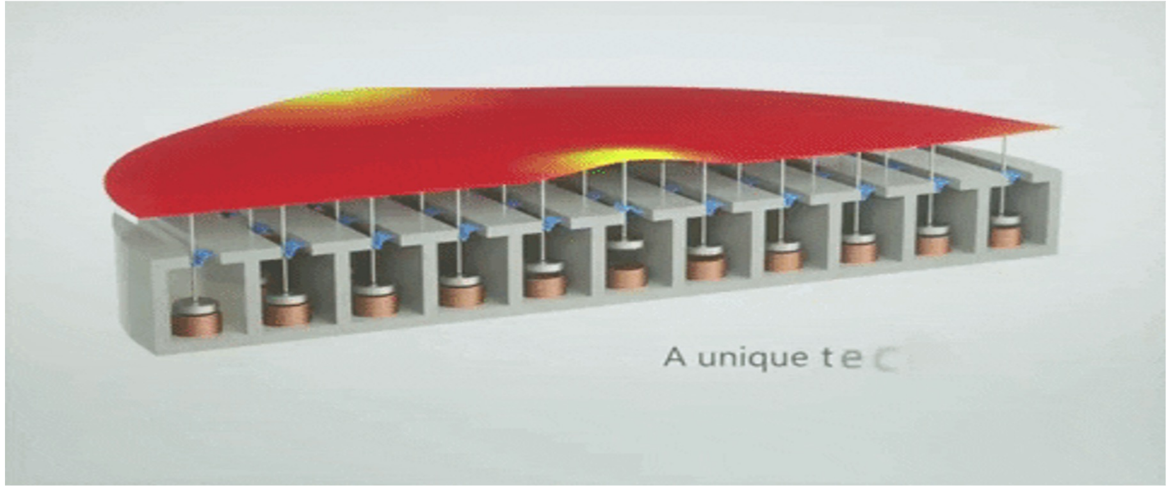
AO Control System



- **Input** is SRTC output – control matrices.
- Computations are a serial of **dense matrix vector multiplications.**
- **Output** is sent to **deformable mirrors** to compensate for wavefront distortions.
- Typical rate of operation is **1kHz.**
- Compute pipeline **latency below 1 millisecond.**
- **Stable time-to-solution** is critical to ensure stable operations (jitter of the order of 10s of μs).

Key Hardware Component: The Deformable Mirrors



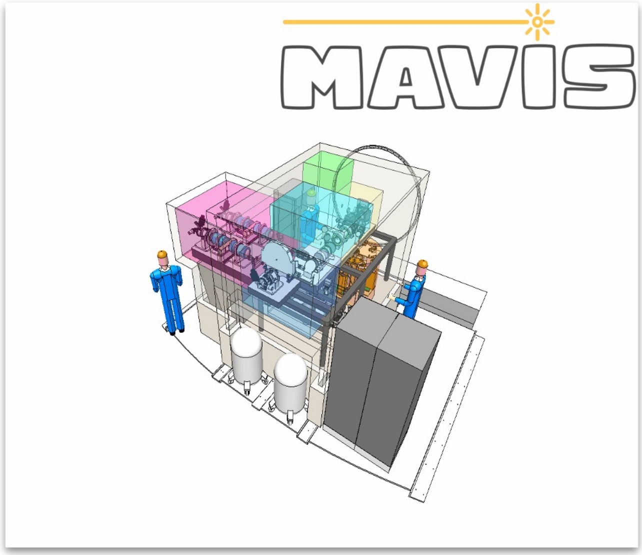


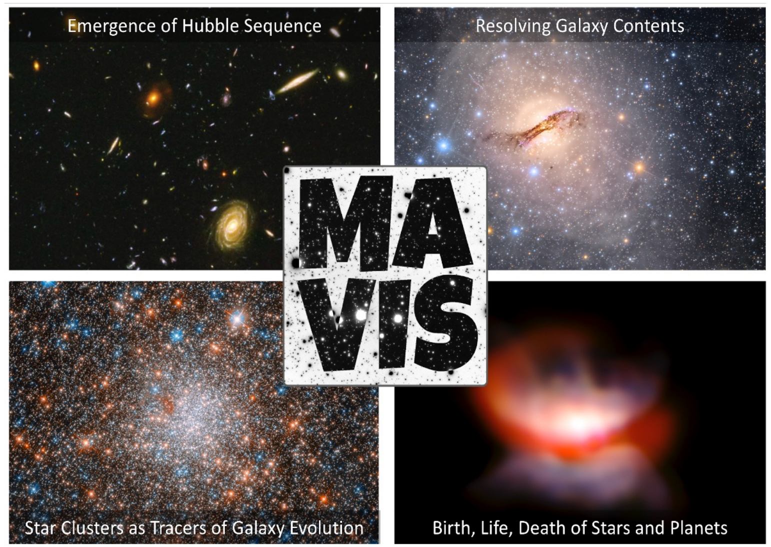
- The physical accuracy of deformable mirrors is limited
- There are potential opportunities to do approximated computing

MAVIS AO System

3rd-generation instrument for the Very Large Telescope

- **Scaling up** the whole AO concept:
  - More actuators
  - Faster control system
- **Fast-track** project:
  - First-light by 2026
- **Deeper & Sharper** than any space-based instruments

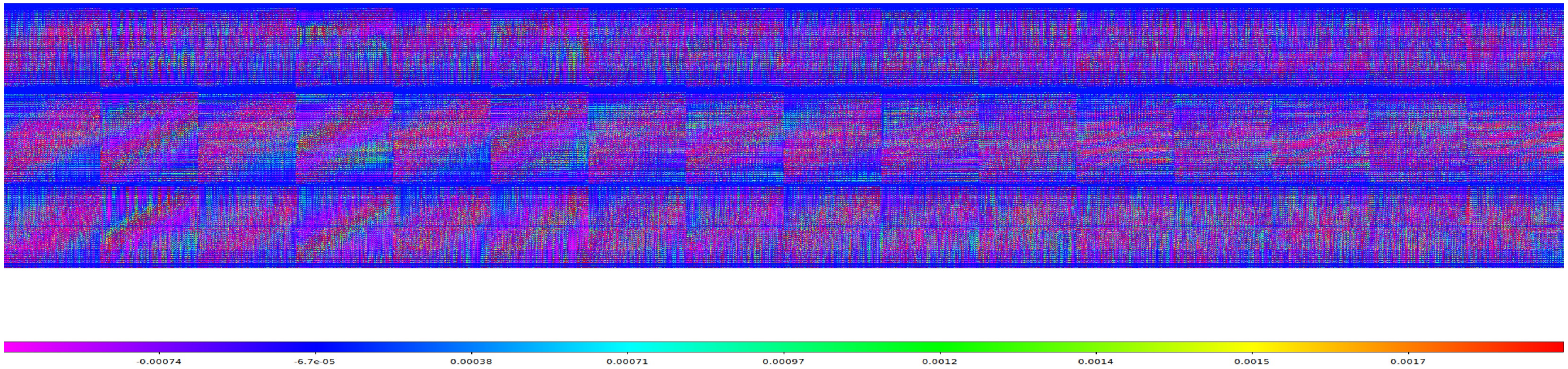




Data structure of Tomographic Reconstructor

Typical reconstruction matrix for MAVIS (tomography + predictive control)

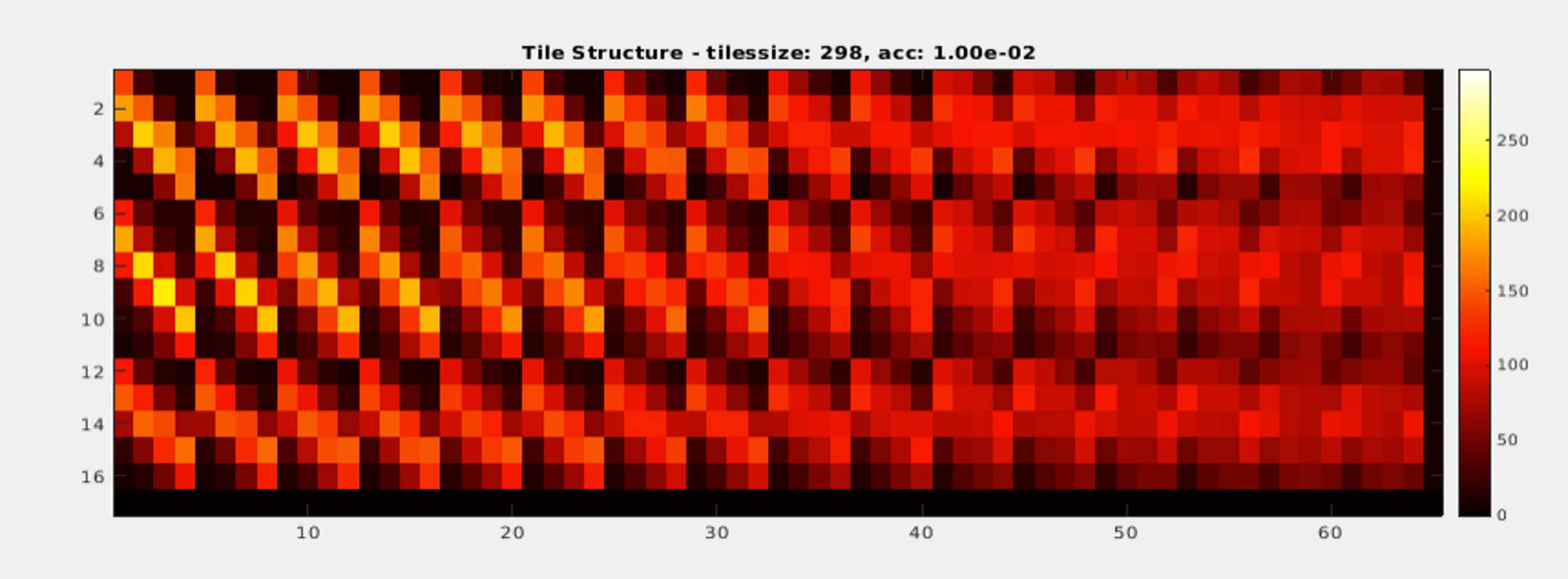
- Mapping WFS measurements (~20k) to actuators (~5k)
- Apparently very structured,connected to system parameters (WFS dimensioning)
- Structure agnostic to turbulence parameters



Rank Analysis

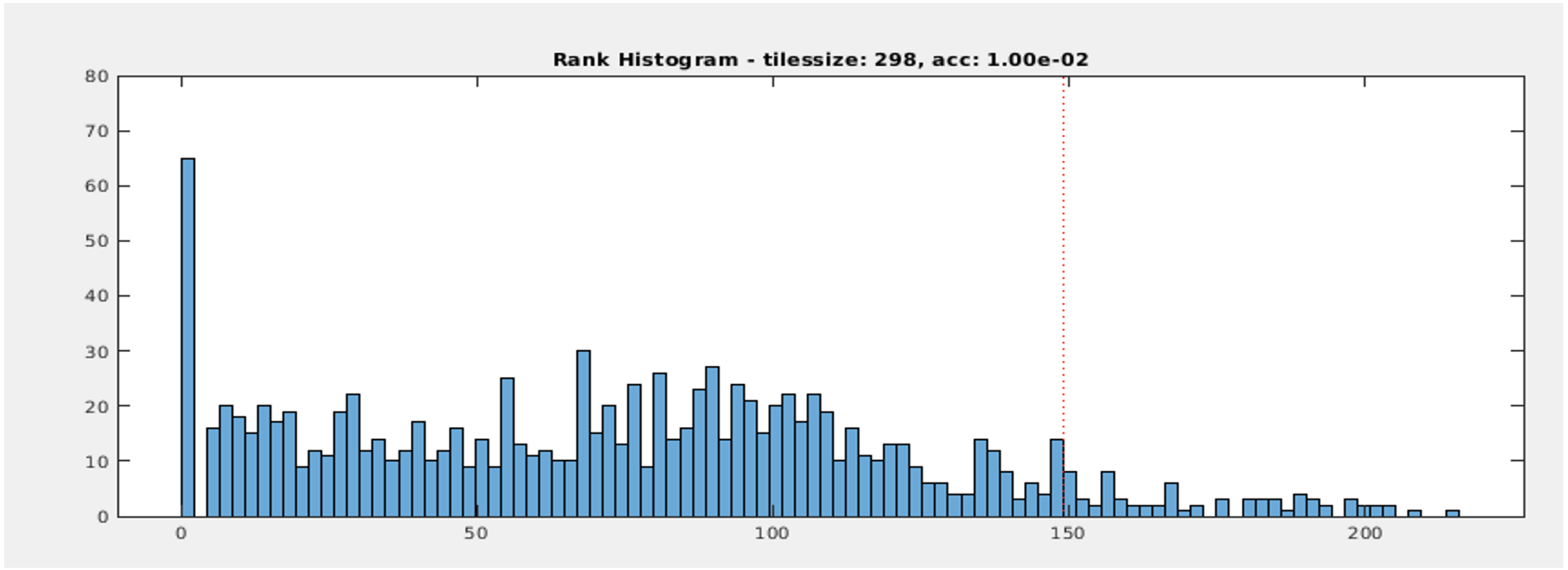
Splitting the matrix into tiles and assessing ranks

- Tiles size aligned with system parameters
- Data sparse, opportunities for low-rank matrix approximation
- Assuming constant tile size (may be sub-optimal



Rank Analysis

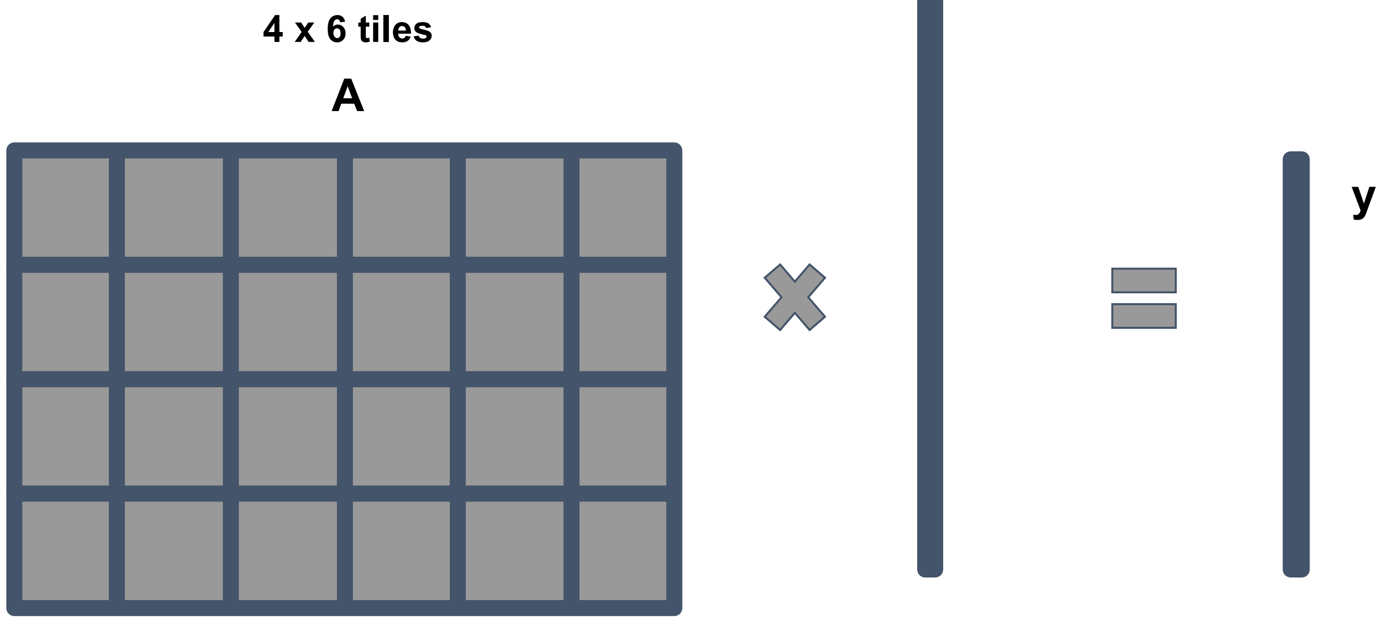
A vast majority of the tiles have low ranks (i.e., smaller than half of the tile size) => **data sparsity structure**, opportunity for low-rank approximations



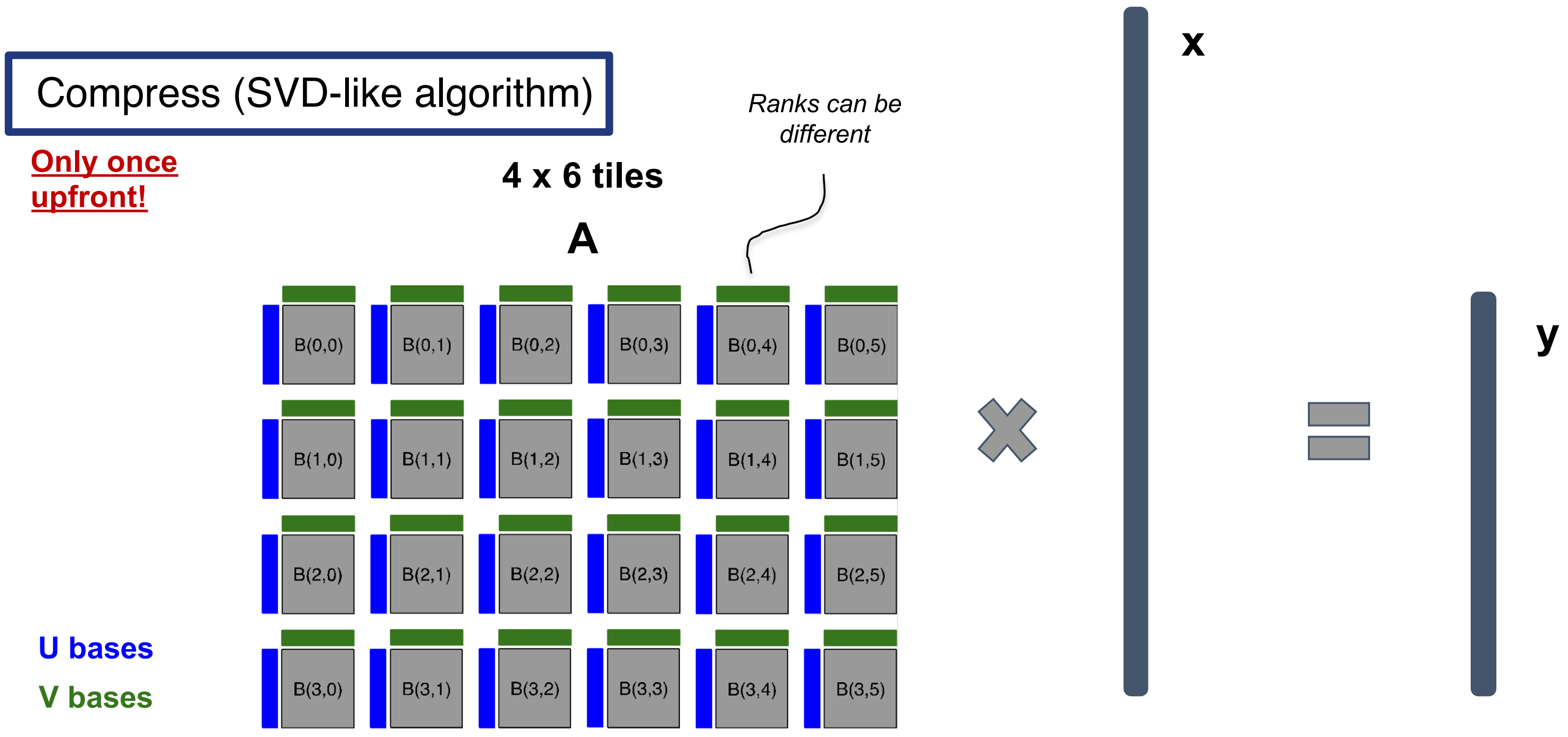
Tile Low-Rank Matrix Vector Multiplication (TLR-MVM)

How to leverage data sparsity?

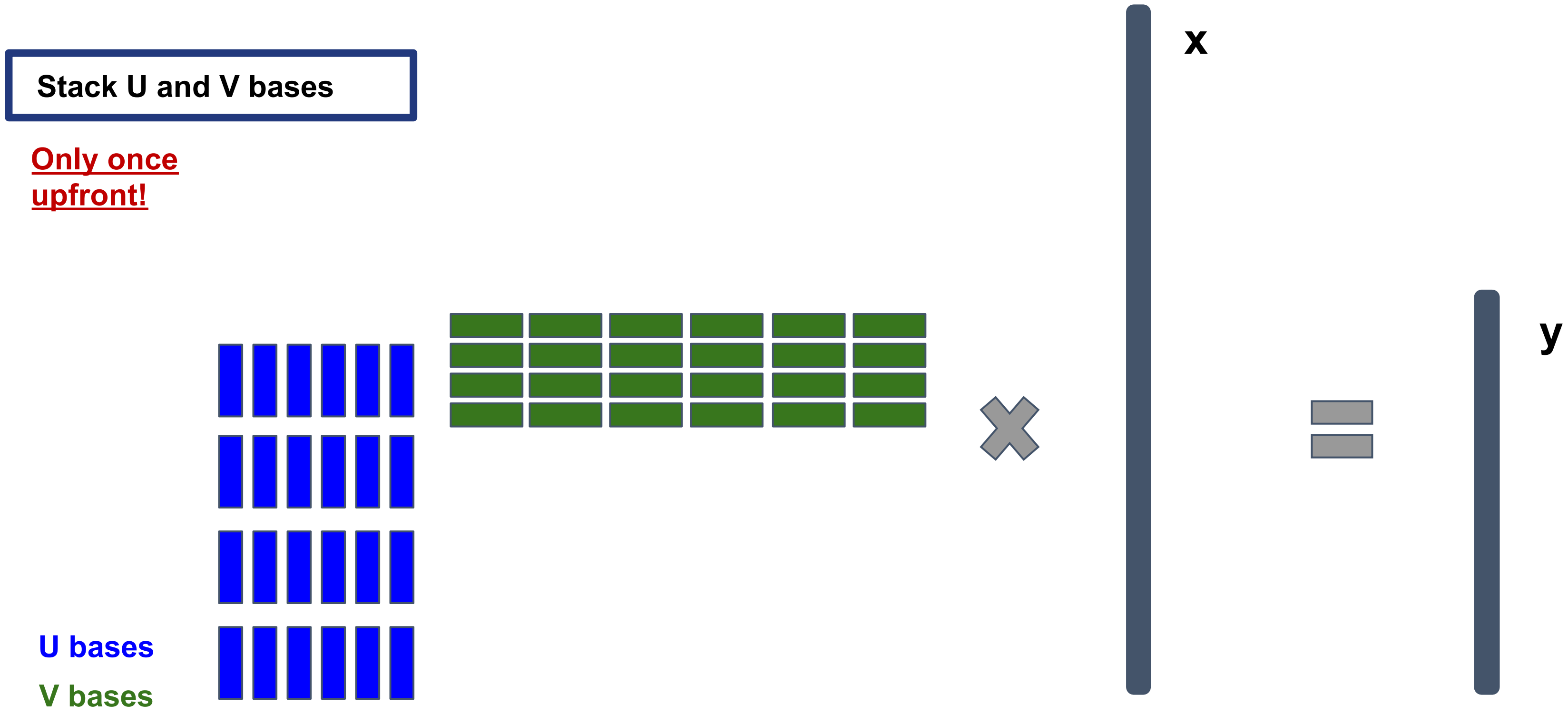
Dense Matrix-Vector Multiplication



TLR-MVM Algorithm Explanation

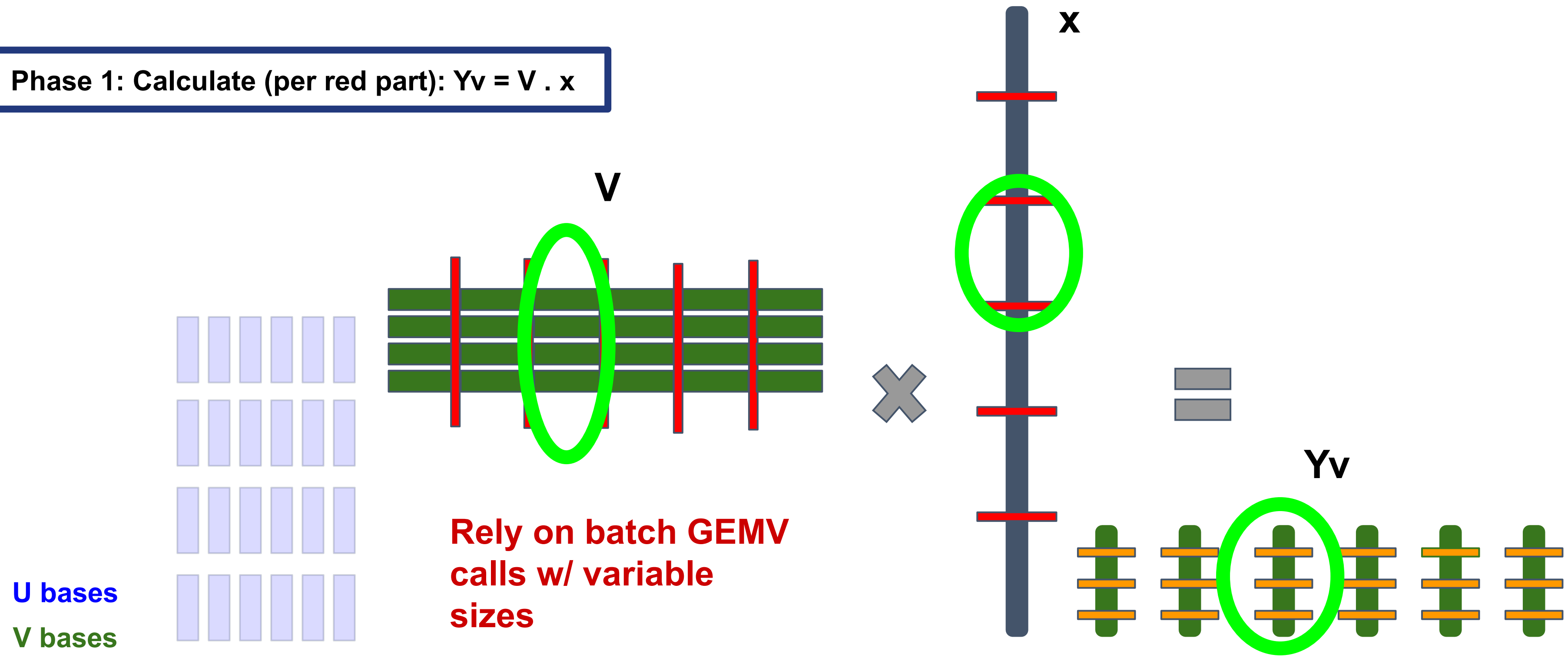


TLR-MVM Algorithm Explanation



TLR-MVM Algorithm Explanation

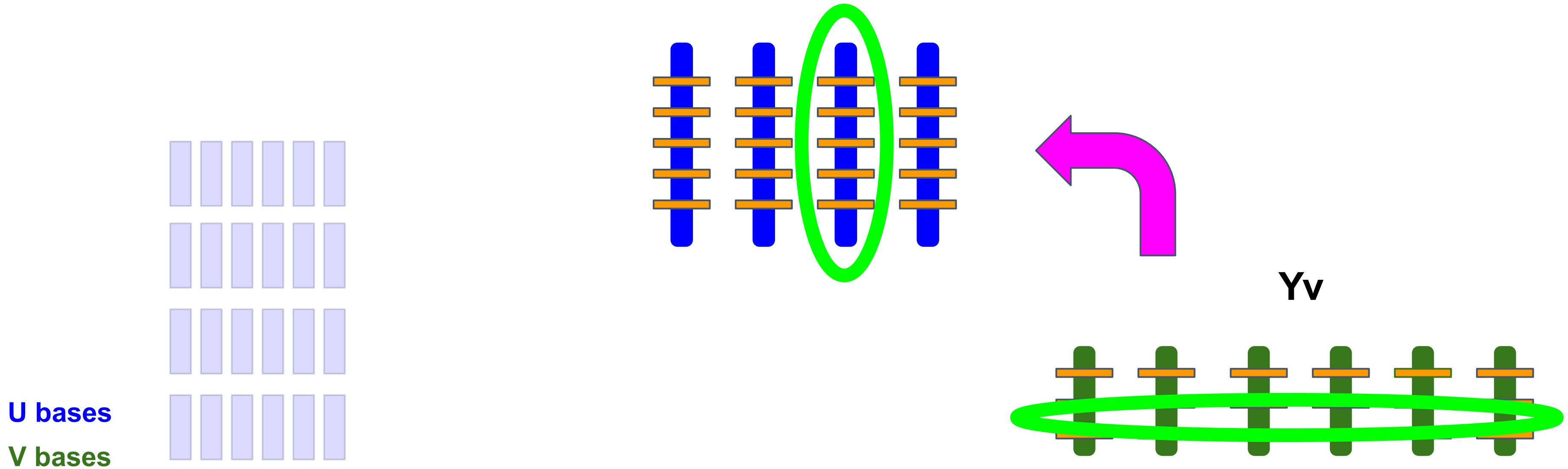
Phase 1: Calculate (per red part): Yv = V . x



Rely on batch GEMV calls w/ variable sizes

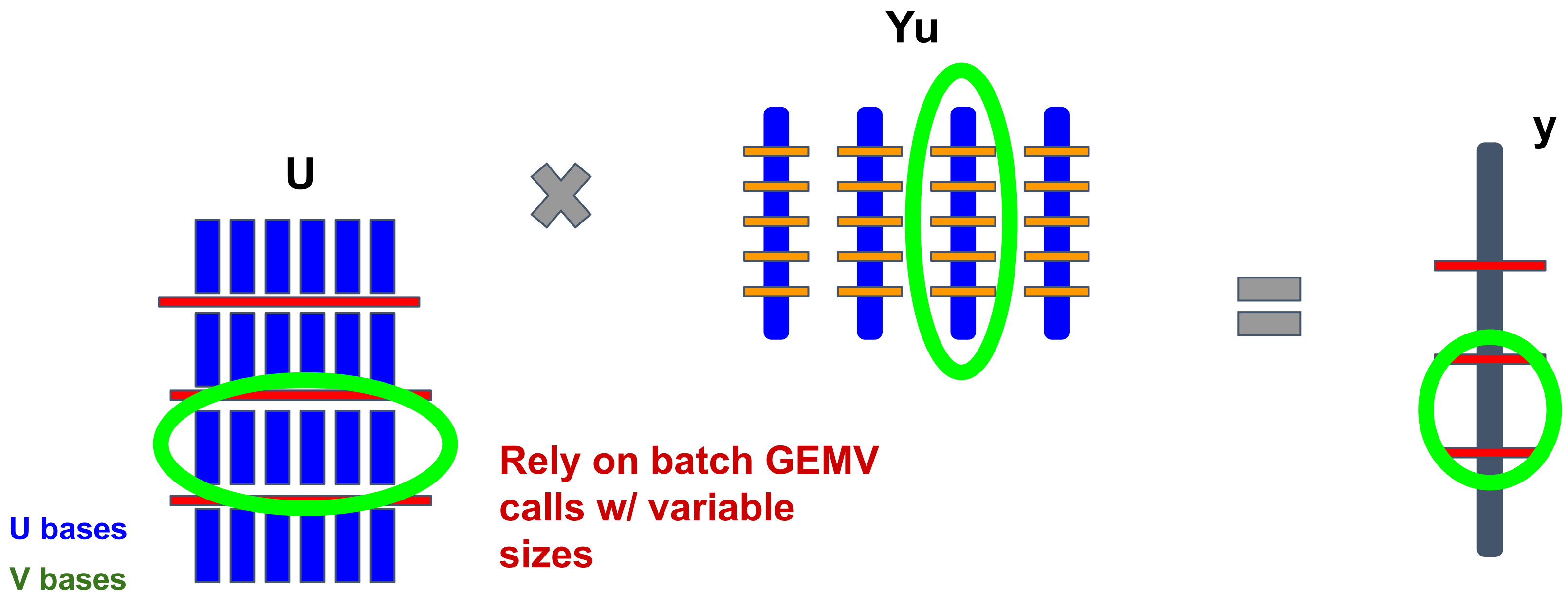
TLR-MVM Algorithm Explanation

Phase 2: reshuffle Yv (V bases) to Yu (U bases)



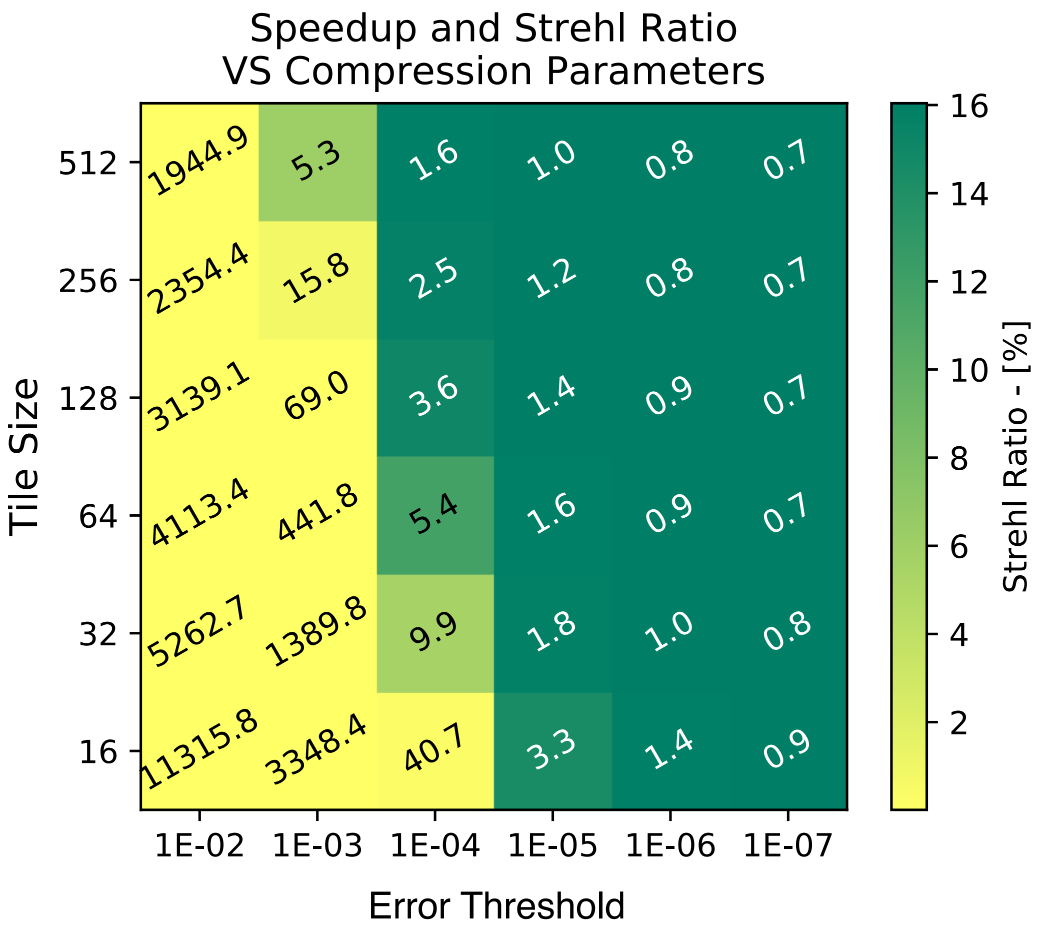
TLR-MVM Algorithm Explanation

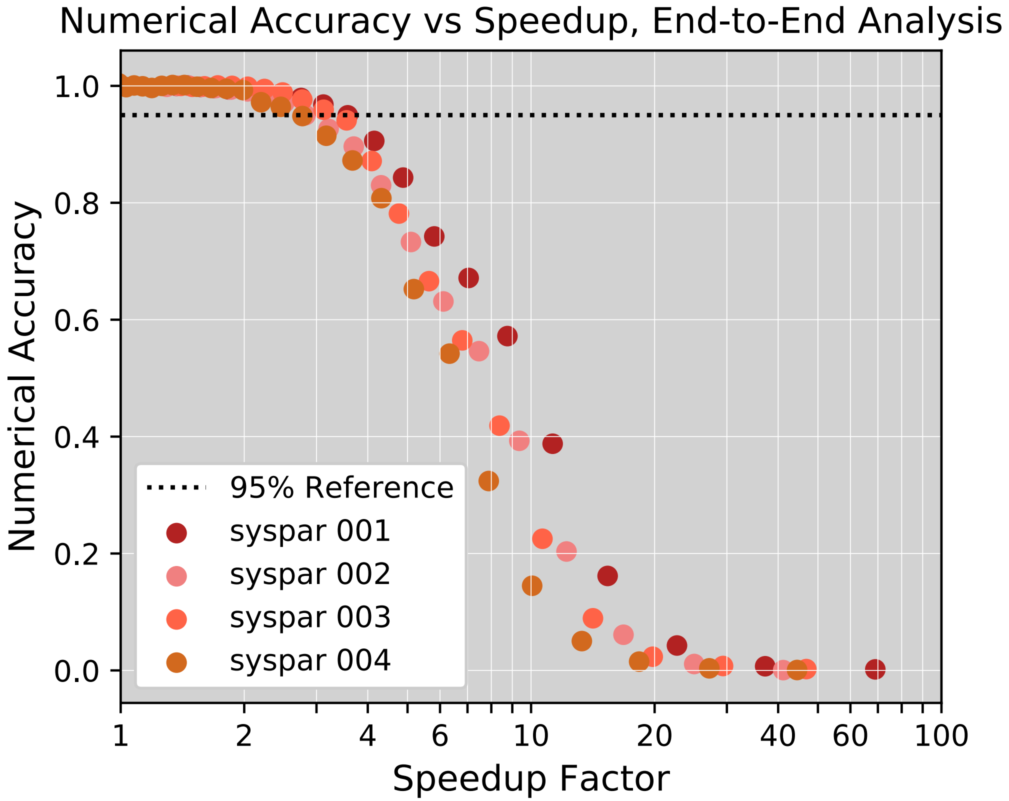
Phase 3: Calculate y = U . Yu



Rely on batch GEMV calls w/ variable sizes

Numerical Accuracy Assessment on MAVIS Datasets





Hardware / Software Specifications

Vendor	Intel	AMD		Fujitsu	NVIDIA	NEC
Family	Cascade Lake	EPYC Rome	Instinct	Primergy A64FX	Ampere GPU	SX-Aurora TSUBASA
Model	6248	7702	MI100	FX1000	A100	B300-8
Node(s)/Card(s)	1	1	1	16	1	8
Socket(s)	2	2	N/A	4	N/A	N/A
Cores	40	128	7680	48	6912	8
GHz	2.5	2.2	1.5	2.2	2.6	1.6
Memory Sustained BW	384GB DDR4 232GB/s	512GB DDR4 330GB/s	32GB HBM2 1.2TB/s	32GB HBM2 800GB/s	40GB HBM2e 1.5TB/s	48GB HBM2 1.5TB/s
LLC Sustained BW	27.5MB 1.1TB/s	**512MB** 4TB/s	8MB 3TB/s	32MB 3.6TB/s	40MB 4.8TB/s	16MB 2.1TB/s
Compiler	Intel compiler 19.1.0	GCC compiler 8.2.0		Fujitsu compiler 4.5.0	NVCC 11.0	NEC compiler 3.1.1
BLAS library	Intel MKL 2020	BLIS 3.0.0		Fujitsu SSL II	cuBLAS 11.0	NEC NLC 2.1.0
MPI library	OpenMPI 4.0.3	OpenMPI 3.1.2		Fujitsu MPI 4.0.1	NCCL 2.0	NEC MPI 2.13.0

x86 **ARM** **Vector Engines**

MPI + OpenMP

Hardware / Software Specifications

Vendor	Intel	AMD		Fujitsu	NVIDIA	NEC
Family	Cascade Lake	EPYC Rome	Instinct	Primergy A64FX	Ampere GPU	SX-Aurora TSUBASA
Model	6248	7702	MI100	FX1000	A100	B300-8
Node(s)/Card(s)	1	1	1	16	1	8
Socket(s)	2	2	N/A	4	N/A	N/A
Cores	40	128	7680	48	6912	8
GHz	2.5	2.2	1.5	2.2	2.6	1.6
Memory Sustained BW	384GB DDR4 232GB/s	512GB DDR4 330GB/s	32GB HBM2 1.2TB/s	32GB HBM2 800GB/s	40GB HBM2e 1.5TB/s	48GB HBM2 1.5TB/s
LLC Sustained BW	27.5MB 1.1TB/s	**512MB** 4TB/s	8MB 3TB/s	32MB 3.6TB/s	40MB 4.8TB/s	16MB 2.1TB/s
Compiler	Intel compiler 19.1.0	GCC compiler 8.2.0		Fujitsu compiler 4.5.0	NVCC 11.0	NEC compiler 3.1.1
BLAS library	Intel MKL 2020	BLIS 3.0.0		Fujitsu SSL II	cuBLAS 11.0	NEC NLC 2.1.0
MPI library	OpenMPI 4.0.3	OpenMPI 3.1.2		Fujitsu MPI 4.0.1	NCCL 2.0	NEC MPI 2.13.0

Accelerators

HIP (rocBLAS) / CUDA (cuBLAS)

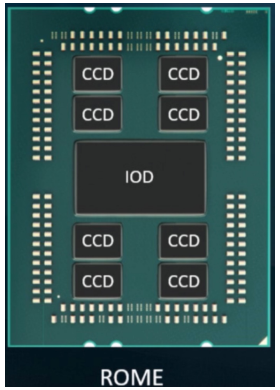
Hardware / Software Specifications

Vendor	Intel	AMD		Fujitsu	NVIDIA	NEC
Family	Cascade Lake	EPYC Rome	Instinct	Primergy A64FX	Ampere GPU	SX-Aurora TSUBASA
Model	6248	7702	MI100	FX1000	A100	B300-8
Node(s)/Card(s)	1	1	1	16	1	8
Socket(s)	2	2	N/A	4	N/A	N/A
Cores	40	128	7680	48	6912	8
GHz	2.5	2.2	1.5	2.2	2.6	1.6
Memory Sustained BW	384GB DDR4 232GB/s	512GB DDR4 330GB/s	32GB HBM2 1.2TB/s	32GB HBM2 800GB/s	40GB HBM2e 1.5TB/s	48GB HBM2 1.5TB/s
LLC Sustained BW	27.5MB 1.1TB/s	**512MB** 4TB/s	8MB 3TB/s	32MB 3.6TB/s	40MB 4.8TB/s	16MB 2.1TB/s
Compiler	Intel compiler 19.1.0	GCC compiler 8.2.0		Fujitsu compiler 4.5.0	NVCC 11.0	NEC compiler 3.1.1
BLAS library	Intel MKL 2020	BLIS 3.0.0		Fujitsu SSL II	cuBLAS 11.0	NEC NLC 2.1.0
MPI library	OpenMPI 4.0.3	OpenMPI 3.1.2		Fujitsu MPI 4.0.1	NCCL 2.0	NEC MPI 2.13.0

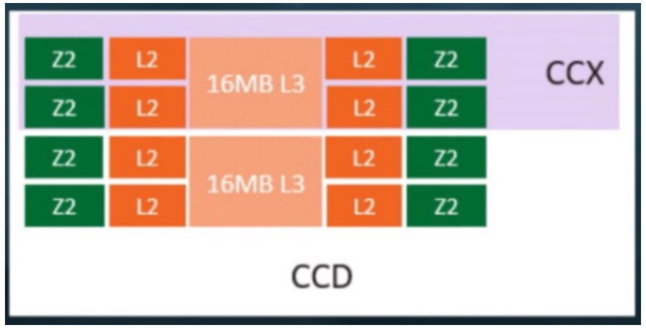
HBM

Hardware / Software Specifications

Vendor	Intel	AMD		Fujitsu	NVIDIA	NEC
Family	Cascade Lake	EPYC Rome	Instinct	Primergy A64FX	Ampere GPU	SX-Aurora TSUBASA
Model	6248	7702	MI100	FX1000	A100	B300-8
Node(s)/Card(s)	1	1	1	16	1	8
Socket(s)	2	2	N/A	4	N/A	N/A
Cores	40	128	7680	48	6912	8
GHz	2.5	2.2	1.5	2.2	2.6	1.6
Memory Sustained BW	384GB DDR4 232GB/s	512GB DDR4 330GB/s	32GB HBM2 1.2TB/s	32GB HBM2 800GB/s	40GB HBM2e 1.5TB/s	48GB HBM2 1.5TB/s
LLC Sustained BW	27.5MB 1.1TB/s	**512MB** 4TB/s	8MB 3TB/s	32MB 3.6TB/s	40MB 4.8TB/s	16MB 2.1TB/s
Compiler	Intel compiler 19.1.0	GCC compiler 8.2.0		Fujitsu compiler 4.5.0	NVCC 11.0	NEC compiler 3.1.1
BLAS library	Intel MKL 2020	BLIS 3.0.0		Fujitsu SSL II	cuBLAS 11.0	NEC NLC 2.1.0
MPI library	OpenMPI 4.0.3	OpenMPI 3.1.2		Fujitsu MPI 4.0.1	NCCL 2.0	NEC MPI 2.13.0

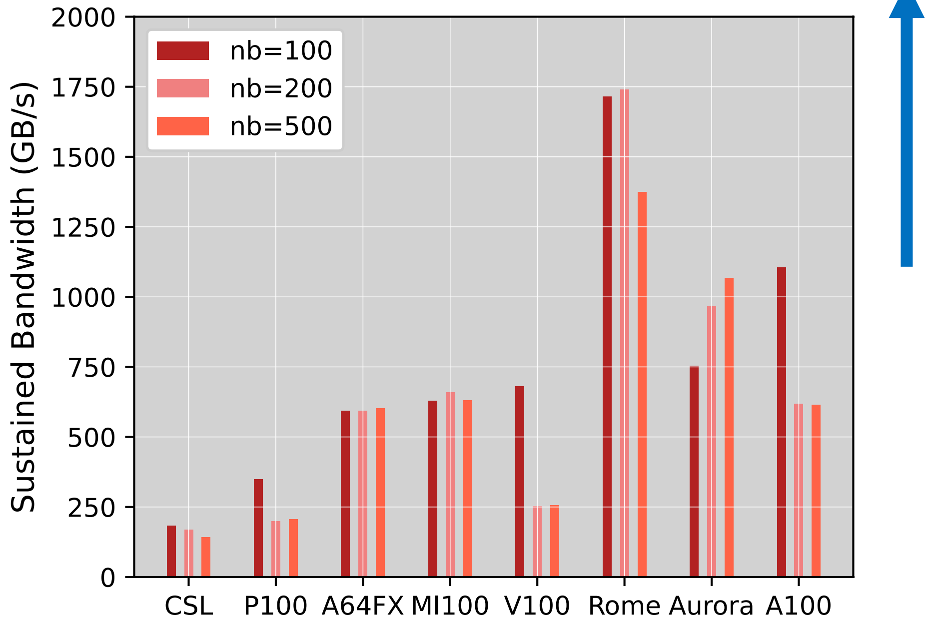


- IOD = "I/O Die" doing all the memory/PCI/other socket traffic
- CCD = "Core Compute Die", a chiplet having compute cores only
- CCX = "Core Compute CompleX", a set of cores sharing a L3 cache



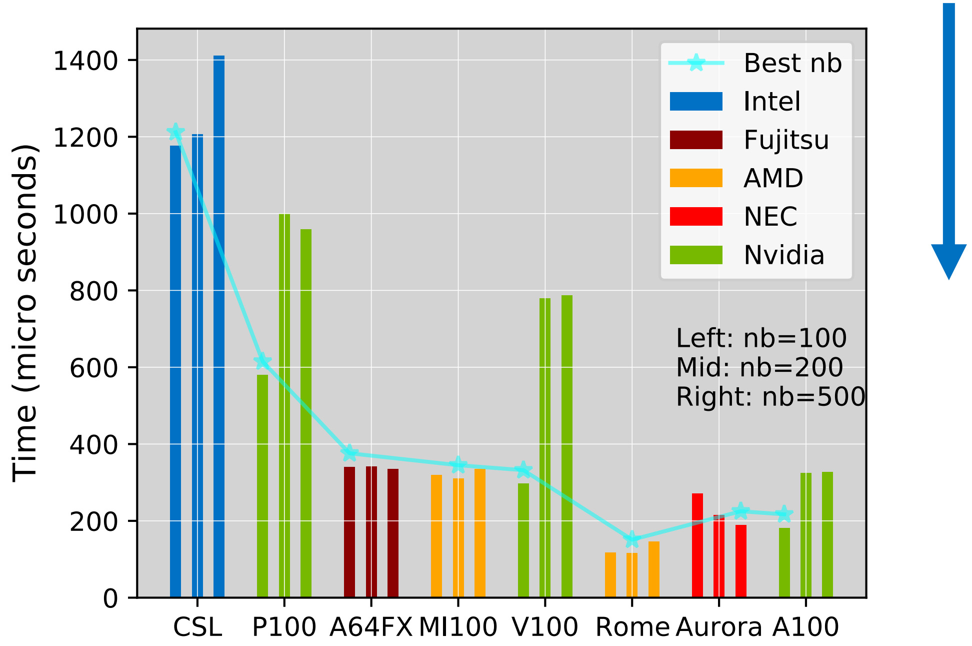
Results on synthetic datasets

Sustained Bandwidth on Synthetic Datasets



Higher is better

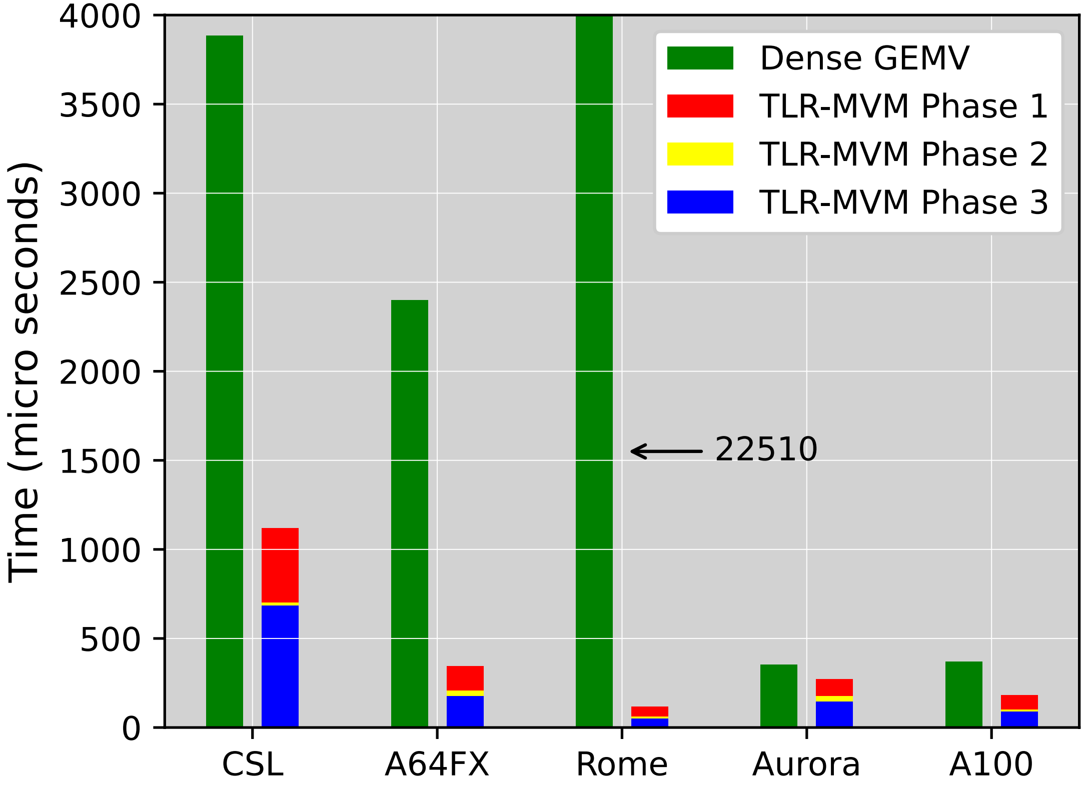
Time to solution on Synthetic Datasets



Rely on batch GEMV calls w/ constant sizes

Lower is better

Dense Vs TLR-MVM: Time Breakdown on Synthetic Datasets

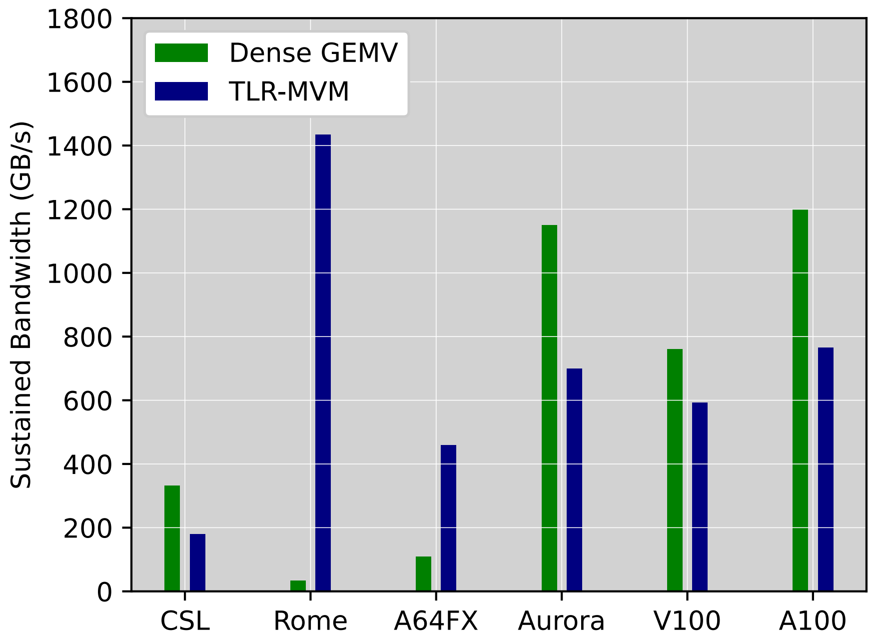


Rely on batch GEMV calls w/ constant sizes

Lower is better

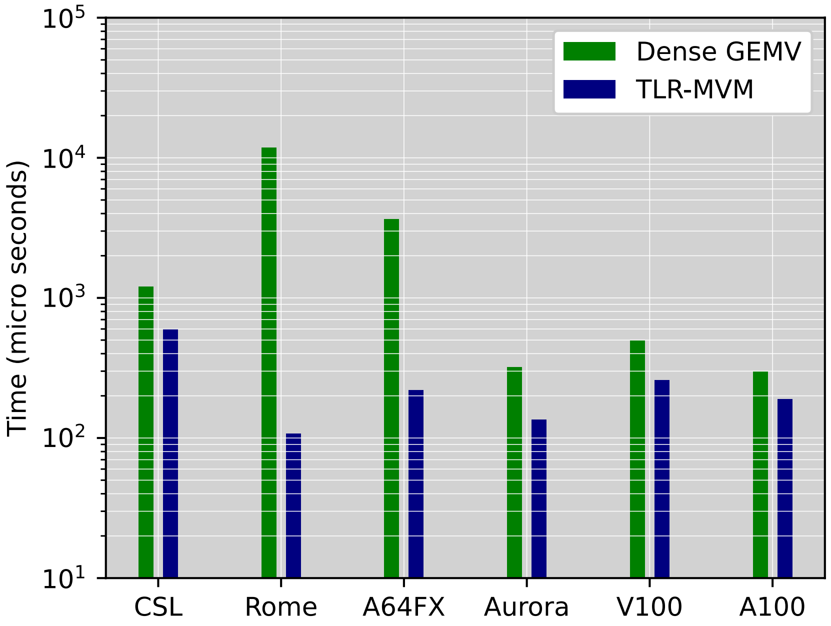
Results on MAVIS datasets

Sustained Bandwidth on MAVIS Datasets



Higher is better

Time to solution on MAVIS Datasets

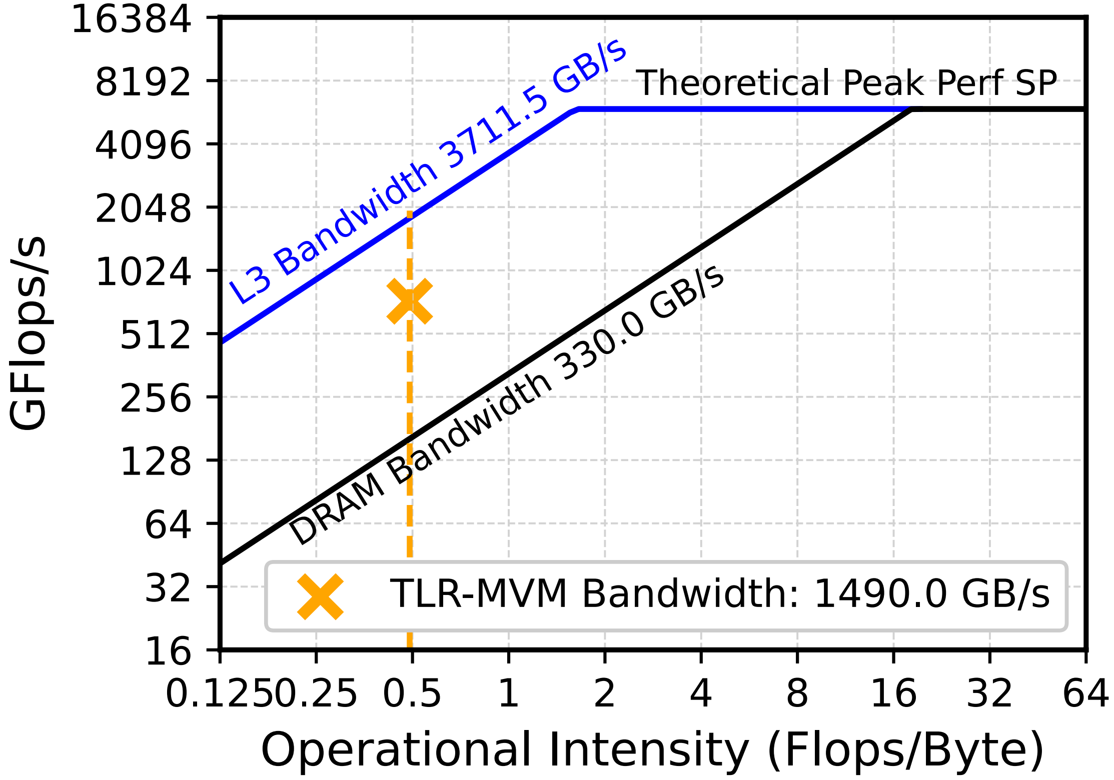


EPYC -> EPIC!

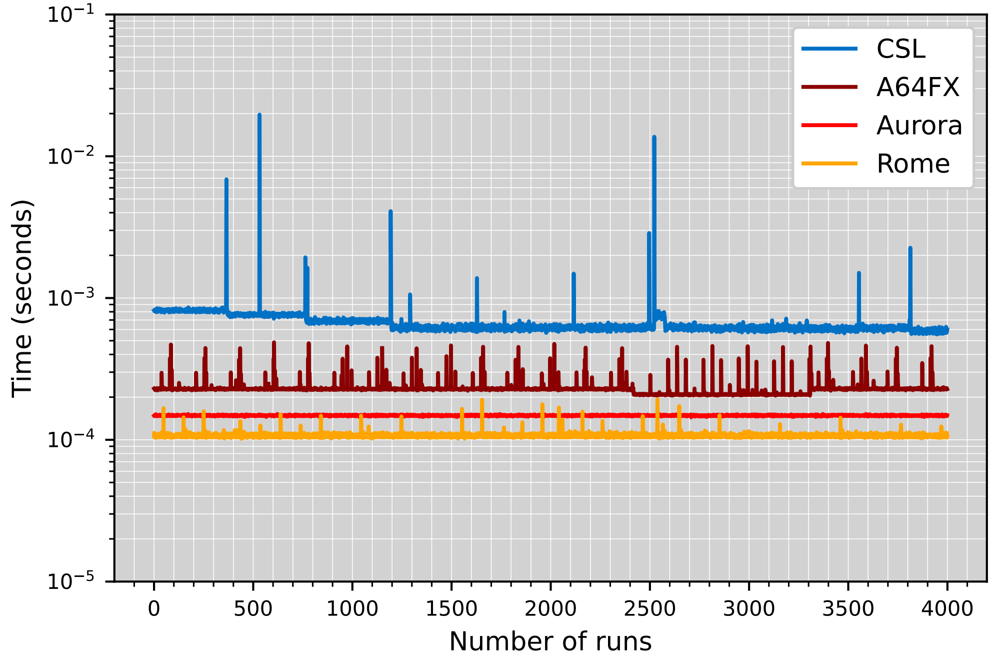
Rely on batch GEMV calls w/ variable sizes

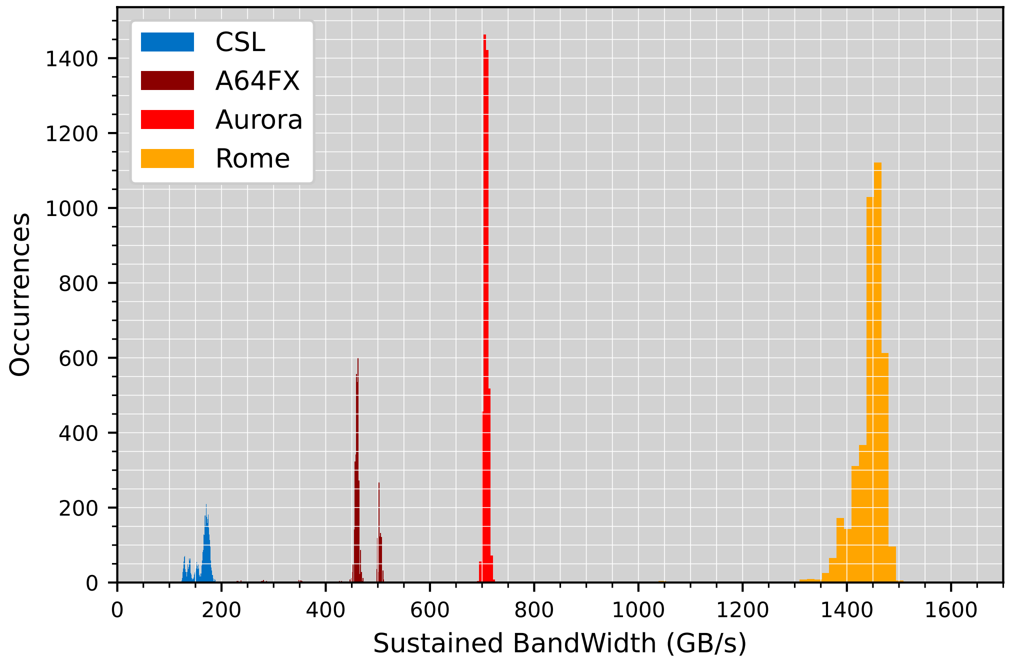
Lower is better

Roofline Performance Model of AMD EYPC Rome



Bandwidth / Time Jitter on MAVIS Datasets





Remarks

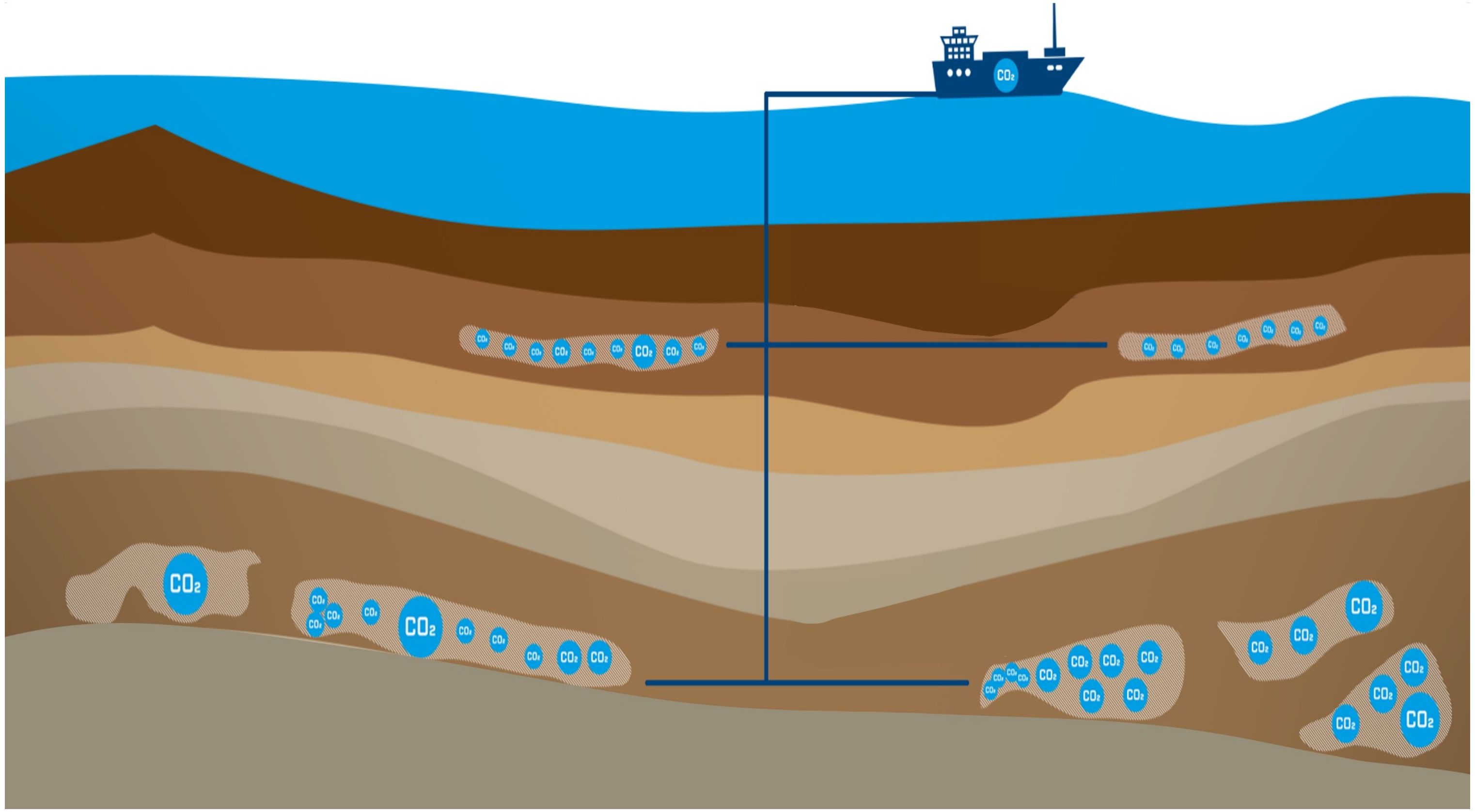
➢ Accelerating the hard real time controller of AO system by

- Tile Low-Rank Matrix Vector Multiplication (TLR-MVM)
- Evaluating the numerical accuracy with compression threshold
- Finding AMD EYPC Rome’s huge LLC has great impact to performance!

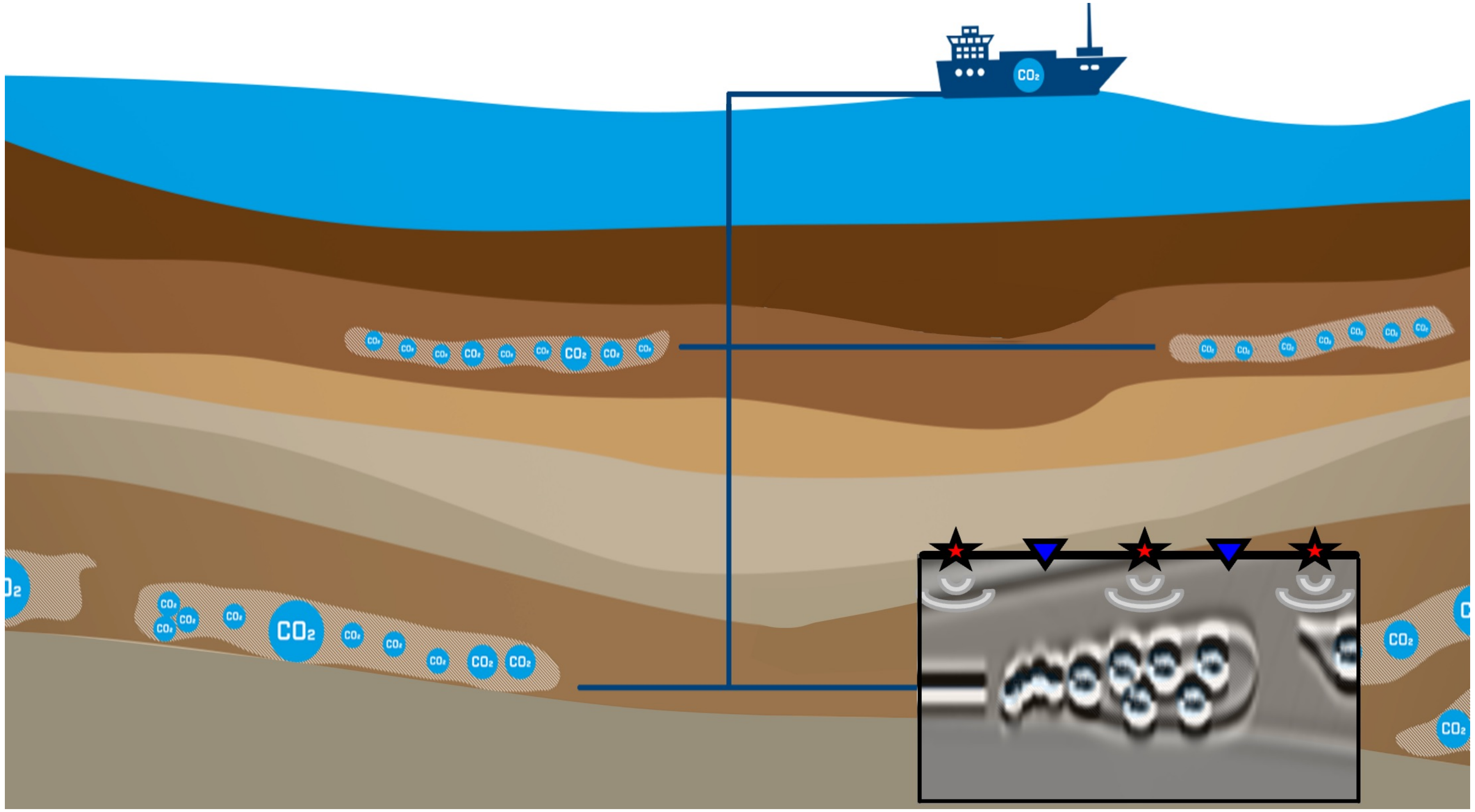
Content

- Introduction
- Adaptive Optics for Ground-based Telescopes
  - Stochastic Levenberg-Marquardt Method in Soft Real-Time Controller
  - Discrete Algebraic Riccati Equation in Soft Real-Time Controller
  - TLR-MVM in Hard Real-Time Controller
- Seismic Redatuming by Inversion Using Marchenko-based Methods
  - Batched TLR-MVM
  - Mixed-Precision Batched TLR-MVM on NVIDIA GPU

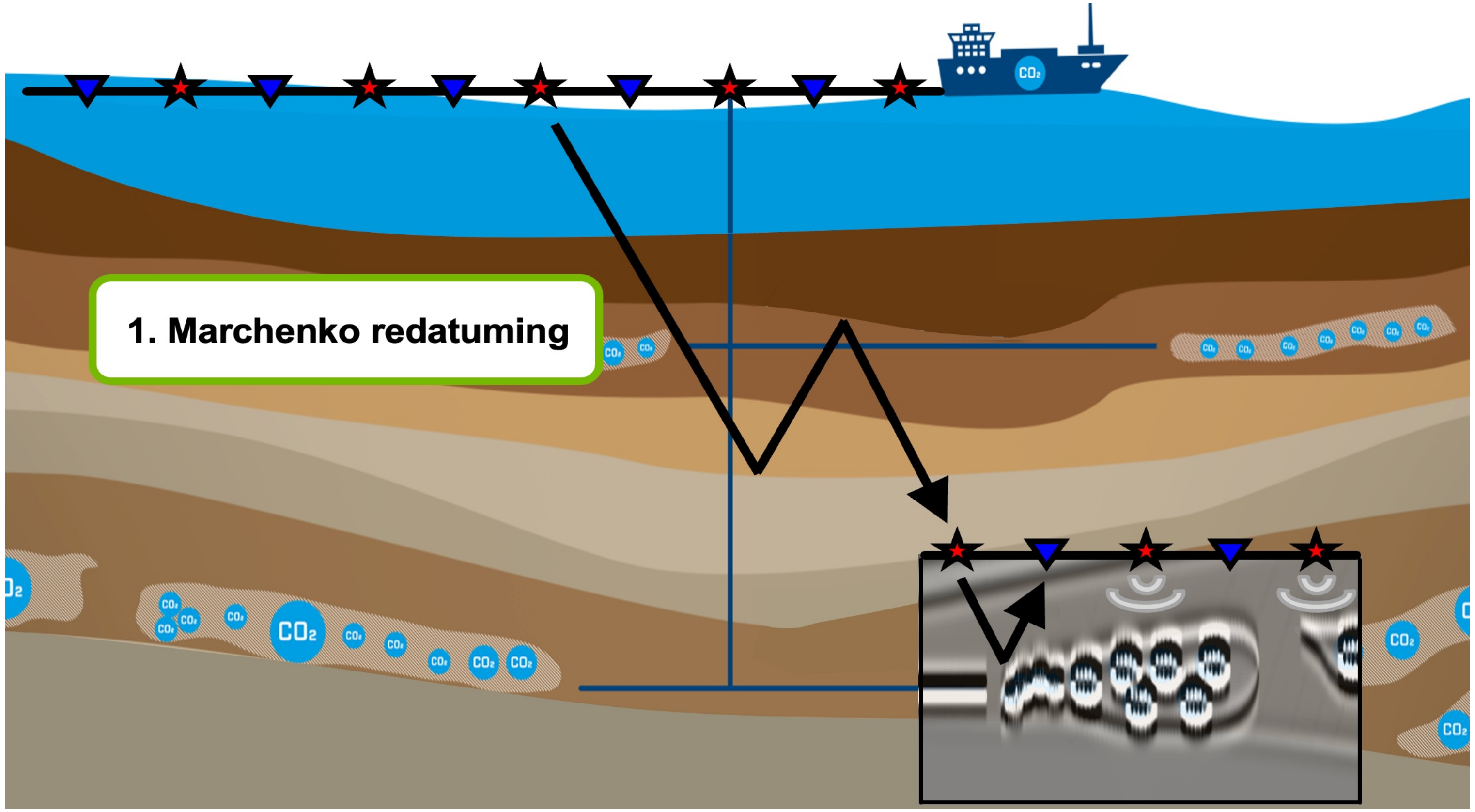
Seismic Imaging – a target-oriented perspective



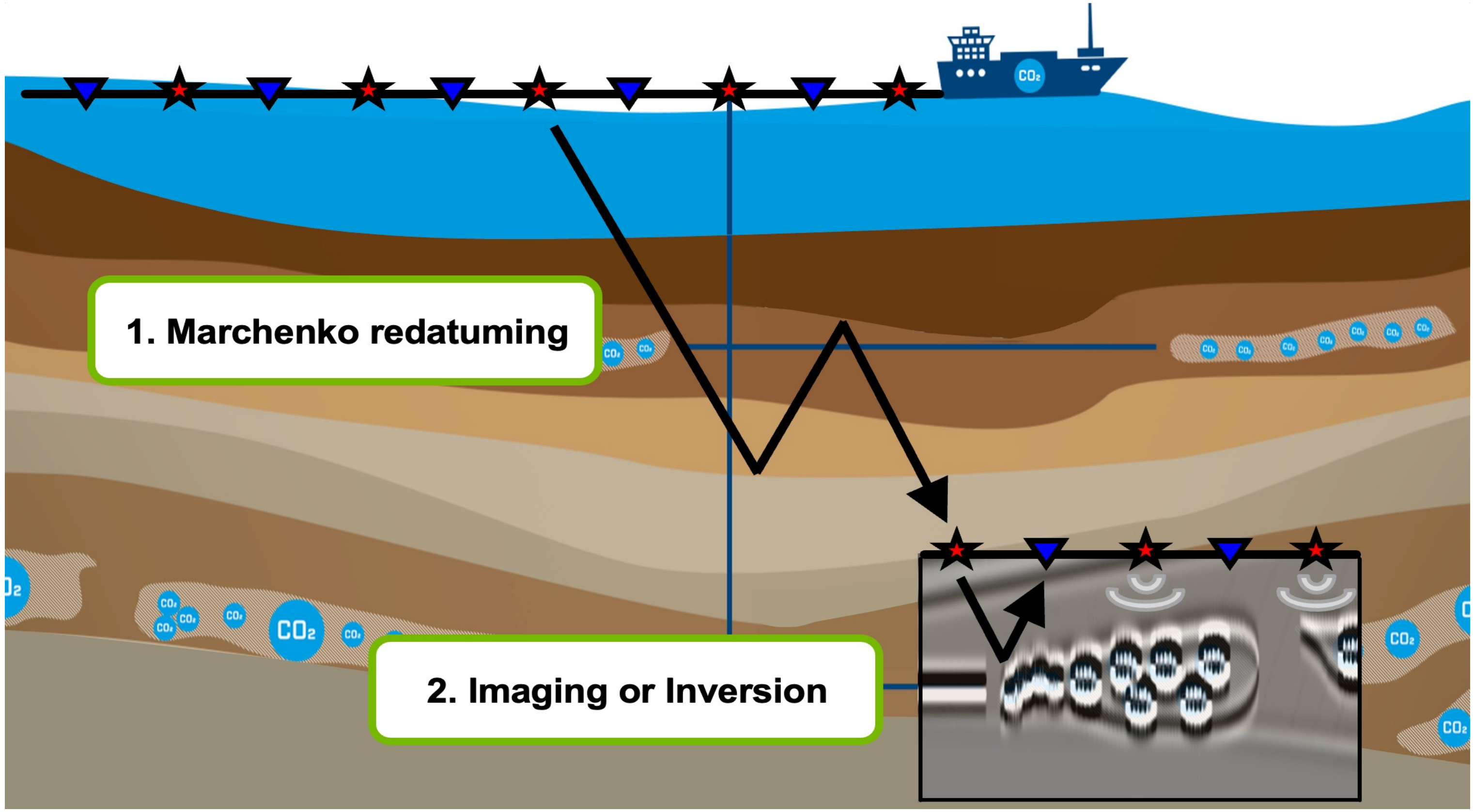
Seismic Imaging – a target-oriented perspective



Seismic Imaging – move the acquisition to the target



Seismic Imaging – image/invert where you care



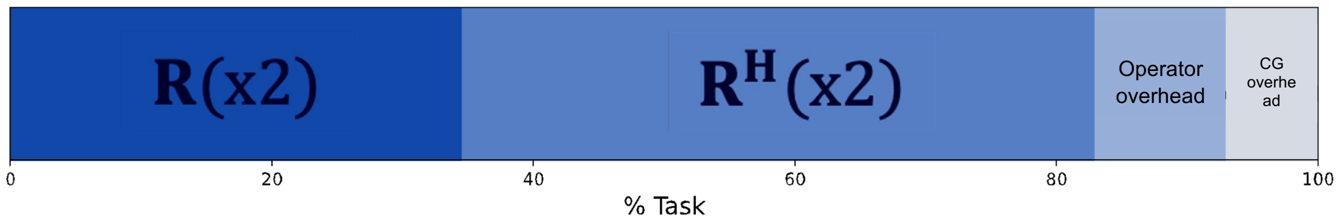
Marchenko Redatuming – mathematical formulation

$$\begin{bmatrix} \Theta R f_d^+ \\ 0 \end{bmatrix} = \begin{bmatrix} I & -\Theta R \\ \Theta R^* & I \end{bmatrix} \begin{bmatrix} f^- \\ f_m^+ \end{bmatrix}$$

R, R*: Multi-dimensional convolution operators (MDC)

Forward: 2 MDC, **Adjoint**: 2 MDC
CGLS (10 iterations): 40 MDC

Task % for one iteration of CG:



Marchenko Redatuming – modelling operator

Multi-dimensional convolution (MDC)

$$\overbrace{\mathbf{F}^{\mathbf{H}}\;\mathbf{I}^{H}_{\omega_{max}}\;\widehat{\mathbf{R}}\cdot\mathbf{I}_{\omega_{max}}\;\mathbf{F}}^{\mathbf{R}}$$

Real IFFT

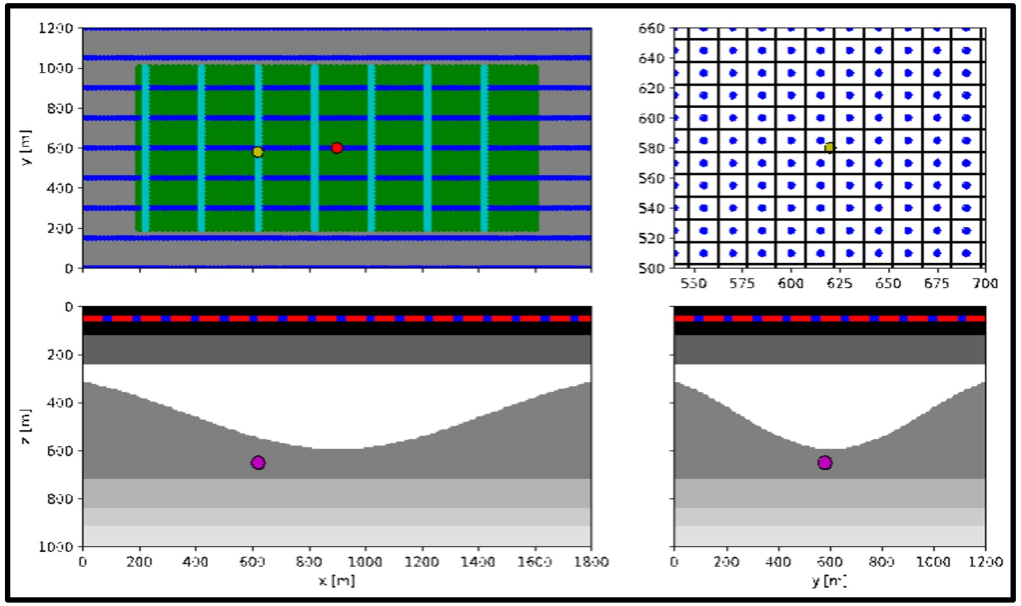
Padding

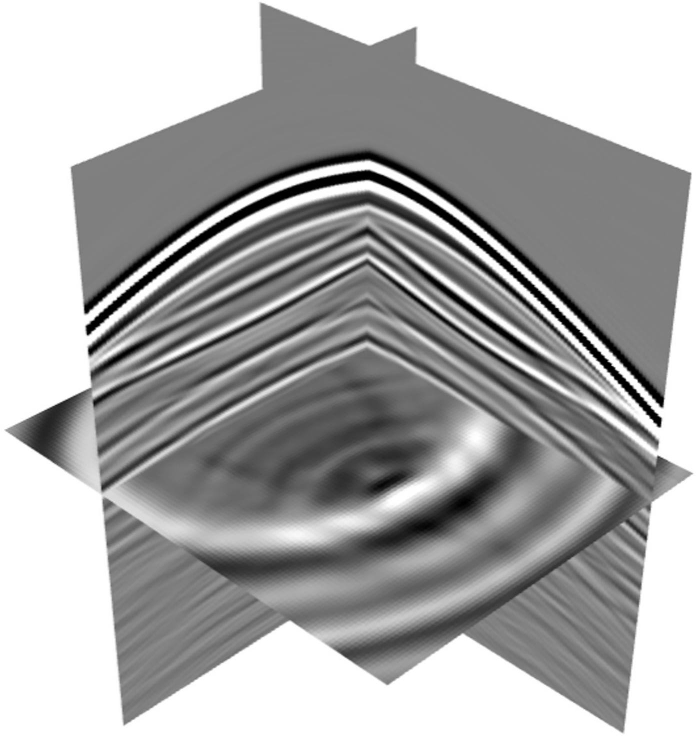
Batch matrix vector multiplication

Restriction

*Real FFT (**r** is real)*

Marchenko Redatuming – dataset





- Subsurface model of size 1.8km x 1.2km x 1km
- Regular carpet of sources and receivers

$N_S = N_R = 9801, \quad dx_S = dx_R = 15m$

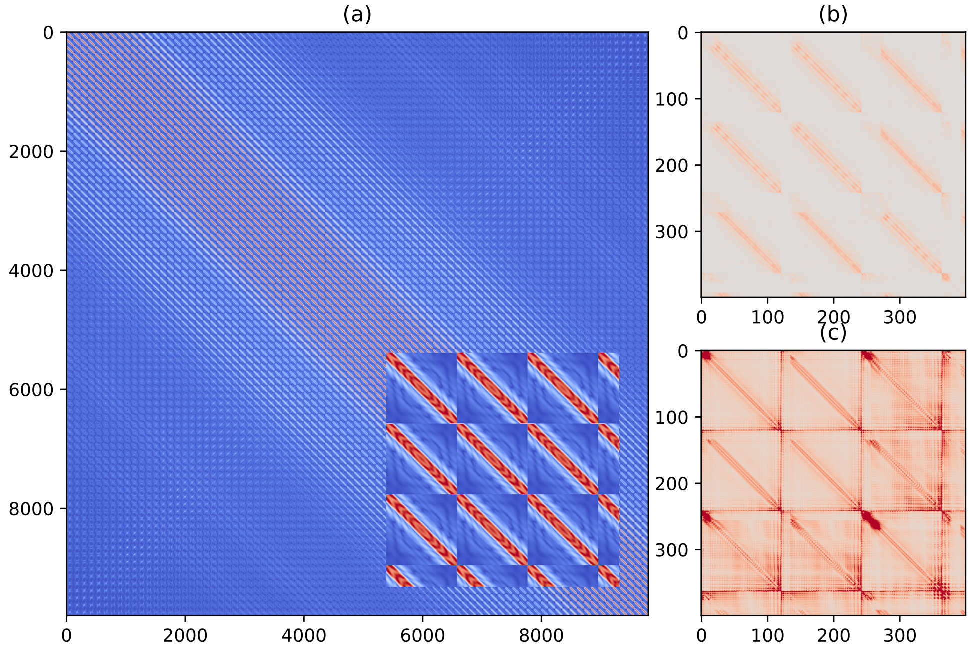
Redatuming carpet @ 650m

Marchenko lines in displays

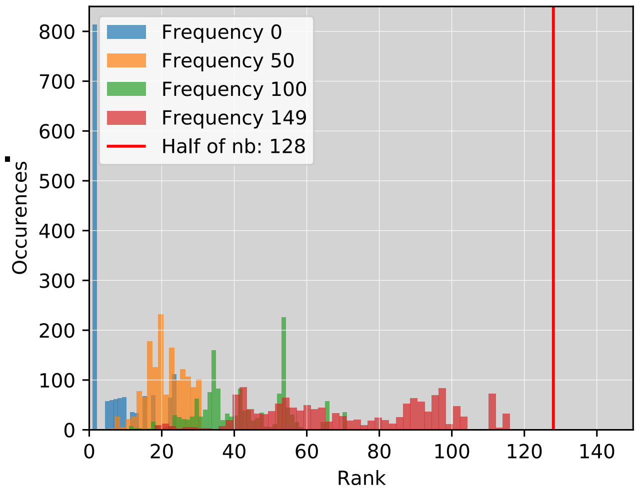
Rank Analysis

The workload is 150 MVMs, where each matrix size is 9801 x 9801. Each matrix corresponds to information from a single frequency.

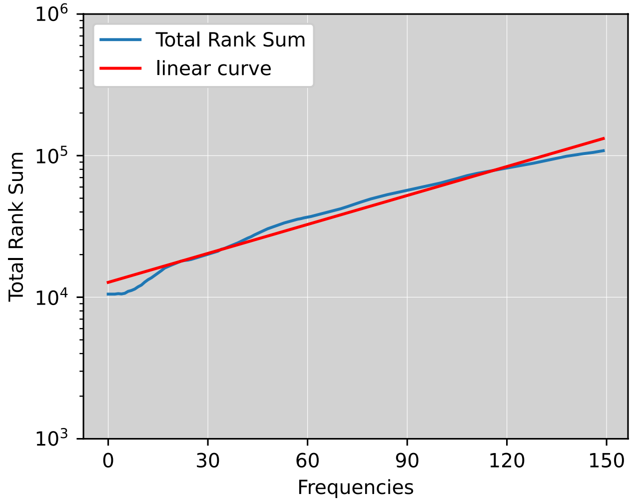
- Low rankness matrix structure of frequency 33 Hz (index=100):



- Rank distribution of tiles: Most matrix ranks are below the critical threshold.

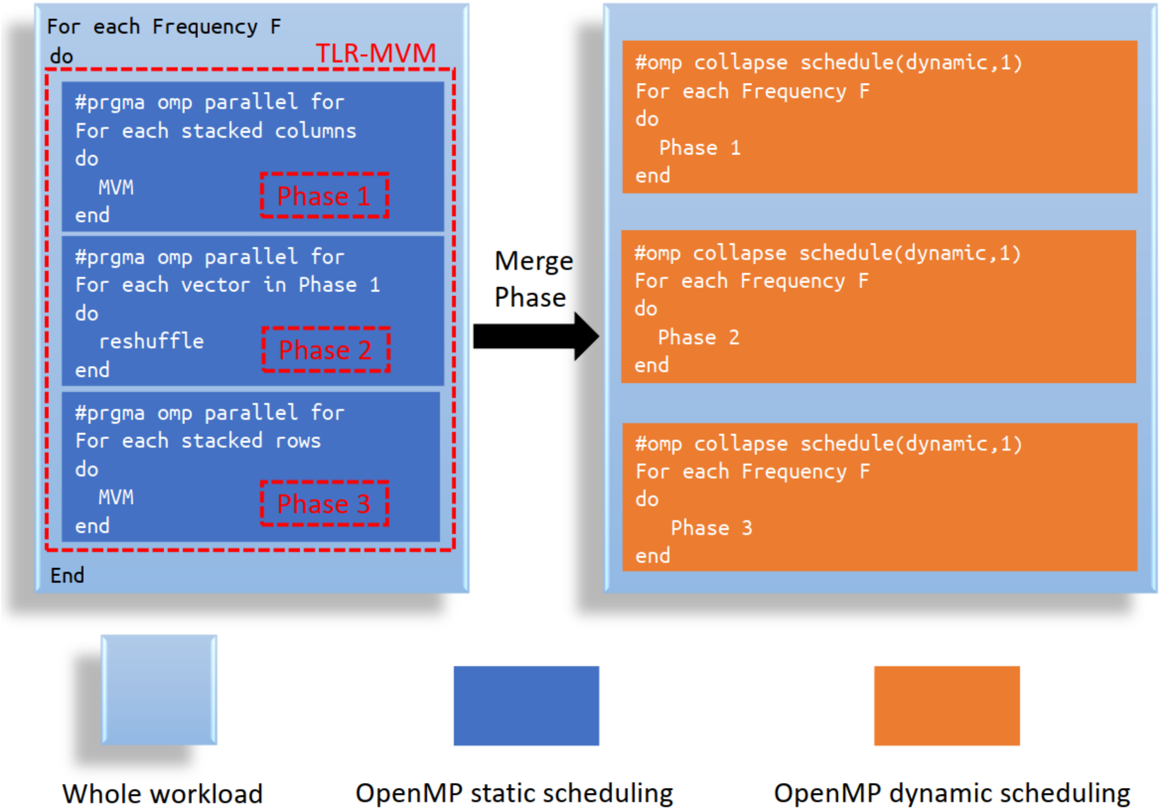


- Rank summation is larger as the matrix frequency index increases.



Load balancing optimizations – Phase Merge Strategy

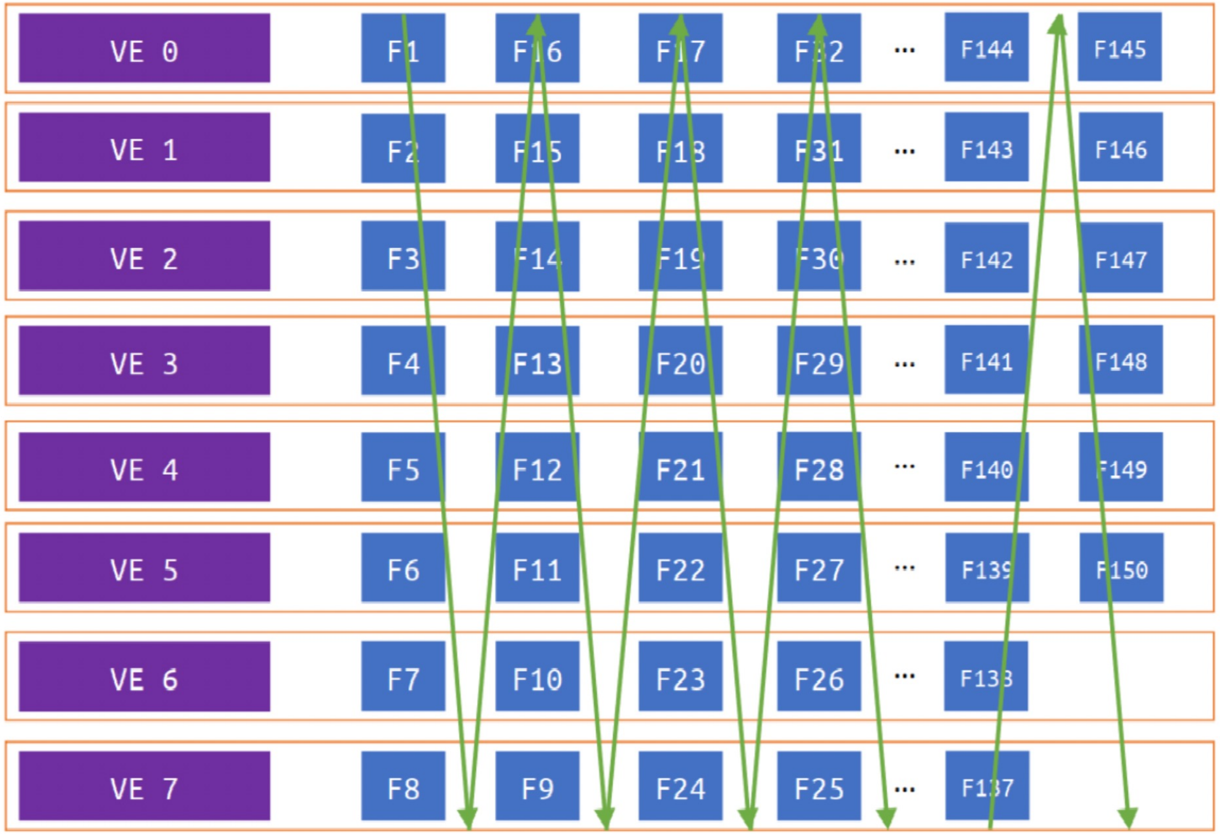
$2F$ synchronization



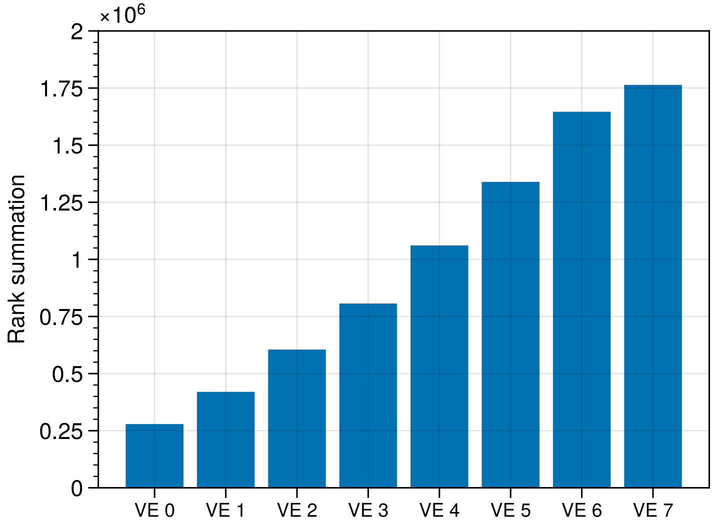
2 synchronization

- Merge phase to reduce synchronization and switch static scheduling to dynamic scheduling for intra-node load balancing.

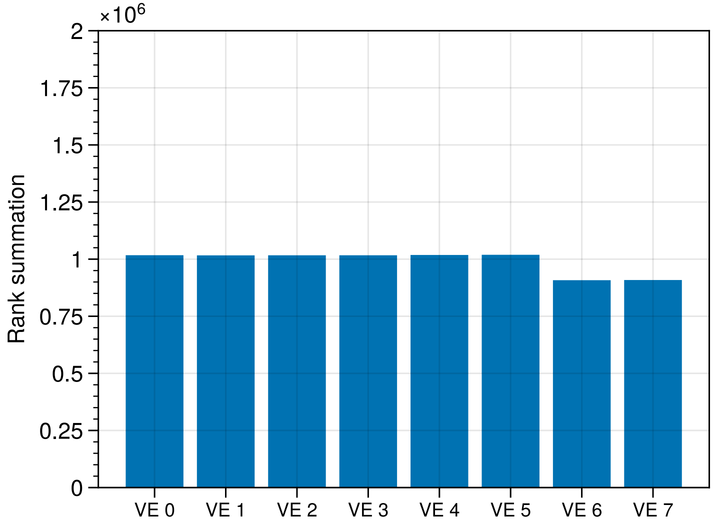
Load balancing optimizations – Zigzag Mapping Strategy



- *Zigzag Mapping strategy* for inter-node load balancing.

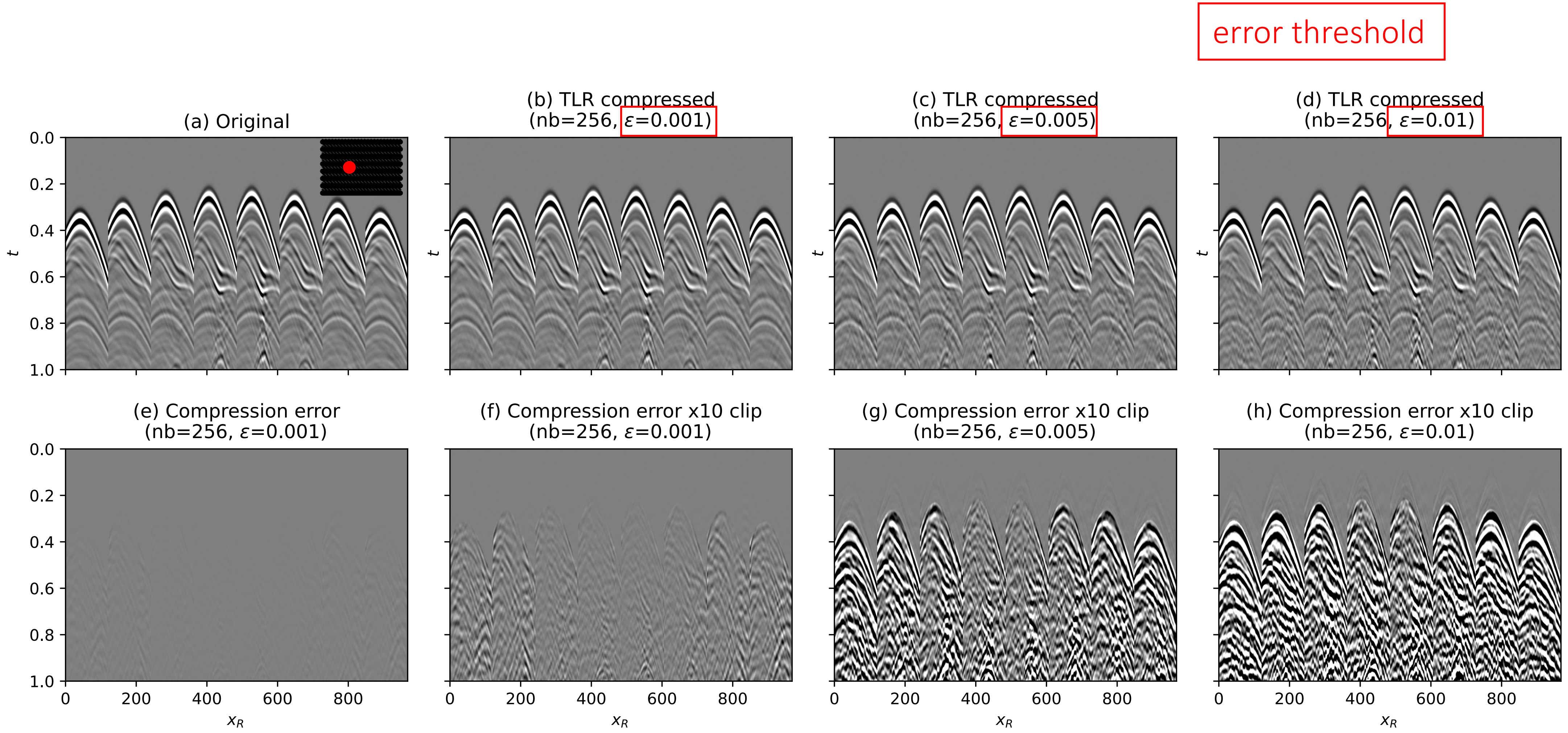


Allocate by chunks



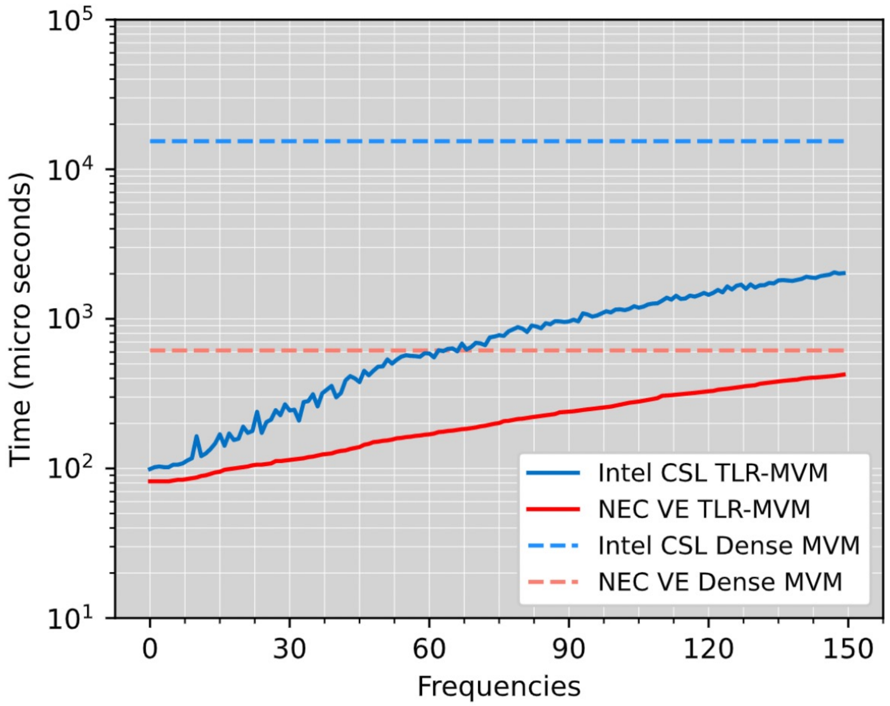
Allocate by Zigzag

TLR-MVM Approximation

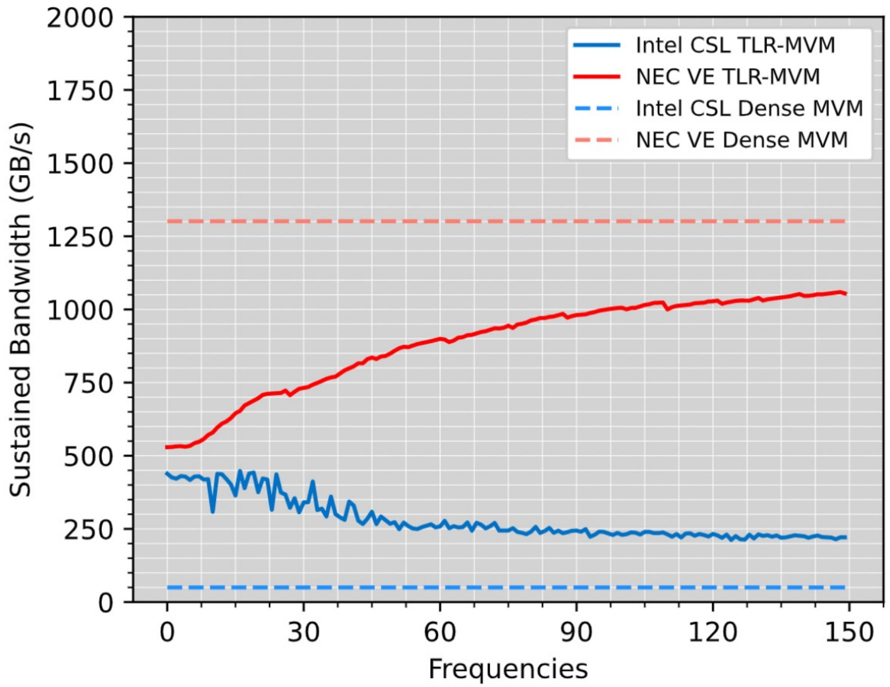


- TLR-MVM introduces negligible error to the final images.

Load imbalance on real seismic dataset

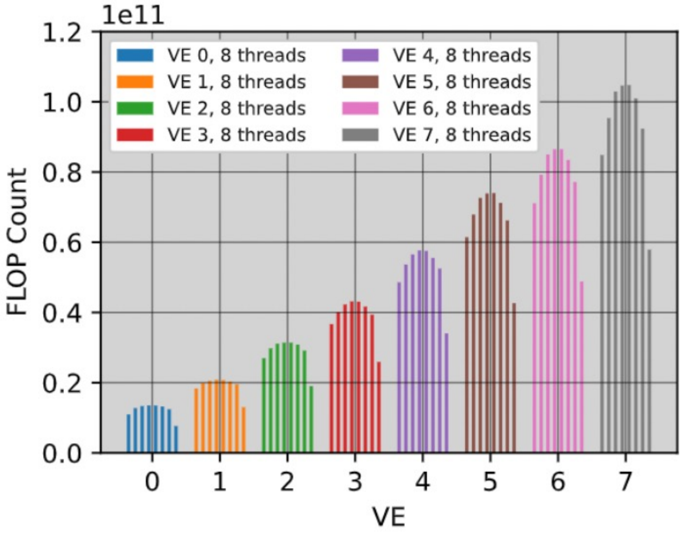


(a) Load imbalance in time

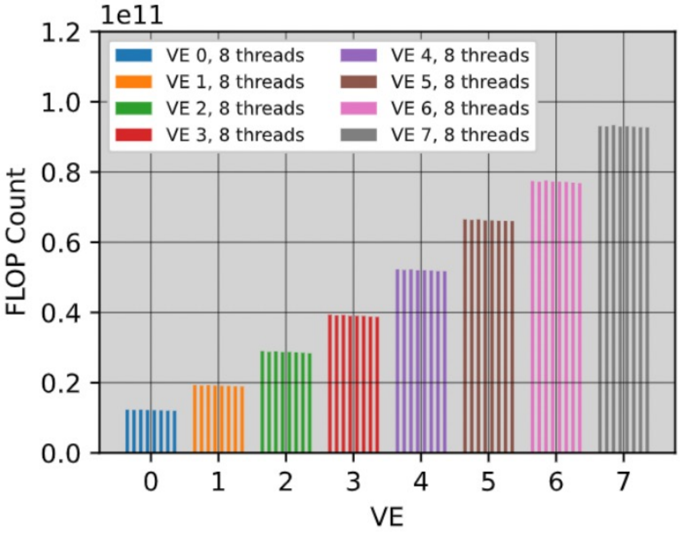


(b) Load imbalance in bandwidth

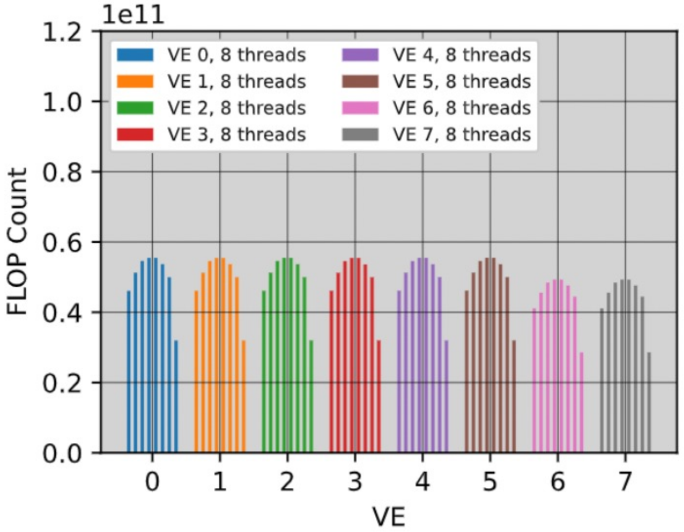
FLOPs counts on 8 VEs using different load balancing strategies



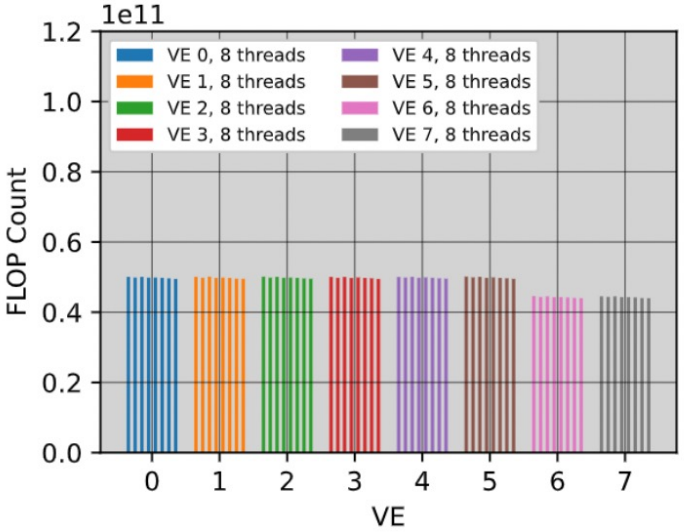
Reference



Merge-Phase

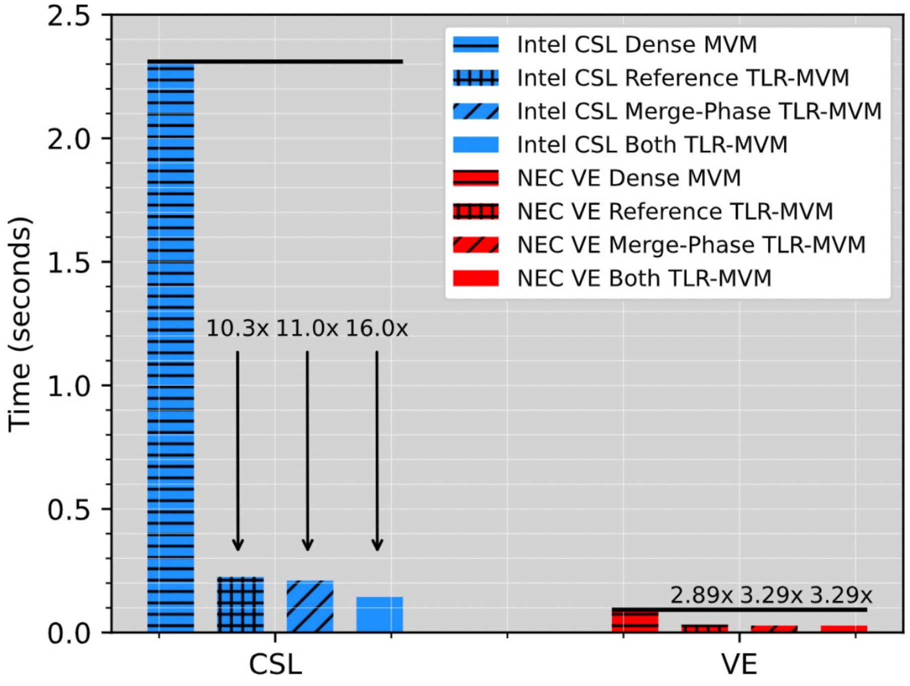


ZigZag mapping

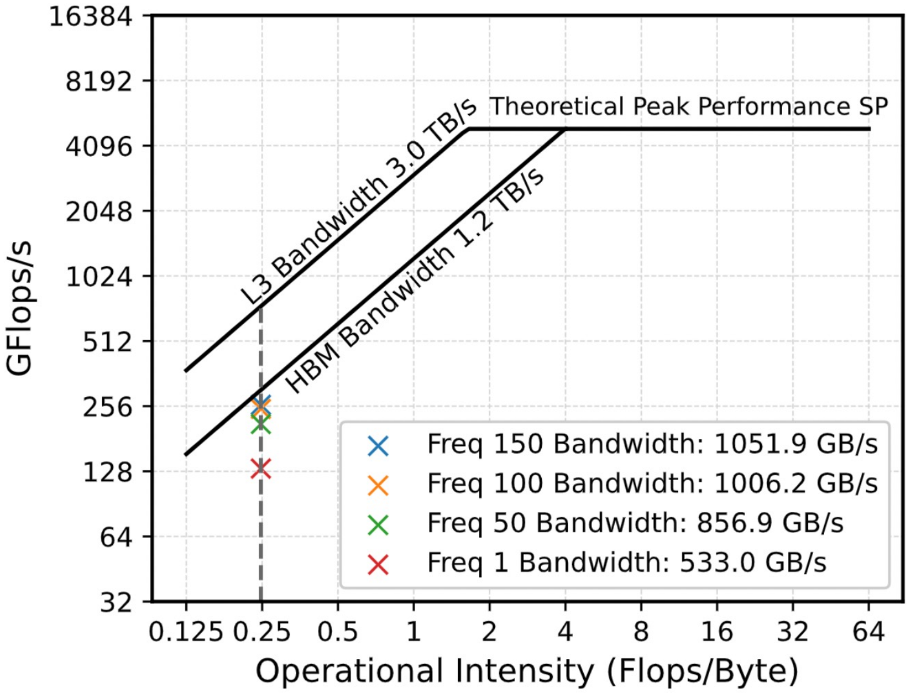


Both (b) and (c)

Performance comparison on Intel and NEC platform

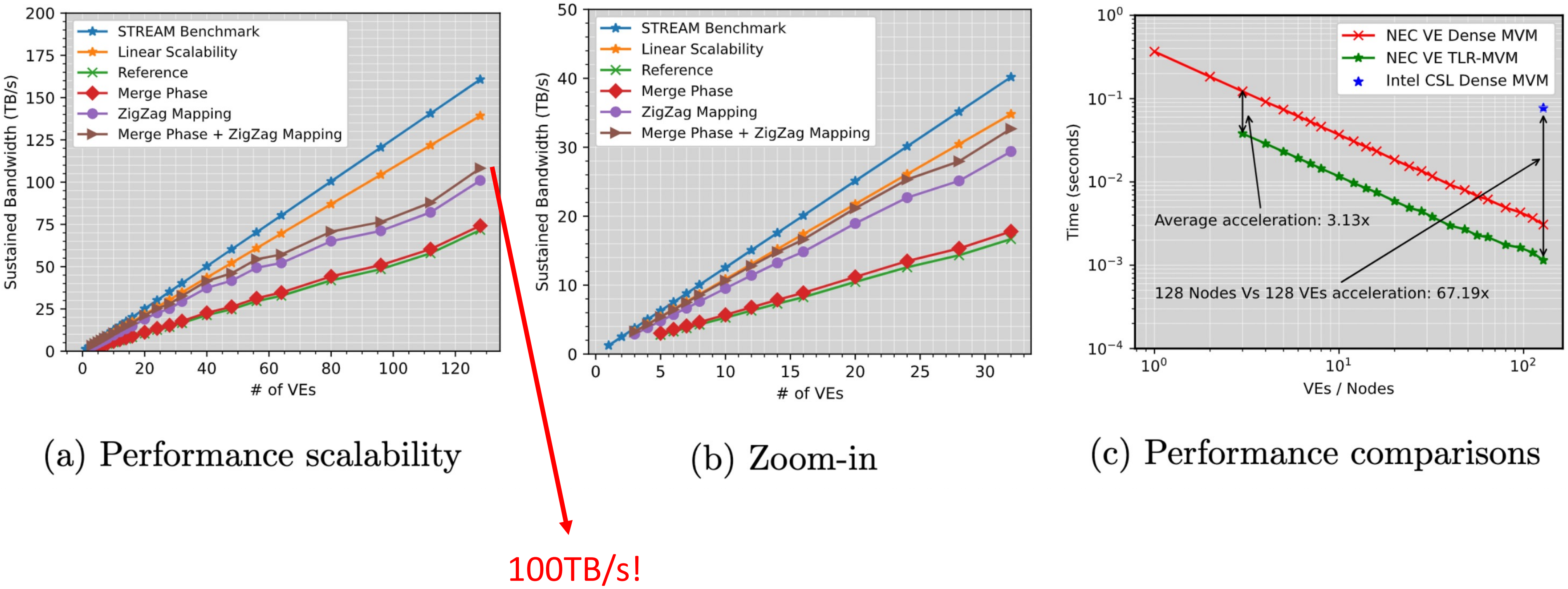


(a) TLR-MVM Vs Dense-MVM



(b) Roofline performance model

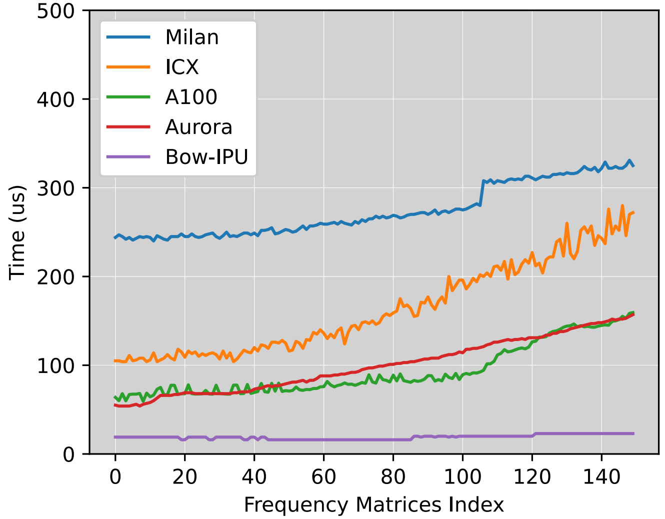
Scalability Results

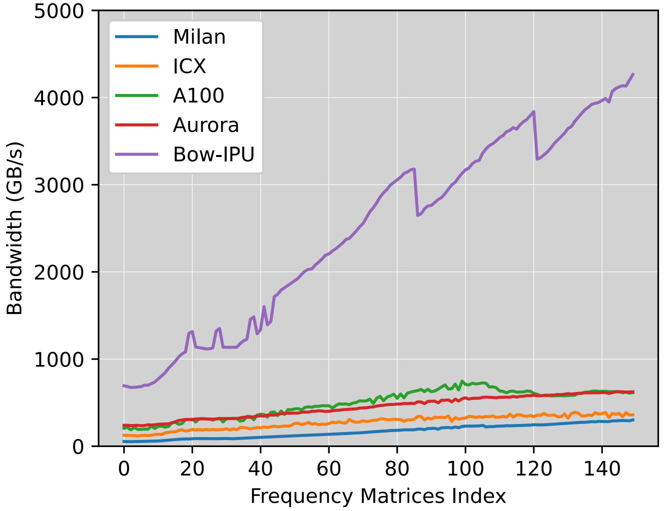


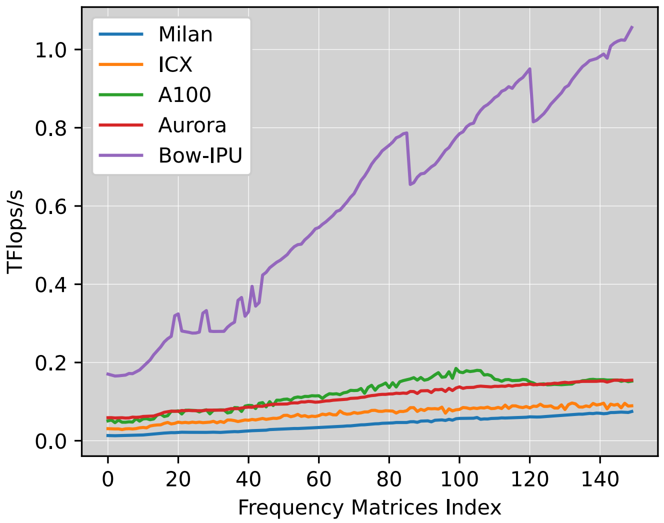
(a) Performance scalability (b) Zoom-in (c) Performance comparisons

TLR-MVM on GraphCore platforms

Will HPC applications benefit from AI accelerators? **Yes!**

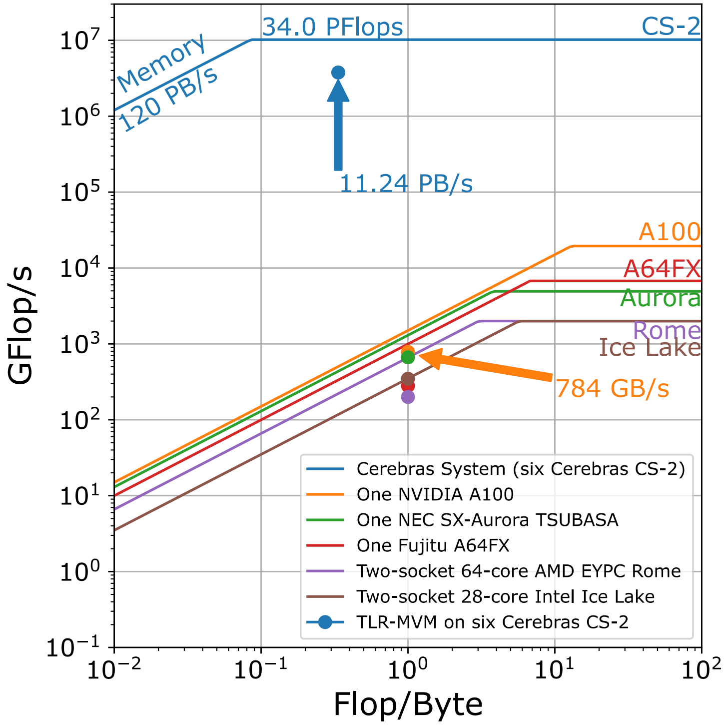


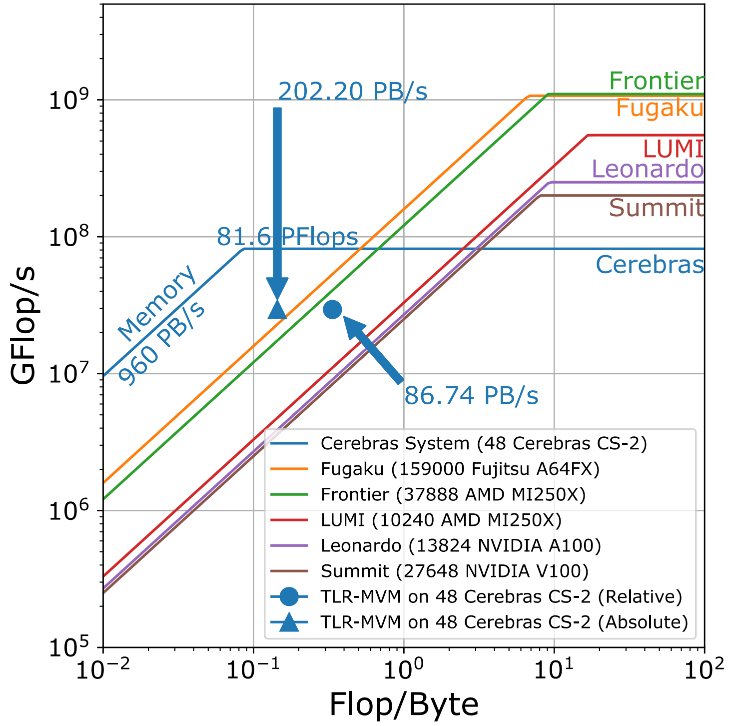




- We conduct the experiments on seismic redautming dataset.
- We execute TLR-MVM for each frequency matrix.
- We have seen a significant performance improvement over other architectures.
- Our implementation scores more than 1 TFlops/s for a memory-bound kernel.

TLR-MVM on Cerebras CS-2 platforms





Remarks

➢ Accelerating the Seismic Redatuming application by

- Designing Batched TLR-MVM
- Evaluating the numerical accuracy and trade-off of compression and acceleration
- Leveraging Merge phase and Zigzag mapping load balancing strategies
- Implement TLR-MVM on GraphCore AI Accelerators

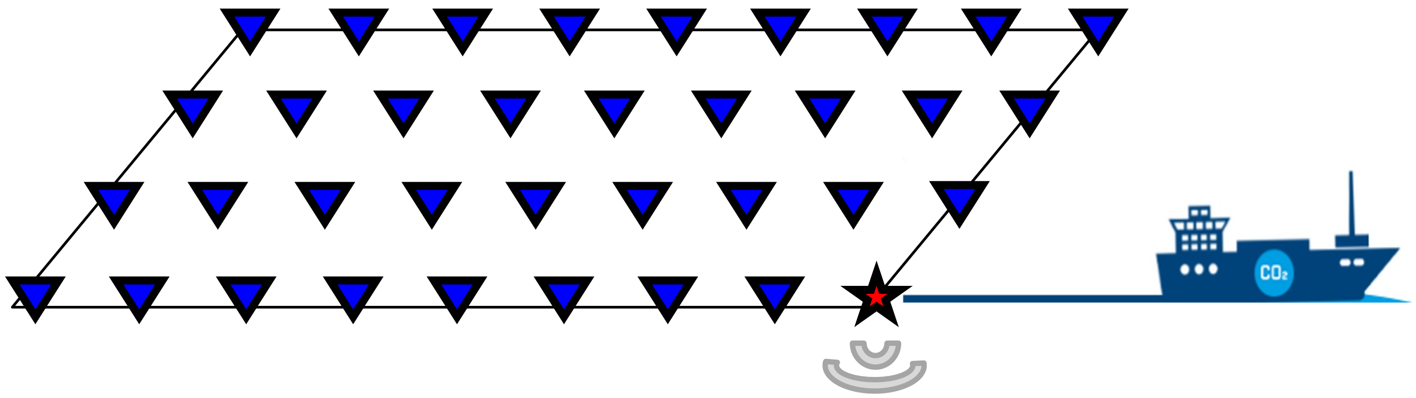
Content

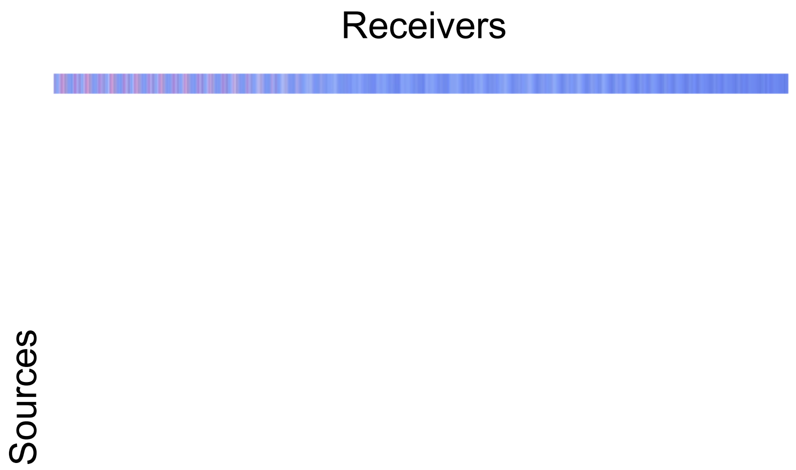
- Introduction
- Adaptive Optics for Ground-based Telescopes
  - Stochastic Levenberg-Marquardt Method in Soft Real-Time Controller
  - Discrete Algebraic Riccati Equation in Soft Real-Time Controller
  - TLR-MVM in Hard Real-Time Controller
- Seismic Redatuming by Inversion Using Marchenko-based Methods
  - Batched TLR-MVM
  - Mixed-Precision Batched TLR-MVM on NVIDIA GPU

Mixed-Precision TLR-MVM in Seismic Redatuming

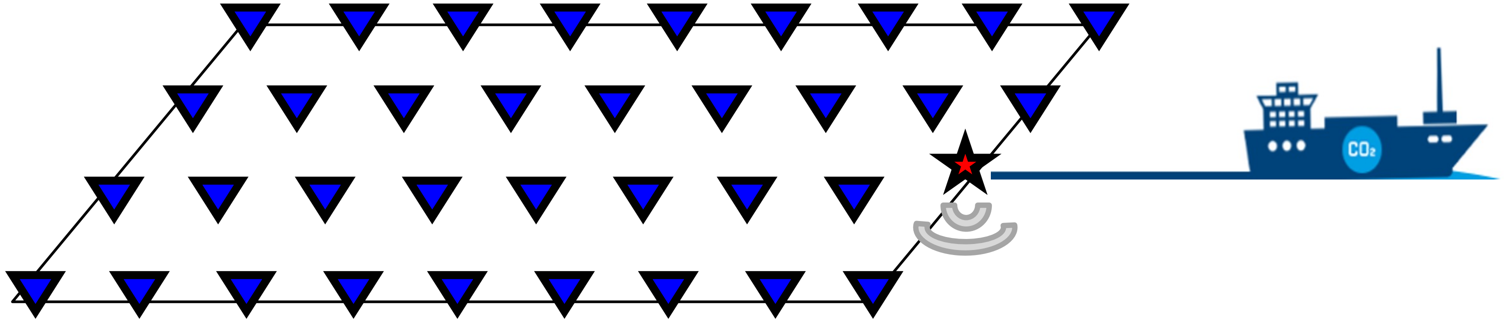
- Distance-aware matrix reordering method – a geometric way to exploit data sparsity
- Explore GPU implementation
- Leveraging lower precision such as FP16 or INT8

Distance-aware matrix reordering method





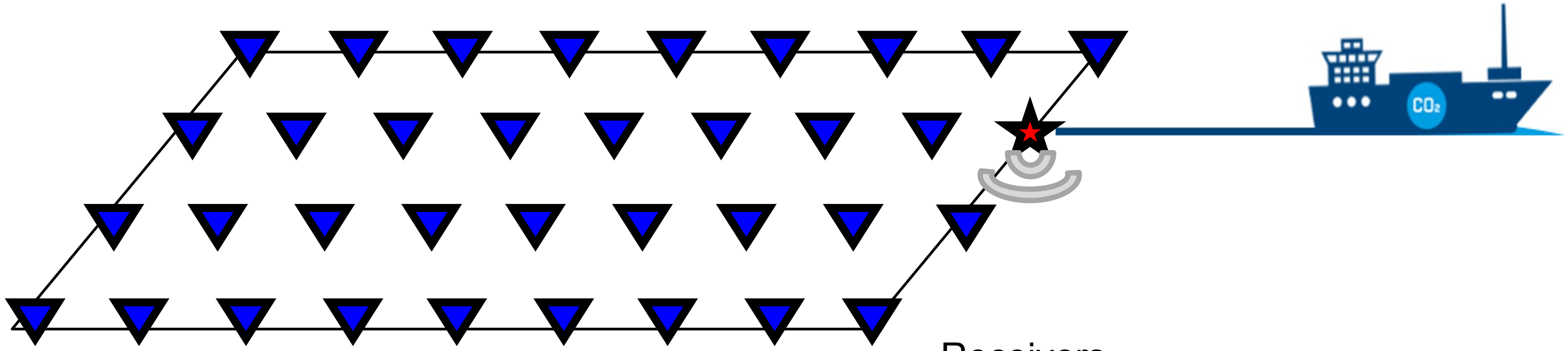
Distance-aware matrix reordering method



Receivers

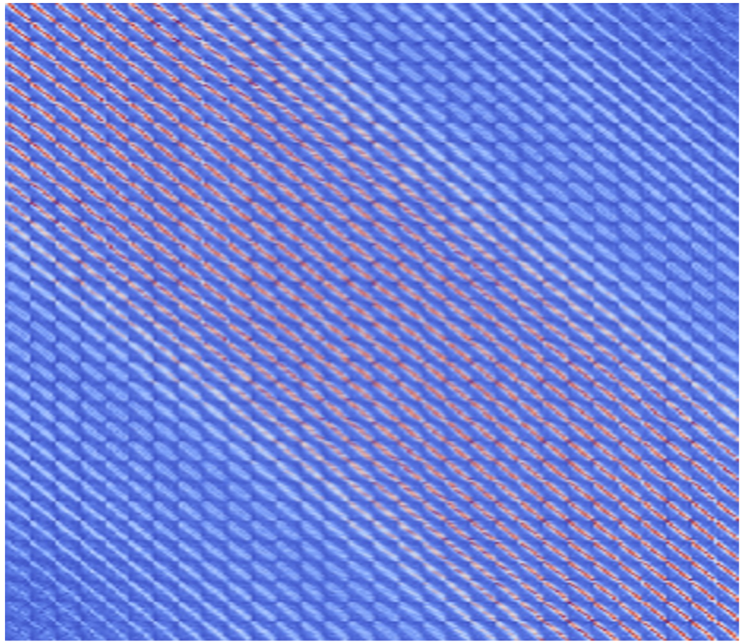
Sources

Distance-aware matrix reordering method



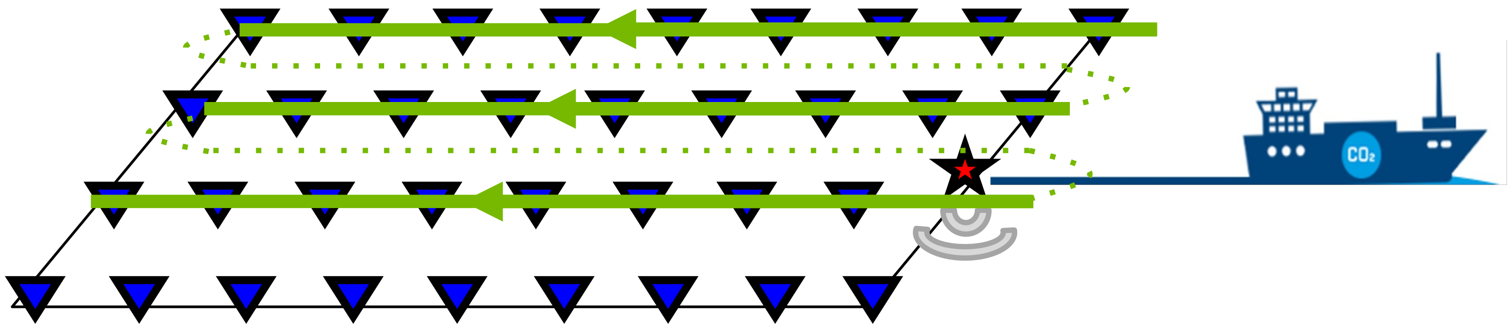
Receivers

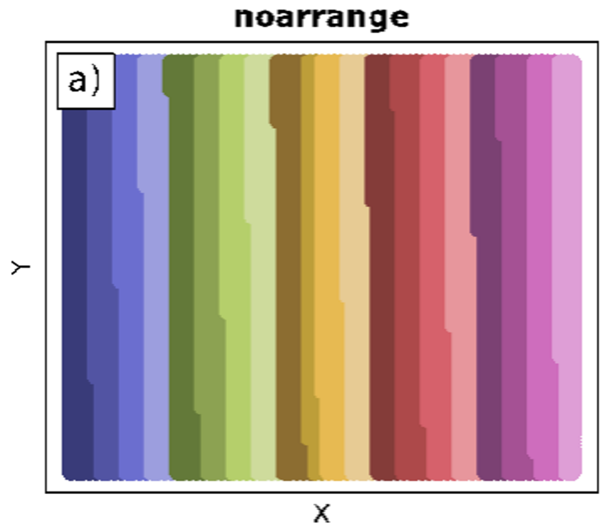
Sources

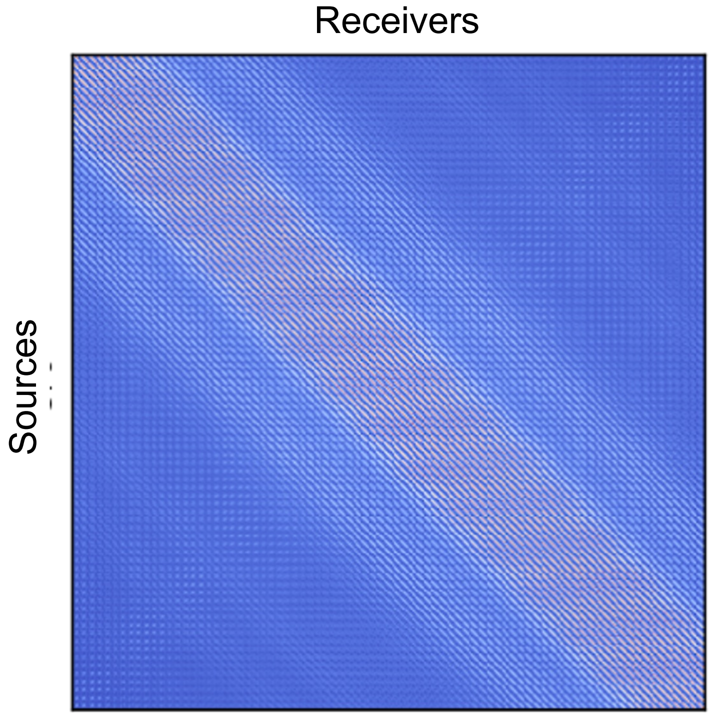


Single frequency matrix

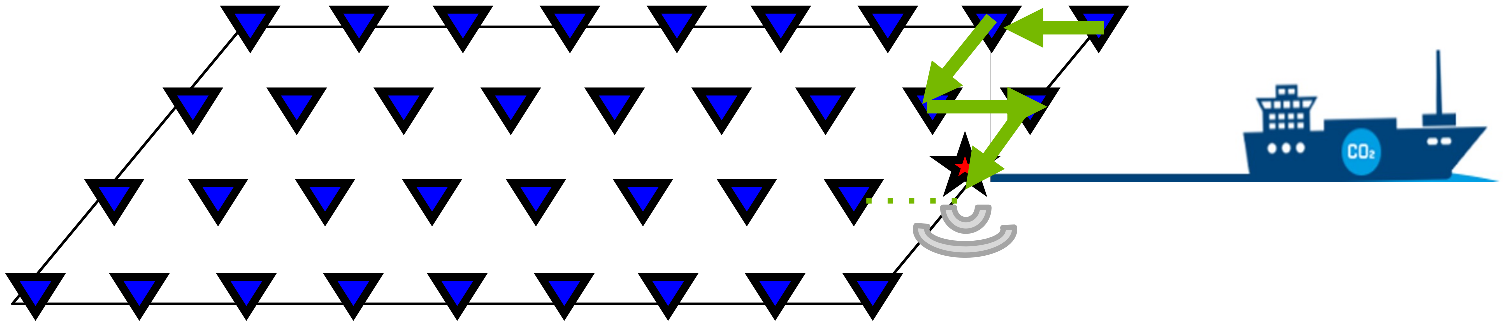
Distance-aware matrix reordering method

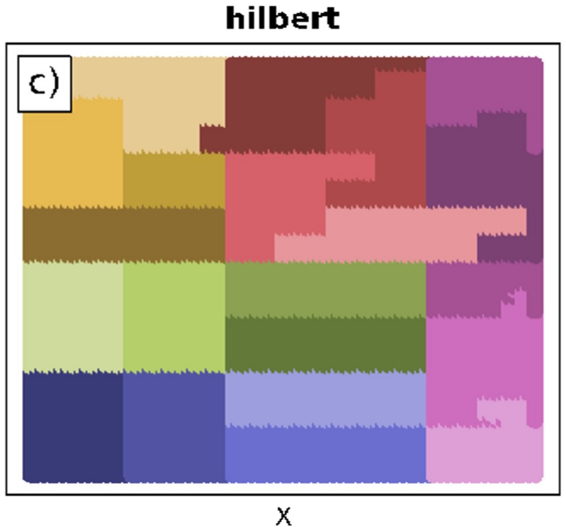


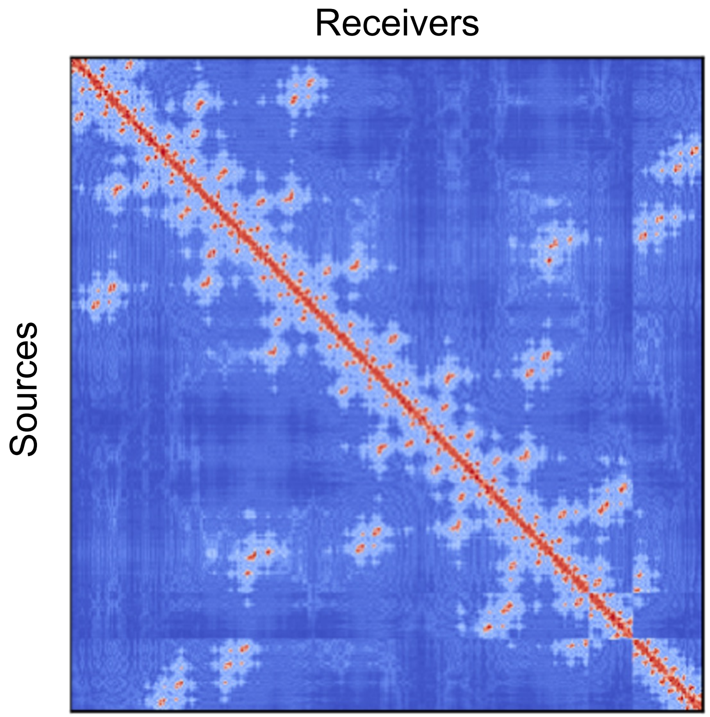




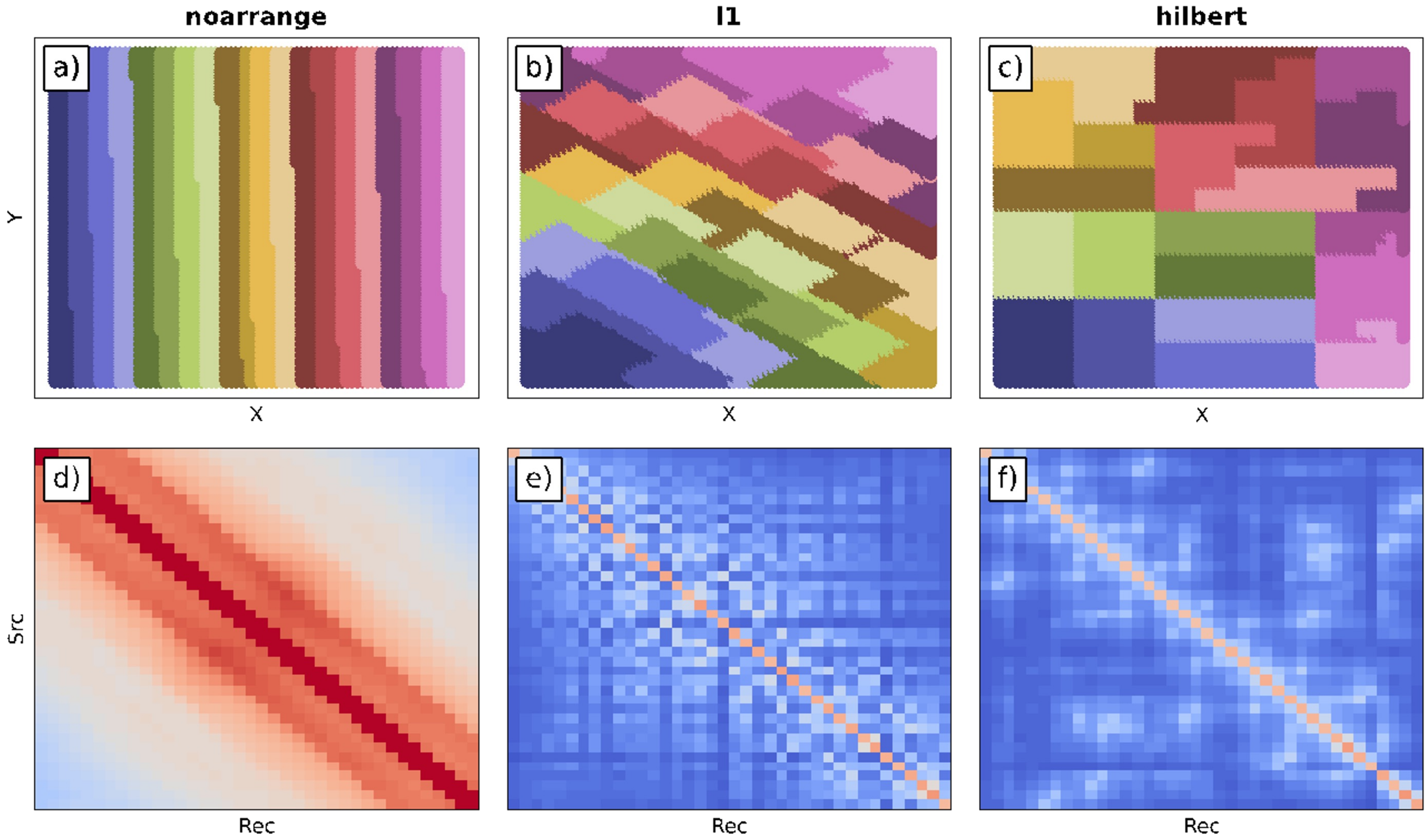
Distance-aware matrix reordering method





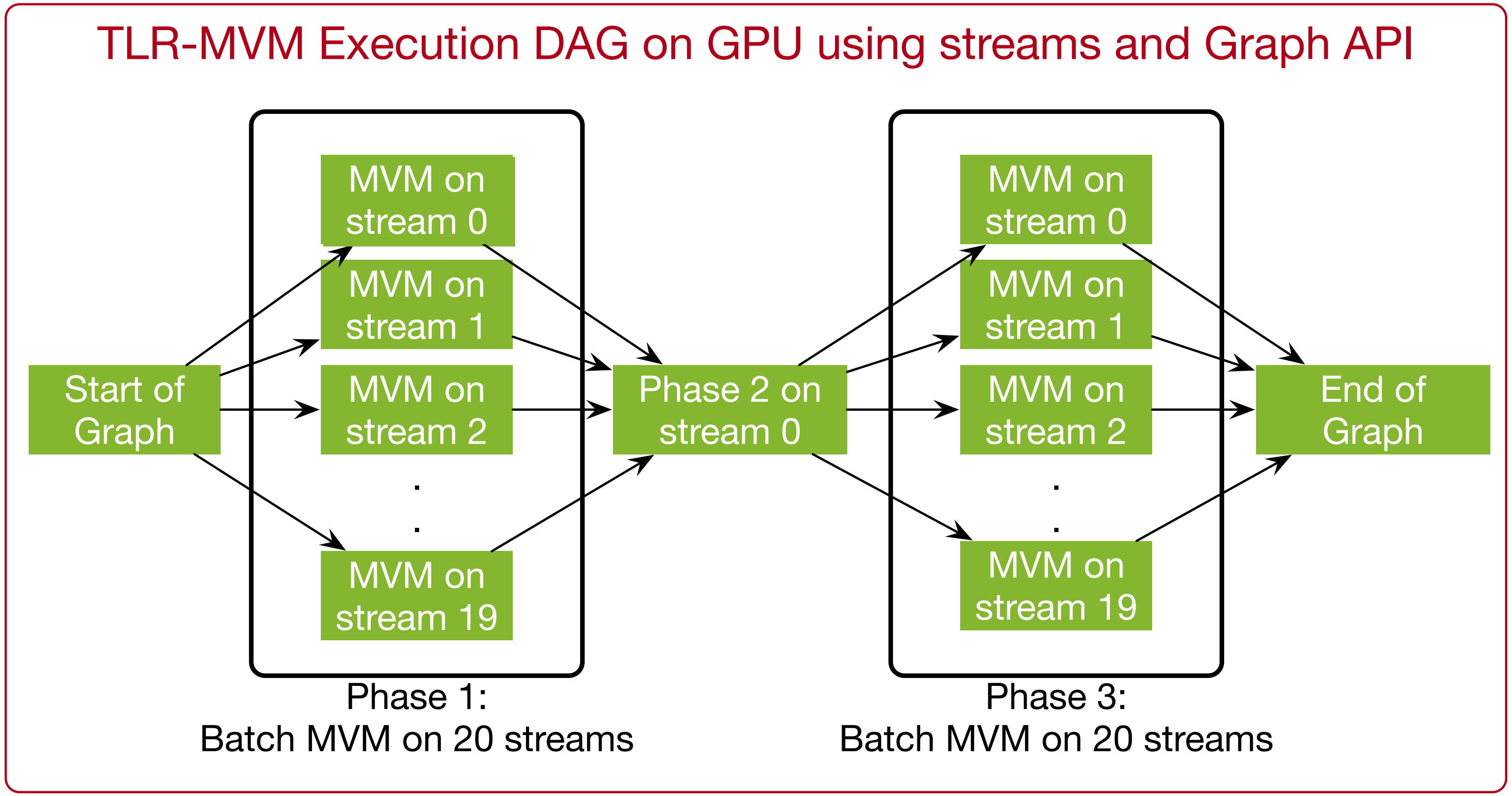


Distance-aware matrix reordering method

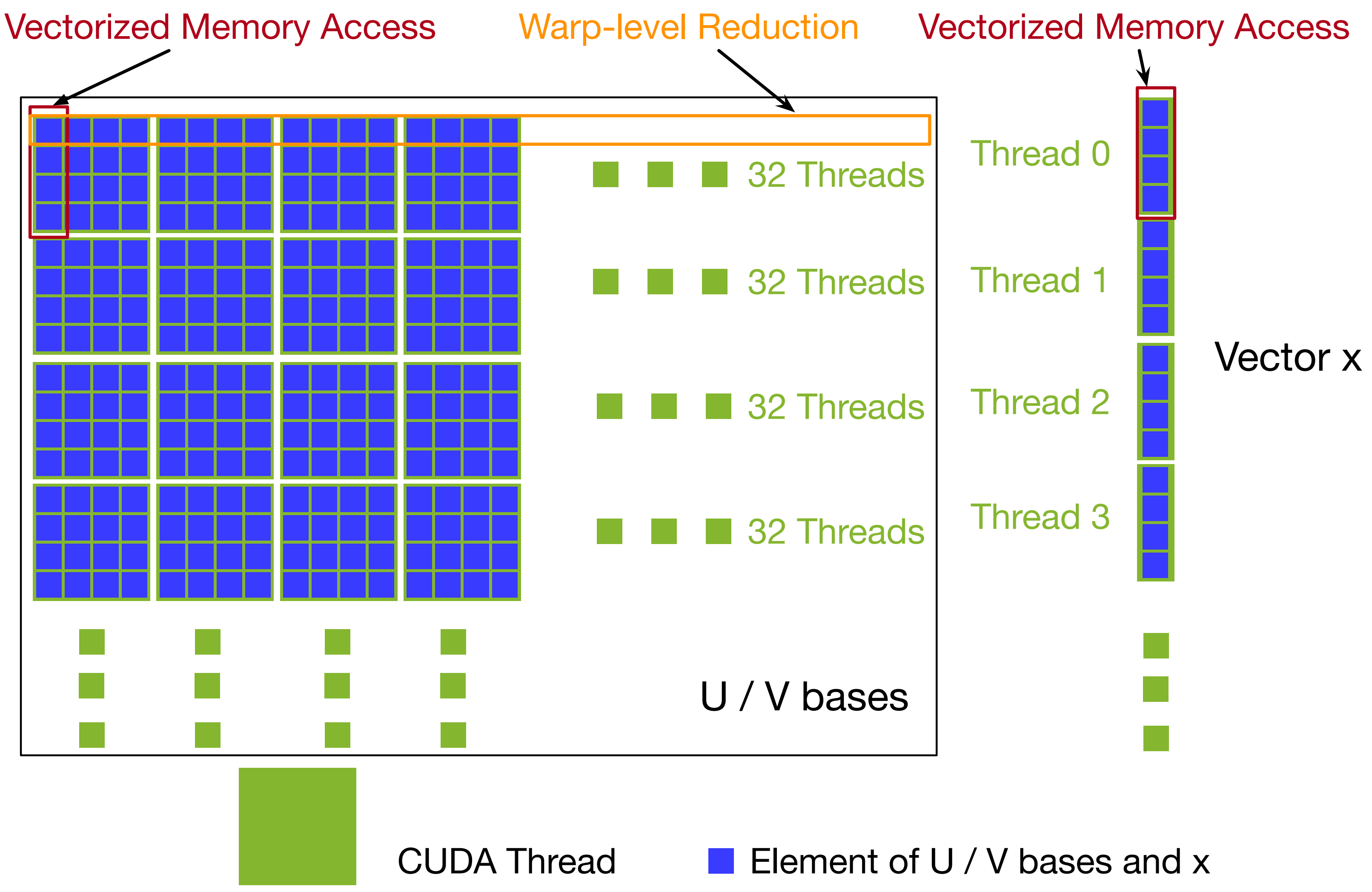


Data size: 29 GB (C. Factor ≈ 3.97) 7.5 GB (C. Factor ≈ 15.36) 6.6 GB (C. Factor ≈ 17.46)

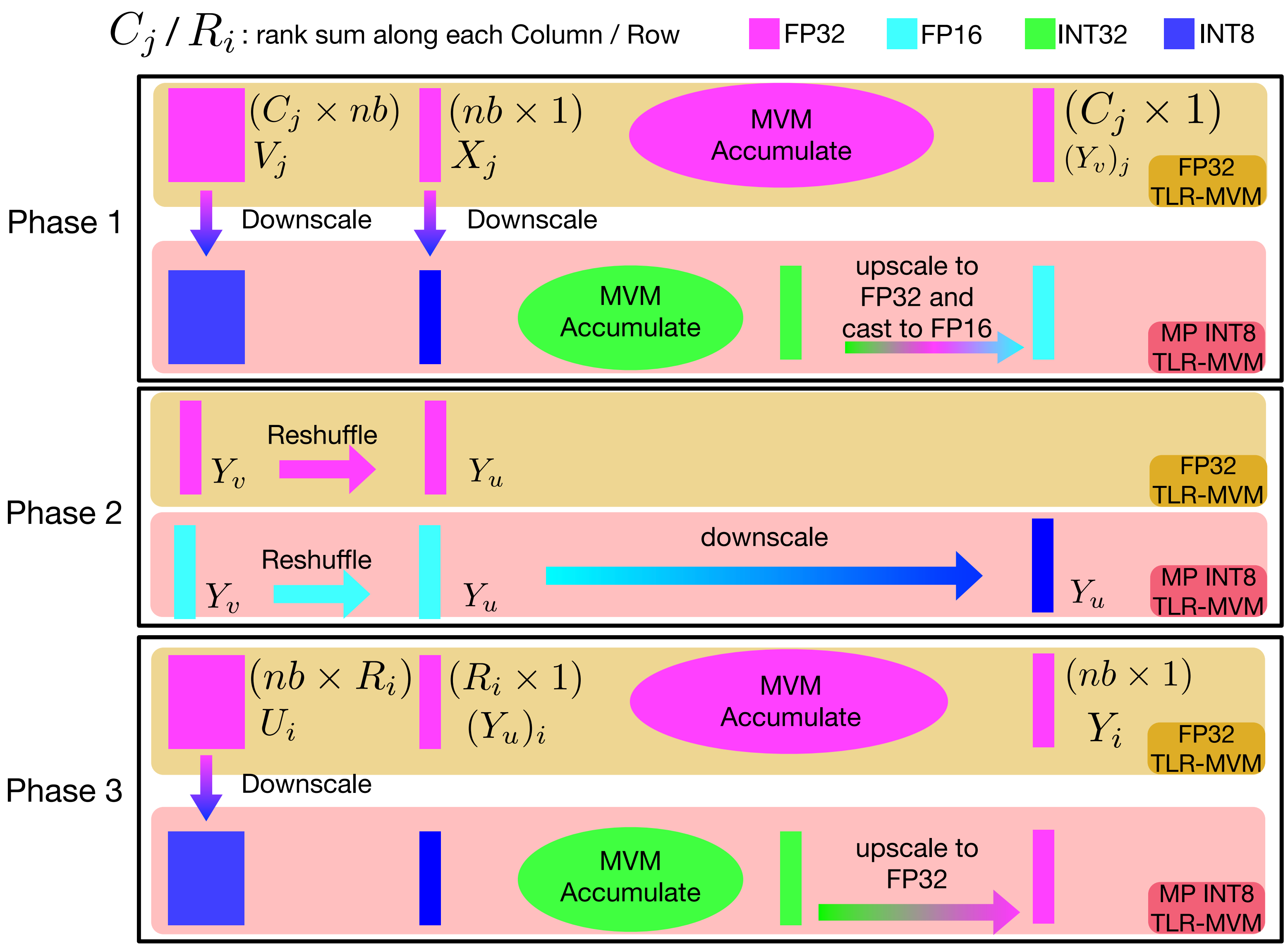
Graph Representation of TLR-MVM



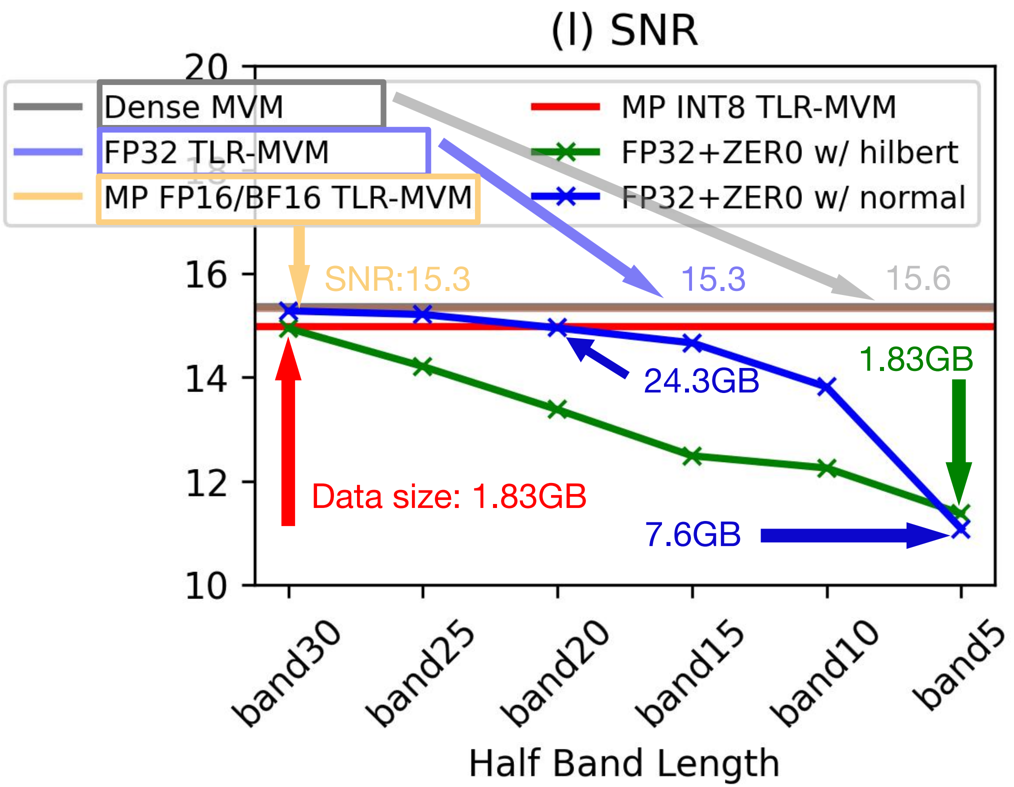
CUDA Threads Layout and Kernel Optimizations



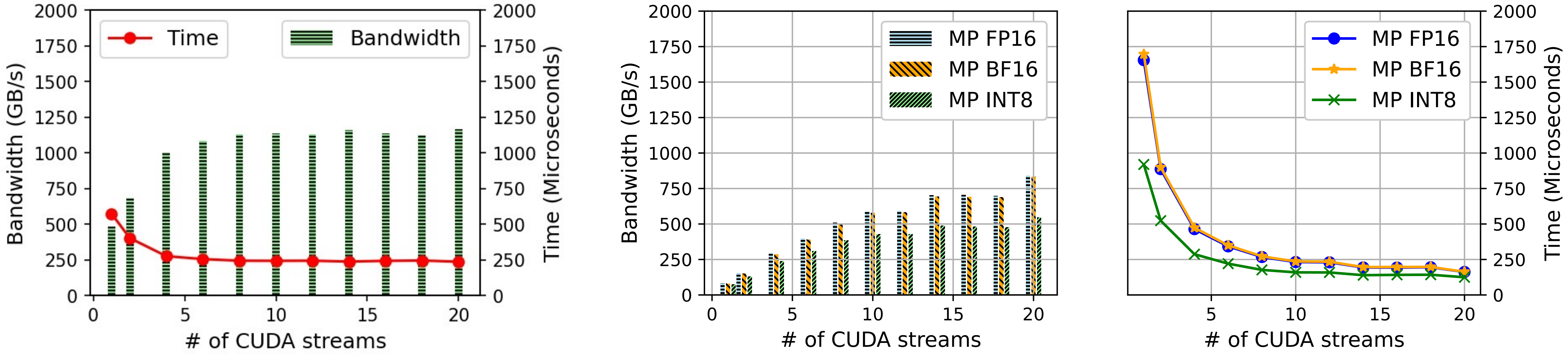
MP INT8 TLR-MVM Explanation



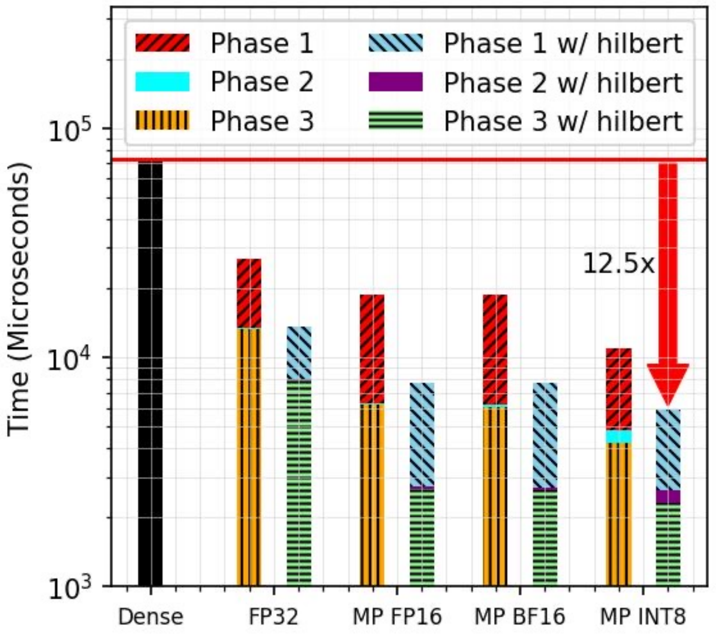
Numerical Accuracy using MP TLR-MVM



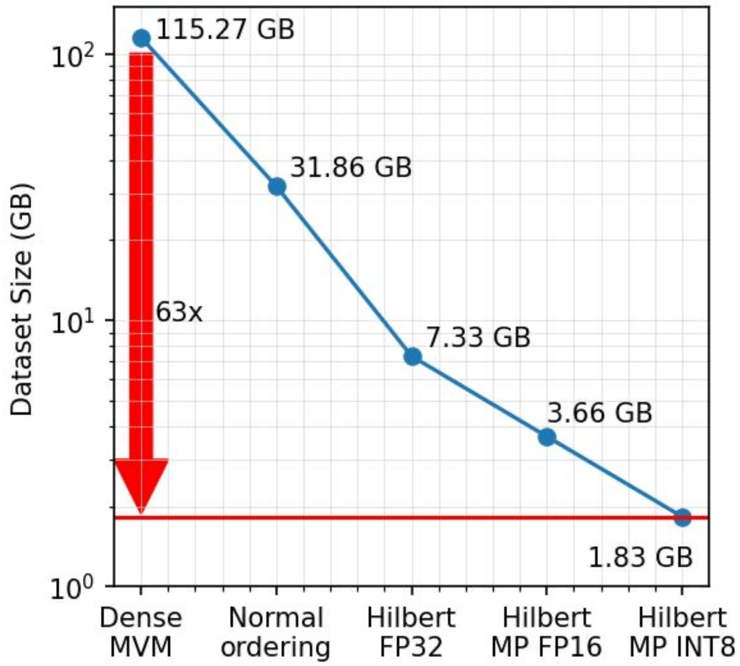
Impact of # of CUDA Streams on Performance



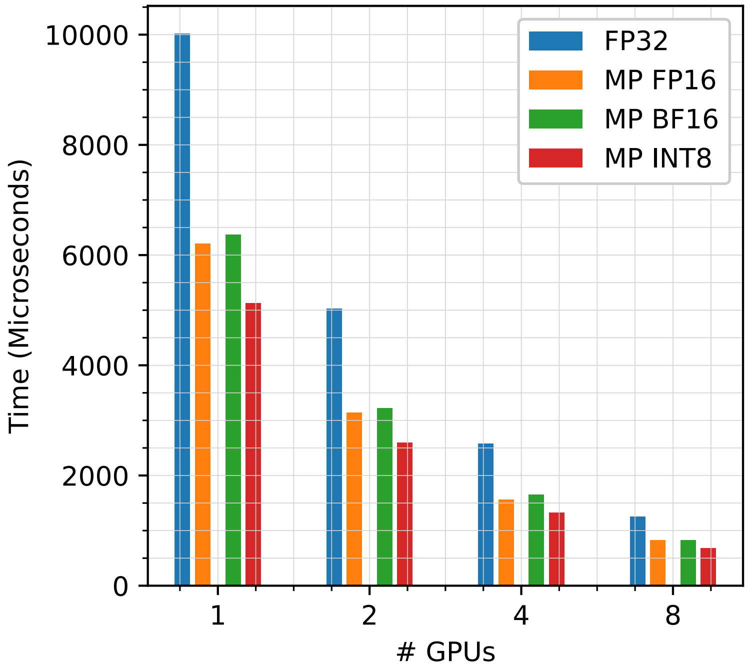
Mixed-Precisions TLR-MVM Results



Time Breakdown

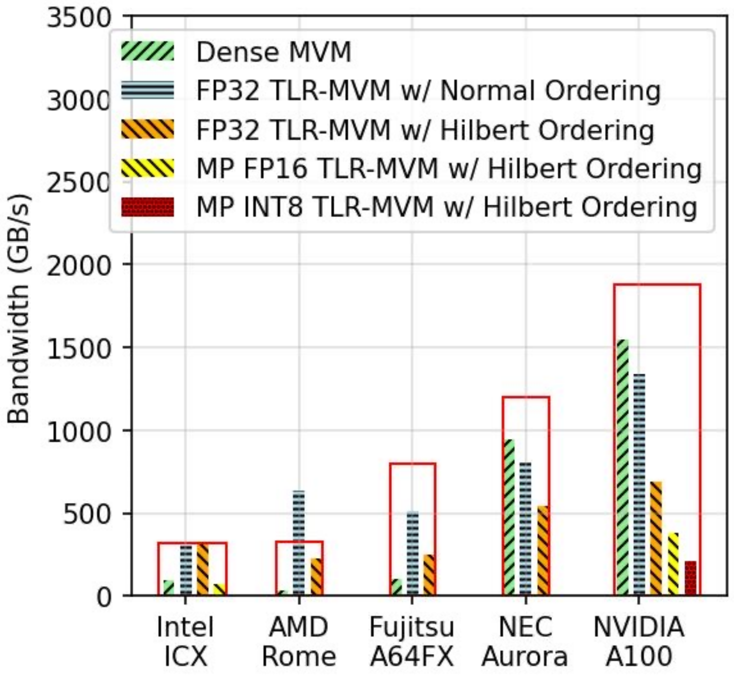


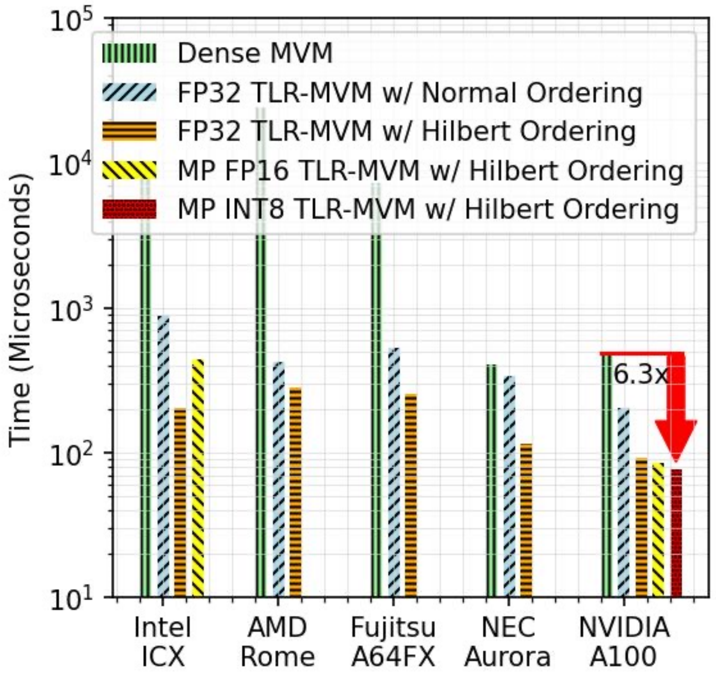
Dataset Reduction



Scalability Results

Comparison with other architectures





Remarks

➢ Accelerating the seismic redatuming applications further by

- Designing Mixed Precision TLR-MVM
- Evaluating the numerical accuracy and trade-off of Mixed Precision TLR-MVM
- Distance-aware matrix reordering method and Hilbert ordering
- CUDA Graph API

Selected Publications

Stochastic Levenberg-Marquardt method (SLM)

- **Yuxi Hong**, El Houcine Bergou, Nicolas Doucet, Hao Zhang, Jesse Cranney, Hatem Ltaief, Damien Gratadour, Francois Rigaut, and David Keyes. 2021. **Outsmarting the Atmospheric Turbulence for Ground-Based Telescopes Using the Stochastic Levenberg-Marquardt Method.** In 27th International Conference on Parallel and Distributed Computing, Lisbon, Portugal, September 1–3, 2021, Proceedings. Springer-Verlag, Berlin, Heidelberg, 565–579. (EuroPar 21) https://doi.org/10.1007/978-3-030-85665-6_35

Tile Low-Rank Matrix Vector Multiplication (TLR-MVM)

- Hatem Ltaief, Jesse Cranney, Damien Gratadour, **Yuxi Hong**, Laurent Gatineau, and David Keyes. 2021. **Meeting the real-time challenges of ground-based telescopes using low-rank matrix computations.** In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 21). Association for Computing Machinery, New York, NY, USA, Article 29, 1–16.https://doi.org/10.1145/3458817.3476225

- Hatem Ltaief, **Yuxi Hong**, Adel Dabah, Rabab Alomairy, Sameh Abdulah, Chris Goreczny, Pawel Gepner, Matteo Ravasi, Damien Gratadour, David Keyes. Steering Customized AI Architectures for HPC Scientific Applications. In International Conference on High Performance Computing (ISC 23). https://doi.org/10.1007/978-3-031-32041-5_7

Selected Publications

Batched TLR-MVM

- **Yuxi Hong**, Hatem Ltaief, Matteo Ravasi, Laurent Gatineau, and David Keyes. 2021. **Accelerating Seismic Redatuming Using Tile Low-Rank Approximations on NEC SX-Aurora TSUBASA.** In Supercomputing Frontiers and Innovations, 8(2), 6-26 (SFI). https://doi.org/10.14529/jsfi210201

- Hatem Ltaief, **Yuxi Hong**, Adel Dabah, Rabab Alomairy, Sameh Abdulah, Chris Goreczny, Pawel Gepner, Matteo Ravasi, Damien Gratadour, David Keyes. Steering Customized AI Architectures for HPC Scientific Applications. In International Conference on High Performance Computing (ISC 23). https://doi.org/10.1007/978-3-031-32041-5_7

- Hatem Ltaief, **Yuxi Hong**, Leighton Wilson, Mathias Jacquelin, Matteo Ravasi, David Keyes. **Scaling the "Memory Wall" for Multi-Dimensional Seismic Processing with Algebraic Compression on Cerebras CS-2 Systems.** (Submitted to ACM Gordon Bell 2023)

- Matteo Ravasi, **Yuxi Hong**, Hatem Ltaief, David Keyes, David Vargas. 2022. **Tile-Low Rank Compressed Multi-Dimensional Convolution and Its Application to Seismic Redatuming Problems** Second International Meeting for Applied Geoscience & Energy (IMAGE 22). https://doi.org/10.1190/image2022-3744978.1

- Matteo Ravasi, **Yuxi Hong**, Hatem Ltaief, David Keyes, David Vargas. 2022. **Large-scale Marchenko imaging with distance-aware matrix reordering, tile low-rank compression, and mixed-precision computations.** Second International Meeting for Applied Geoscience & Energy (IMAGE 22). https://doi.org/10.1190/image2022-3744978.1

- **Yuxi Hong**, Matteo Ravasi, Hatem Ltaief, David Keyes. 2023**. Can tile low-rank compression live up to expectations? An application to 3D multi-dimensional deconvolution.** The International Meeting for Applied Geoscience & Energy (Accepted by IMAGE 23).

Selected Publications

Mixed Precision Batched TLR-MVM

- Yuxi Hong, Hatem Ltaief, Matteo Ravasi, Laurent Gatineau, and David Keyes. 2022. **High-Performance Seismic Redatuming by Inversion Using Algebraic Compression and Mixed Precisions.** (Submitted to IJHPCA)

Related Software Releases

SHIPS

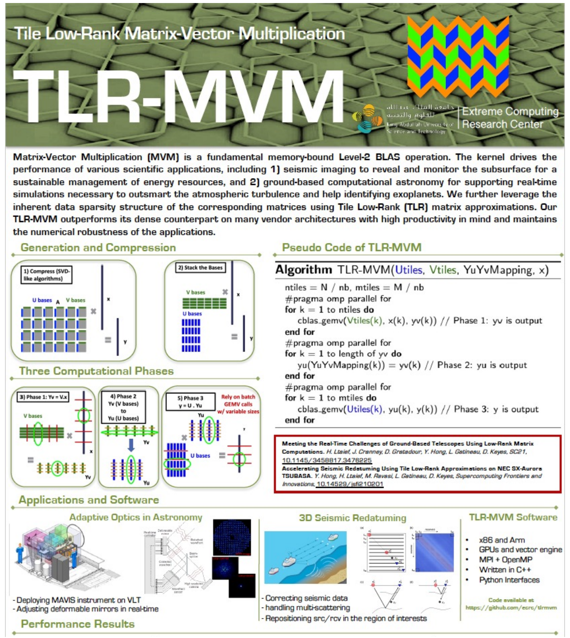
High Performance AO control system software

SLM inside

DARE

DARE

TLR-MVM



TLR-MVM

TLR-MDC

Tile-Low Rank Multi-Dimensional Convolution: fast MDC modelling and inversion for seismic applications

Seismic Redatuming framework

Thank you Q & A

Content

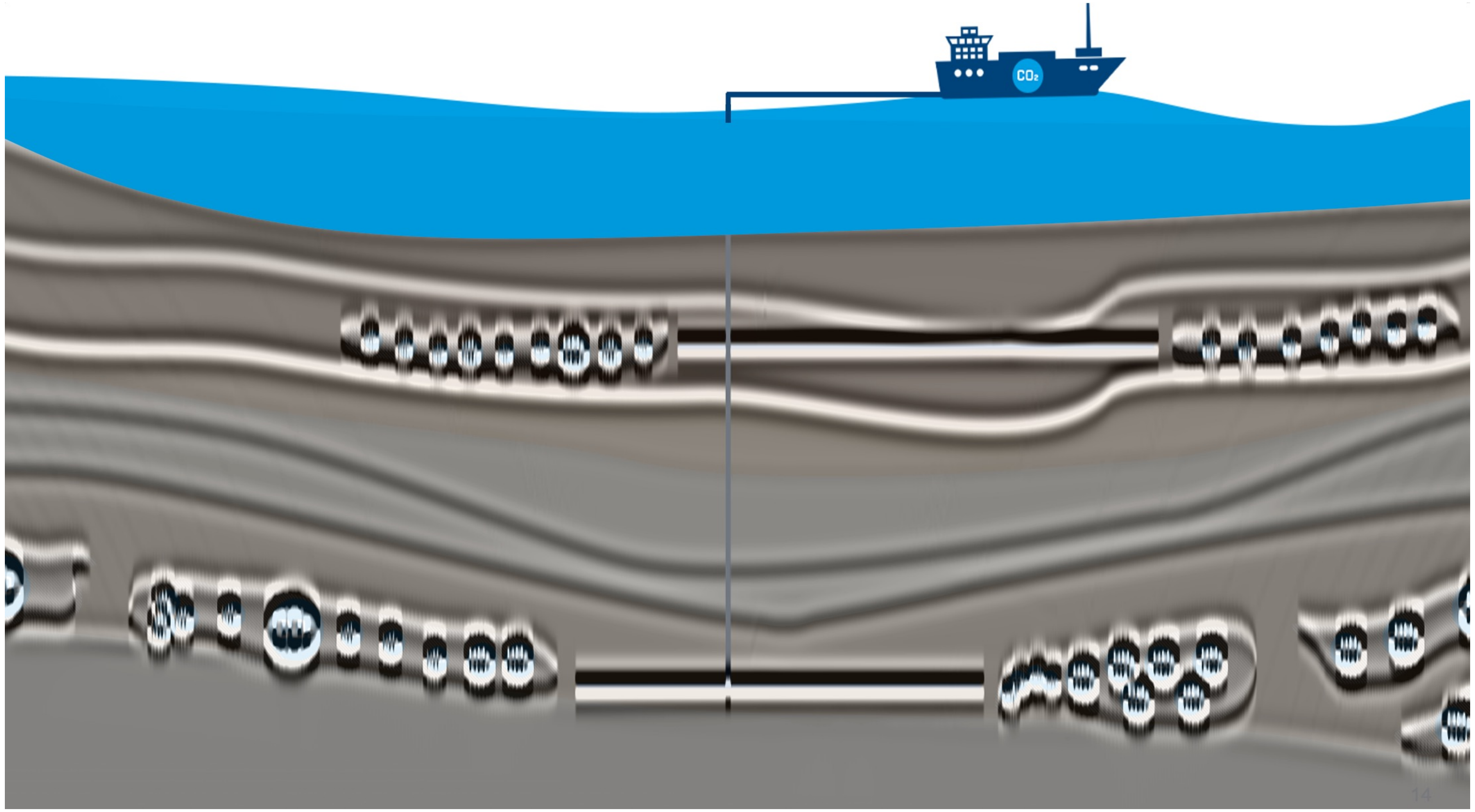
- Introduction

- Computational Astronomy for Ground-based Telescopes
  - Soft Real-Time Controller
  - Hard Real-Time Controller

- Seismic Redatuming Inversion Using Marchenko-based Methods
  - Batched TLR-MVM
  - Mixed-Precision Batched TLR-MVM

Seismic Imaging – an explorer perspective

King Abdullah University
of Science and Technology



Acknowledgement

David Keyes

- Ph.D. Advisor

Matteo Ravasi

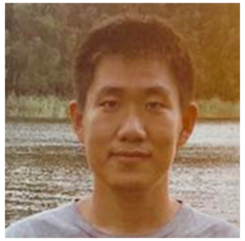
- Ph.D. Advisor

Hatem Ltaief

- Ph.D. Advisor

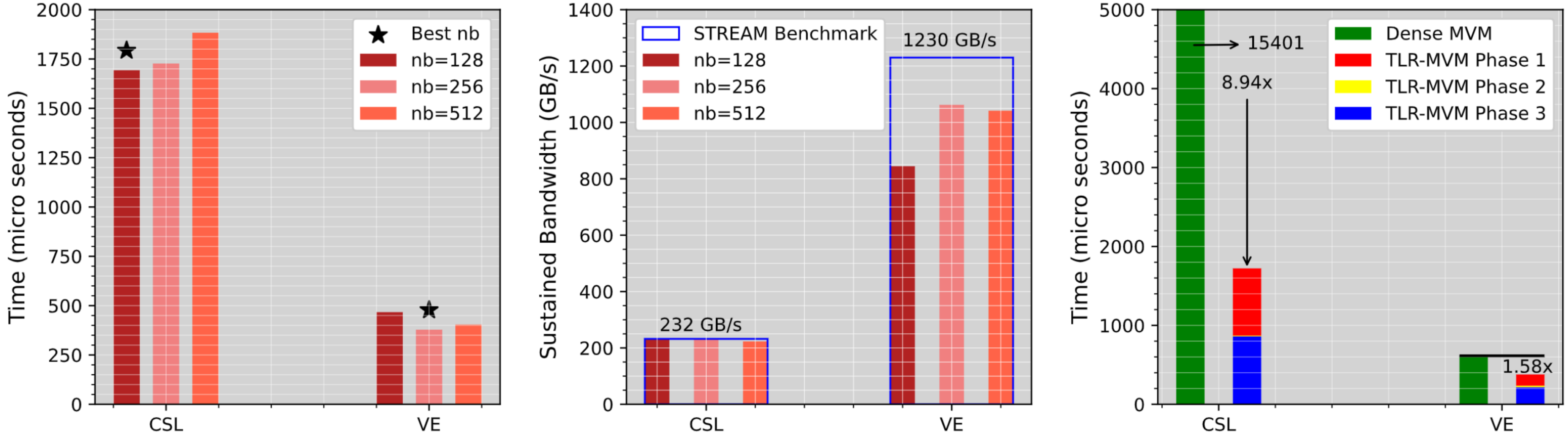
Damien Gratadour

- Collaborator



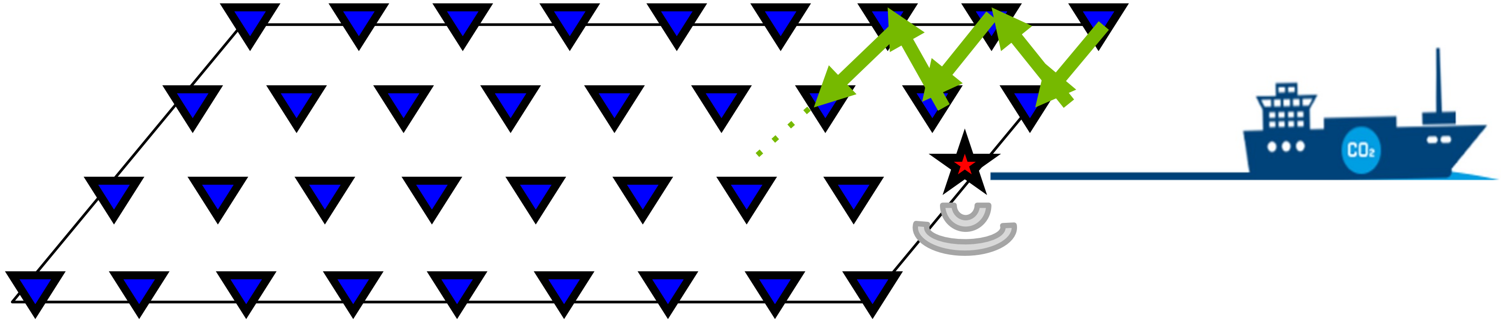
King Abdullah University
of Science and Technology

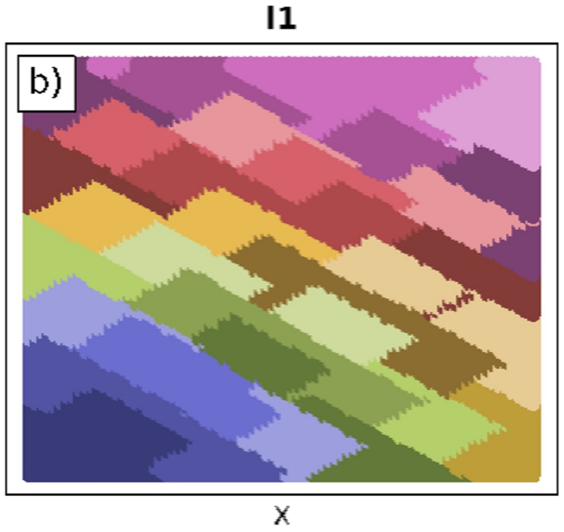
Results on synthetic datasets

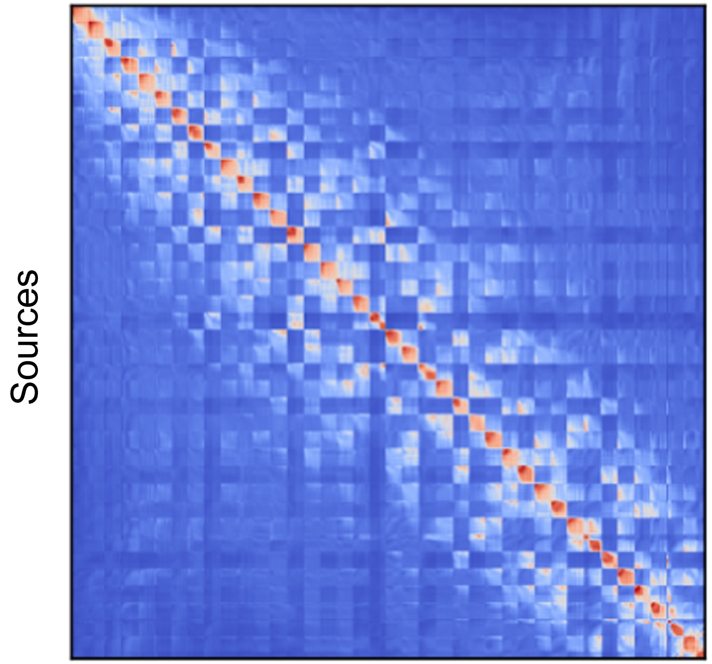


(a) Time to solution (b) Memory bandwidth (c) Time breakdown

Distance-aware matrix reordering method

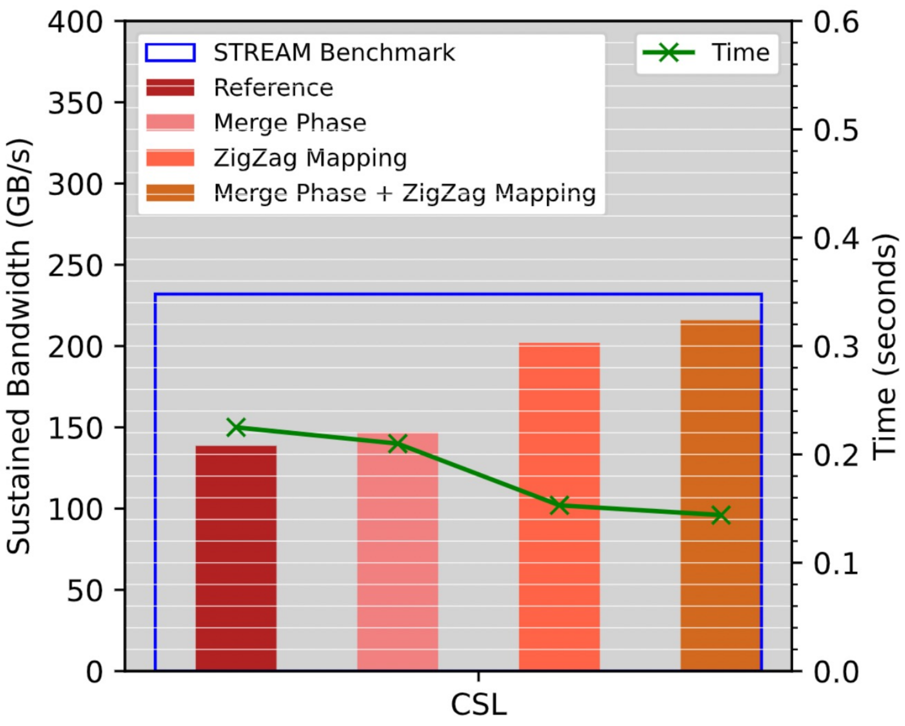




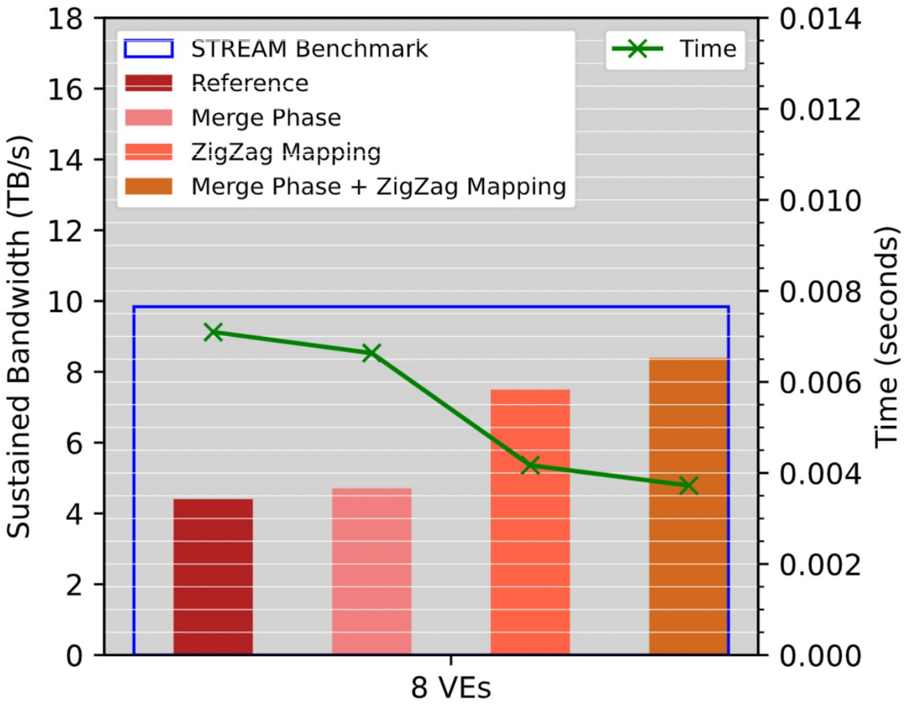


Load Balancing Strategies Results

King Abdullah University of Science and Technology



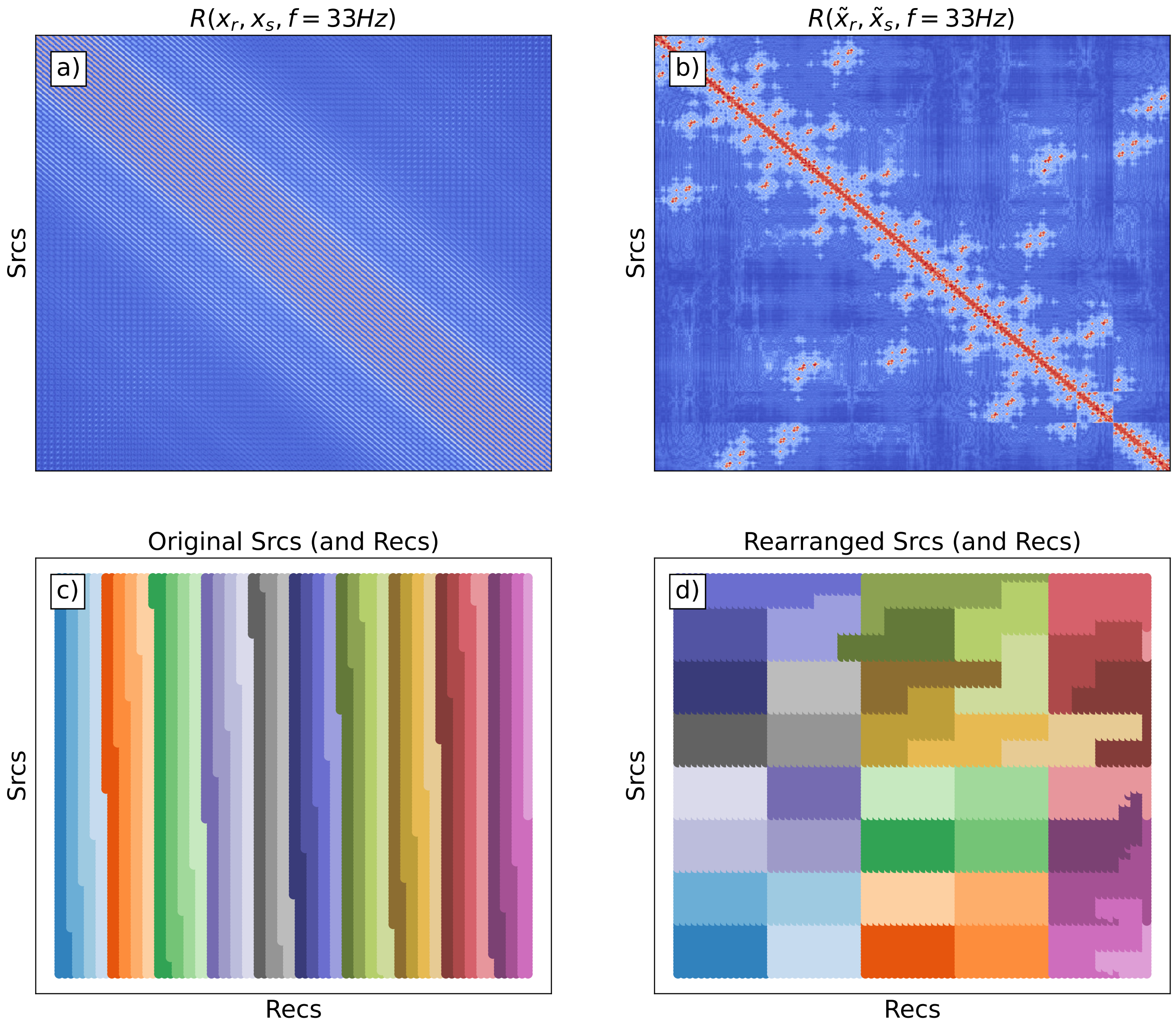
(a) Bandwidth/time on Intel CSL



(b) Bandwidth/time on 8 NEC VEs

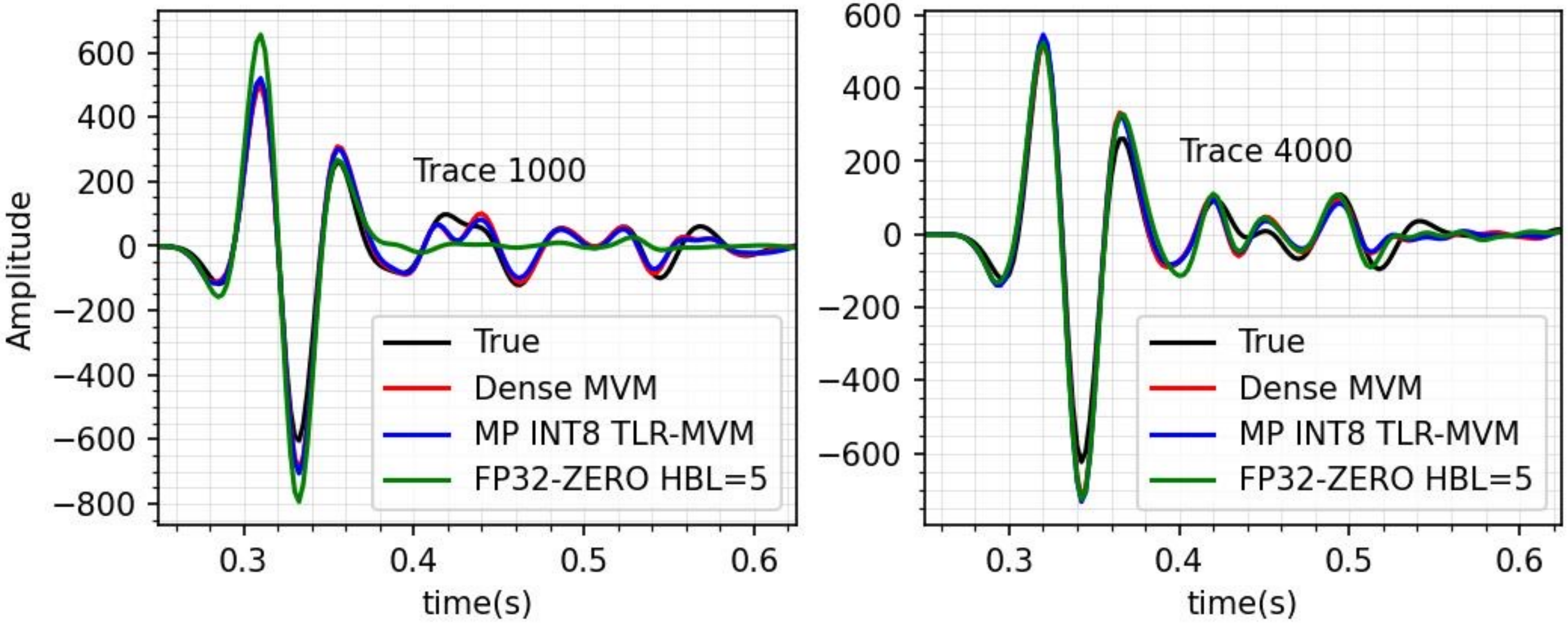
Distance-aware matrix reordering method

King Abdullah University of Science and Technology



Numerical Accuracy – Trace Comparison

King Abdullah University of Science and Technology



BOW-IPU Architectures

IPU-Tiles™

1472 independent IPU-Tiles™ each with an IPU-Core™ and In-Processor-Memory™

IPU-Core™

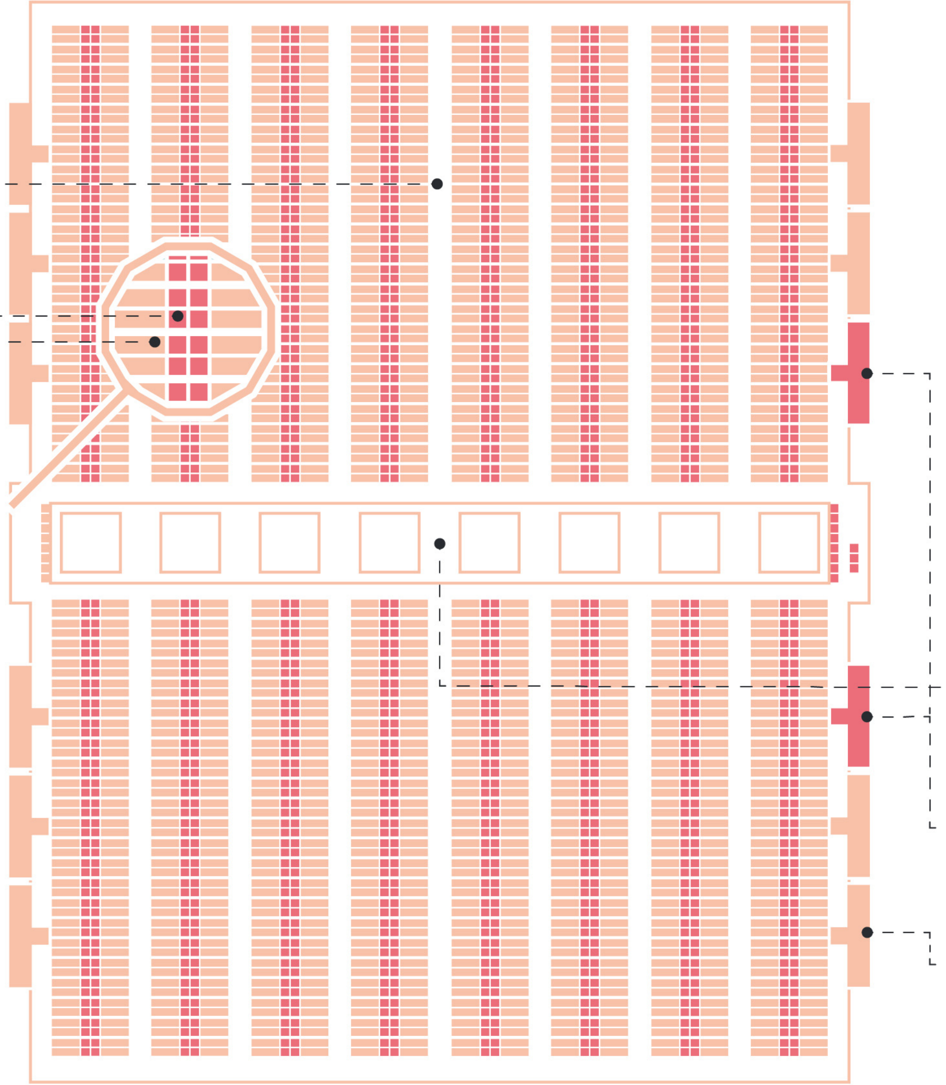
1472 independent IPU-Core™

8832 independent program threads executing in parallel

In-Processor-Memory™

900MB In-Processor-Memory™ per IPU

65TB/s memory bandwidth per IPU



IPU-Exchange™

11 TB/s all to all IPU-Exchange™
Non-blocking, any communication pattern

PCIe

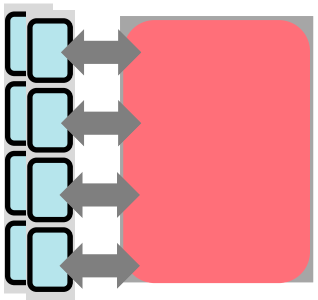
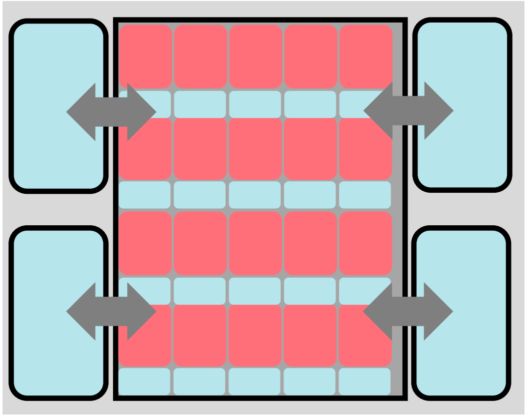
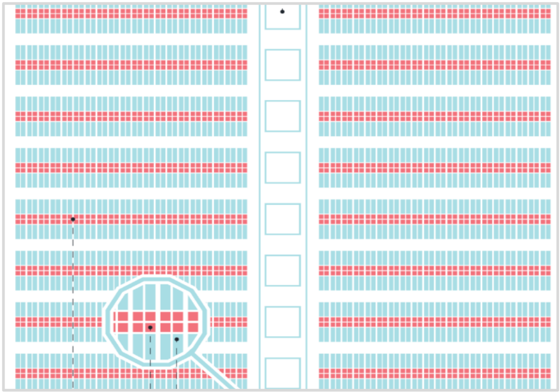
PCI Gen4 x16
64 GB/s bidirectional bandwidth to host

IPU-Links™

10 x IPU-Links,
320GB/s chip to chip bandwidth

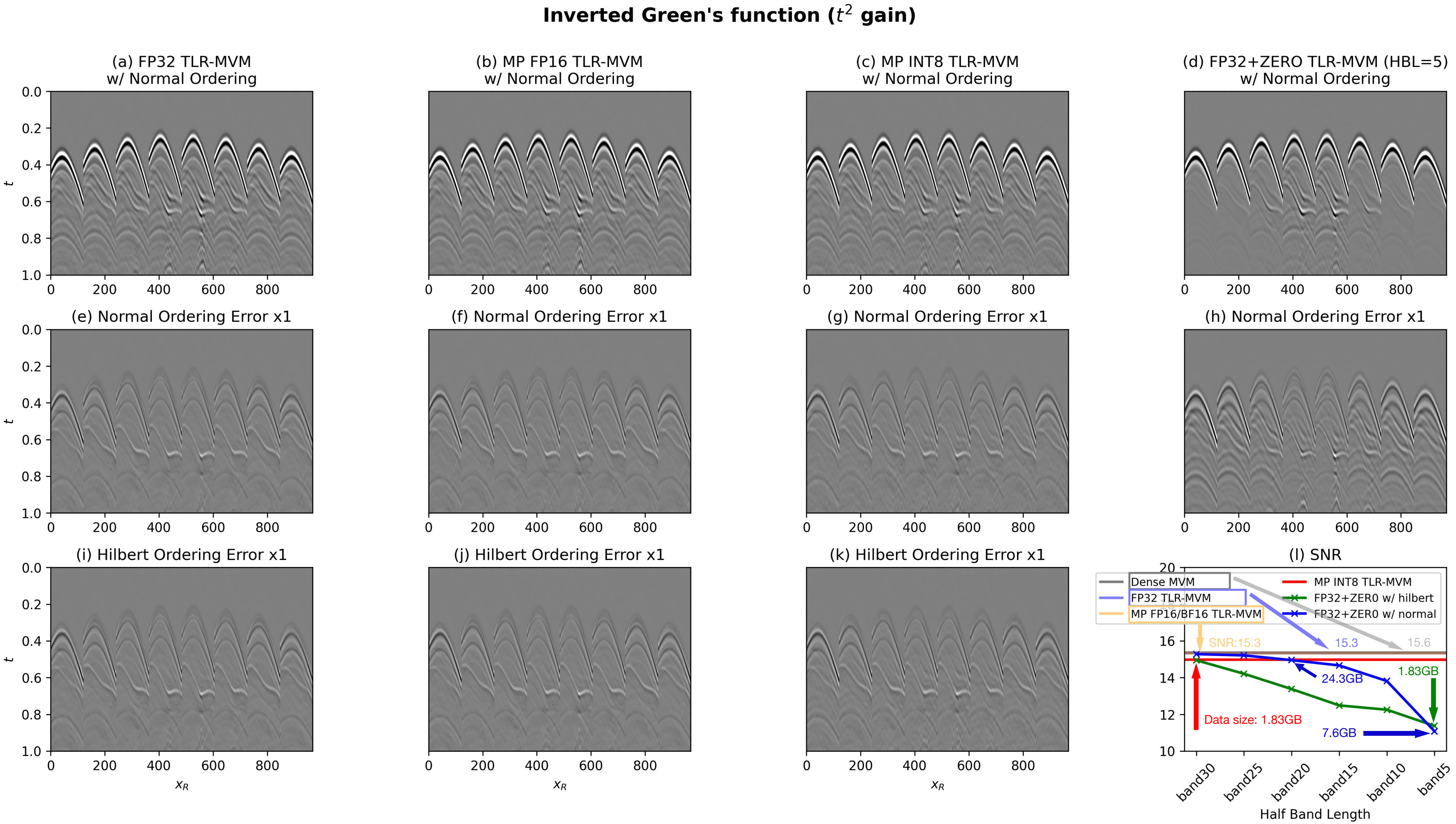
* Slides reference: GraphCore non-NDA marketing slides

GraphCore Design

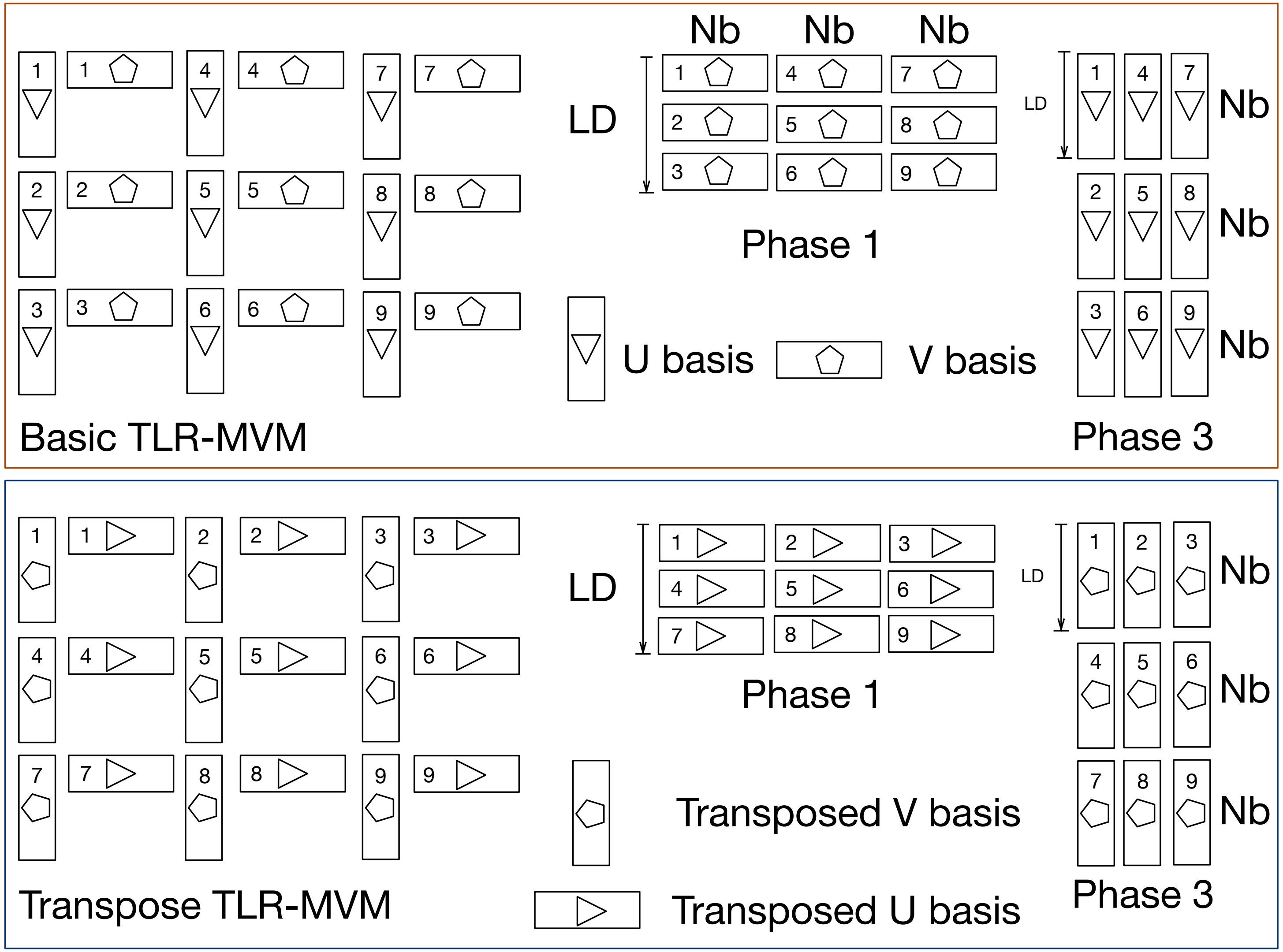
	CPU	GPU	IPU
Parallelism	Designed for scalar processing	SIMD/SIMT architecture. Designed for large blocks of dense contiguous data	**Massively parallel MIMD architecture. High performance/efficiency for future ML trends**
Processor Memory			
Memory Access	Off-chip memory	Model and Data spread across off-chip and small on-chip cache and shared memory	**Main Model & Data in tightly coupled large locally distributed SRAM**

* Slides reference: GraphCore non-NDA marketing slides

Numerical Accuracy using MP TLR-MVM



Transpose TLR-MVM Explanation



Theory

Assumption 1. *($\mathcal{L}$-smoothness) The function f is $\mathcal{L}$ smooth if its gradient is $\mathcal{L}$-Lipschitz continuous, that is, for all $x, y \in \mathbb{R}^d$,*

$$f(x) \le f(y) + \nabla f(y)^\top (x-y) + \frac{\mathcal{L}}{2}\|x-y\|^2.$$

Assumption 2. *(Boundness of stochastic gradient) The stochastic gradient is bounded by $G > 0$, that is,*

$$\mathbb{E}\left[\|\tilde{g}^k\|\right]^2 \le G^2.$$

The latter inequality implies $\mathbb{E}\left[\|\tilde{g}^k\|\right] \le G$ (using Jensen's inequality).

Assumption 3. *The function F Jacobian is bounded by $\kappa_J > 0$, that is,*

$$\mathbb{E}\left[\|\nabla F(x^k)\|\right] \le \kappa_J.$$

The latter Assumption implies $\mathbb{E}\left[\|J^k\|\right] \le \kappa_J$, independently from S^k.

Theorem 1. *Let Assumptions 1, 2 and 3 hold. Let $K > 0$, $\mu_0 > 0$ and $\mu = \mu_0\sqrt{K+1}$, then*

$$\frac{\sum_{k=0}^K \mathbb{E}\|\nabla f^k\|^2}{K+1} \le \frac{C}{\sqrt{K+1}},$$

where

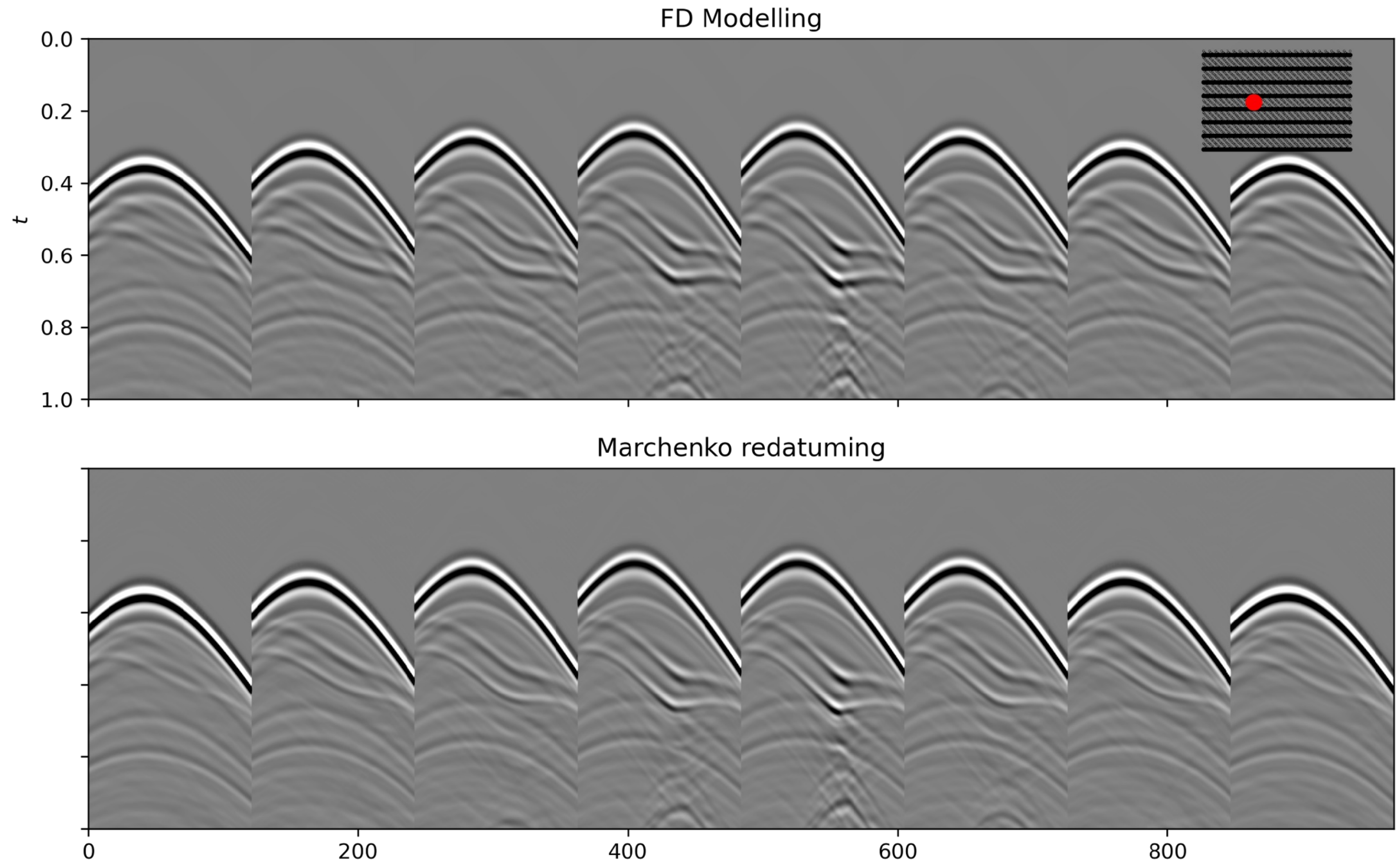
$$C := \mu_0 f^0 + \frac{2\kappa_J^2 G^2 + \mathcal{L}G^2}{2\mu_0}.$$

Corollary 1. *Let Assumptions 1, 2 and 3 hold. Let $K > 0$, $\mu_0 > 0$ and $\mu = \mu_0\sqrt{K+1}$, then*

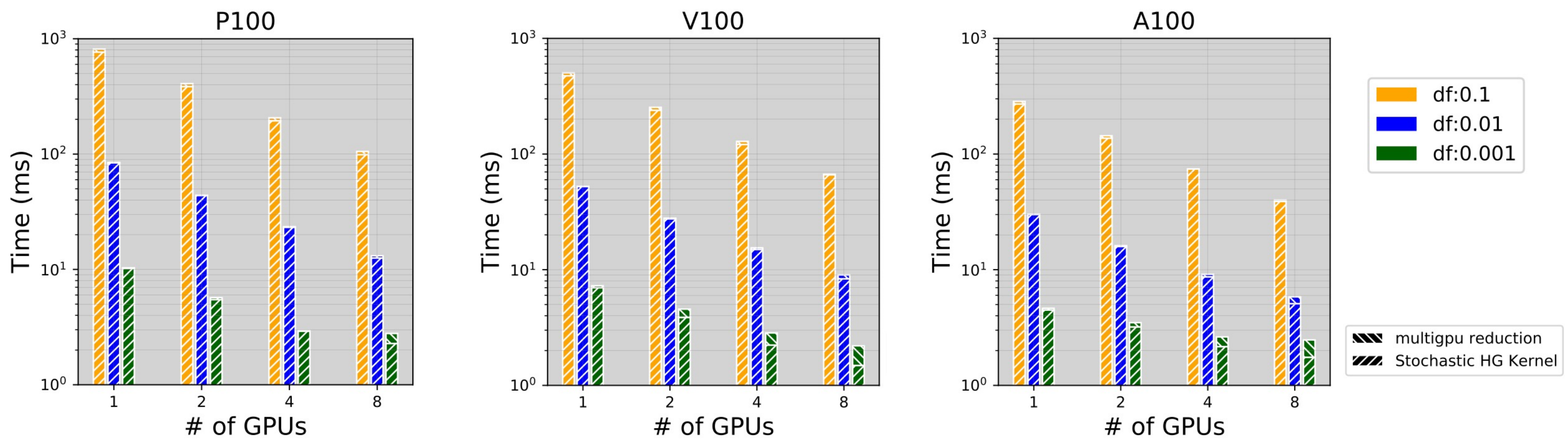
$$\boxed{\min_{k\in\{0,\dots,K\}} \mathbb{E}[\|\nabla f^k\|^2] \le \mathcal{O}\left(\frac{1}{\sqrt{K+1}}\right).}$$

- The Minimum of the gradient norm square over iterations is of the order of $O(\frac{1}{\sqrt{K+1}})$, which is the classical complexity bound know for SGD and its variants.

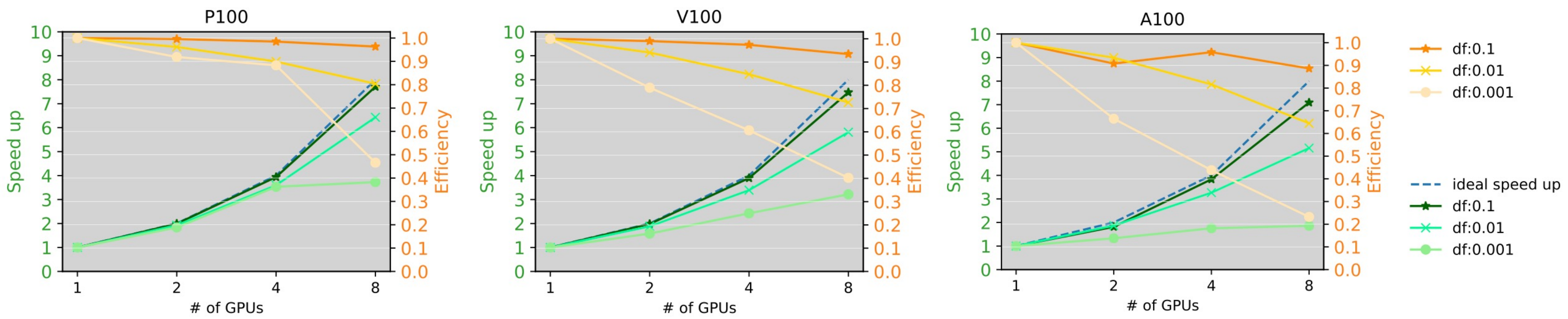
Marchenko Redatuming – dataset



Stochastic Levenberg-Marquardt method analysis



Time breakdown of 1 iteration SLM method



Speedup of SLM method

Problem Definition and Stochastic Levenberg-Marquardt Method

BERKELEY LAB

$$F(x) = vec(C_{i,j}^{ana}(x) - C_{i,j}^{num})$$

# of elems ≈ 40k x 40k

10 layer analytical atmosphere model suggested by European Southearn Observatory:
- Trubulence strength,
- Bi-dimentional wind velocity

30 parameters in all

Measurement numerical data is collected accumulatively every 15s to 30s to get higher enough signal-noise ratio.

Algorithm 1 SLM algorithm with fixed regularization.

Initialization: : Choose initial x^0. Choose a constant μ. Generate a random index sequence $Rand_{seq}$. Choose a data fraction df to decide the samples size used per iteration. Record initial $\|\tilde{g}^0\|_\infty$. Choose stopping criteria ϵ.

1: **for** $k = 1, 2, ..., K$ **do**
2: Get random indices from $Rand_{seq}$
3: Calculate ${J^k}^\top J^k$ and $\tilde{g}^k$ using GPU
4: If $\|\tilde{g}^k\|_\infty \;/\; \|\tilde{g}^0\|_\infty \leq \epsilon$, exit algorithm
5: Solve linear system: $({J^k}^\top J^k + \mu I)\delta x = \tilde{g}^k$
6: Update x^{k+1}: $x^{k+1} = x^k - ({J^k}^\top J^k + \mu I)^{-1}\tilde{g}^k = x^k - \delta x$
7: **end for**

Approximated Hessian *Fixed Regularization 1e-6*

Problem definition

$$\min_{x \in R^d} f(x) := \frac{1}{2}||F(x)||^2 = \frac{1}{2}\sum_{i=1}^{N} F_i(x)^2$$

$f: R^d \to R$ $F: R^d \to R$

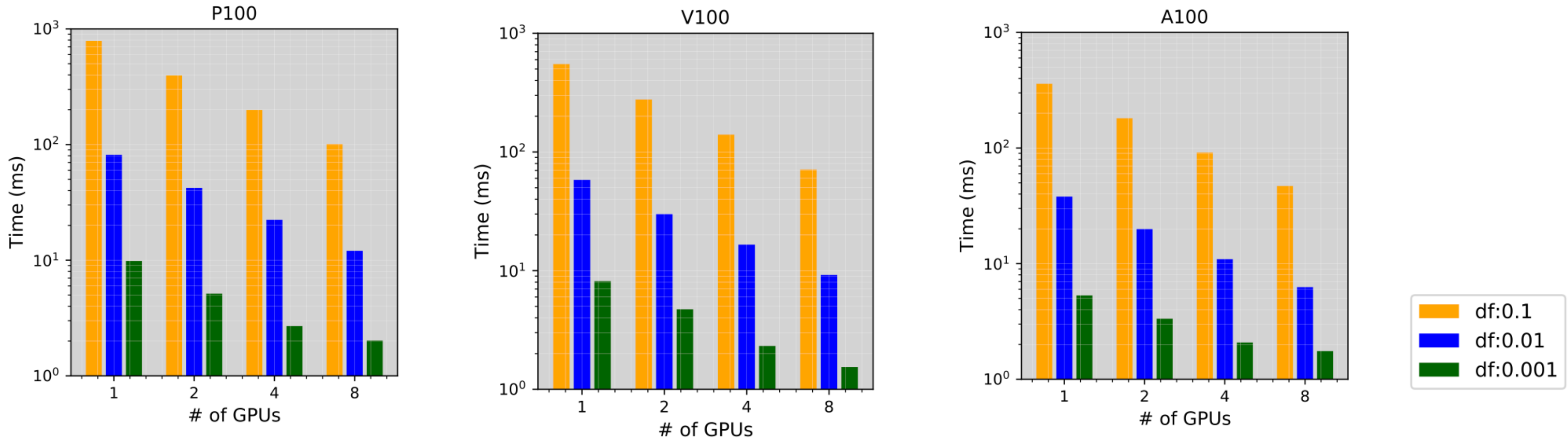
$J^k = S^k \nabla F(x^k)$ $\tilde{g}^k = \nabla F(x^k) S^k F^k$

Stochastic Jacobian vector *Stochastic gradient vector*

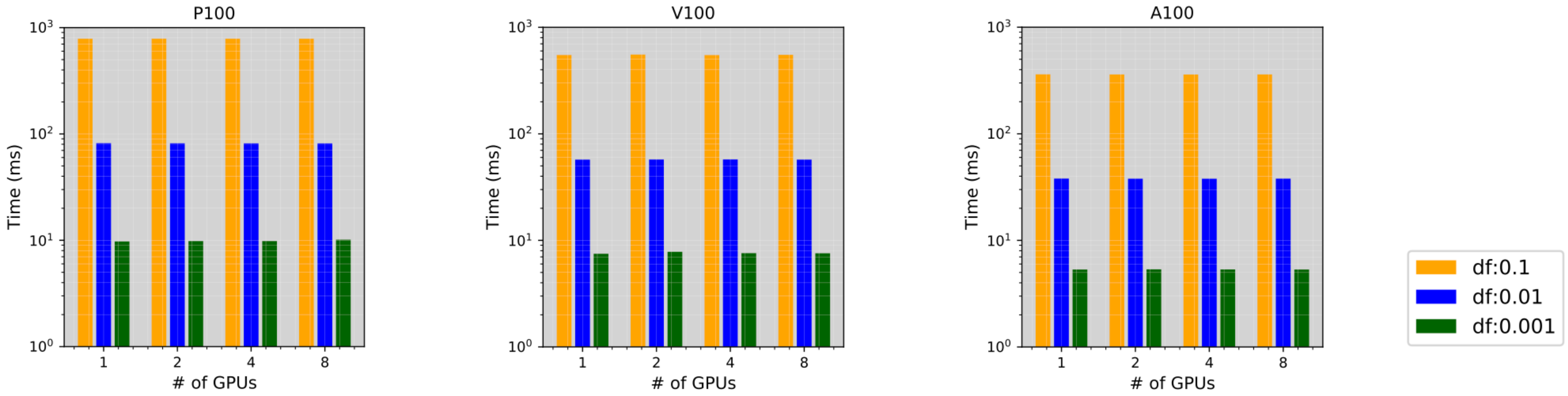
STOA: Levenberg-marquardt Method

Our SLM leverages data sparsity of the matrix and use a sub-sampling method to solve the problem. It randomly selects items inside Covariance matrix to form the approximated gradient and Hessian.

Kernel Performance Across NVIDIA GPU Hardware Generations

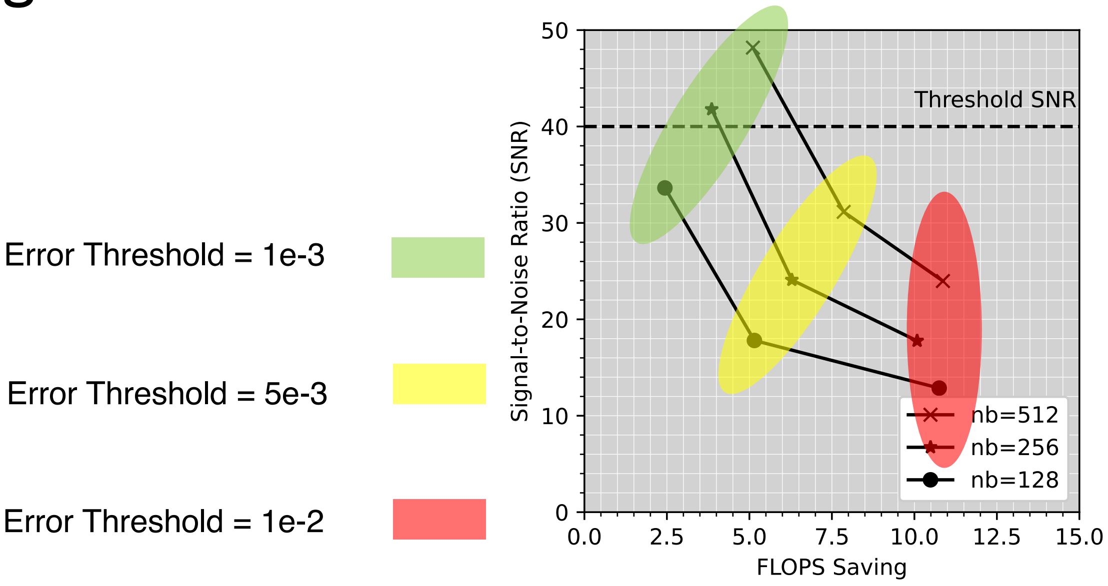


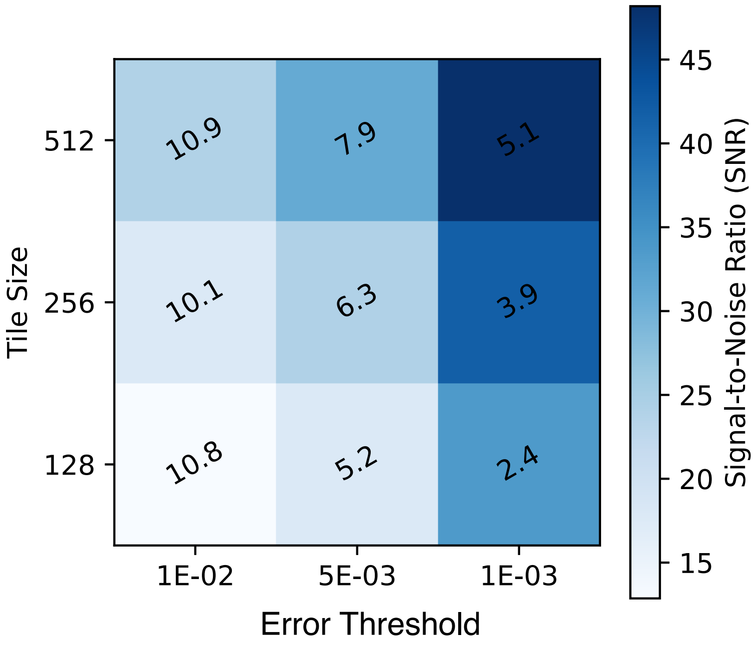
Strong Scaling Results



Weak Scaling Results

Application SNR vs different algorithmic configurations





- There are 2 parameters for user to tune the algorithms to trade off FLOPS saving and accuracy.
- In seismic application, signal-to-noise ratio (SNR) is used to quantify the quality of the results. We test on different tile sizes (nb) and accuracy thresholds. In the left figure, from left to right the error threshold is 1e-3, 5e-3, and 1e-2.
- We set 40 as the SNR threshold and find two eligible configurations. We conduct subsequent experiments using nb = 256 and error threshold 1e-3.

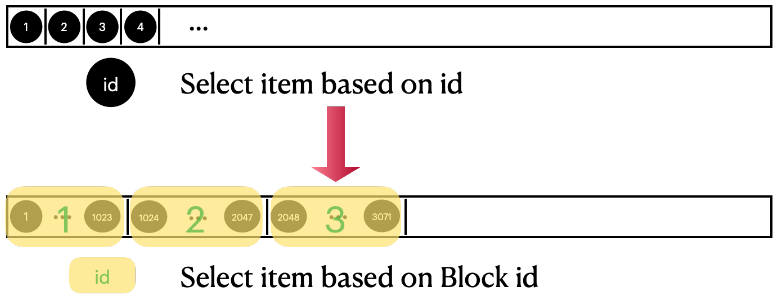
Implementation

- **Stochastic Hessian and Gradient Kernel Design (the most time consuming kernel in SLM)**

  We design stochastic HG kernel to compute approximate Hessian and gradient. Each CUDA thread is responsible for one sample in the optimization problem. We get numerical Jacobian using finite difference approximation.

- **Block Random Index**

  If we select index randomly, we will have irregular memory access issue. We group the index together and select the Index by group id to have coalesced memory access pattern.



- **Reduction Optimization**

  We use NVIDIA cub library to perform block-level reduction. Then we use atomic operation for global reduction.