Evaluating asynchronous schwarz solvers for Exascale

Pratik Nayak
pratik.nayak@kit.edu

Hartwig Anzt
hartwig.anzt@kit.edu

Steinbuch Center for Computing,
Karlsruhe Institute for Technology

6th November 2019

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

KIT
Karlsruhe Institute of Technology

Objectives

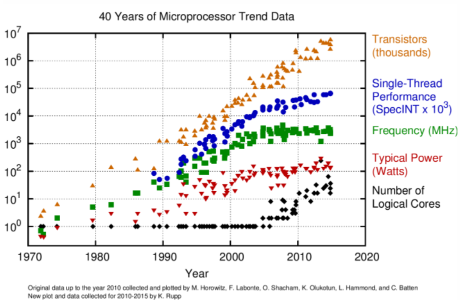
1. What our objectives are:
   - Study asynchronous iterative algorithm behaviour.
   - Use Schwarz methods as a test-bed due to the simplicity of the algorithm
   - In particular, study multi-GPU, multi-node problems.

Motivation

Motivation and Background

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**Frontera** - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR , Dell EMC Texas Advanced Computing Center/Univ. of Texas United States	448,448	23,516.4	38,745.9	
6	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384

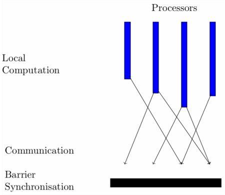
(a) Top 6



(b) Computing trends

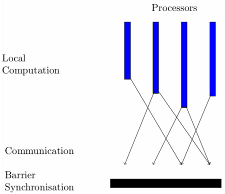
Motivation and Background

1. Bulk synchronous model of parallel execution (Most algorithms today).
   - A known task graph.
   - Needs regular synchronization between processes.
   - Not feasible for large number of processes.

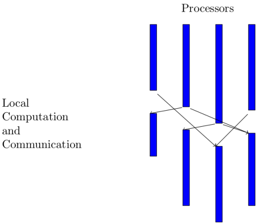


Motivation and Background

1. Bulk synchronous model of parallel execution (Most algorithms today).
   - A known task graph.
   - Needs regular synchronization between processes.
   - Not feasible for large number of processes.

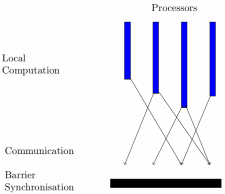


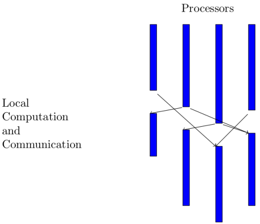
2. Asynchronous model of execution (Where algorithms need to be).
   - Ideally, no synchronization.
   - Feasible and possibly necessary for large number of processes.
   - Possibly unknown task graph.



Motivation and Background

1. Bulk synchronous model of parallel execution (Most algorithms today).
   - A known task graph.
   - Needs regular synchronization between processes.
   - Not feasible for large number of processes.
2. Asynchronous model of execution (Where algorithms need to be).
   - Ideally, no synchronization.
   - Feasible and possibly necessary for large number of processes.
   - Possibly unknown task graph.
3. Partial Synchronization model (A compromise, future work).
   - Partially regular synchronization between processes.
   - Might work for large number of processes.
   - Partially known task graph.





Interlude

Back of the envelope.

1. Consider the NVIDIA V100 GPU, current state of the art HPC GPU. Amount of memory: 32GB.

Back of the envelope.

1. Consider the NVIDIA V100 GPU, current state of the art HPC GPU. Amount of memory: 32GB.
2. What is the maximum size (number of rows) of a matrix that can be stored on the GPU for computing its solution with a given right hand side?

Back of the envelope.

1. Consider the NVIDIA V100 GPU, current state of the art HPC GPU. Amount of memory: 32GB.

2. What is the maximum size (number of rows) of a matrix that can be stored on the GPU for computing its solution with a given right hand side?

3. Assume: CSR matrix storage format: Let N be the number of rows, `nnz` be the number of non-zeros represented as $\sigma * N^2$, with σ as the sparsity factor. This is a conservative estimate without consideration for the auxillary vectors. CSR stores two array of length of `nnz` and one array with length N. For a solution, we need an additional N for right hand side array and the solution array.

   Total number of bytes:

   $$\texttt{num\_bytes} = (8 \times N + 16 \times \sigma N^2) + 2 \times (8 \times N)$$

   So to get the maximal number of rows solve:

   $$16 \times \sigma N^2 + 24 \times N = 3.2 \times 10^{10}$$

Back of the envelope.

This gives

Table: Maximal number of rows.

	Sparsity factor, σ	**Number of rows, N**
	0.02	3.2e5
SuiteSparse-avg $\longleftarrow$	0.002	1e6
	0.0002	3e6

Our case of Laplace

- Similar calculation gives: `stencil_size/number of rows` (For 5 point stencil $\approx 5/N$)
- Therefore, Laplace is an hard problem for Schwarz type solvers.

The Schwarz algorithm

Iterative methods.

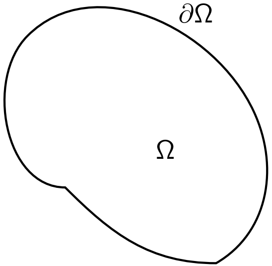


Figure: Generic Domain

Problem:

$$\mathcal{L}x = f \ in \ \Omega; \qquad \mathcal{B}x = g \ on \ \partial\Omega$$

Linear system:

$$Ax = f$$

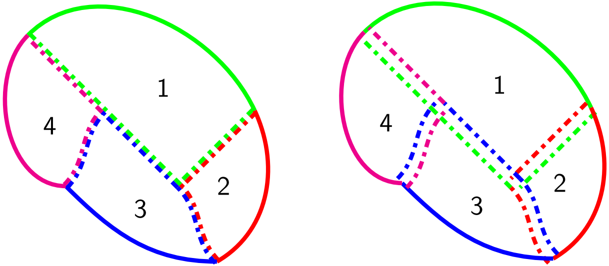
Iterative methods

Stationary iterative method:

$$x^{k+1} = Bx^{k} + c$$

For convergence, $\rho(B) < 1$.

Domain decomposition methods



(a) Non-overlapping domains (b) Overlapping domains

Figure: Overlapping and non-overlapping domains

Schwarz methods

1. Initially used to prove convergence of the Poisson problem for general domains (Schwarz, 1870). Slow convergence.
2. Gained popularity with parallel computers.
3. Solve each subset(subdomain) independently and communicate between each "iteration".
4. Restricted additive Schwarz methods: An improvement of the parallel version of the Schwarz method for faster convergence.

Restricted Additive Schwarz methods

Used widely as a preconditioner:

$$M_{RAS}^{-1} = \sum_{j}^{N} \tilde{R}_j^T A_j^{-1} R_j$$

Group unknowns into subsets:

$$x_j = \tilde{R}_j x, \;\; j = 1, \ldots, N$$

$\tilde{R}_j$ is the rectangular Restriction matrices which corresponds to a non-overlapping decomposition.

Restricted Additive Schwarz

Compute using the full overlapped sub-matrix, but update only your locally associated values.

Restricted Additive Schwarz methods

RAS:

$$x_p^{k+1} = x_p^k + \sum_j^N \tilde{R}_p(R_j f - (R_j A R_j^T)^{-1} R_j x^k)$$

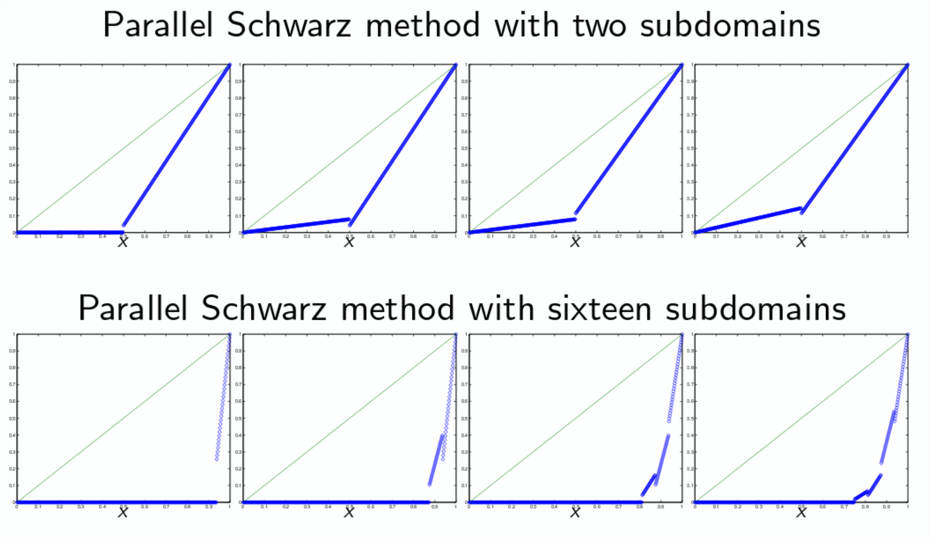
Advantages:

1. Saves communication compared to Additive Schwarz.
2. Reduced iteration count compared to Additive Schwarz.

Source: Why Restricted Additive Schwarz Converges Faster than Additive Schwarz, Efstathiou and Gander, 2003

Interlude 2

The problem with Schwarz.



Source: Martin Gander, University of Geneva

The problem with Schwarz - Fixes

Optimized schwarz

- Allow for better exchange of information at the boundaries. Modify the interface exchange from zeroth order function to a first order function.
- Method loses generality, more complicated for general problems.

Coarse grid

- Coarse grid preconditioning: Borrow idea of a coarse grid preconditioner from Multi-grid.

Source: Martin Gander, University of Geneva

Implementation and Experimentation

The Schwarz iterative solver.

Algorithm 1 Schwarz Iterative solver

procedure ITERATIVE SOLUTION(A, x, b)
  procedure INITIALIZATION
    Partition matrix ▷ 1D / objective based
    Distribute data
    Initialize data
  procedure SOLVE
    while *iter* $<$ *max_iter* or until convergence **do**
      Locally solve the matrix ▷ Iterative / direct
      Exchange boundary information
      Update boundary information
      Check for Convergence ▷ Centralized(Tree based)/Decentralized ...
    Gather the final solution vector

Experimentation Parameters

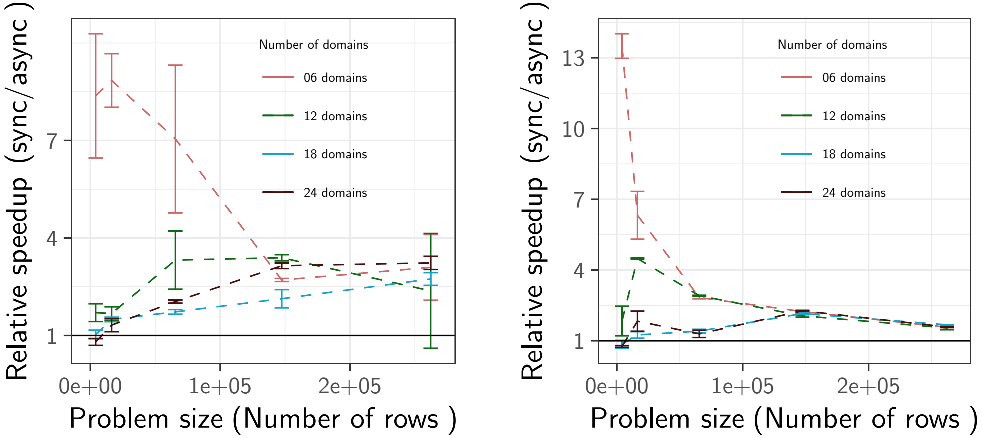
1. Everything implemented with Ginkgo.
2. Experiments performed on Summit, ORNL.
    1. 6 GPU's per node, NVIDIA Tesla V100's.
3. Global convergence is tree-based (Yamazaki et.al, 2019).
4. RDMA communication with MPI-onesided functions.
5. Partitioning with METIS / simple 1D.
6. Test problem: Laplace 5 point stencil.

Ginkgo

Source: Performance of asynchronous optimized schwarz with one-sided communication, Yamazaki et.al

Results

Asynchronous speedup

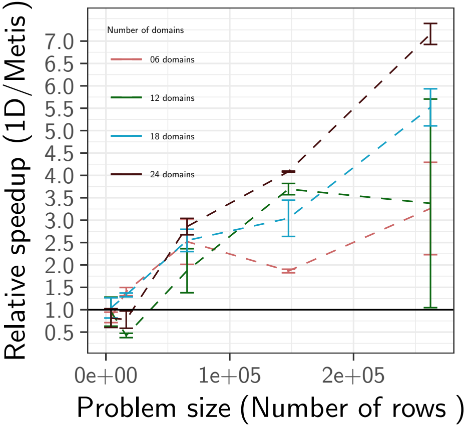


(a) Metis partitioning

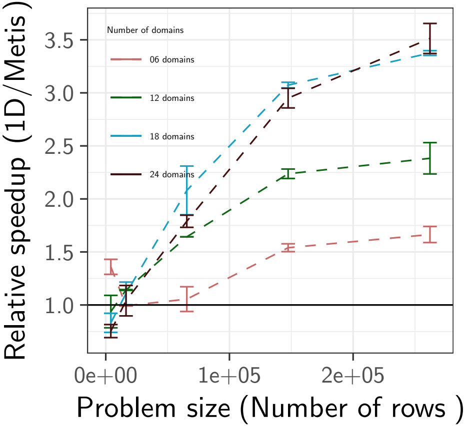
(b) Naive partitioning

Figure: Speedup of the asynchronous schwarz for different partitionings

Effect of Partitioning



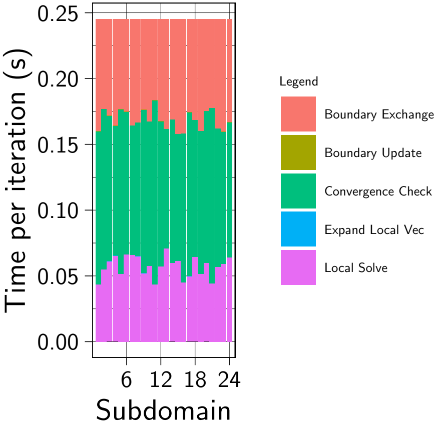
(a) Asynchronous



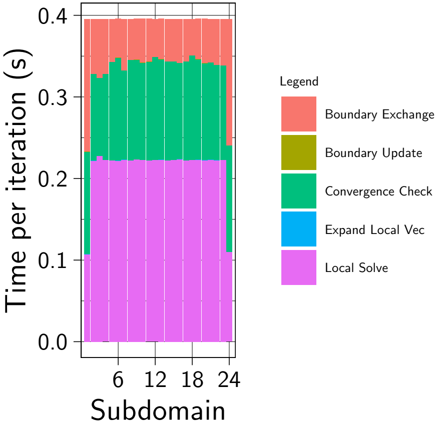
(b) Synchronous

Figure: Speedup of the METIS partitioning over Naive 1D partitioning.

Effect of Partitioning: Communication patterns



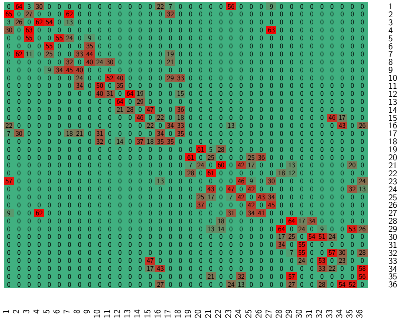
(a) Metis



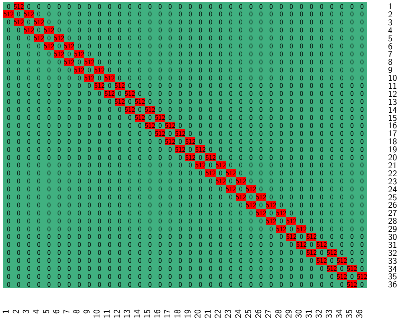
(b) Naive 1D

Figure: Split up of function timings - Naive 1D and Metis

Effect of Partitioning: Communication patterns



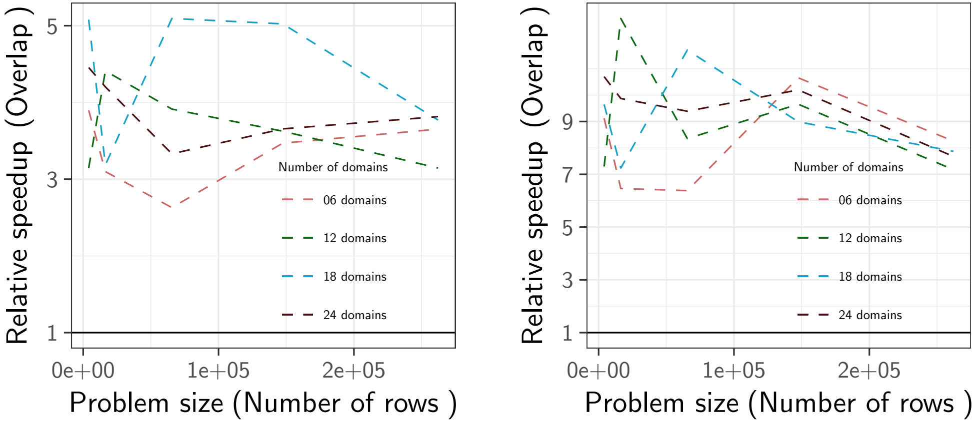
(a) Metis



(b) Naive 1D

Figure: Split up of function timings - Naive 1D and Metis

Effect of Overlap - METIS - Twosided

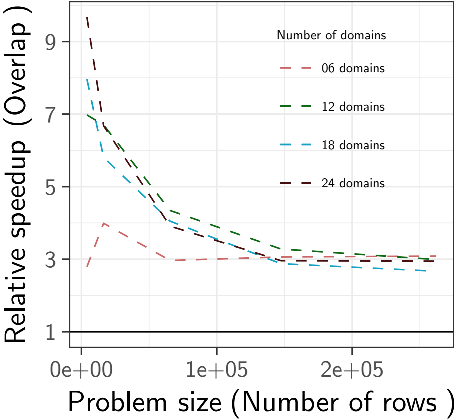


(a) 8 elements v/s 16 elements

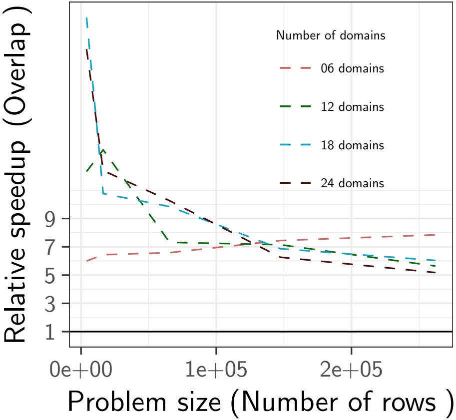
(b) 2 elements v/s 16 elements

Figure: Effect of overlap - METIS partitioning

Effect of Overlap - Naive 1D - Twosided



(a) 8 elements v/s 16 elements



(b) 2 elements v/s 16 elements

Figure: Effect of overlap - Naive 1D partitioning

Summary and Future work

Summary

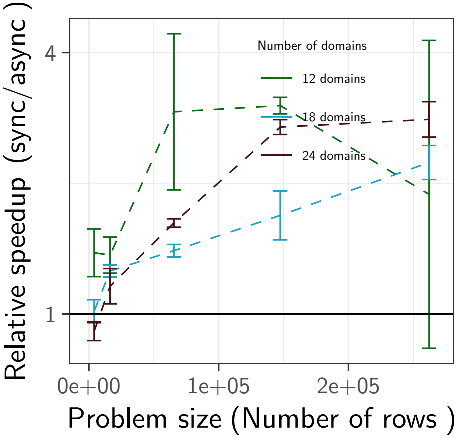
- Asynchronous methods can improve the overall time to solution.
- Communication pattern and load balancing are important factors.
- The plain Schwarz method does not scale well, particularly with a regular 1D communication setup.

Future Work

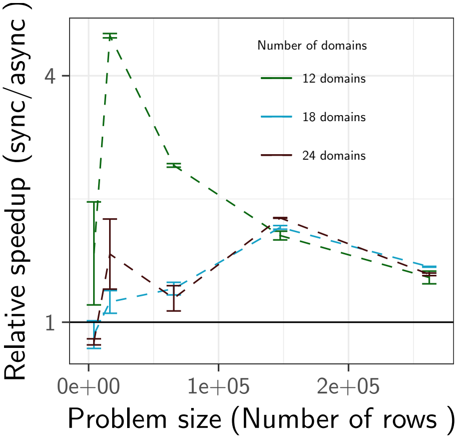
- Extend the algorithm to Optimized Schwarz.
- Use hybrid-CPU-GPU approach.
- Event based communication for only periodic information transfer.

Backup

Asynchronous speedup - Zoomed in



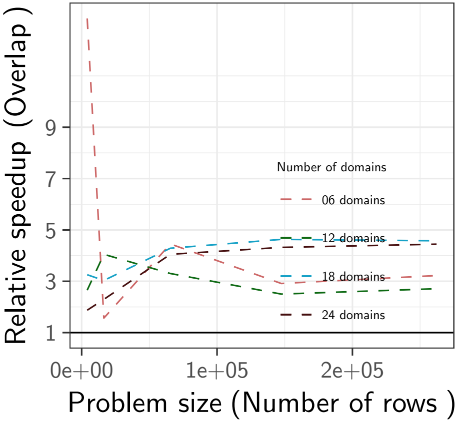
(a) Metis partitioning



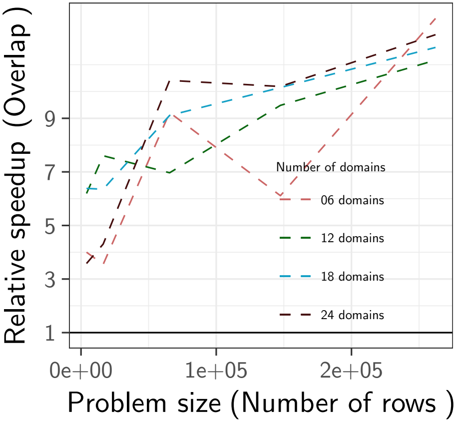
(b) Naive partitioning

Figure: Speedup of the asynchronous schwarz for different partitionings

Effect of Overlap - METIS - Onesided



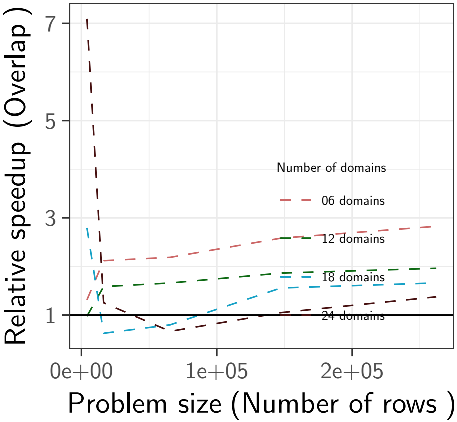
(a) 8 v/s 16



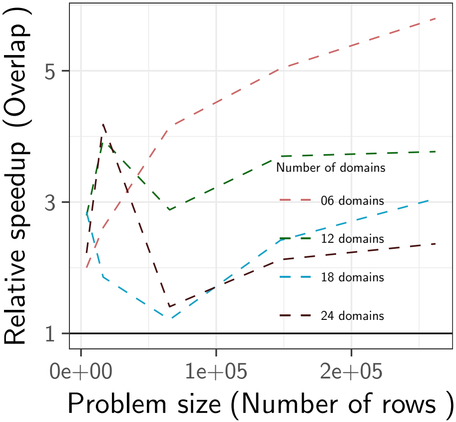
(b) 2 v/s 16

Figure: Effect of overlap - METIS partitioning

Effect of Overlap - Naive 1D - Onesided



(a) 8 v/s 16



(b) 2 v/s 16

Figure: Effect of overlap - Naive 1D partitioning