Network-Oblivious Algorithms

Francesco Silvestri

Department of Information Engineering, University of Padova, Italy
francesco.silvestri@dei.unipd.it
www.dei.unipd.it/~silvest1

ABSTRACT

Communication is a major factor determining the performance of algorithms on current parallel computing systems. Reducing the communication requirements of algorithms is then of paramount importance, if they have to run efficiently on physical machines. Recognition of this fact has motivated a large body of results in algorithm design and analysis, but these results do not yet provide a coherent and unified theory of the communication requirements of computations. One major obstacle toward such a theory lies in the fact that communication is defined only with respect to a specific mapping of a computation onto a specific machine structure. Furthermore, the impact of communication on performance depends on the latency and bandwidth properties of the machine. In this scenario, algorithm design, optimization, and analysis can become highly machine dependent, which is undesirable from the economical perspective of developing efficient and portable software.

It is natural to wonder whether algorithms can be designed that, while independent of any machines, are nevertheless efficient for a wide set of machines. In other words, we are interested in exploring the world of efficient *network-oblivious* algorithms, in the same spirit as the exploration of *cache-oblivious* algorithms [2]. We develop a framework where the concept of network-obliviousness and of algorithmic efficiency are precisely defined. In this framework, a network-oblivious algorithm is designed in a model of computation where the only parameter is the problem's input size. Then, the algorithm is evaluated on a model with two parameters, capturing parallelism and granularity of communication. We show that, for a wide class of network-oblivious algorithms, optimality in the latter model implies optimality in a block-variant of the Decomposable BSP model, which effectively describes a wide class of parallel platforms. We illustrate our framework by providing optimal network-oblivious algorithms for a few key problems, and also establish some negative results.

This abstract is based on results appeared in [1].

References

[1] G. Bilardi, A. Pietracaprina, G. Pucci, and F. Silvestri. Network-Oblivious Algorithms. In *Proc. of 21st International Parallel and Distributed Processing Symposium*, 2007.

[2] M. Frigo, C. Leiserson, H. Prokop, and S. Ramachandran. Cache-Oblivious Algorithms. In *Proc. of 40th Symp. on Foundations of Computer Science*, 1999.

NETWORK-OBLIVIOUS ALGORITHMS

G. Bilardi, A. Pietracaprina, G. Pucci, F. Silvestri

{bilardi,capri,geppo,silvest1}@dei.unipd.it

DEI - Università di Padova

ACG

Advanced Computing Group

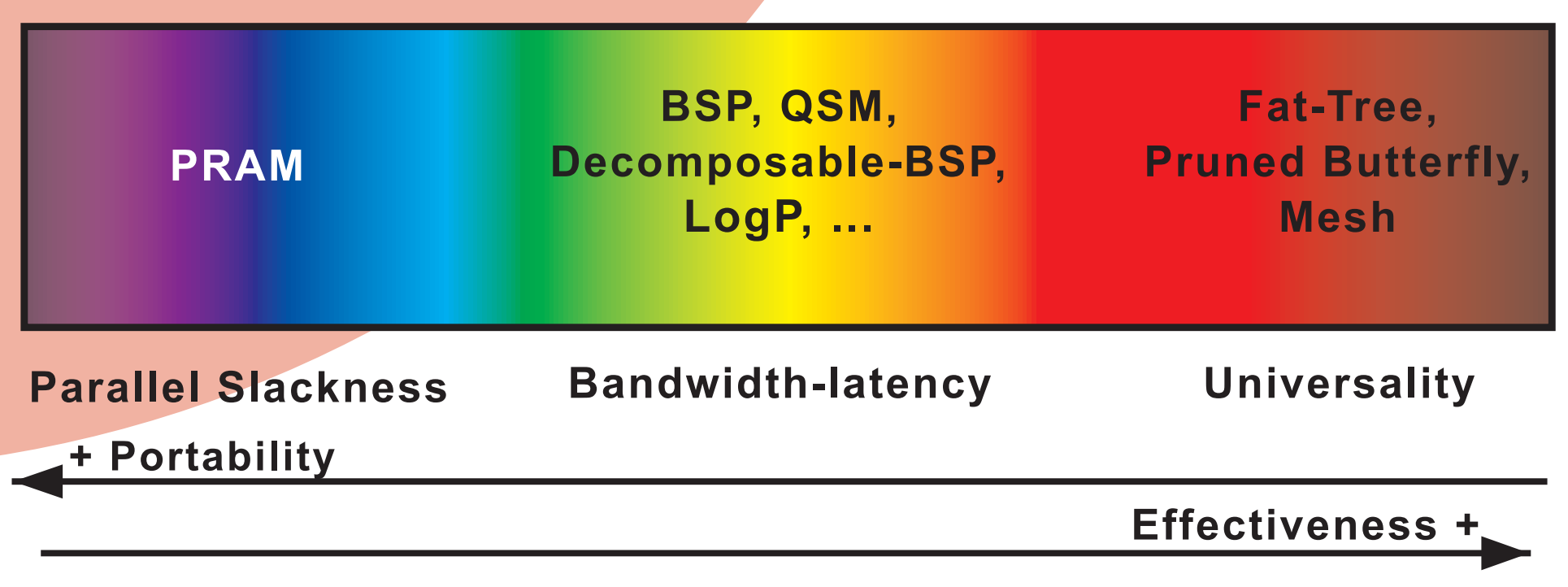
SADA07

Helsinki, Finland

28 May - 1 June 2007

COMMUNCATION COST

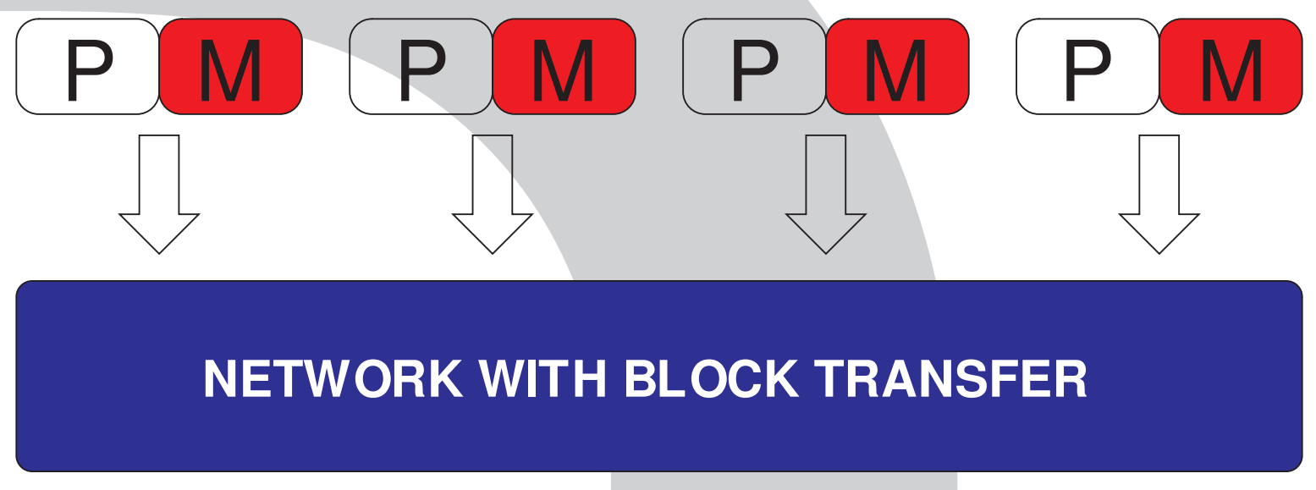
- Communication heavily affects the efficiency of parallel algorithms
- Communication costs depend on interconnection topology and other machine-specific characteristics
- Models of computation for parallel algorithm design aim at striking some balance between portability and effectiveness



OUR RESULTS

- Notion of network-oblivious algorithm
- Framework for design, analysis, and execution of network-oblivious algorithms
- Network-oblivious algorithms for case study applications: matrix multiplication and transposition, FFT and sorting
- Impossibility result for matrix transposition

EVALUATION MODEL $\mathrm{M}(p, B)$

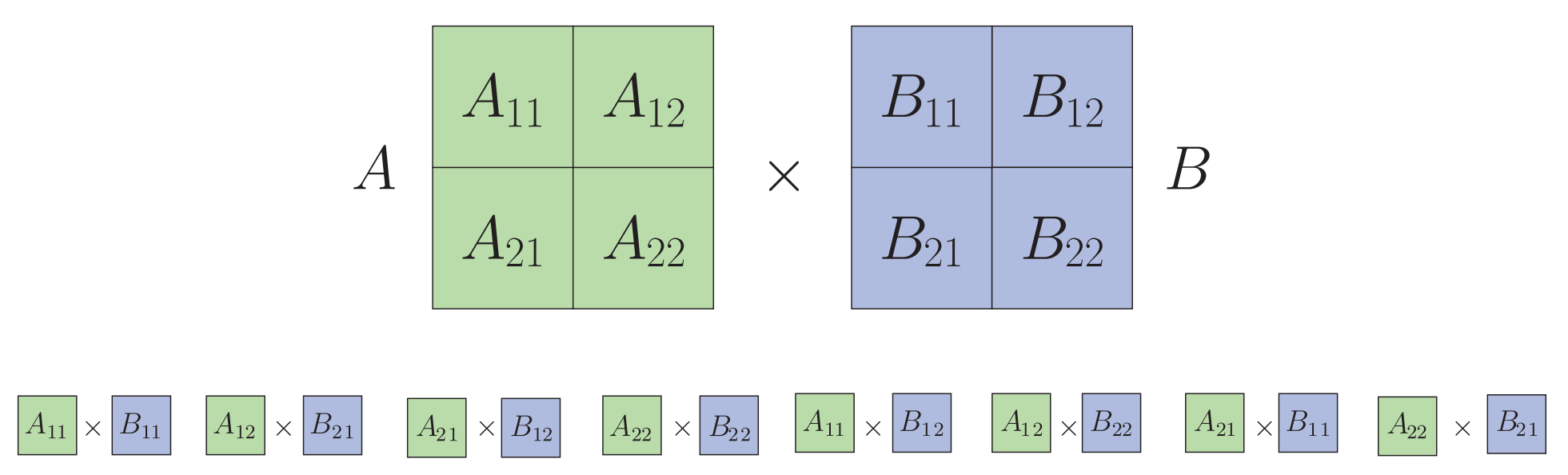


- $\mathrm{M}(p, B)$ is a $\mathrm{M}(p)$ where:
  - Data exchanged between two PEs travel within blocks of B words
  - Block-degree $h^s(p,B)$: maximum number of blocks sent/received by a PE in a superstep s
  - Communication complexity of A: $\sum_{\forall s \text{ of } A} h^s(p,B)$
- Execution of an $\mathrm{M}(n)$-algorithm on an $\mathrm{M}(p, B)$:
  - Every PE of $\mathrm{M}(p, B)$ simulates a segment of n/p consecutive PEs of $\mathrm{M}(n)$
  - Communications between PEs of $\mathrm{M}(n)$ in the same segment ⇒ local computations in $\mathrm{M}(p, B)$

DEFINITION

A network-oblivious algorithm A for æ is optimal if, ∀ instance of size n and ∀ p ñ n and Bò 1, the execution of A on an $\mathrm{M}(p, B)$ yields an algorithm with asymptotically minimum communication complexity among all $\mathrm{M}(p, B)$-algorithms for æ

MATRIX MULTIPLICATION



Problem: multiplying two $\sqrt{n} \times \sqrt{n}$ matrices, A and B

1) Row-major distribution of A and B among the n PEs

2) Subdivision of the problem into 8 subproblems

3) Each subproblem is solved in parallel within a distinct segment of $n/8$ PEs

- Optimal communication complexity $\Theta\left(\frac{n}{Bp^{2/3}}\right)$ on $\mathrm{M}(p, B)$ for B ñ n/p
- Optimal communication time on D-BSP$(p, \boldsymbol{g}, \boldsymbol{B})$ for B_i ñ n/p
- $\Theta\left(p^{1/3}\right)$ memory blow-up, unavoidable if minimal communication is sought

This work was supported in part by MIUR of Italy under project MAINSTREAM, and by EU under the EU/IST Project 15964 AEOLUS

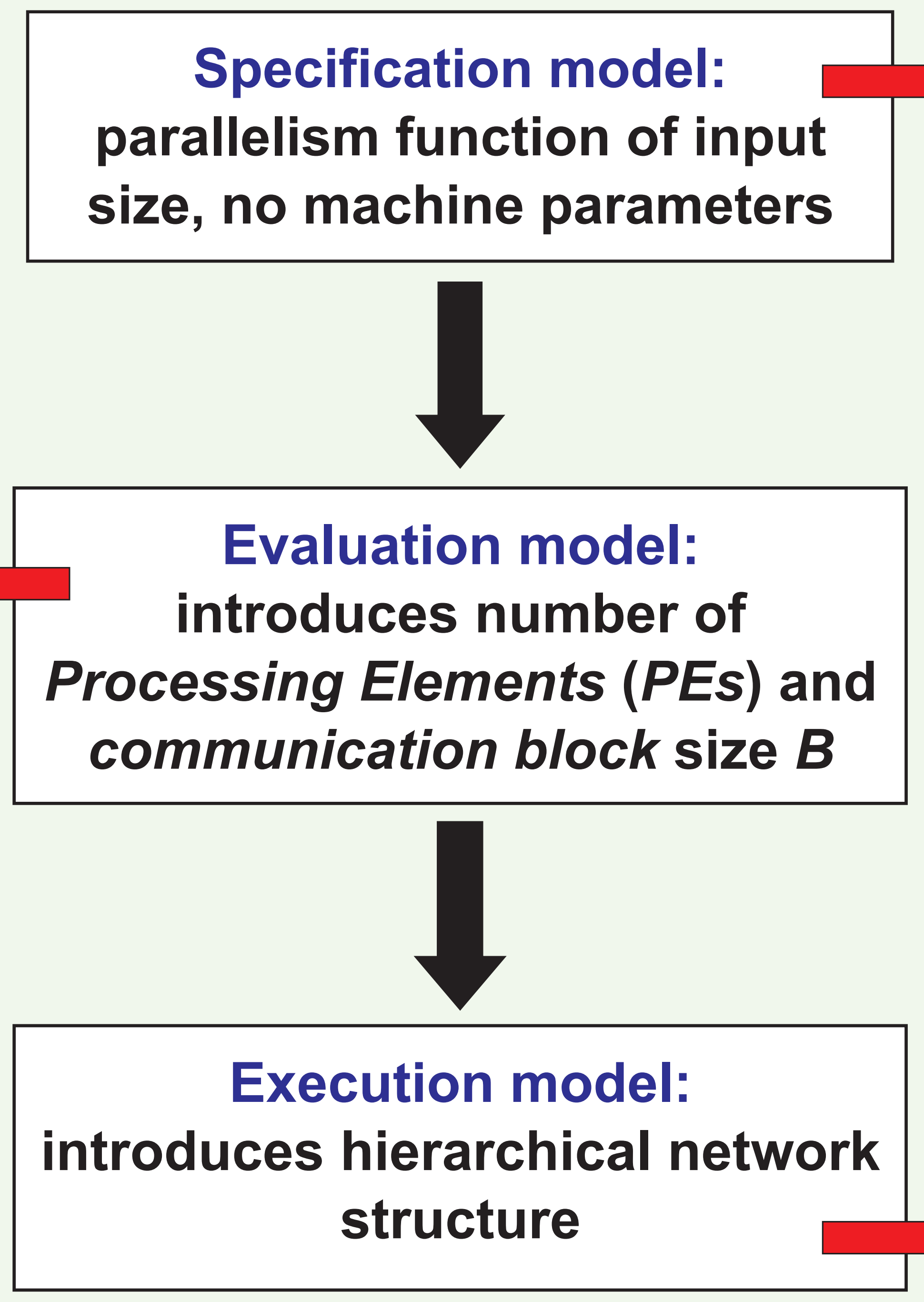
OBLIVIOUSNESS

- Broad consensus on bandwidth-latency models:
  - Parameters capture relevant machine characteristics
  - Logarithmic number of parameters sufficient to achieve high effectiveness (e.g., D-BSP) [Bilardi *et al.*, 99]

QUESTION

Can we design efficient parallel algorithms oblivious to any machine/model parameters?

FRAMEWORK FOR NETWORK-OBLIVIOUS ALGORITHMS



Specification model: parallelism function of input size, no machine parameters

Evaluation model: introduces number of *Processing Elements* (*PEs*) and *communication block* size *B*

Execution model: introduces hierarchical network structure

OPTIMALITY RESULT

THEOREM

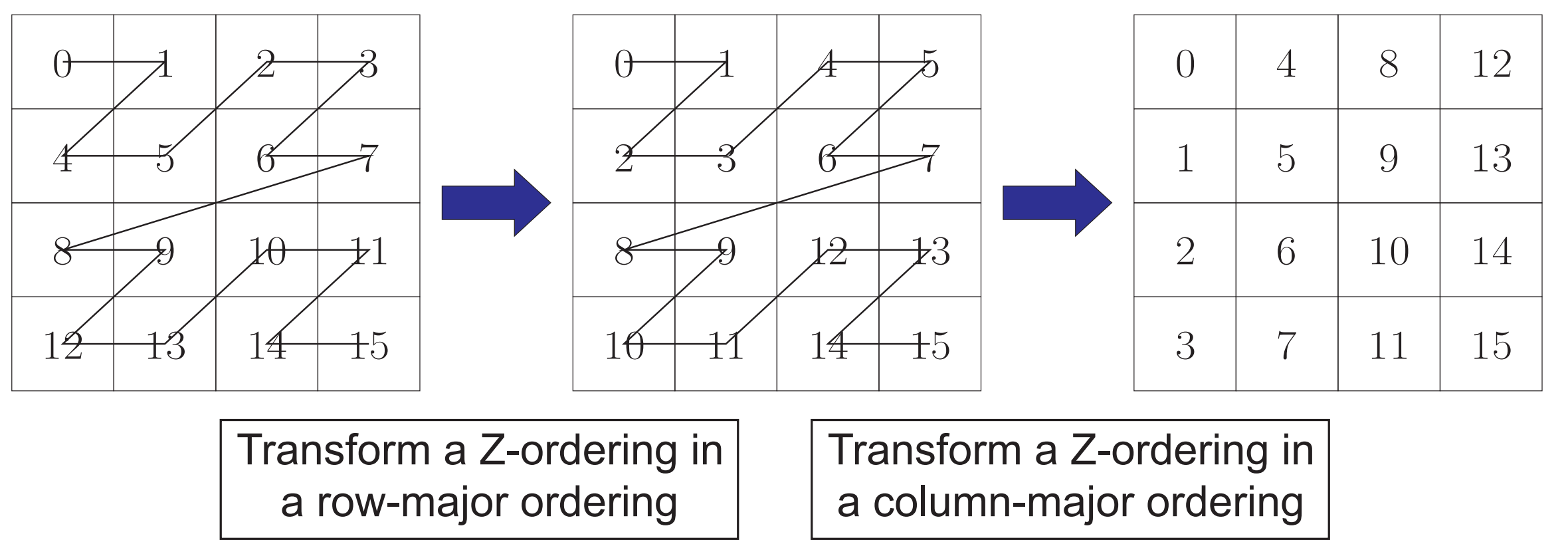
An optimal network-oblivious algorithm A exhibits an asymptotically optimal communication time when executed on a D-BSP$(p, \boldsymbol{g}, \boldsymbol{B})$ with p ñ n under the following conditions:

Wiseness: *for each superstep of A, its communications are either **almost all local** or **almost all non-local** w.r.t. D-BSP$(p, \boldsymbol{g}, \boldsymbol{B})$ PEs*

Fullness: *all communicated blocks are **almost full***

Remark: The actual wiseness and fullness conditions specified in the paper are less restrictive

MATRIX TRANSPOSTION



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

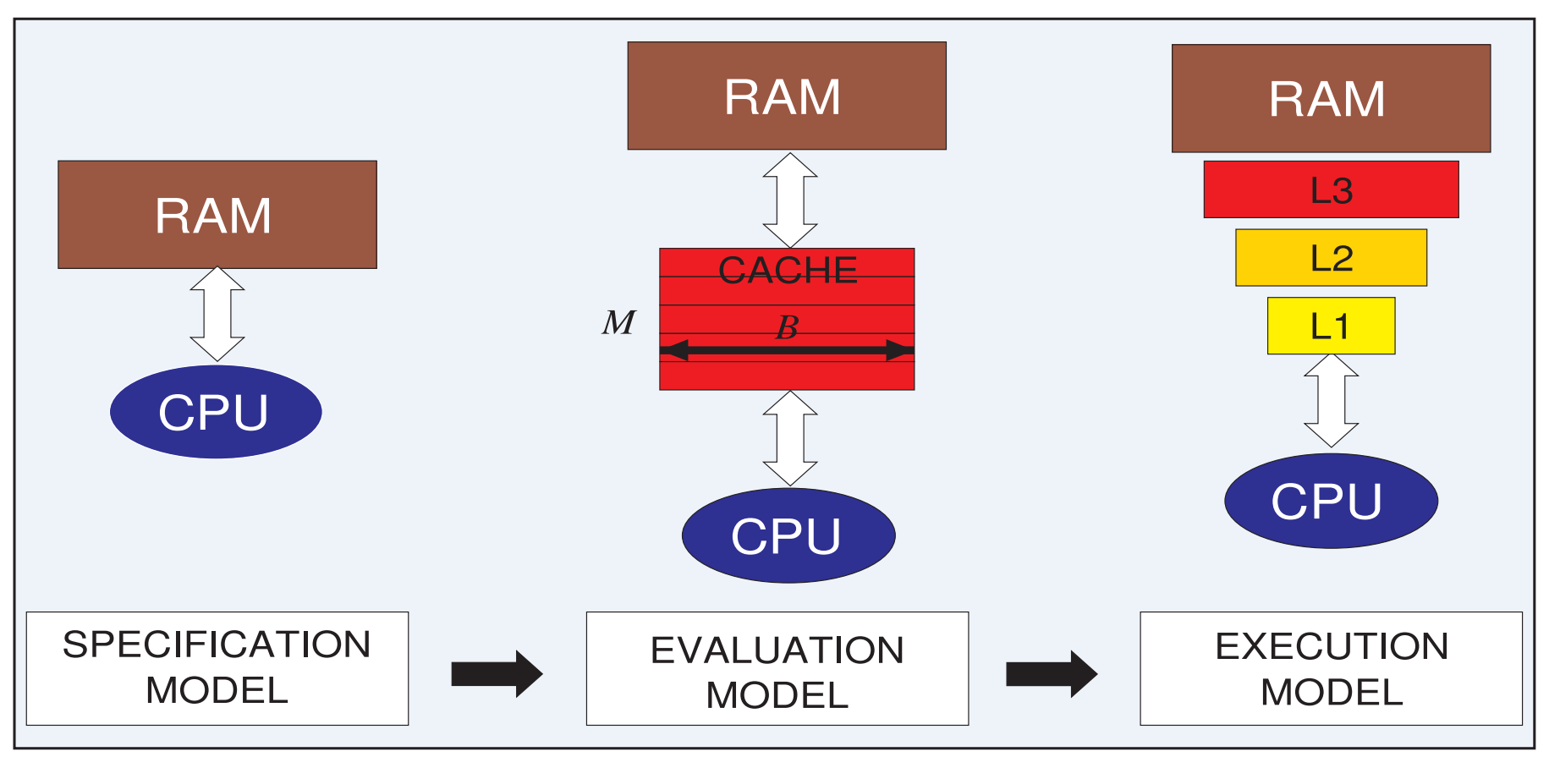
Transform a Z-ordering in a row-major ordering

Transform a Z-ordering in a column-major ordering

Problem: transpose a $\sqrt{n} \times \sqrt{n}$ matrix

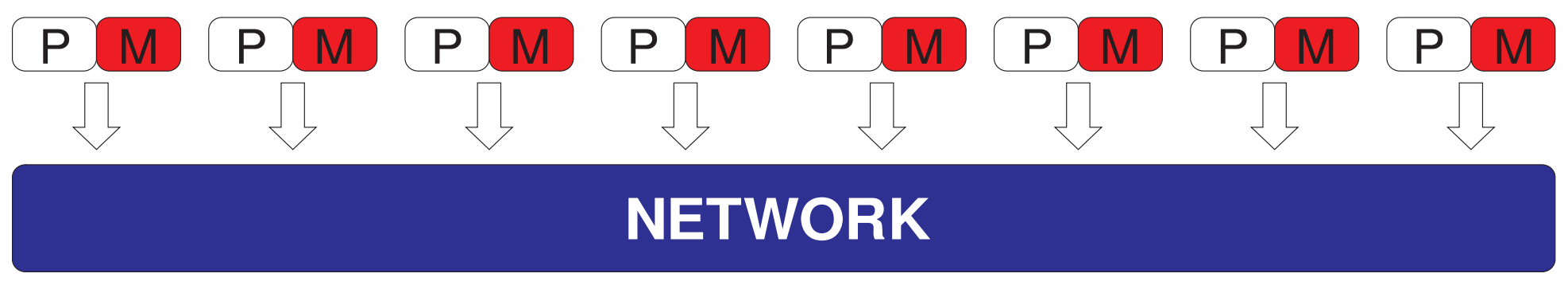
- We use a two-step algorithm based on Z-Morton ordering
- Optimal communication complexity $\Theta\left(\frac{n}{pB}\right)$ on $\mathrm{M}(p, B)$ for $B \leq \sqrt{n/p}$
- Optimal communication time on D-BSP$(p, \boldsymbol{g}, \boldsymbol{B})$ for $B_i \leq \sqrt{n/p}$
- Constraint $B \leq \sqrt{n/p}$ reminiscent of the tall-cache assumption
  - Necessary to achieve cache-oblivious optimality for the matrix transposition problem [Silvestri, 06]

CACHE-OBLIVIOUS ALGORITHMS



- Parameters M, B are not used for algorithm design
- Optimality in a cache-RAM hierarchy implies optimality in a multilevel cache hierarchy

SPECIFICATION MODEL $\mathrm{M}(n)$



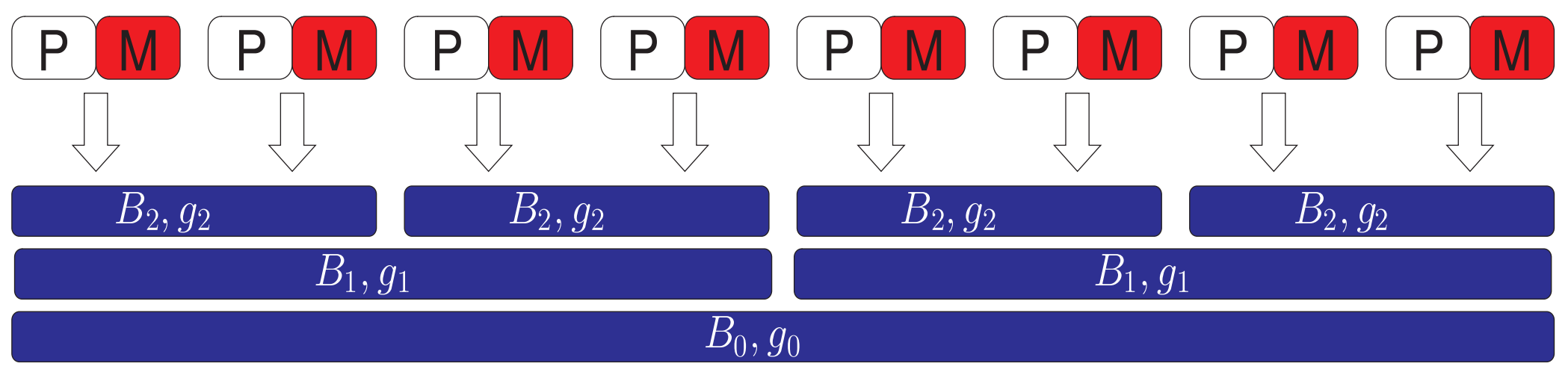
- n Processing Elements (PEs)
- An algorithm *A* is a sequence of supersteps
- In a superstep each PE can:
  - perform operations on local data
  - send/receive messages to/from PEs
- The $\mathrm{M}(n)$ is a BSP [Valiant, 90] with no banwidth and latency parameters

DEFINITION

A network-oblivious algorithm for a problem 0 is an $\mathrm{M}(n)$-algorithm where n is a function of the input size

- Algorithm specification is:
  - independent of network topology
  - independent of the actual number of processors

EXECUTION MODEL D-BSP$(p,\boldsymbol{g},\boldsymbol{B})$



- p Processing Elements (PEs)
- Recursive decomposition into i-clusters of $p/2^i$ PEs, 0 ¡ $i < \log p$
- An algorithm A is a sequence of labeled supersteps
- In an i-superstep, a PE can:
  - Perform operations on local data
  - Send/receive messages to/from PEs in its i-cluster
- A D-BSP$(p, \boldsymbol{g}, \boldsymbol{B})$ is an $\mathrm{M}(p, \cdot)$ with a hierarchical network structure
  - $\boldsymbol{g}=(g_0,$ ¿ $, g_{\log p - 1})$, $\boldsymbol{B}=(B_0,$ ¿ $, B_{\log p - 1})$
  - g_i ⇒ reciprocal of the bandwidth in an i-cluster
  - B_i ⇒ block size for communications in an i-cluster
- Communication time of an i-superstep: $h^s(p,B_i)g_i$
- Communication time of A: $\sum_{\forall s \text{ of } A} h^s(p, B_i)g_i$
- An $\mathrm{M}(p, \cdot)$-algorithm can be naturally translated in a D-BSP$(p, \boldsymbol{g}, \boldsymbol{B})$-algorithm by suitably labeling each superstep

FFT AND SORTING

- Fast Fourier Transform of n elements (FFT(n)):
  - The algorithm exploits the recursive decomposition of the FFT(n) dag into $\sqrt{n}$ FFT$(\sqrt{n})$ subdags
  - Optimal algorithm on $\mathrm{M}(p,B)$ for p ñ n and $B \leq \sqrt{n/p}$
- Sorting of n keys:
  - The algorithm is based on a recursive *Columnsort*
  - Optimal algorithm on $\mathrm{M}(p,B)$ for p ñ $n^{1-ε}$, ∀ constant ε and $B \leq \sqrt{n/p}$

IMPOSSIBILITY RESULT

THEOREM

There is no network-oblivious matrix transposition algorithm such that ∀ p ñ n and B ñ n/p, its execution on $\mathrm{M}(p, B)$ achieves optimal communication complexity