On performance analysis of heterogeneous parallel algorithms

Alexey Lastovetsky, Ravi Reddy
Presented by Joe Mayo

Homogeneous Parallel Algorithms (HomoPAs)

- Assumes a homogeneous environment
  - All processors have the same performance/speed
  - Communication costs are the same between nodes
- Performance analysis of HomoPAs well studied
  - PRAM, BSP, LogP
- Analysis accompanied by relatively small number of experiments
  - To show that the analysis is correct
  - To show the algorithm is actually faster

Heterogeneous Network Environment

- Processors have different performance properties
  - Processor Speed
  - Memory
- Communication is not constant
  - Network may force serial communication
  - Network may allow parallel communication
  - t_{comm} may be constant for a subset of nodes
  - t_{comm} may incur different cost for linking between subsets

Heterogeneous Parallel Algorithms (HeteroPAs)

- No adequate and practical model exists yet to predict execution time with satisfactory accuracy
- Often analyzed experimentally
  - Due to irregular nature of HeteroPAs, often not as convincing as when done with HomoPAs
  - May change when run on different environment
- Design often reduces to the problem of optimal data distribution
  - Proven to be NP-complete

Proposed Approach

- HeteroPA is often just a modification of a HomoPA
- Compare HeteroPA with its HomoPA prototype
- Asses the heterogeneous modification rather than as an isolated entity
- Compare efficiency of HeteroPA on heterogeneous network with efficiency of HomoPA prototype on homogeneous network
  - Homogeneous network has same aggregate performance as heterogeneous network

Equivalent Network Environments

- Postulate: HeteroPA cannot be more efficient than its HomoPA prototype
- Homogeneous network equivalent to heterogeneous if:
  1. Aggregate communication characteristics are the same
  2. Same number of processors
  3. Speed of each processor = average speed of processors on heterogeneous network
- Heterogeneous algorithm is considered *optimal* if its efficiency is the same as that of its homogeneous prototype

Heterogeneous Communication Networks

- Analysis works on a special case:
  - Network Assumptions:
    - Composed of segments
    - Fast homogeneous comm within segment
    - Interconnected via slower comm links
    - Limit links between segments to serial comm

Heterogeneous Communication Networks

- Analysis works on a special case:
  - HeteroPA Design Assumptions:
    - Relatively rare point-to-point comm ops
    - Large computation between comm ops
    - Comm op consists of passing long messages
- More general cases are left for future research

Analysis Problem Description

- Two sets of processors:
  - Both sets consist of same number of processors
  - First set (Homogeneous Network):
    - All processors are identical
    - All processors belong to same homogeneous comm segment
    - Aggregate performance is same as that of the second set
  - Second set (Homogeneous Network):
    - Includes processors of different speeds that span several comm segments
    - Average speed of point-to-point comm is same as speed of point-to-point comm for the first set

Analysis Problem Description

- n = # procs in first set
- v = speed of processors in first set
- s = comm speed of segment for the first set

- P = Set of processors in second set
- m = # comm segments: S_1, S_2,…,S_m
- s_i = comm speed between procs in segment S_i
- n_i = # procs in segment S_i
- v_{ij} = speed of *j*th processor in segment S_i
- s_{ij} = speed of comm link between segments S_i and S_j

Analysis Problem Description

Average speed of point-to-point communications between processors in second set should be equal to speed of point-to-point communications in the first:

$$\frac{\sum_{i=1}^{m} s_i \times \frac{n_i \times (n_i - 1)}{2} + \sum_{i=1}^{m} \sum_{j=i+1}^{m} n_i \times n_j \times s_{ij}}{\frac{n \times (n-1)}{2}} = s,$$

Analysis Problem Description

Total number of processors in each set should be equal:

$$\sum_{i=1}^{m} n_i = n,$$

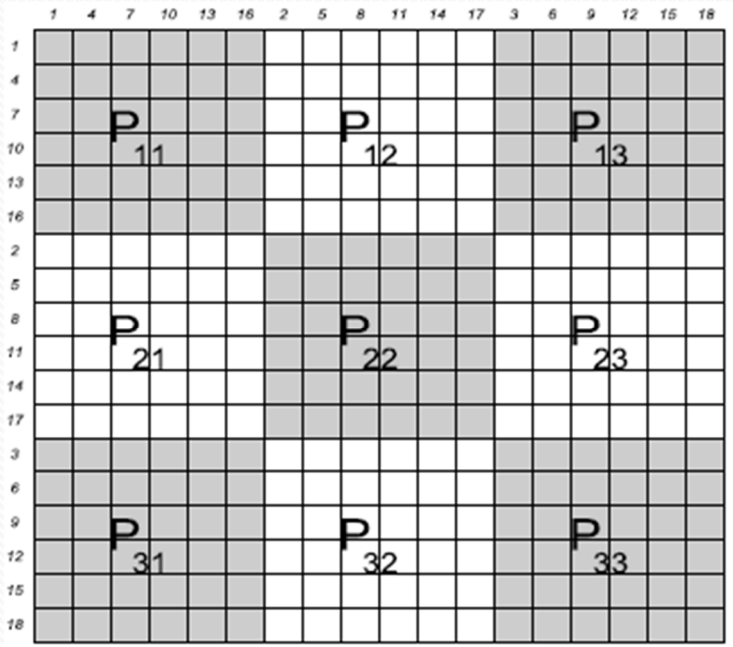
Aggregate performance of processors should be equal:

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} v_{ij} = n \times v$$

Matrix Multiplication: HomoPA Prototype

Data Partitioning:

- Unit of computation: an r x r block of matrix elements
- Each matrix (A,B,C) are partitioned identically into *p* equal squares, called general blocks
- Each processor responsible for calculating its part of C
- Processors arranged in a 2D *mxm* grid, where $m = \sqrt{p}$

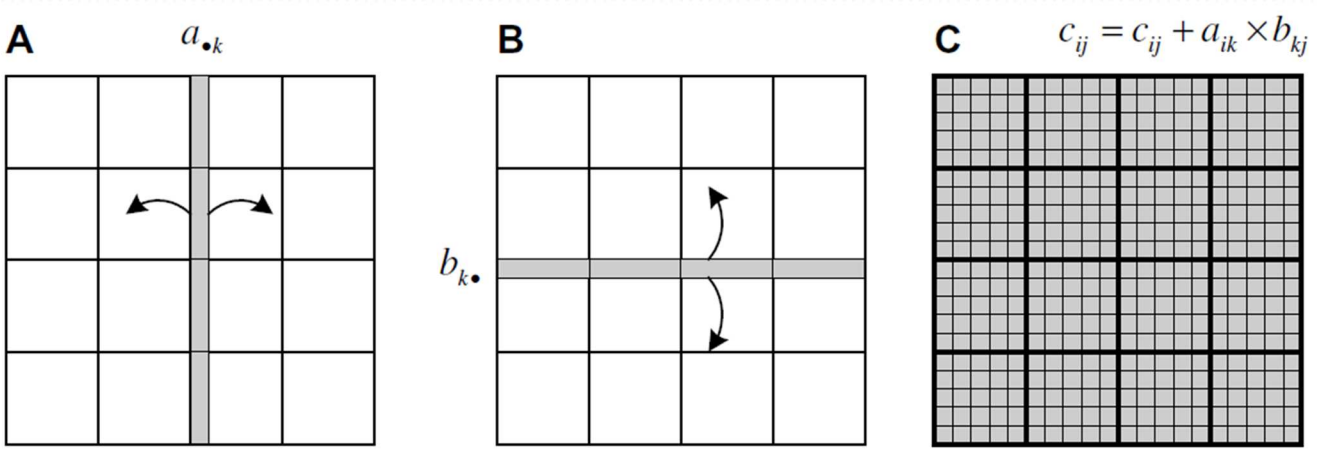


Matrix Multiplication: HomoPA Prototype

Algorithm consists of k = n/r steps

At each step k:

- Pivot column of matrix A broadcasted horizontally
- Pivot row of matrix B broadcasted vertically
- Each processor, P_{ij}, updates each unit in its C general block: c_{ij} += a_{ik} x b_{kj}



Matrix Multiplication: HomoPA Prototype

After n/r steps:

$$c_{ij} = \sum_{k=1}^{\frac{n}{r}} a_{ik} \times b_{kj},$$

i.e., $C = A \times B$.

Matrix Multiplication: HomoPA Prototype

- Strong dependence between successive steps
  - Since r is usually much less than m: at step k+1, PiK will again be the root of the broadcast
  - At step k+1, the broadcast involving the row of processors Pi* cannot start until processor PiK completes the broadcast at step k
  - Think about how an implementation must use a buffer
    - Cannot reuse buffer until previous broadcast has completed
  - Hinders parallel execution

Matrix Multiplication: HomoPA Prototype

- Removing strong dependence:
  - Within the same set of processors
  - Broadcast ops at successive steps have different root processors
  - Answer:
    - 2-D Block Cyclic Distribution

2-D Block Cyclic Distribution

- *m*x*m* processor grid
- Each element in A,B, and C is a square *r*x*r* block
- Blocks scattered in a cyclic fashion along both dimensions
  - Blocks aij, bij, cij will be mapped to Pij so that
    - I = (i-1) mod m + 1
    - J = (j-1) mod m + 1

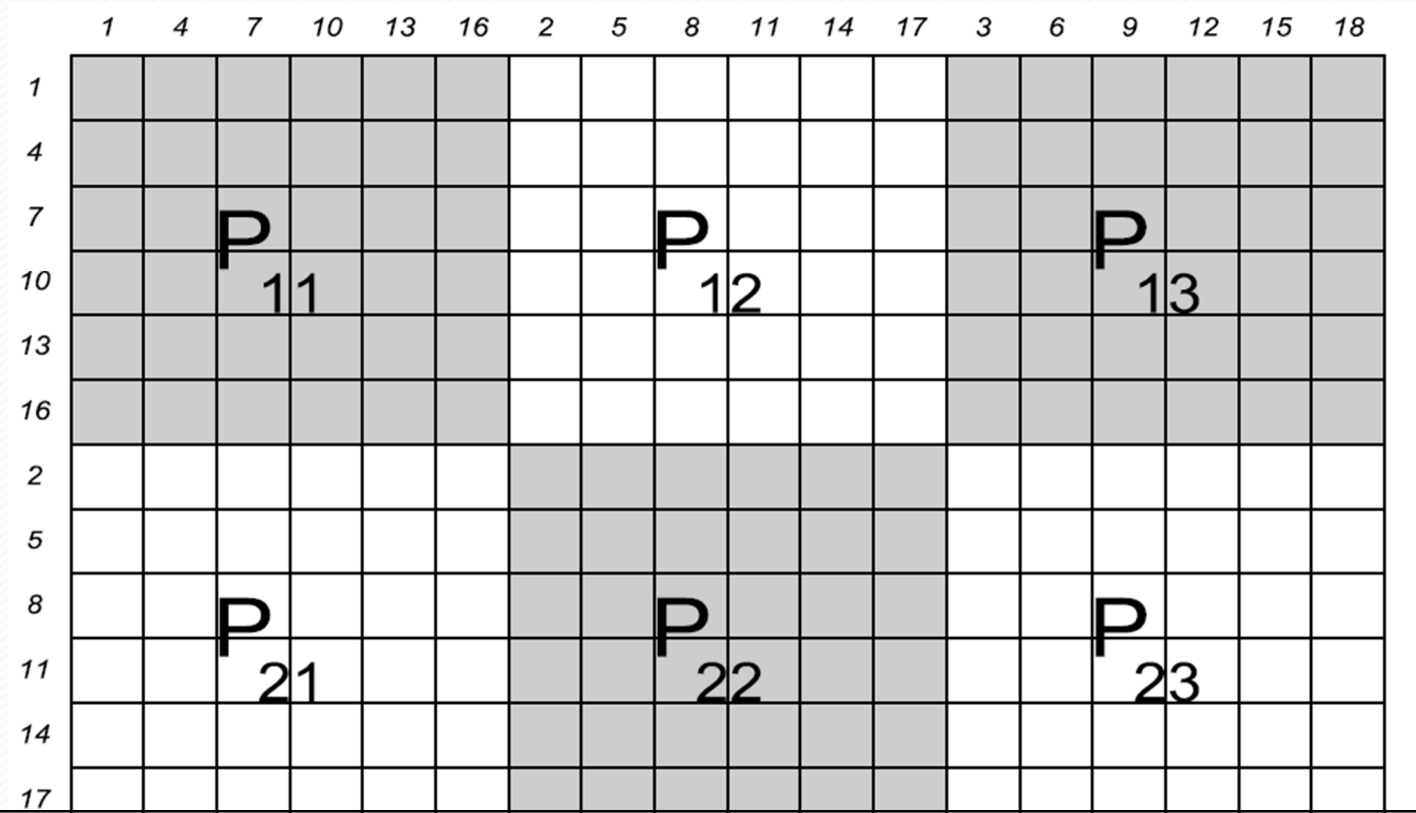
2-D Block Cyclic Distribution

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}
2	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{11}	P_{11}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}
3	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{11}	P_{11}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}
4	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}
5	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}
6	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}
7	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}
8	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}
9	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}
10	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}
11	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}
12	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}
13	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}
14	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}
15	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}
16	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}
17	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}
18	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}

	1	2	3	4	5	6	7	8
1	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}
2	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{11}
3	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{11}
4	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}
5	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}
6	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}	P_{33}	P_{31}	P_{32}
7	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}	P_{13}	P_{11}	P_{12}
8	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}	P_{23}	P_{21}	P_{22}

2-D Block Cyclic Distribution

Element area covered by each process is the same as with the static partitioning described before:



Parallel Matrix Multiplication on Heterogeneous Platform

- New problem:
  - Each processor may have a different speed
  - Leads to unbalanced workload
- Objective:
  - Keep units of computation to *r*x*r* unit blocks
  - # unit blocks allocated to each processor should be proportional to its speed
  - Still the *optimal partitioning problem*

Heterogeneous 2-D Block Cyclic Distribution

- Let relative speed of processor P_{ij} is a real positive number, s_{ij}, so that

$$\sum_{i=1}^{m}\sum_{j=1}^{m} s_{ij} = 1$$

- The area of rectangle allocated to P_{ij} should be:
  - $s_{ij} \times n^2$

Heterogeneous 2-D Block Cyclic Distribution

- Generalized blocks:
  - New size:

    (rxl) x (rxl), where $m <= l <= n/r$
  - Partitioned into m^2 blocks
    - Each block assigned to a different processor
    - Area of each block is proportional to the speed of the processor that stores that block
    - Therefore unequal block sizes
  - Each generalized block is partitioned the same

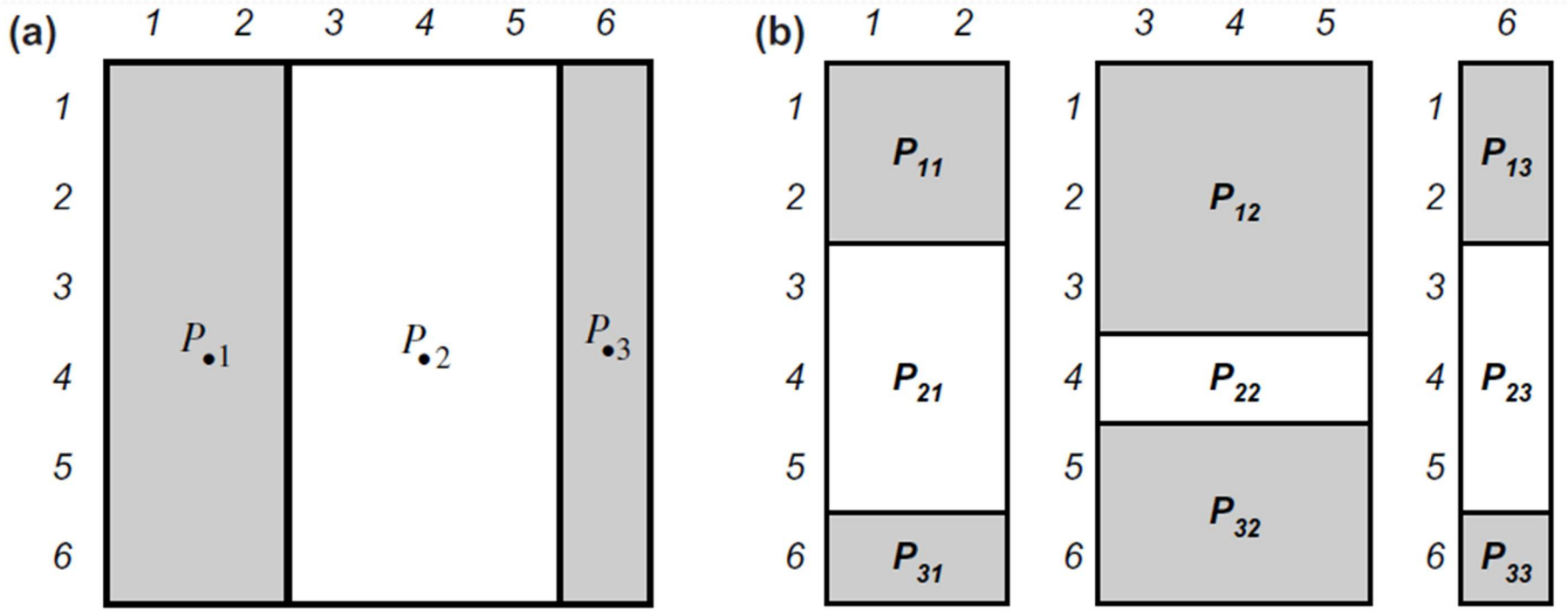
Heterogeneous 2-D Block Cyclic Distribution

- Partitioning of a Generalized Block:
  - *l*x*l* square of element blocks
  - Each element block is a square of *r*x*r* matrix elements
  - *l*x*l* square partitioned into *m* vertical slices
    - Area of *j*th slice is proportional to $\sum_{i=1}^{m} s_{ij}$
  - Each slice is then partitioned independently into *m* horizontal slices
    - Area of *i*th horizontal slice in *j*th vertical slice is proportional to s_{ij}
  - End up with *i*th horizontal slice in *j*th vertical slice is assigned to P_{ij}

Heterogeneous 2-D Block Cyclic Distribution

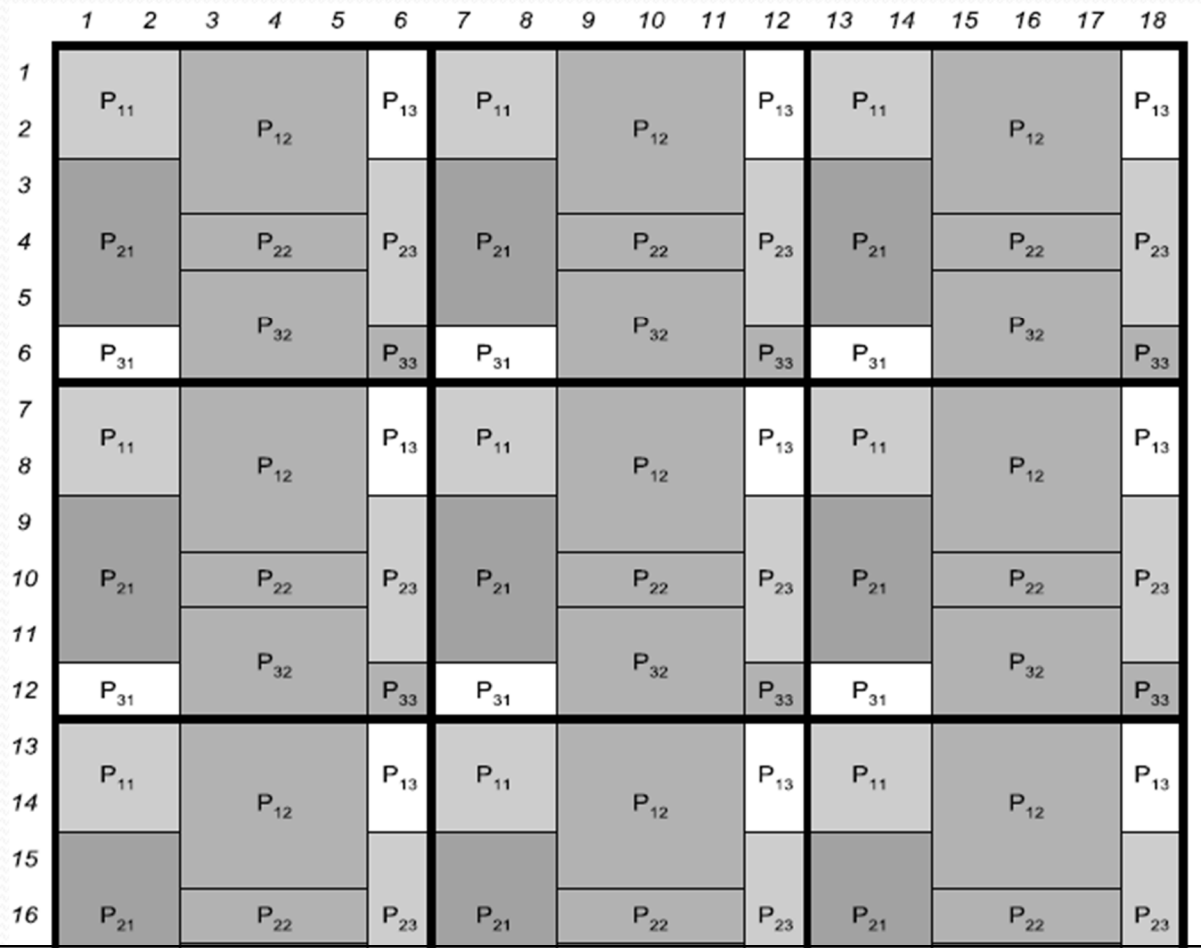
(a) Vertical slices

(b) Horizontal slices



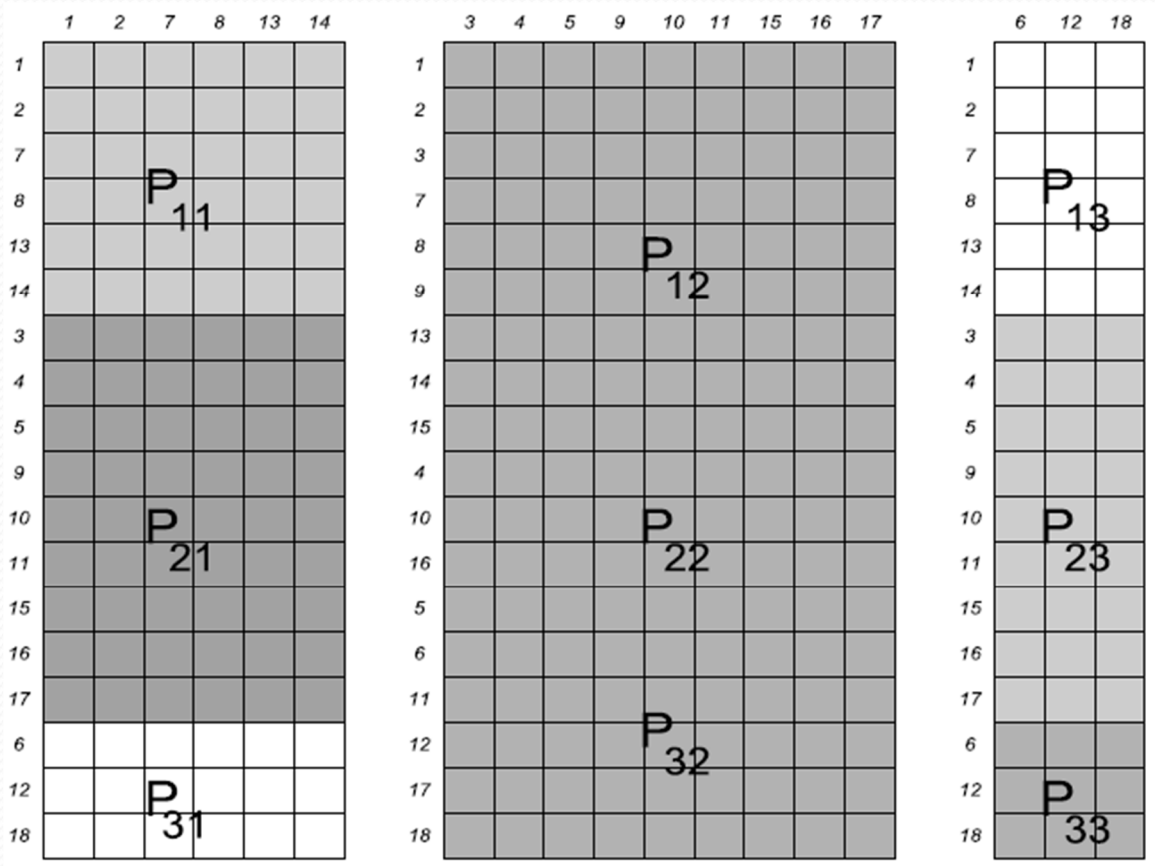
Heterogeneous 2-D Block Cyclic Distribution

Each generalized block has the same partioning:



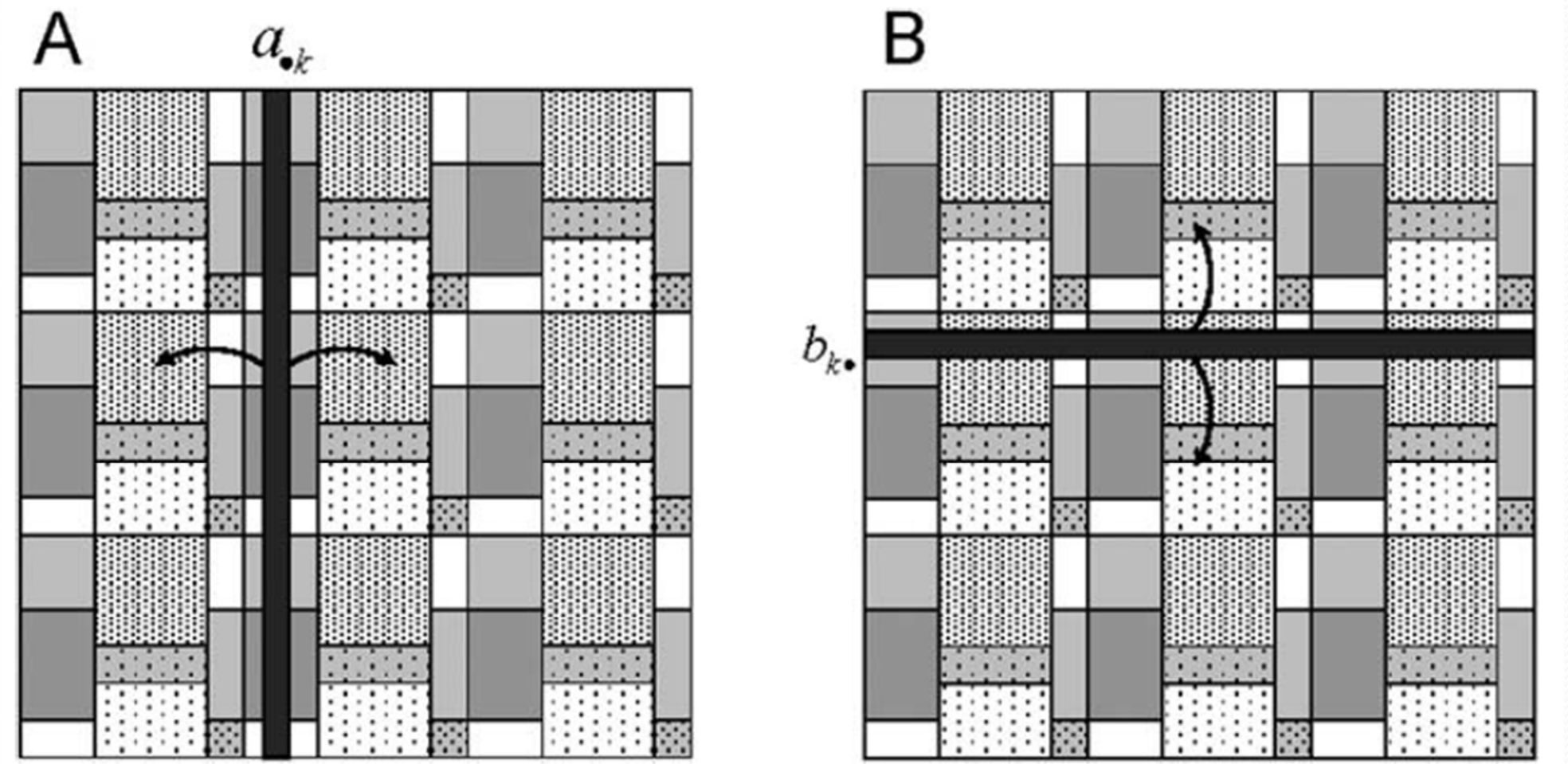
Heterogeneous 2-D Block Cyclic Distribution

Area assigned to each processor is proportional to its relative speed:



Heterogeneous 2-D Block Cyclic Communication

- Still have the same number of comm ops:



Effect of parameter l

- Range: $[m, \frac{n}{r}]$
- Controls two conflicting aspects of the algorithm:
  - Accuracy of load balancing
  - Level of potential parallelism in execution of successive steps of the algorithm
- The greater l
  - the better the load of processors is balanced
  - The stronger the dependence between successive steps, hindering parallel execution of steps

Effect of parameter *l*

- When $l = \frac{n}{r}$,
  - Distribution provides best possible balance of load to processors
  - Turns into pure 2-D block distribution
  - Lowest possible level of parallel execution on successive steps

Effect of parameter *l*

- When $l = m$,
  - Distribution identical to homogeneous distribution
  - Doesn't do any load-balancing at all
  - Provides highest possible level of parallel execution on successive steps

- Optimal value of l somewhere in between m and $\frac{n}{r}$

Heterogeneous Parallel Algorithm Summary

- The core is partitioning of a generalized block into uneven rectangles
- Easily generalized for arbitrary 2-D processor arrangement
- Optimal partitioning a square into rectangles
  - Of any shape or arrangement
  - Proved to be NP-complete

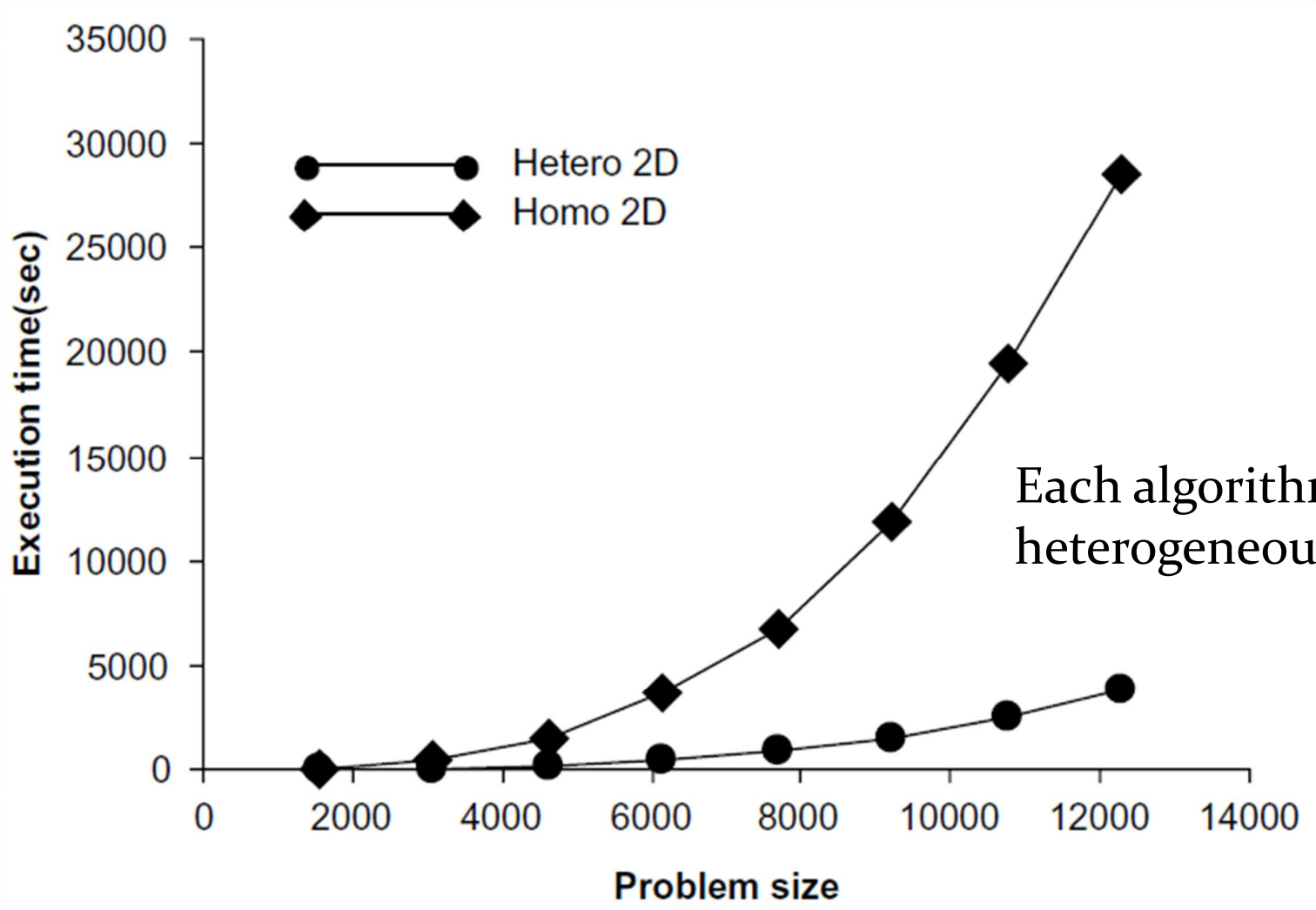
Assessment of HeteroPA

- Compare HeteroPA with its HomoPA prototype
- Assume n,m and r are the same
- Both algorithms consist of $\frac{n}{r}$ successive steps
- At each step
  - Equivalent comm ops are performed by each algorithm
  - If l is big enough, each processor of heterogeneous platform performs volume of computation proportional to its speed

Assessment of HeteroPA

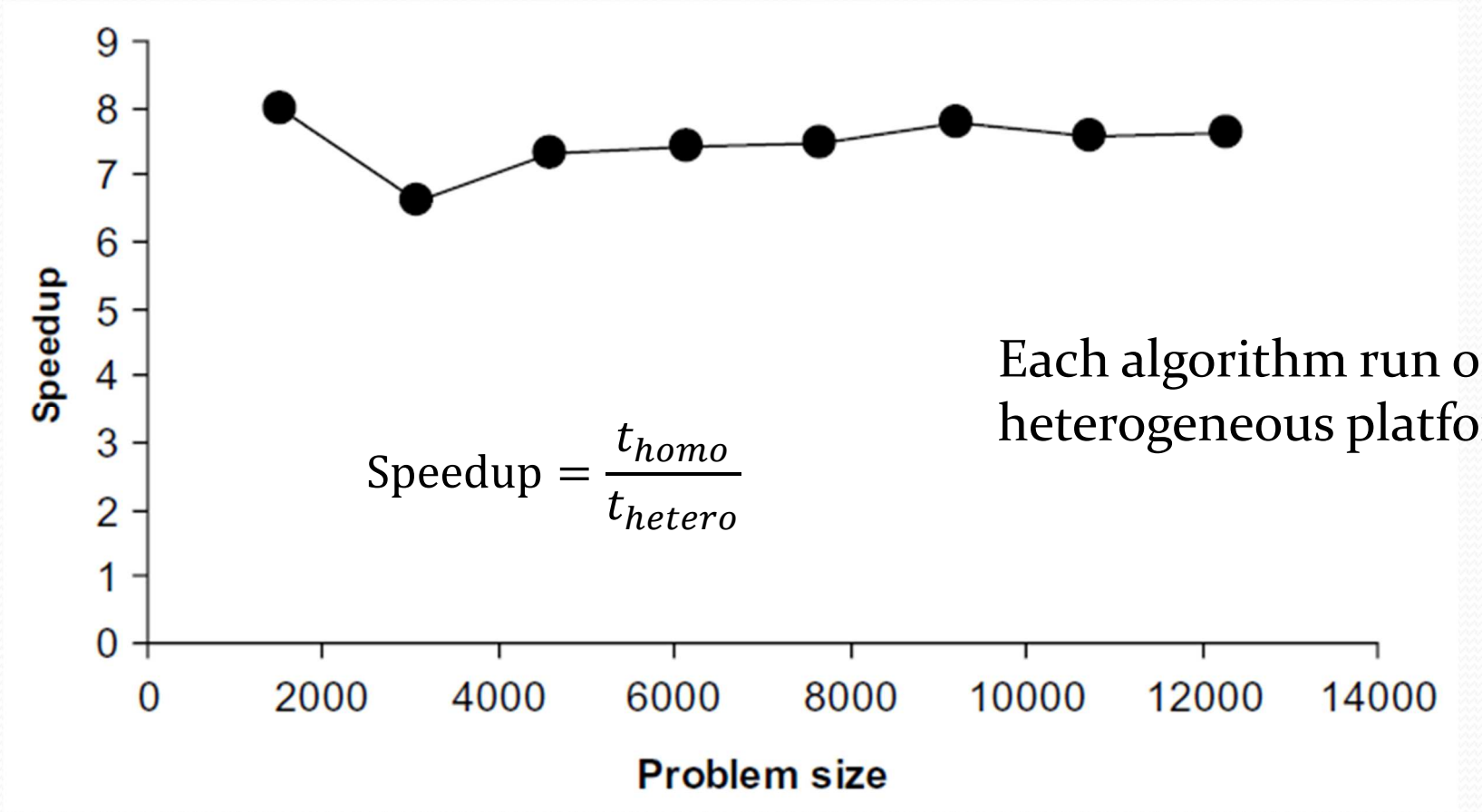
- Only source of less efficiency is due to lower level of potential overlapping of comm ops at successive steps
  - A bigger ration between max/min processor speed, the lower is this level
- If communication layer serializes data packages
  - HeteroPA has approximately same efficiency as HomoPA
  - Then HeteroPA is the optimal modification of its HomoPA prototype

Experimental Results



Each algorithm run on heterogeneous platform

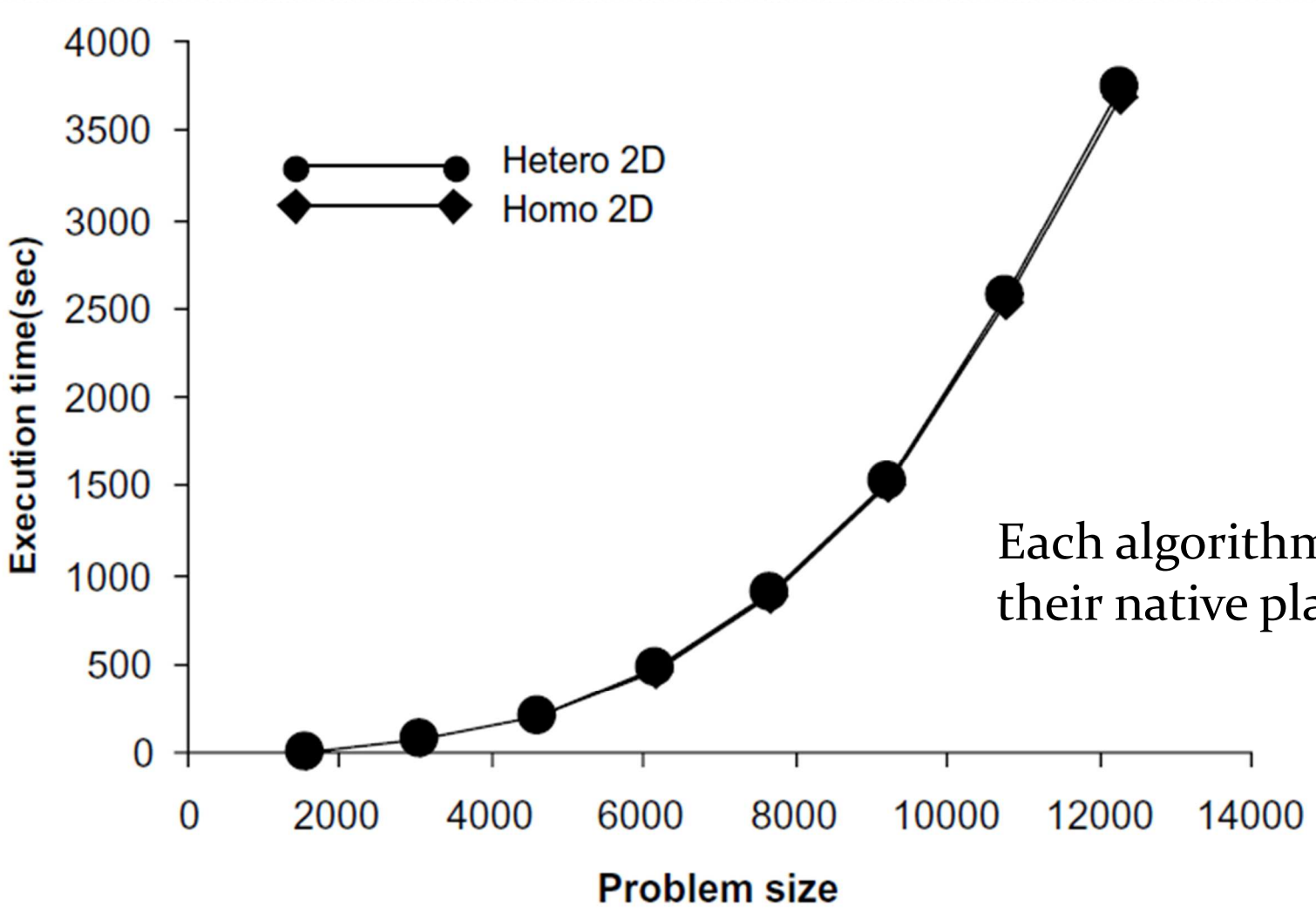
Experimental Results



$$\text{Speedup} = \frac{t_{homo}}{t_{hetero}}$$

Each algorithm run on heterogeneous platform

Experimental Results



Each algorithm run on their native platform

Experimental Results

- Observations
  - Speedup due to overlapping of comm ops is not very significant
  - Speed of processors in heterogeneous network doesn't differ too much
    - additional dependence between steps is very weak

Other Experiments

- Tried using different values for *r* and different mappings of processors onto the *r*x*r* grid
  - Optimal mapping should arrange processors in a non-increasing order of their speed along each row/column
- Different shapes of general blocks (*p*x*q)*
  - Ex: 2x6, 3x4, etc
  - Compared with different values for *r* and *l*
  - Shape does in fact influence efficiency of the algorithm if far off from optimal values of *r* and *l*
  - Doesn't influence execution time of the algorithm if *r* and *l* are optimal

Summary

- Design of heterogeneous parallel algorithms is typically reduced to the problem:
  - Optimal data partitioning of some mathematical object such as a set or matrix
- Small number of experiments in a heterogeneous environment
- Compare efficiency to that of a HomoPA prototype run on an equally powerful homogeneous environment