Strategies for Parallelizing the Solution of Rational Matrix Equations

J.M. Badía[1] P. Benner[2] M. Castillo[1] H. Fassbender[3]
R. Mayo[1] E.S. Quintana-Ortí[1] G. Quintana-Ortí[1]

[1]Depto. Ingeniería y Ciencia de Computadores
Univ. Jaume I, Spain.

[2]Fakultät für Mathematik, Technische Universität Chemnitz, Germany;

[3]Technische Universität Braunschweig, Institut *Computational Mathematics*, Germany;

Parallel Computing 2007 (ParCo2007)

Main goal

Compare different strategies for parallelizing the solution of numerical problems

- Levels of granularity: Application, linear algebra operations, numerical kernel.
- Programming models: message-passing, shared memory.
- Difficulty of programming. Transparency to the programmer.
- Sequential and parallel libraries and tools: ScaLAPACK, LAPACK, BLAS, MPI.

Outline

1. Rational Matrix Equation

2. Sequential algorithm

3. Parallel algorithms
    - Multithreading algorithm (MT)
    - Message-passing multiprocess algorithm (MP)
    - Message-passing two processes algorithm (2P)
    - Two-processes hybrid algorithm (HB2P)
    - Multiprocess hybrid algorithm (HBMP)

4. Conclusions

Outline

Rational Matrix Equation (RME)

$$X = Q + LX^{-1}L^T$$

$Q \in \mathbb{R}^{n\times n}$ is symmetric positive definite, $L \in \mathbb{R}^{n\times n}$ is nonsingular

$X_+ \in \mathbb{R}^{n\times n}$: positive definite symmetric solution.

- **Applications**: Analysis of stationary Gaussian reciprocal processes.
- **Examples**:
  - Steady-state distribution of temperature along a heated beam subjected to random loads.
  - Ship surveillance problem. Prediction of its trajectory.

Structure-preserving Doubling Algorithm (SDA)

- Based on the solution of a Discrete-time Algebraic Riccati equation.
- Iterative method.
  - Quadratic convergence rate.
  - Fair numerical stability.
  - Cubic computational cost on n.
- It can be efficiently parallelized.

Outline

Sequential algorithm

$L_0 = \widehat{L}\,, \quad Q_0 = \widehat{Q} + \widehat{P}\,, \quad P_0 = 0$

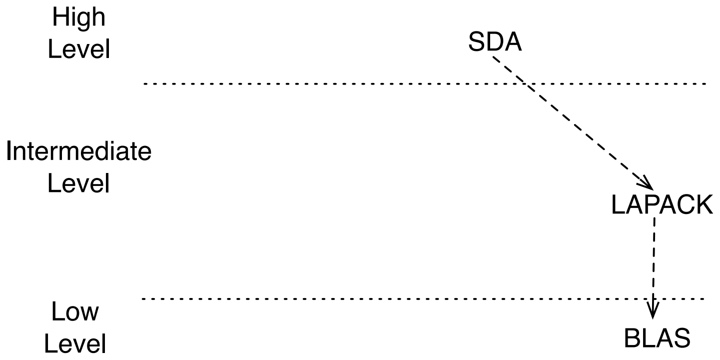
for $i = 0, 1, 2, \ldots$

$Q_i - P_i = C_i^T C_i$	DPOTRF	$(n^3/3)$
$A_C = L_i^T C_i^{-1}$	DTRSM	(n^3)
$C_A = C_i^{-T} L_i^T$	DTRSM	(n^3)
$Q_{i+1} = Q_i - C_A^T C_A$	DGEMM	(n^3)
$P_{i+1} = P_i + A_C A_C^T$	DGEMM	(n^3)
$L_{i+1} = A_C C_A$	DGEMM	$(2n^3)$

until convergence

Total cost $\approx 6.3n^3$ *flops*

Levels of granularity



Outline

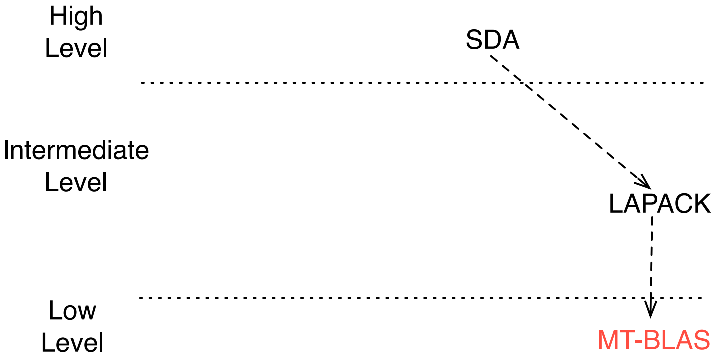
Parallelizing Strategies

Exploit three levels of granularity:

- **High level**: Overlap steps of the SDA Algorithm.
- **Medium level**: Linear Algebra routines: Linear system solving, Cholesky factorization, etc.
- **Low level**: Numerical kernels: matrix products, matrix-vector products, etc.

Multithreading algorithm (MT)

Lazy programmer



Hardware environment: set

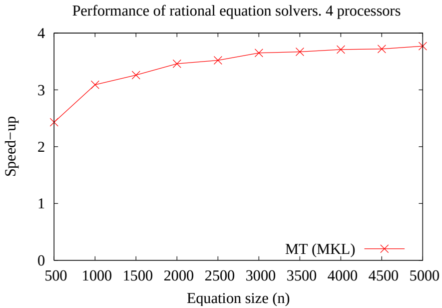
SGI Altix 350

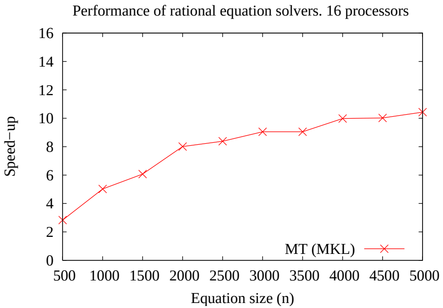
- DSM multiprocessor.
- 16 Itanium2 (1.5GHz.) processors (6 MB L3 cache).
- 32 GBytes of DSM.
- cc-NUMA architecture.
  - SGI NUMAlink interconnect.
  - Ring topology.
  - 8 dual-processor nodes.

Software environment

- Programming language: Fortran.
- Libraries and tools:
  - SGI ScaLAPACK for Distributed Shared Memory: SDSM.
  - SGI implementation of MPI.
  - Two multithreaded versions of LAPACK+BLAS:
    - SGI Scientific Computing Software Library: SCSL.
    - Intel Math Kernel Library: MKL.

Experimental results



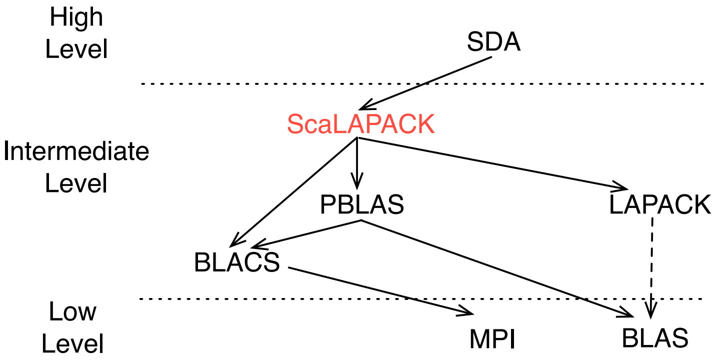


Comments

- Can only be used in shared memory multiprocessors.
- Low performance when many processors are used. Possible causes:
  - Fine-grain parallelism.
  - Unique copy of the data in memory. Cost of the data transference to maintain the coherence of the data.

Could we improve the performance with a little more work?

Message-passing multiprocess algorithm (MP)



Multiprocess algorithm

Data distribution

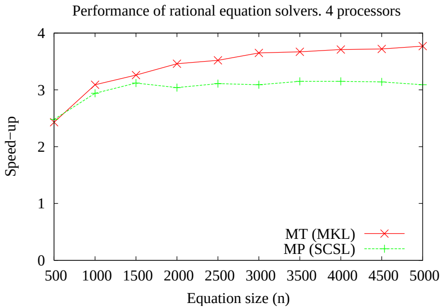
$L_0 = \widehat{L}\,,\quad Q_0 = \widehat{Q} + \widehat{P}\,,\quad P_0 = 0$

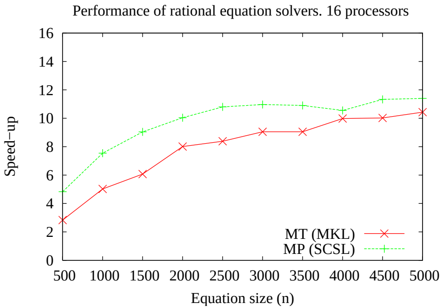
for $i = 0, 1, 2, \ldots$

$Q_i - P_i = C_i^T C_i$	PDPOTRF
$A_C = L_i^T C_i^{-1}$	PDTRSM
$C_A = C_i^{-T} L_i^T$	PDTRSM
$Q_{i+1} = Q_i - C_A^T C_A$	PDGEMM
$P_{i+1} = P_i + A_C A_C^T$	PDGEMM
$L_{i+1} = A_C C_A$	PDGEMM

until convergence

Experimental results



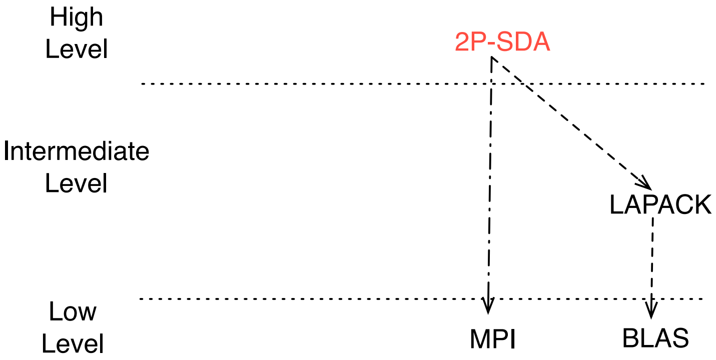


Comments

- Can be used on shared and distributed memory multiprocessors.
- Worse performance than MT algorithm with few processors.
- Improves the performance of MT algorithm with more processors.
  - Coarser-grain parallelism than MT algorithm.
  - Distributed data among "local" memories. Larger locality to access data ⇒ Less data transference cost.

Could we improve the performance with a little more work? Again

Message-passing two processes algorithm (2P)



Two processes algorithm

$L_0 = \widehat{L}\,, \quad Q_0 = \widehat{Q} + \widehat{P}\,, \quad P_0 = 0$

for $i = 0, 1, 2, \ldots$

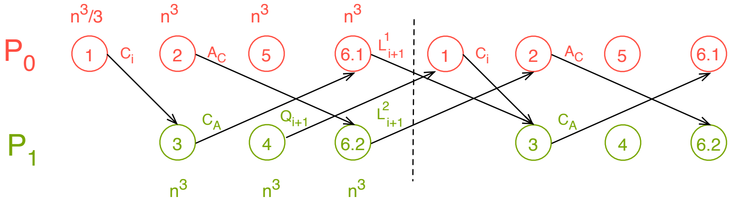
1. $Q_i - P_i = C_i^T C_i$	DPOTRF $(n^3/3)$
2. $A_C = L_i^T C_i^{-1}$	DTRSM (n^3)
3. $C_A = C_i^{-T} L_i^T$	DTRSM (n^3)
4. $Q_{i+1} = Q_i - C_A^T C_A$	DGEMM (n^3)
5. $P_{i+1} = P_i + A_C A_C^T$	DGEMM (n^3)
6.1. $L_{i+1} = A_C C_A$	DGEMM (n^3)
6.2. $L_{i+1} = A_C C_A$	DGEMM (n^3)

until convergence

P_0 *cost* $\approx 3.3n^3$ *flops*

P_1 *cost* $\approx 3n^3$ *flops*

Dependencies and Communications

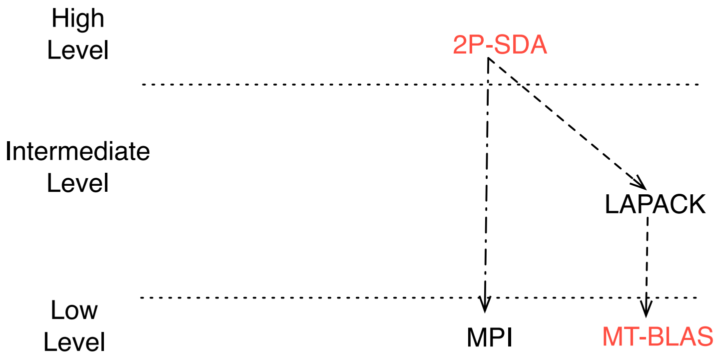


Comments

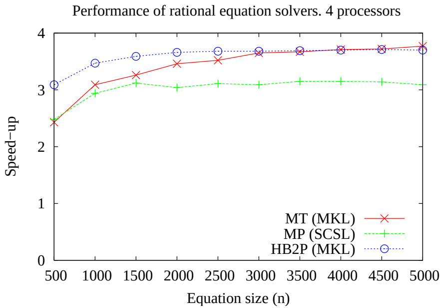
- Can be used on shared and distributed memory multiprocessors.
- Exploits the highest level parallelism.
- Uses only two MPI processes.

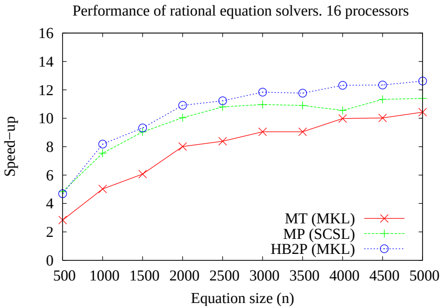
Could we take profit of more than two processors?

Two-processes hybrid algorithm (HB2P)

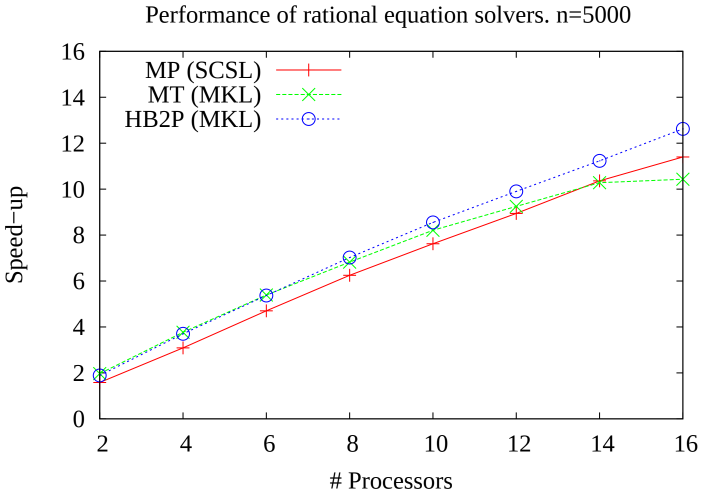


Experimental results





Experimental results

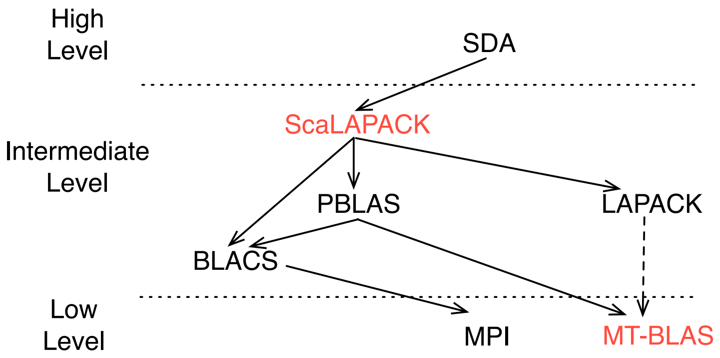


Comments

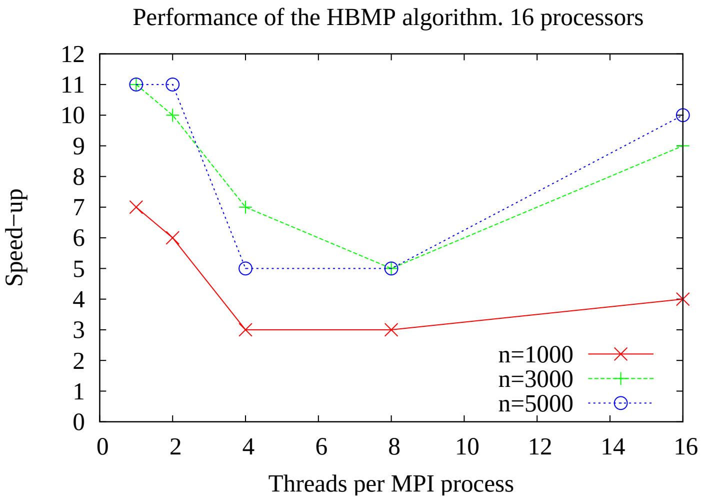
- Can be used on shared and distributed memory multiprocessors.
- Similar performance than MT algorithm with few processors. Better with small problems.
- Improves MP and MT algorithms with 16 processors. Possible causes.
  - Exploits two different levels of parallelism: processes and threads.
  - Increases the locality of MT algorithm $\Rightarrow$ Decreases data transference cost.

Could we take combine two different levels of parallelism? Last try.

Multiprocess hybrid algorithm (HBMP)



Experimental results



Comments

- The exact behaviour depends on the size of the problem.
- The hybrid algorithm (HBMP) is always improved by one of the "pure" algorithms: MT or MP algorithm.

Outline

Conclusions

- We have applied five different parallel strategies to solve the Rational Matrix Equation.
- The algorithms exploit and combine different levels of parallelism and apply different programming models.
- Studying the dependencies of the algorithm and applying parallelism to the highest level pays off.
- The best results are obtained with a hybrid algorithm that combines processes with threads.
- The speed-up scales almost linearly with the number of processors with large problems.
- The algorithms use standard libraries ensuring the portability of the codes.