Efficient Parallel Out-of-core Matrix Transposition

Sriram Krishnamoorthy
Gerald Baumgartner
Daniel Cociorva
Chi-Chung Lam
P Sadayappan

Department of Computer and Information Science
The Ohio State University

Outline

- The Problem
- Motivation
- Related Work
- I/O characteristics analysis
- Out-of-core Matrix Transposition
- Results
- Conclusion

The Problem

- To transpose an *NxN* ($N=2^n$) array stored in disk in row major order.
- Main memory: *M* ($=2^m$) elements, $M<N^2$, *M=O(N)*.
- Element-wise approach – expensive.
- Approach – Read some portion of data, permute and write to disk.
- A number of passes through the whole array.

Motivation

- Key operation in many scientific computations. Eg: multi-dimensional FFT.
- Tensor Contraction Engine (TCE) - Automatic synthesis system for quantum chemistry computations.
- TCE-synthesized programs might need to transform output from another tool (eg. NWChem) to match its access pattern.

Related Work

- Block transposition – Single pass, out-of-place. $O(N^{3/2})$ I/O operations.
- Eklundh – Fundamental algorithm, in-place. *logN* passes and *O(NlogN)* I/O operations.
- Kaushik et al. - Out-of-place. Reduces the number of read operations.
- Suh and Prasanna – Out-of-place. Achieves minimum number of I/O operations.

Related Work (Contd.)

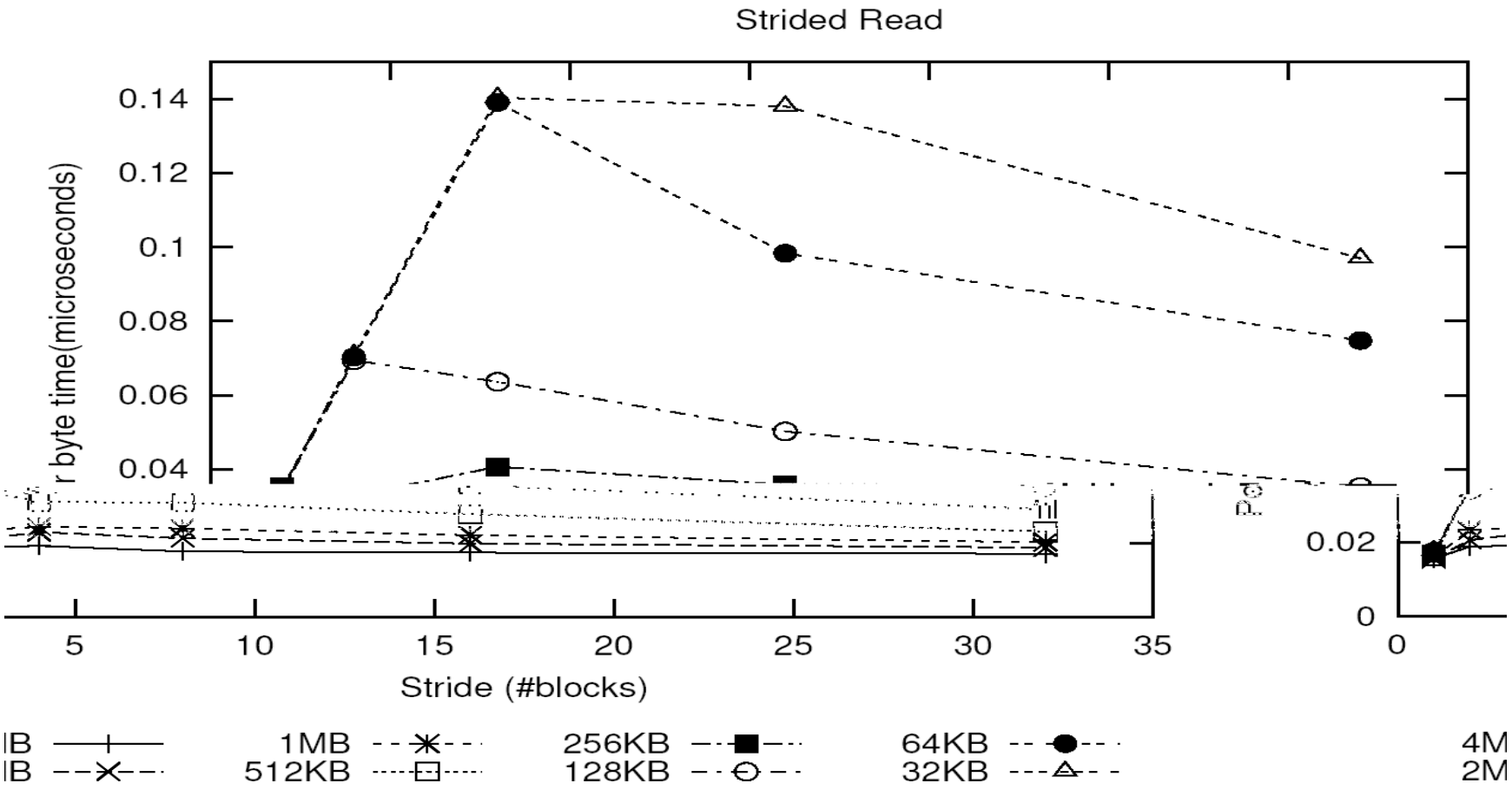
- Focus of all algorithms - number of I/O operations.
- Lesser number I/O operations → larger I/O size.
- I/O size proportional to matrix dimensions
- Total execution time and system I/O characteristics not considered.

I/O characteristics analysis

- Out-of-matrix transposition involves strided I/O.
- Measured the per-byte read and write times for different block sizes and different strides of access.
- Analyzed the effect of stride on I/O time.

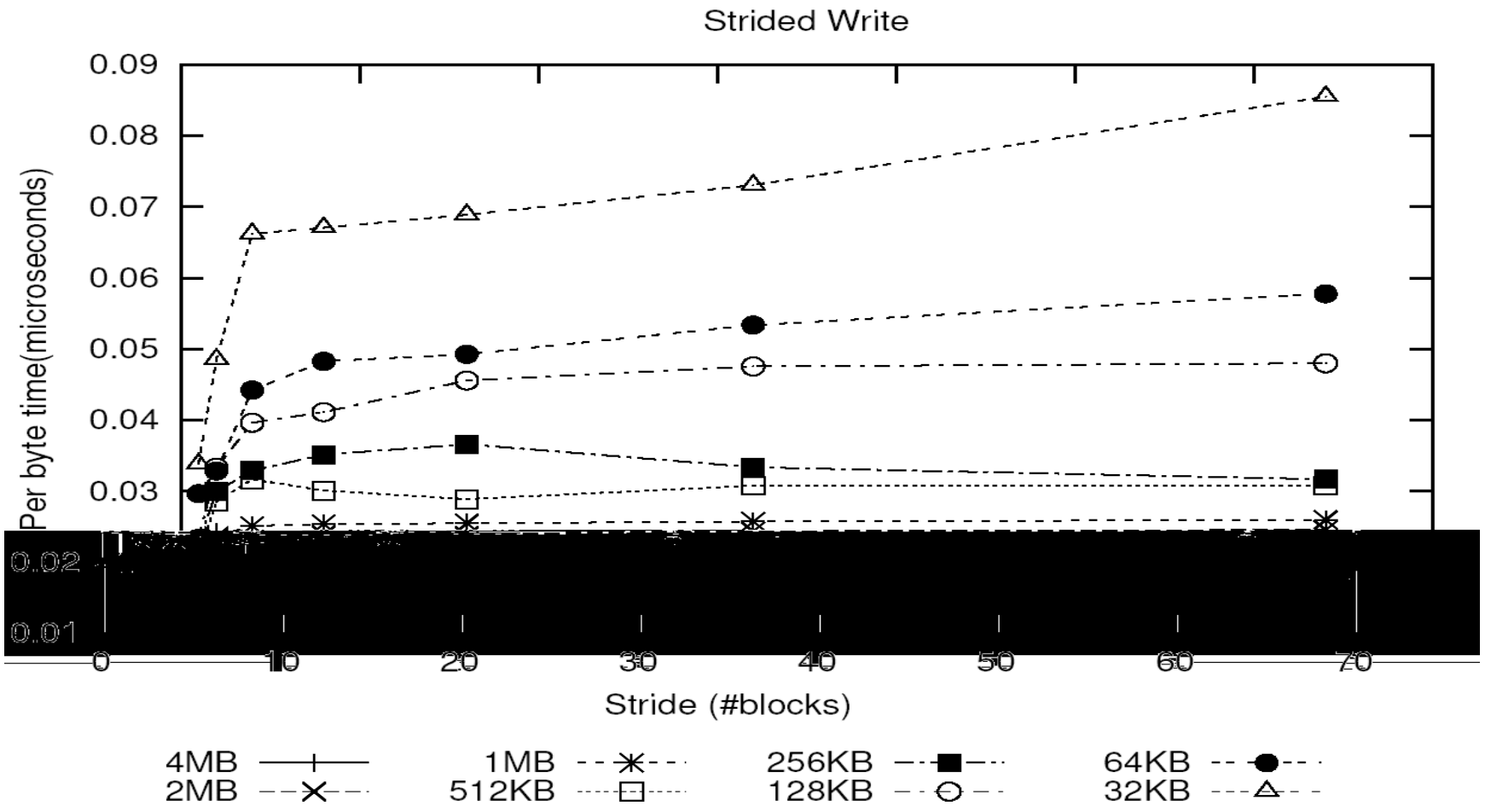
System	Configuration			
	Processor	Memory	OS	Compiler
IA32 Cluster	Dual AMD Athlon MP (1.533GHz)	2GB	linux 2.4.20	pgcc 4.0-2
Itanium2 Cluster	Dual Itanium2 (900MHz)	4GB	linux 2.4.18	gcc 2.96

I/O Characteristics - Read Times



Itanium2 Cluster

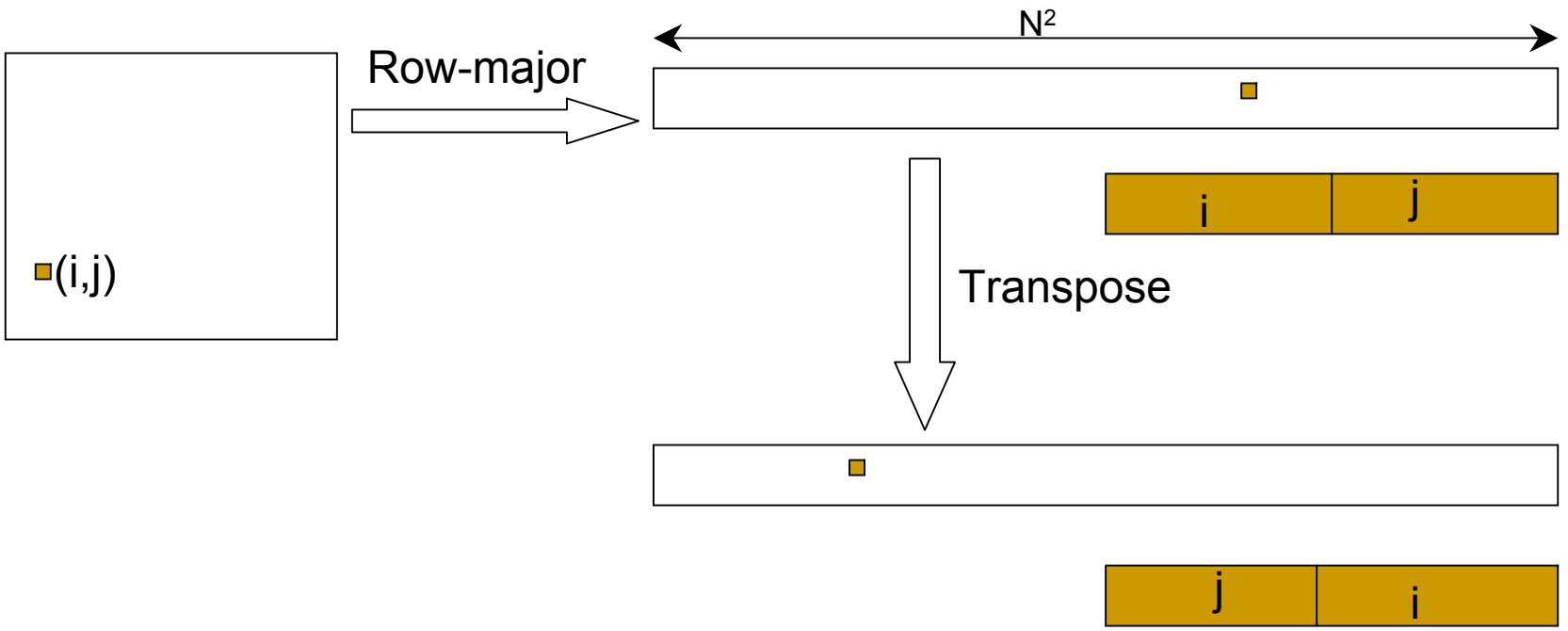
I/O Characteristics - Write Times



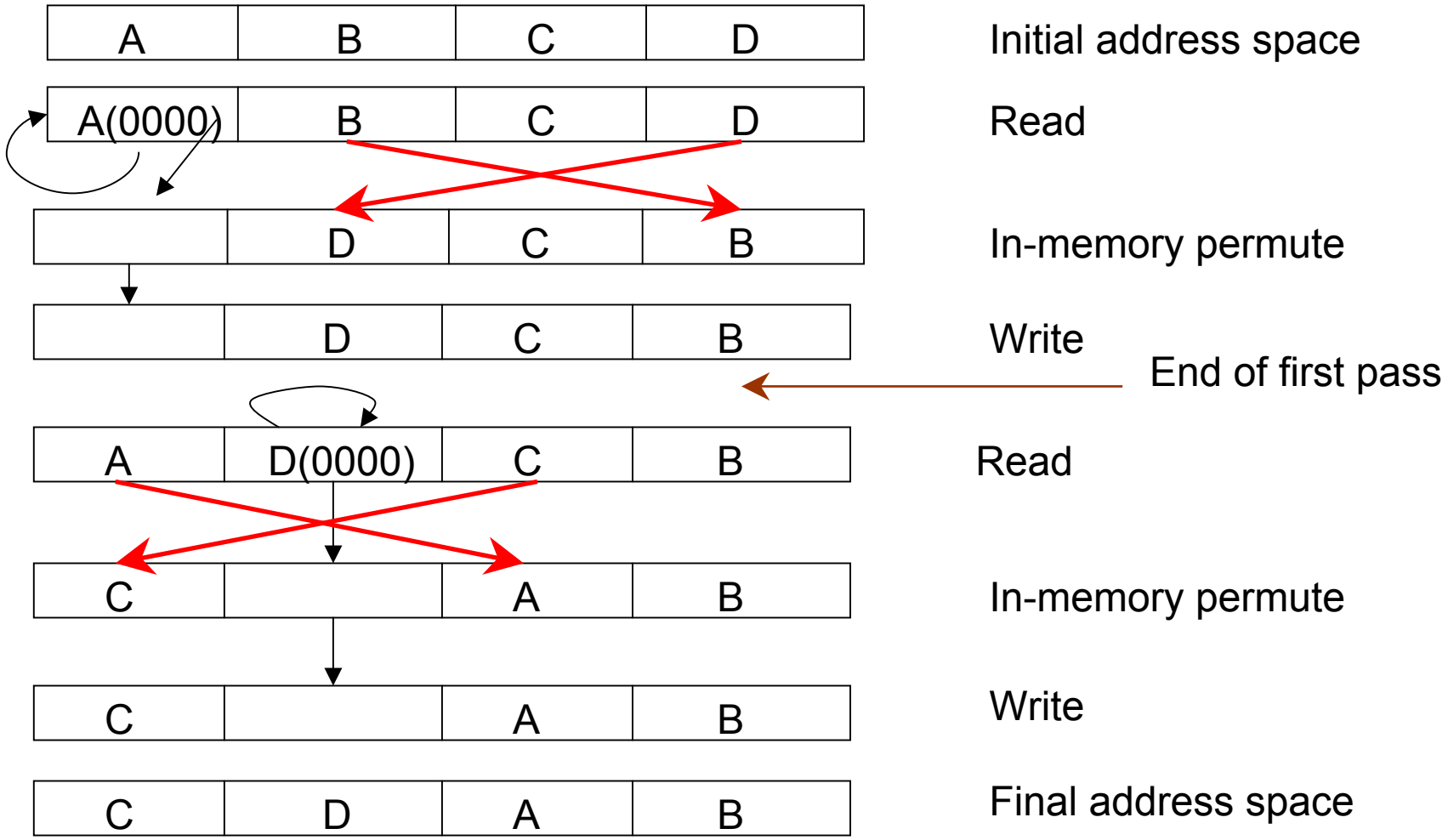
Itanium2 Cluster

Out-of-core Matrix Transposition - Intuitive Representation

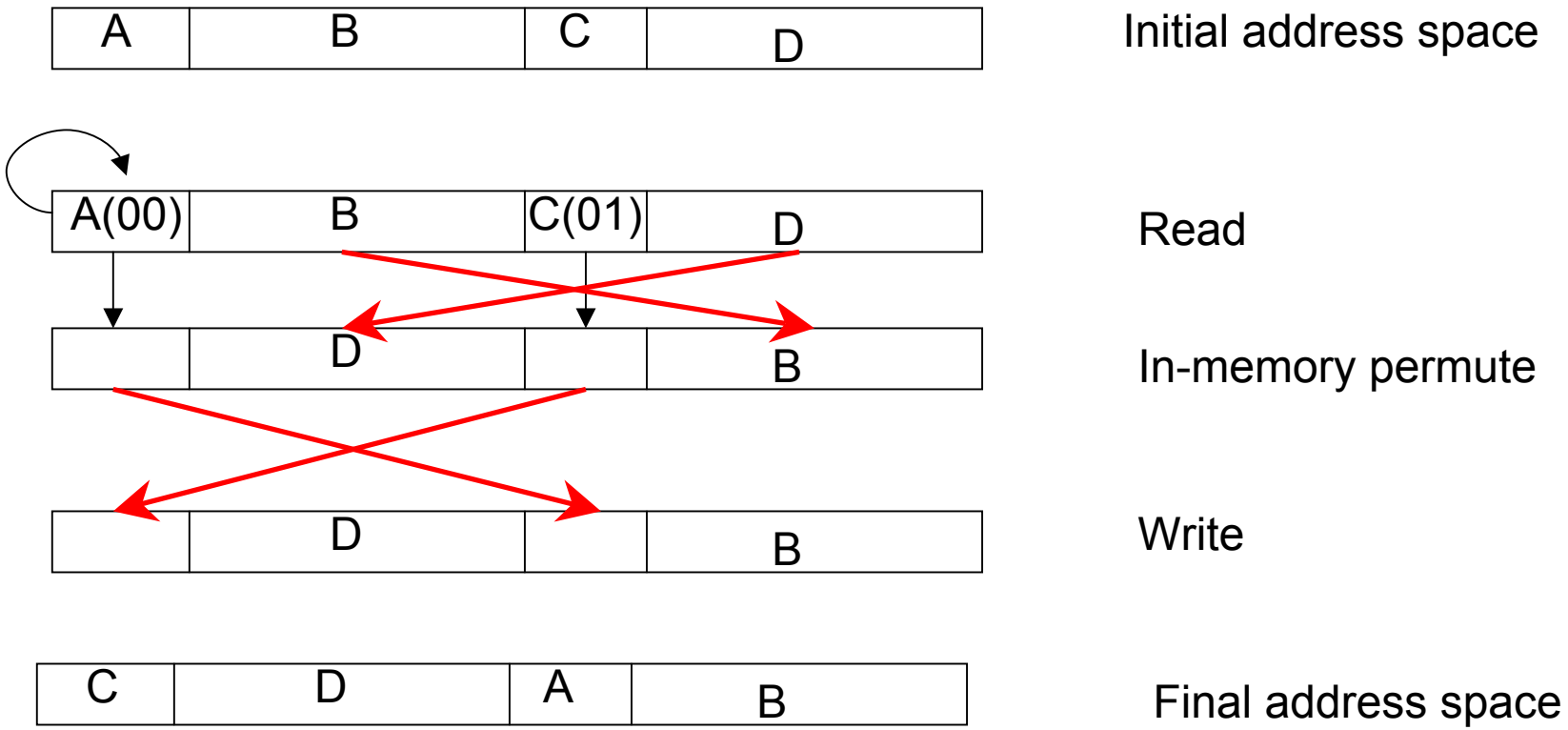
- Corresponds to transformation of the address space of elements.



Eklundh's Algorithm



Our Algorithm



Out-of-core Matrix Transposition - Formulation

- Tensor Product Notation. $A \oplus B = \begin{pmatrix} A0 \\ 0B \end{pmatrix}$
- Involves three types of transformations:
  - Read
  - In-memory permutation
  - Write
- All existing transposition algorithms formulated using this notation.
- A generic transposition algorithm derived.

Out-of-core Matrix Transposition – Formulation (Contd.)

$$T = \prod_{i=t-1}^{i=0} W_i P_i R_i$$

$$R_i = A_{2*n-r} \oplus I_r \quad r \leq m$$

$$P_i = I_{2*n-m} \oplus B_m$$

$$W_i = C_{2*n-w} \oplus I_w \quad w \leq m$$

(1) **for** $i = 0$ **to** $t - 1$

(2) **for** $j = 0$ **to** $2^{2*n-m} - 1$

(3) Read M elements at address $R_i^{-1}(j)$ /*Might involve multiple reads*/

(4) Permute data in memory according to P_i.

(5) Write M elements at address $W_i(j)$ /*Might involve multiple writes*/

Parallel Out-of-core Matrix Transposition

- Formulation of multi-processor array redistribution
- Parallel algorithm - composing sequential transposition and array redistribution.

Intuition for Parallel Algorithm

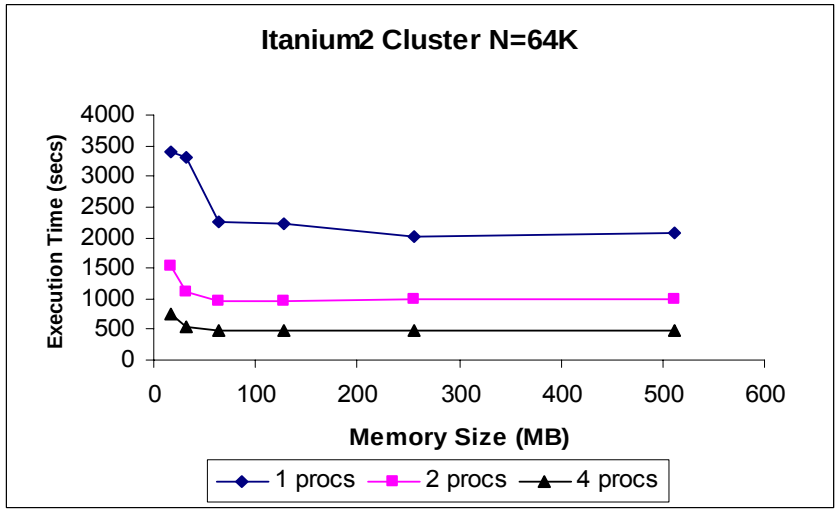
- Distribution of the array among processors ≡ certain address bits form processor identifier.
- Processor identifier - transformed only by communication.
- Remaining bits - transformed as in the sequential algorithm.
- The first pass of parallel algorithm combines first pass of sequential algorithm and array redistribution.

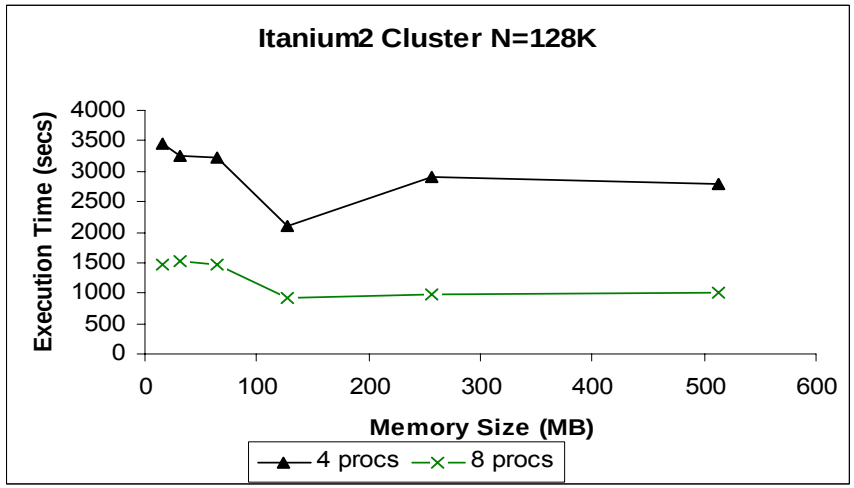
Results – Sequential Algorithms

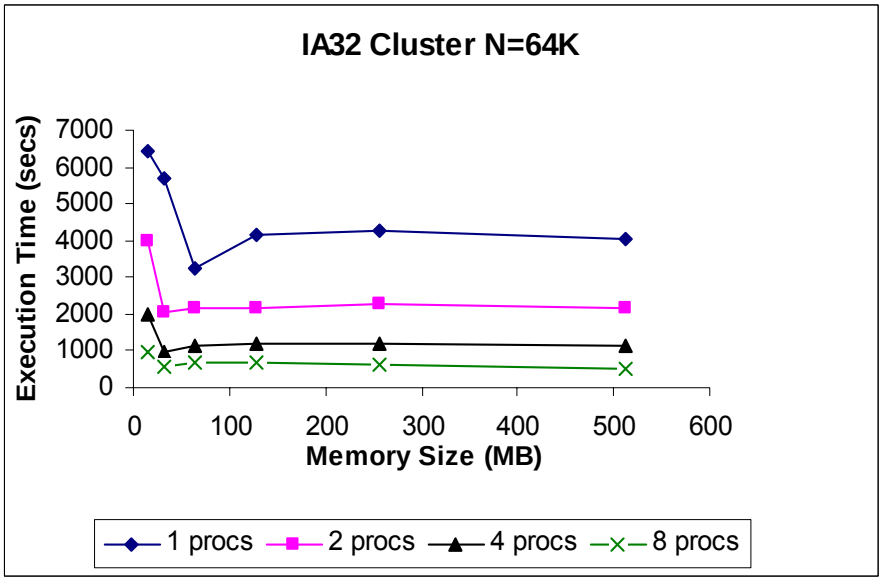
Algorithm	Memory Size					
	32MB	64MB	128MB	256MB	512MB	1GB
Our	2897	2378	2529	2298	2237	2007
Suh's	14437	10811	9756	11464	2623	2516
Kaushik's	7750	4423	3944	4138	4051	4057

Itanium2 Cluster N=64K (Array Size=16GB)

Results – Parallel Algorithm







Conclusion

- Out-of-core matrix transposition - viewed as a index transformation of the address of matrix elements.
- Existing algorithms formulated and a generic algorithm derived.
- Parallel algorithm derived - attempts to minimize overall execution time.
- Experimental results show improved performance and scalability of our algorithm.