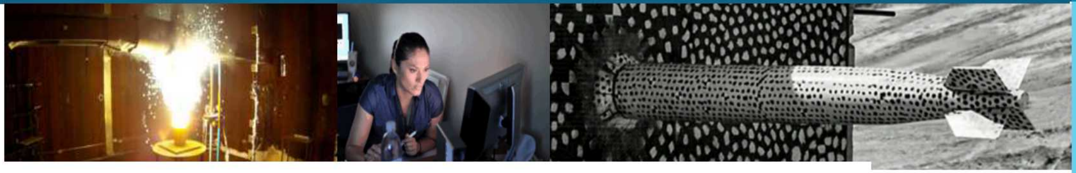
SAND2020-1689C

2D Block Cyclic Partitioning for Sparse Matrices



Seher Acer ✦
Erik G. Boman ✦
Cevdet Aykanat △

✦ : Sandia National Labs △ : Bilkent University

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

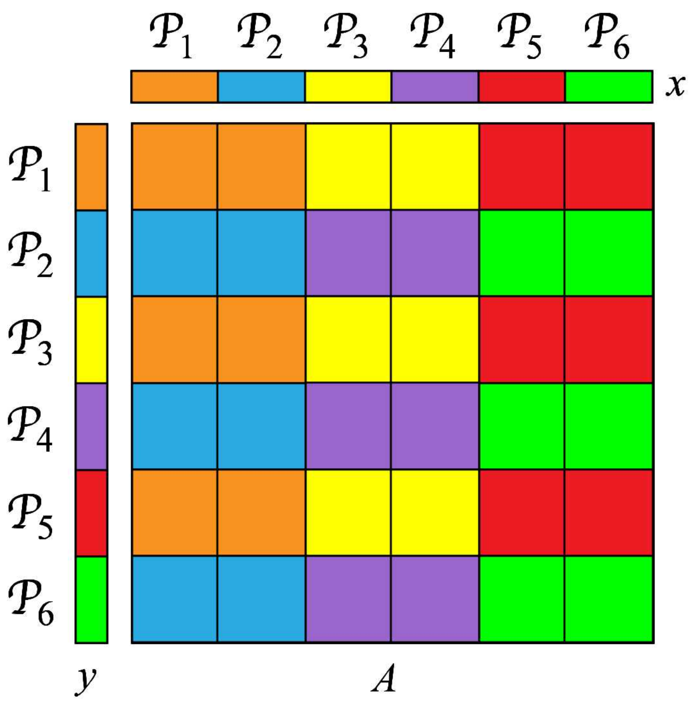
Outline

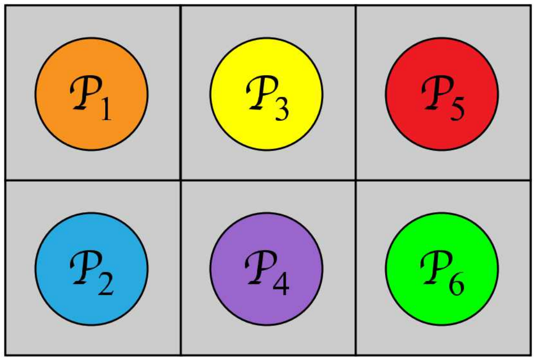
- 2D Block Cyclic Partitioning
- Extension to Rectangular Matrices
- Refinement Heuristic for Improving Load Balance
- Experimental Results
- Conclusion

2D Block Cyclic Partitioning [1]

2-phase partitioning for SpMV $y = Ax$

- 1st phase: partitioning vectors x and y
  - A K-way graph/hypergraph partitioning
    - Vertex i = row i and entries x_i and y_i
  - x and y have the same partition: conformal
- 2nd phase: partitioning matrix A
  - Determine $K \times K$ blocks in matrix A
  - Assume virtual $K = Q \times R$ process layout
  - Assign Q blocks to each of R processes
  - Cycle along the row dimension



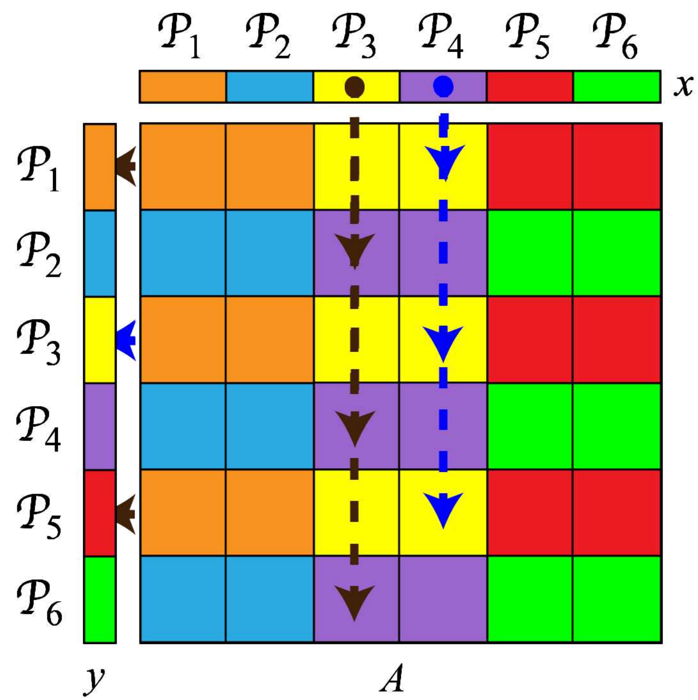


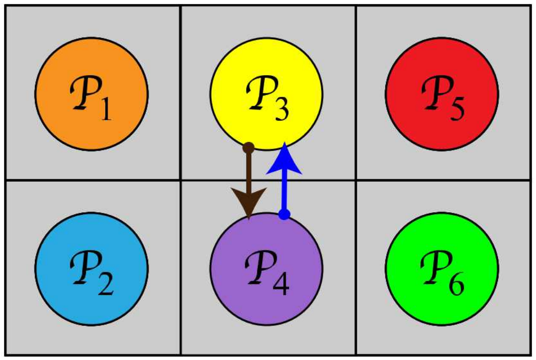
[1] **Boman et al.**, "Scalable matrix computations on large scale-free graphs using 2D graph partitioning", SC'13.

2D Block Cyclic Partitioning [1]

Why is it good?

- Communications occur only along
  - The columns of the virtual mesh
    - At most $Q-1$ messages while **expand**ing x entries
  - The rows of the virtual mesh
    - At most $R-1$ messages while **fold**ing y entries
  - Maximum message count = $O(\sqrt{K})$
    - Critical for scale-free graphs/matrices
- Graph partitioning in the first phase
  - Addresses communication volume
  - Tries to balance computational workload
- Cheap compared to the checkerboard hypergraph partitioning model [2]





[1] **Boman et al.**, "Scalable matrix computations on large scale-free graphs using 2D graph partitioning", SC'13.

[2] **Catalyurek and Aykanat**, "A hypergraph-partitioning approach for coarse-grain decomposition", SC'01.

2D Block Cyclic Partitioning [1]

However…

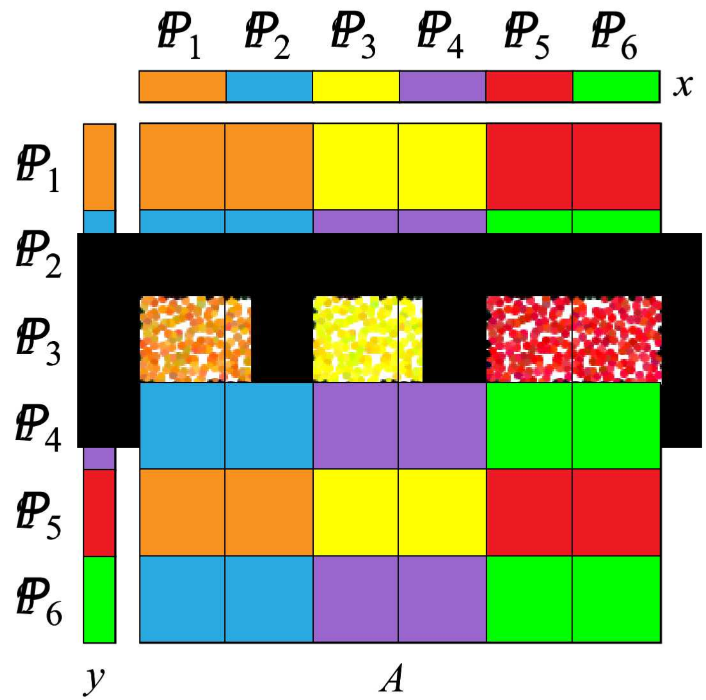
- Only for conformal vector partitions
  - There can be scale-free rectangular matrices!
- Graph partitioning in the first phase does not correctly capture computational load

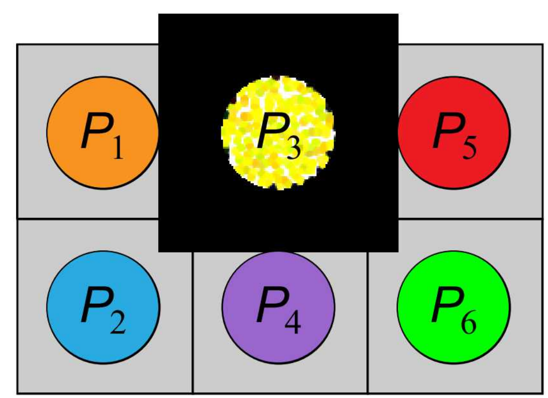
Extending the model to nonconformal partitions

Proposing refinement heuristic to reduce computational imbalance

(Communication volume was already

- Not expected to dominate the runtime
  - Focus is scale-free graphs/matrices!
- Not captured correctly due to using a graph instead of a hypergraph [2])



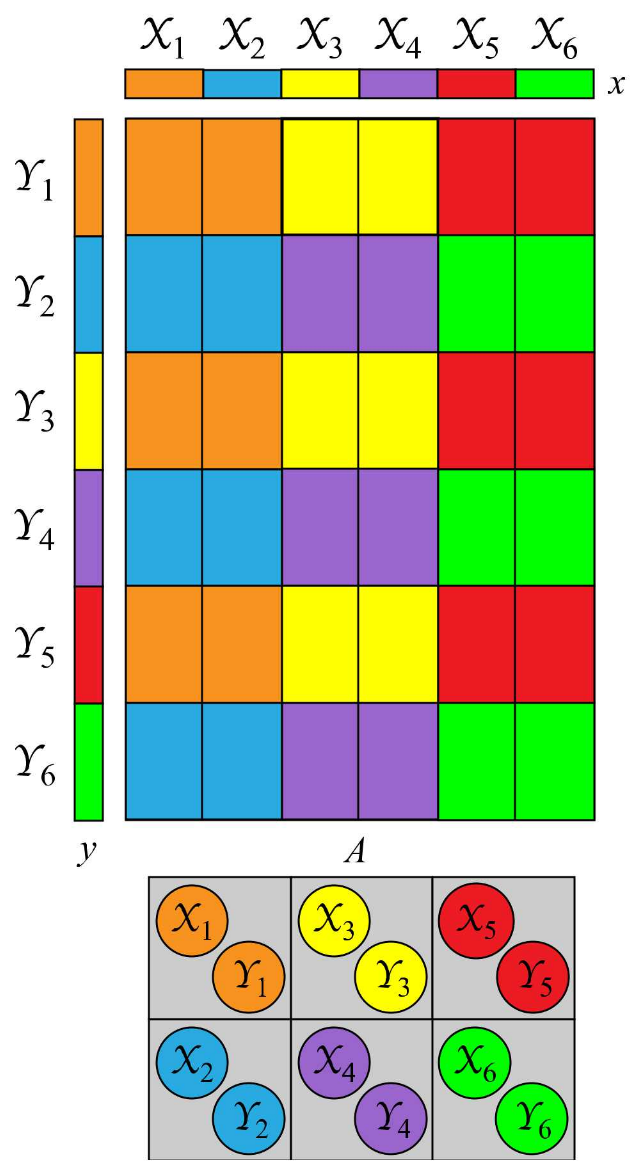


[1] **Boman et al.**, "Scalable matrix computations on large scale-free graphs using 2D graph partitioning", SC'13.
[2] **Catalyurek and Aykanat**, "A hypergraph-partitioning approach for coarse-grain decomposition", SC'01.

Extension to Nonconformal Partitions

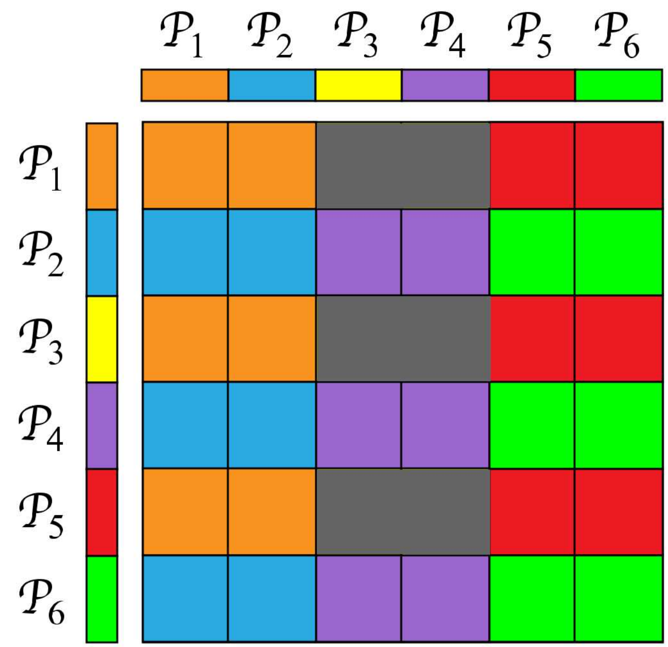
How to find smart partitions for x and y?

- Use a bipartite graph [1] in the first phase
  - row i = row vertex v_i^r
  - column j = column vertex v_j^c
  - nonzero $a_{i,j}$ = edge between v_i^r and v_j^c
  - 2-constraint partitioning
    - 1st weight of v_i^r: number of nonzeros in row i
    - 2nd weight of v_j^c: 1
  - Partition on y = partition of row vertices
  - Partition on x = partition of column vertices



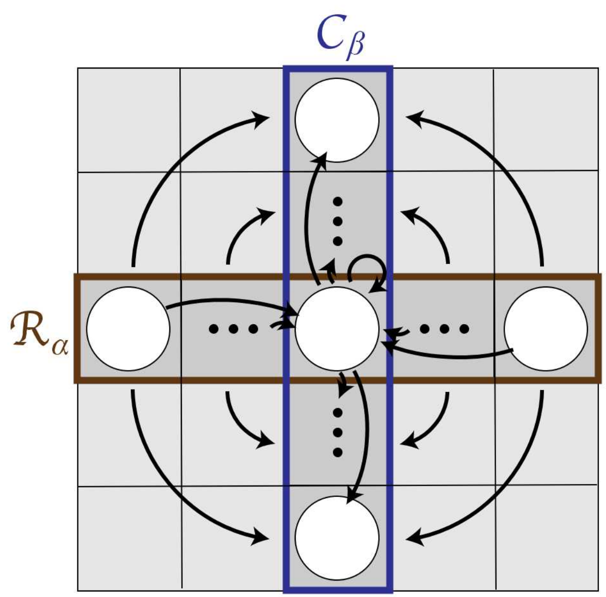
[1] **Hendrickson and Kolda**, "Partitioning rectangular and structurally unsymmetric sparse matrices for parallel processing", SISC, 2000.

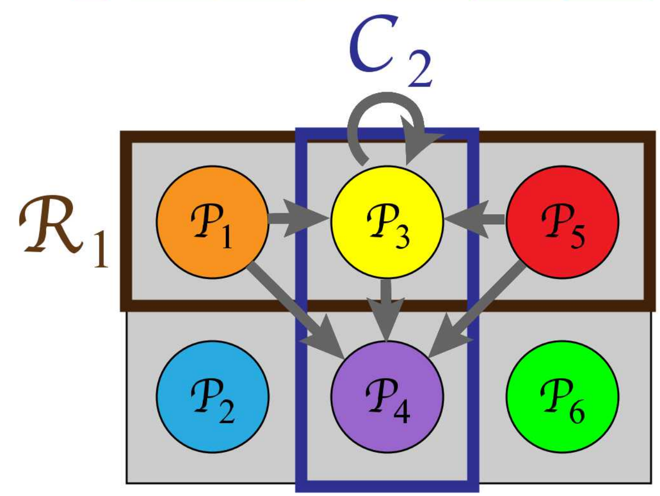
Proposed Refinement Heuristic



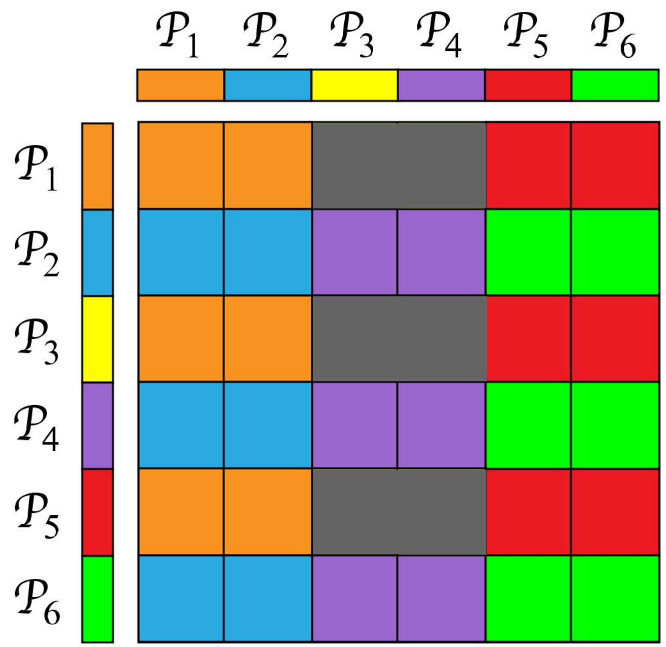
Formulation

- $load(\alpha, \beta)$ = number of nnzs assigned to $p_{\alpha,\beta}$
  - Consider $load(1,2)$ in the figure
- $\mathcal{R}_\alpha$ = parts assigned to row α
- $\mathcal{C}_\beta$ = parts assigned to column β
- $load(\alpha, \beta) = nz(\mathcal{R}_\alpha, \mathcal{C}_\beta)$





Proposed Refinement Heuristic

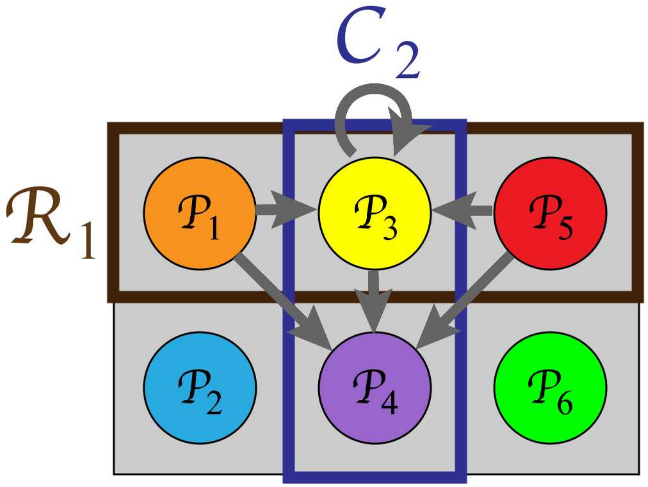


Objective:

- minimize $\max_{\alpha,\beta} load(\alpha, \beta)$

Algorithm:

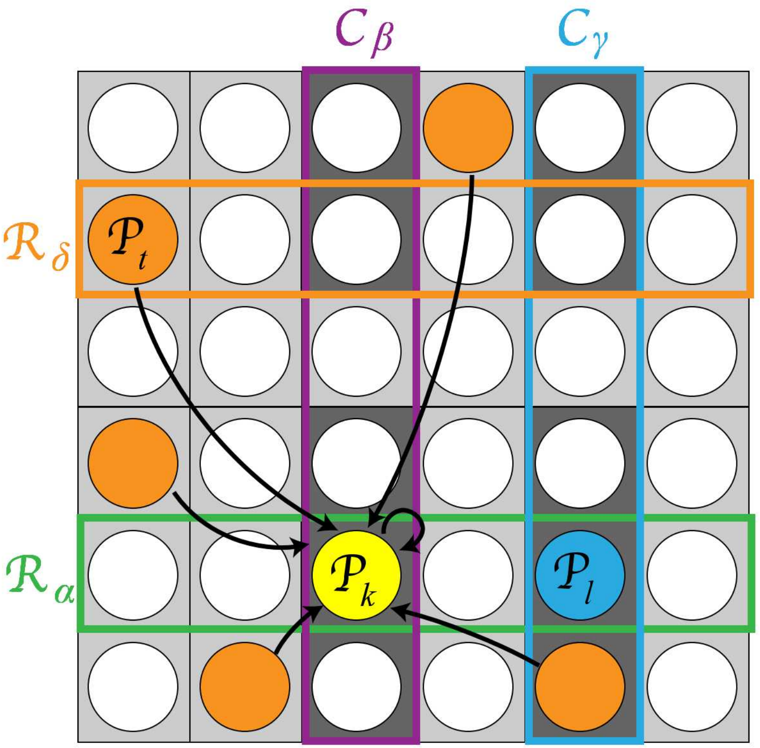
- start with an initial part-to-process mapping
- while not converged
  - find a process $p_{\alpha,\beta}$ with the maximum load
  - let $p_{\alpha,\beta} = map(\mathcal{P}_k)$
  - for each other part $\mathcal{P}_\ell \in \mathcal{R}^\alpha$ (mapped to row α)
    - compute the gain of swapping $\mathcal{P}_k$ with $\mathcal{P}_\ell$
  - for each other part $\mathcal{P}_\ell \in \mathcal{C}^\beta$ (mapped to column β)
    - compute the gain of swapping $\mathcal{P}_k$ with $\mathcal{P}_\ell$
  - perform a swap with the maximum gain



Proposed Refinement Heuristic

A horizontal swap of $\mathcal{P}_k$ and $\mathcal{P}_\ell$

- Let $p_{\alpha,\beta} = map(\mathcal{P}_k)$ and $p_{\alpha,\gamma} = map(\mathcal{P}_\ell)$
- for each $\mathcal{P}_t$ s.t. $nz(\mathcal{P}_t, \mathcal{P}_k) \neq 0$
  - Let $\mathcal{P}_t \in \mathcal{R}_\delta$
  - $load(\delta, \beta) \leftarrow load(\delta, \beta) - nz(\mathcal{P}_t, \mathcal{P}_k)$
  - $load(\delta, \gamma) \leftarrow load(\delta, \gamma) + nz(\mathcal{P}_t, \mathcal{P}_k)$
- for each $\mathcal{P}_t$ s.t. $nz(\mathcal{P}_t, \mathcal{P}_\ell) \neq 0$
  - Let $\mathcal{P}_t \in \mathcal{R}_\delta$
  - $load(\delta, \beta) \leftarrow load(\delta, \beta) + nz(\mathcal{P}_t, \mathcal{P}_\ell)$
  - $load(\delta, \gamma) \leftarrow load(\delta, \gamma) - nz(\mathcal{P}_t, \mathcal{P}_\ell)$



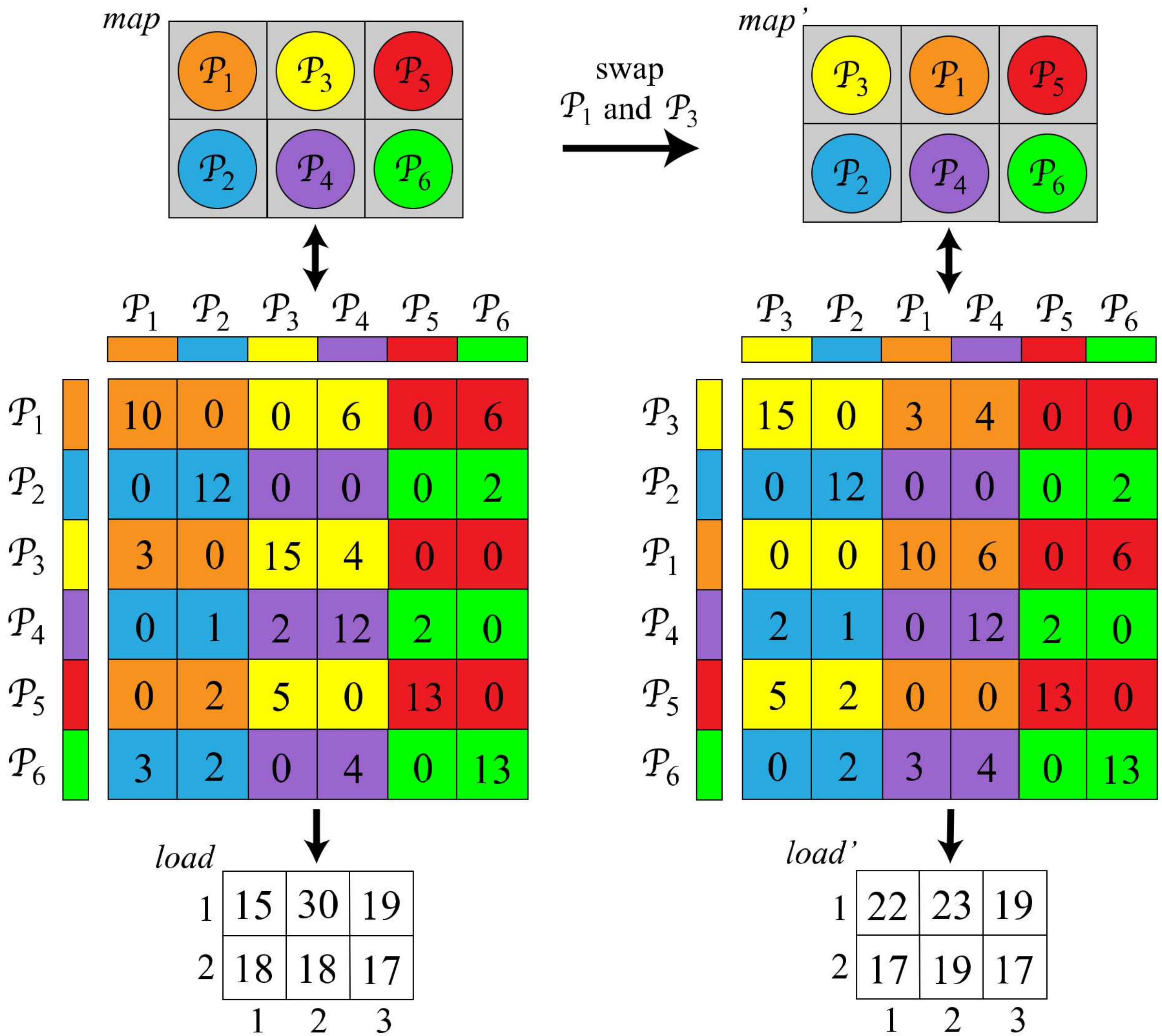
Cost Analysis:

1) cost(swap) = $O(M)$, where M is max part degree
2) cost(swap) = cost(computeGain)
3) cost(iteration) = cost(findMax) + $\sqrt{K}$ cost(swap) = $O(K + M\sqrt{K})$

avg no of iterations = 10

Proposed Refinement Heuristic

Swap example:



Experiments

Datasets

- Scale-free matrices from the SuiteSparse matrix collection [1]
- Scale-free: at least one dense row/column in the matrix
- Dense: at least 1% of entries are nonzero
- Three datasets
  - sym: 34 symmetric matrices
  - squ: 77 square but not symmetric matrices
  - rec: 32 rectangular matrices

[1] **Davis and Hu**, "The University of Florida sparse matrix collection", ACM TOMS, 2011.

Experiments

2D block cyclic partitioning with nonconformal vector partitions

- Baseline: 1D bipartite graph partitioning [1]
- Proposed: uses the baseline model in its first phase

Normalized results w.r.t. the baseline model [1]						
dataset	K	maximum computation	communication volume		number of messages	
			maximum	average	maximum	average
squ	64	0.93	1.75	1.28	**0.31**	0.43
	256	0.76	1.89	1.26	**0.20**	0.45
	1024	0.54	1.44	1.21	**0.14**	0.58
	4096	0.33	0.68	1.16	**0.10**	0.77
rec	64	1.19	1.68	1.67	**0.25**	0.35
	256	1.20	1.18	1.43	**0.16**	0.36
	1024	1.08	0.85	1.25	**0.13**	0.50

[1] **Hendrickson and Kolda**, "Partitioning rectangular and structurally unsymmetric sparse matrices for parallel processing", SISC, 2000.

Experiments

Refinement heuristic on

- Baseline: 2D block cyclic (2DBC) partitioning with conformal partitions
- with standard graph partitioning in the first phase

Normalized results w.r.t. the baseline 2DBC						
dataset	K	maximum computation	communication volume		number of messages	
			maximum	average	maximum	average
sym	64	**0.91**	1.03	1.02	1.00	1.01
	256	**0.83**	1.03	1.02	1.01	1.01
	1024	**0.80**	1.05	1.03	1.01	1.02
	4096	**0.75**	1.05	1.03	1.00	1.03
squ	64	**0.92**	1.04	1.05	1.01	1.03
	256	**0.88**	1.08	1.05	1.02	1.02
	1024	**0.81**	1.07	1.04	1.01	1.02
	4096	**0.83**	1.05	1.02	1.00	1.01

Experiments

Refinement heuristic on

- Baseline: 2DBC partitioning with nonconformal vector partitions
- with bipartite graph partitioning model in the first phase

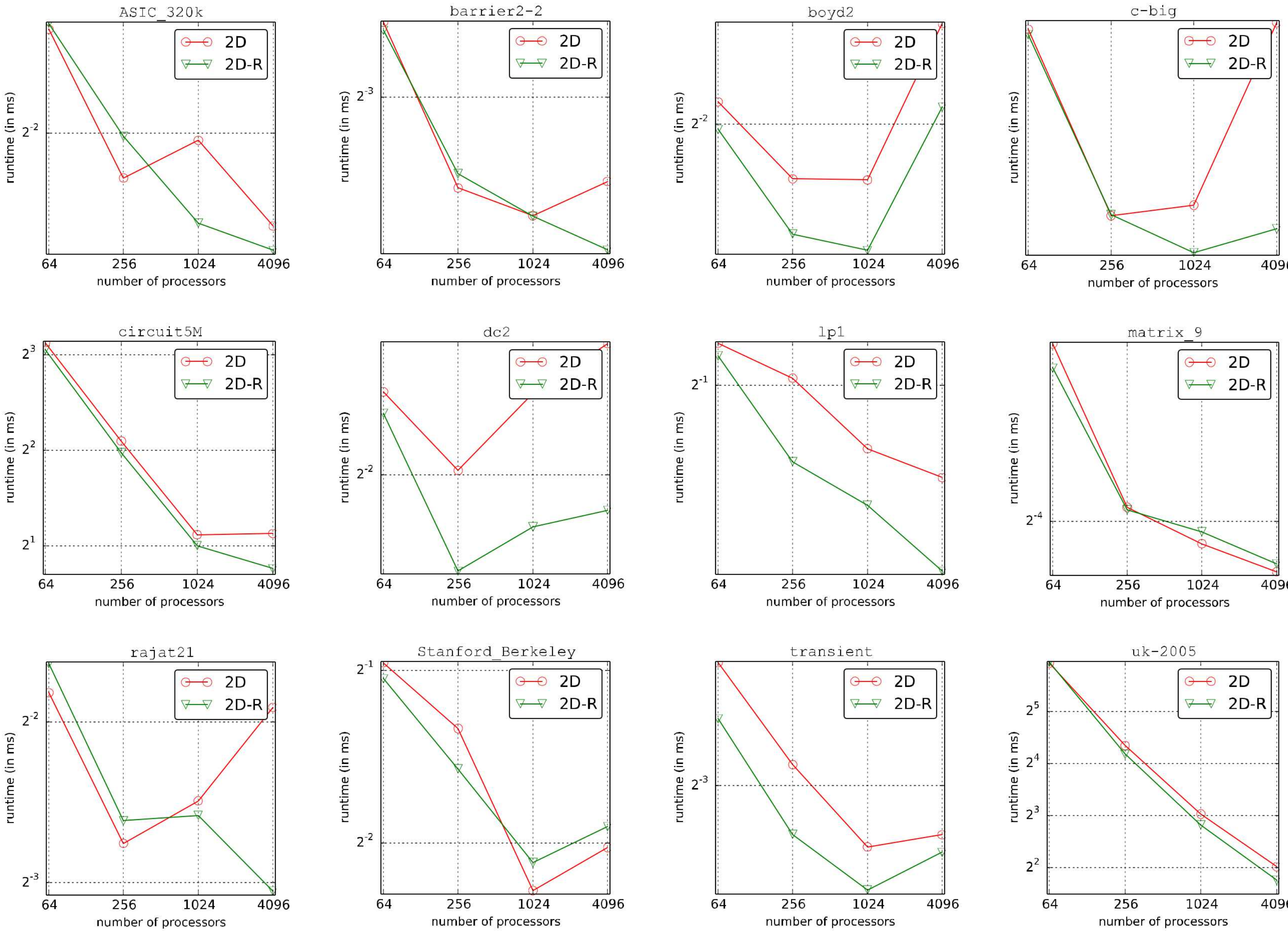
Normalized results w.r.t. the baseline 2DBC						
dataset	K	maximum computation	communication volume		number of messages	
			maximum	average	maximum	average
sym	64	**0.89**	1.65	1.22	1.01	1.03
	256	**0.88**	1.28	1.07	1.01	1.01
	1024	**0.87**	1.09	1.02	1.01	1.01
	4096	**0.85**	1.05	1.01	1.00	1.00
squ	64	**0.90**	1.60	1.24	1.00	1.02
	256	**0.87**	1.27	1.07	1.01	1.02
	1024	**0.86**	1.04	1.02	1.00	1.01

Experiments

Running time of the refinement heuristic:

Heuristic runtime normalized w.r.t. the graph partitioning runtime				
K	conformal partitions (standard graph model)		nonconformal partitions (bipartite graph model)	
	sym	squ	squ	rec
64	0.8%	0.8%	0.3%	0.3%
256	0.8%	0.7%	0.4%	0.4%
1024	1.6%	1.3%	0.7%	1.2%
4096	9.9%	6.1%	3.3%	-

Experiments



Conclusion

- 2D block cyclic partitioning method
- Extended it to nonconformal vector partitions
  - Up to 90% improvement in maximum message count
- Proposed a refinement heuristic to improve balance
  - Up to 25% improvement in computational imbalance