Network-Oblivious Algorithms

Gianfranco Bilardi, Andrea Pietracaprina,
Geppino Pucci, Michele Scquizzato and Francesco Silvestri

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

Overview

- Background
- Summary of results
- Framework for network-oblivious algorithms
- Case studies:
  - Some network-oblivious optimal algorithms
  - An impossibility result
- Conclusions

Background: communication 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

Background: models of parallel computation

PRAM

BSP, QSM
Decomposable-BSP
LogP …

Fat-Tree,
Pruned Butterfly,
Mesh

Parallel slackness | **Bandwidth-latency** | **Universality**

← + Portability -

\- Effectiveness + →

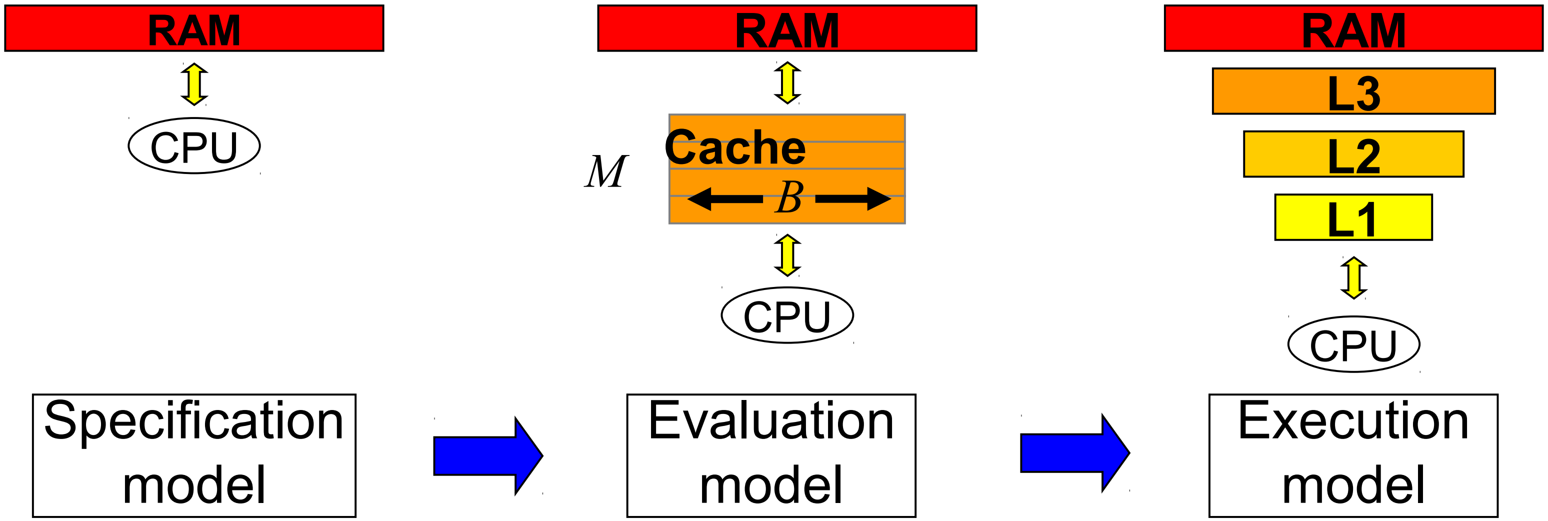
Background: obliviousness

- 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?

Background: cache-oblivious framework

- Cache-oblivious framework [Frigo *et al.*, 99]



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

Summary of 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, sorting)
- Impossibility result for matrix transposition

Framework for network-oblivious algorithms

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

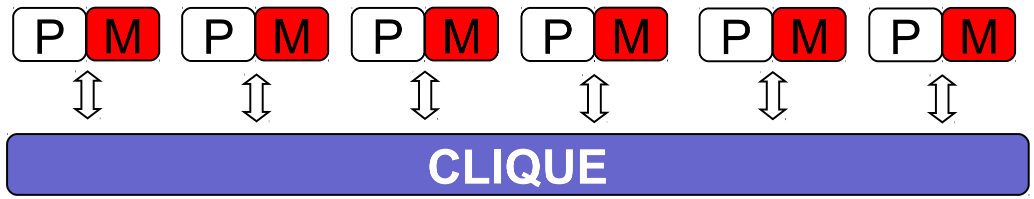
Evaluation model: introduces number of PEs p and communication block size B

Execution model: introduces hierarchical network structure

Main technical result: for a wide class of network-oblivious algorithms, optimality in the evaluation model implies optimality in the execution model

Specification model

- Specification model $M(n)$:



  - n Processing Elements (PEs)
  - An algorithm A is a sequence of supersteps, separated by barriers
  - In a superstep, each PE can:
    - Perform operations on local data
    - Send/receive messages to/from PEs
  - Note that $M(n)$ is a BSP [Valiant, 90] with maximum (reciprocal) bandwidth (g=1) and no latency (l=0)

Network-oblivious algorithm

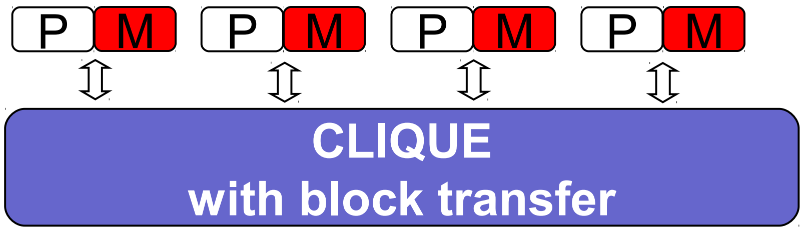
Definition: A network-oblivious algorithm for a problem Π is an $\mathrm{M}(n)$-algorithm, where n (parallelism) is a function of the input size

Remarks: algorithm specification is

- independent of network topology
- independent of actual number of processors

Evaluation model

- Evaluation model M(*p*, *B*):



- M(*p*, *B*) is an 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 any PE in a superstep *s*
  - Communication complexity of A:

$$\Sigma_{s \in A}\, h^s(p, B)$$

Evaluation model (cont'd)

Execution of an $M(n)$-algorithm on an $M(p, B)$:

- Every $M(p, B)$-PE simulates a segment of n/p consecutive $M(n)$-PEs
- Communications between $M(n)$-PEs in the same segment $\Rightarrow$ local computations in $M(p, B)$.

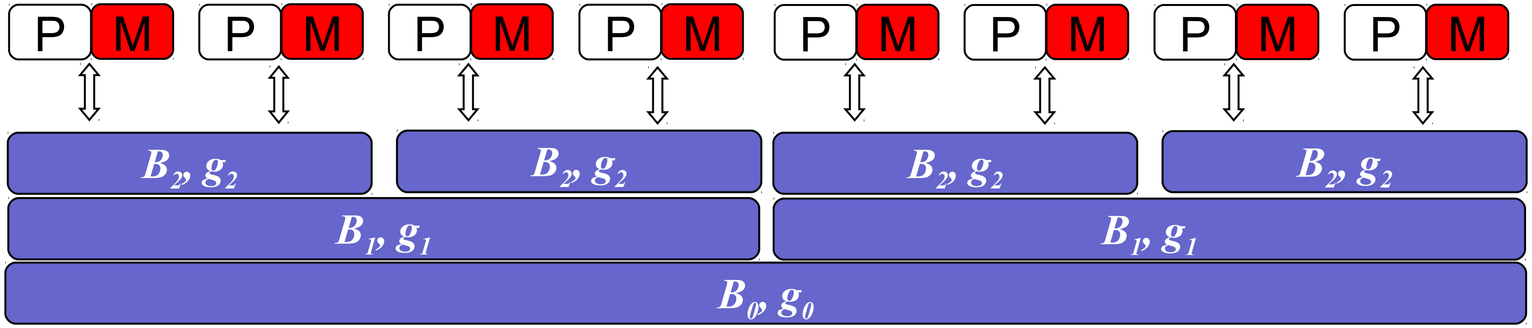
Evaluation model (cont'd)

Optimal network oblivious algorithm A for Π

$\forall$instance of size n, $\forall p \leq n$, and $\forall B \geq 1$,
the execution of A on an M(p, B) attains
asymptotically minimum communication complexity
among all M(p, B)-algorithms for Π

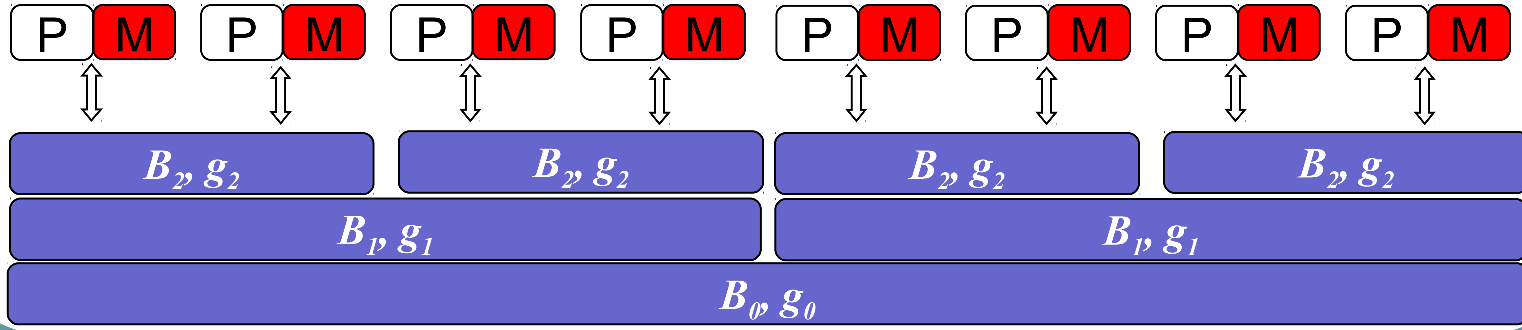
Execution model

- Execution model D-BSP(p, g, B) [De la Torre *et al.*, 96]:
  - p Processing Elements (PEs)
  - Recursive decomposition into i-clusters of $p/2^i$ PEs, $0 \leq i < \log p$
  - 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



Execution model (cont’d)

- Execution model D-BSP(p, $\boldsymbol{g}$, $\boldsymbol{B}$) [De la Torre *et al.*, 96]:
  - an M(p, ·) with a hierarchical network structure
  - p Processing Elements (PEs)
  - Recursive decomposition into i-clusters of $p/2^i$ PEs, $0 \le i < \log p$
  - $\boldsymbol{g}=(g_0, \ldots, g_{\log p -1})$, $\boldsymbol{B}=(B_0, \ldots, B_{\log p -1})$:
  - g_i : bandwidth parameter in an i-cluster
  - B_i : block size for communications in an i-cluster



Execution model (cont'd)

- An algorithm A is a sequence of labeled supersteps separated by barriers
- In an i-superstep, a PE can:
  - Perform operations on local data
  - Send/receive messages to/from PEs in its i-cluster
- Block degree of i-superstep s: $h^s(p, B_i)$
- Communication time of i-superstep s: $h^s(p, B_i) \cdot g_i$
- Communication time of A: sum of comm. times of constituent supersteps

Remark: an M(p, ·)-algorithm can be naturally translated into a D-BSP(p, $\mathbf{g}$, $\mathbf{B}$)-algorithm by suitably labeling each superstep

Optimality result

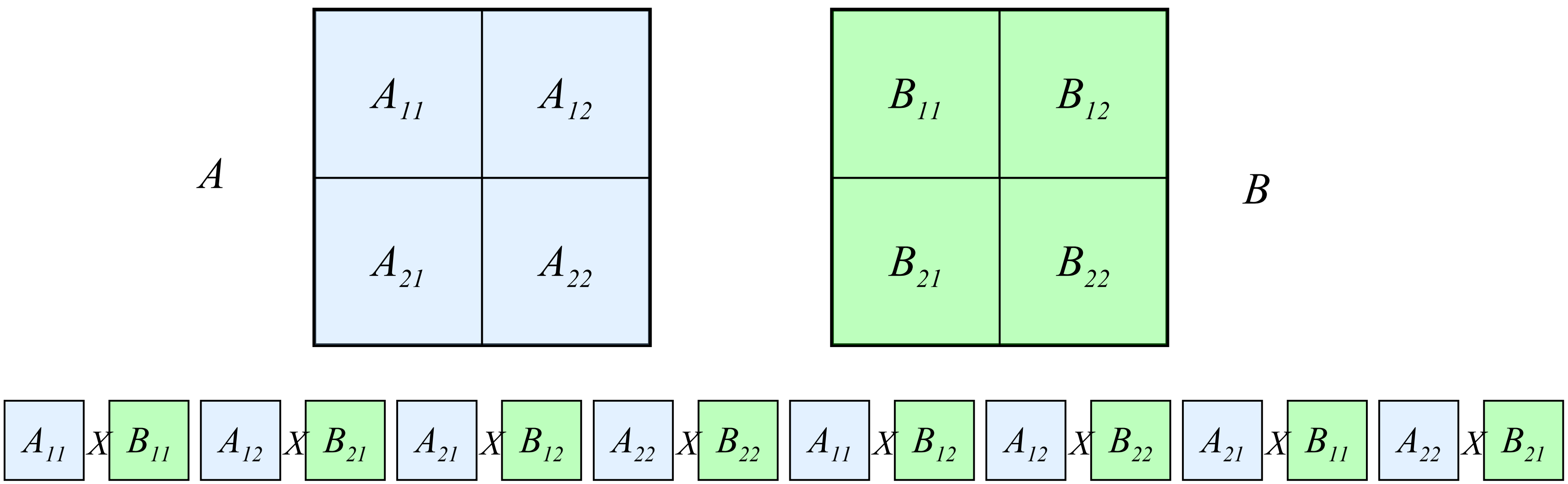
Theorem: an optimal network-oblivious algorithm A exhibits an asymptotically optimal communication time when executed on any D-BSP(p, $\boldsymbol{g}$, $\boldsymbol{B}$) with $p \leq n$ under the following conditions:

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

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

Matrix Multiplication

- **Problem:** multiplying two $\sqrt{n} \times \sqrt{n}$ matrices, A and B.
- Initial row-major distribution of A and B among the n PEs



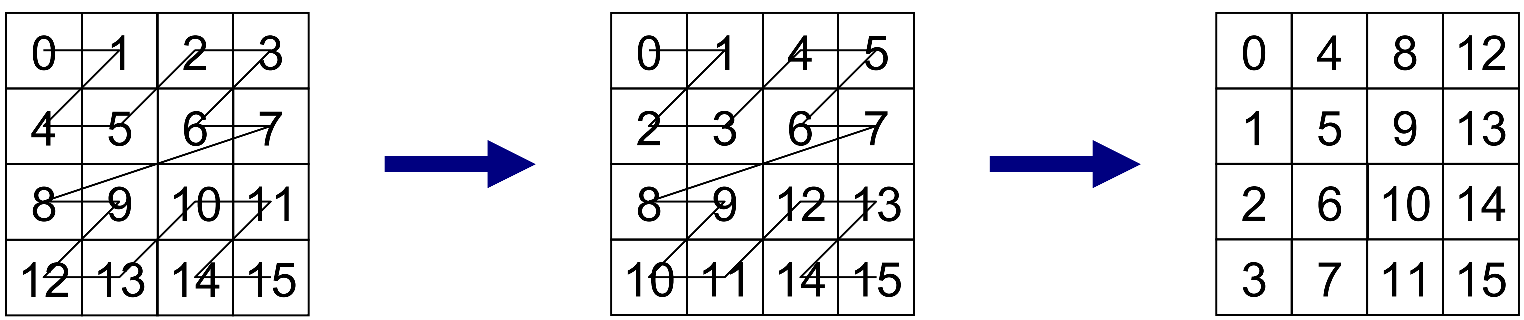
- 8 subproblems
- Solve each subproblem in parallel within a distinct segment of $n/8$ PEs

Matrix Multiplication (cont'd)

- When executed on an $\mathrm{M}(p, B)$:
  - Optimal communication complexity $O(n/p^{2/3})$
- By the previous theorem, this algorithm is also optimal in a $\text{D-BSP}(p, \mathbf{g}, \boldsymbol{B})$, as long as $B_i \leq n/p$
- The algorithm requires $\Theta(n/p^{1/3})$ memory blow-up, unavoidable if minimal communication is sought
- A different recursive strategy yields
  - Constant memory blow-up
  - Communication complexity $O(n/p^{1/2})$: optimal under constant memory blow-up constraint [Irony *et al.*, 04]

Matrix Transposition

- The naïve one-step algorithm doesn't exploit the block feature.
- Two-step algorithm based on Z-Morton ordering.



Transform a Z-ordering in a row-major ordering

Transform a Z-ordering in a column-major ordering

- If $B \le \sqrt{\frac{n}{p}}$, optimal communication complexity $\Theta\left(\frac{n}{pB}\right)$

Matrix Transposition: impossibility result

- Constraint $B \leq \sqrt{\frac{n}{p}}$ reminiscent of the tall-cache assumption in [Frigo *et al.*, 99] (necessary to achieve cache-oblivious optimality for the matrix transposition problem [Silvestri, 06]).

- Can we remove the assumption on the block size? No!

Theorem: There is no network-oblivious matrix transposition algorithm such that for each $p \leq n$ and $B \leq n/p$, its execution on $\mathrm{M}(p, B)$ achieves optimal communication complexity $\Omega(n/(pB)(1+f(n,p,B)))$

FFT and Sorting

- Fast Fourier Transform of n elements ($\mathrm{FFT}(n)$):
  - Network-oblivious algorithm exploits the recursive decomposition of the $\mathrm{FFT}(n)$ dag into $\sqrt{n}$ $\mathrm{FFT}(\sqrt{n})$ subdags
  - Optimal algorithm for $p \leq n$ and $B \leq \sqrt{\frac{n}{p}}$
- Sorting of n keys:
  - Network-oblivious algorithm based on a recursive version of *Columnsort*.
  - Optimal algorithm for $p \leq n^{1-\varepsilon}$ $\forall$ constant ε and $B \leq \sqrt{\frac{n}{p}}$

Conclusions

Our contribution:

- Notion of network-oblivious algorithms:
  - Independent of actual number of processors and of communication block size
  - Independent of interconnection network topology.
- Framework for design, analysis, and execution of network-oblivious algorithms.
- Optimality: general theorem and specific results for prominent case studies

Conclusions (cont'd)

Further research:

- Network-oblivious algorithms for other key problems
- Broaden the spectrum of machines for which network-oblivious optimality translates into optimal time
- Lower bound techniques to limit the level of optimality of network-oblivious algorithms

QUESTION: are p and B the *right* parameters to be oblivious to?