Parallel Hierarchical Matrices

Ronald Kriemann

joint work with L. Grasedyck and S. Le Borne

Max-Planck-Institute for Mathematics in the Sciences Leipzig

ALA2006

Düsseldorf July 24–27, 2006

Outline

Outline

Introduction

Problem

Fast solution of

$$Ax = b$$

with $A \in \mathbb{C}^{I \times I}$ being a matrix defined by a PDE or integral operator in $\Omega \subset \mathbb{R}^d$.

Problem

Fast solution of

$$Ax = b$$

with $A \in \mathbb{C}^{I \times I}$ being a matrix defined by a PDE or integral operator in $\Omega \subset \mathbb{R}^d$.

Solution

1. Represent A as an $\mathcal{H}$-matrix,
2. Factorise $A = LU$ using LU decomposition,
3. Solve equation

Introduction

Problem

Fast solution of

$$Ax = b$$

with $A \in \mathbb{C}^{I \times I}$ being a matrix defined by a PDE or integral operator in $\Omega \subset \mathbb{R}^d$.

Solution

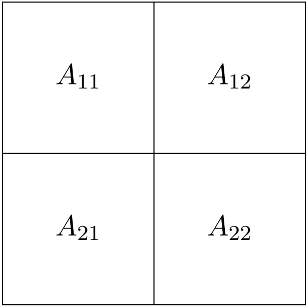
1. Represent A as an $\mathcal{H}$-matrix,
2. Factorise $A = LU$ using LU decomposition,
3. Solve equation

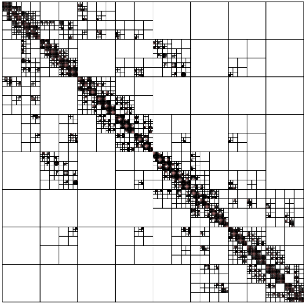
Condition: do each step on a parallel machine with p processors

Outline

Bisection

Block Structure





Algorithm

1. factorise $A_{11} = L_{11}U_{11}$, (Recursion)
2. solve $A_{12} = L_{11}U_{12}$ and $A_{21} = L_{21}U_{11}$, (involves matrix mult.)
3. update $A_{22} = A_{22} - L_{21} \cdot U_{12}$, (matrix mult.)
4. factorise $A_{22} = L_{22}L_{22}$ (Recursion)

Bisection

Parallelisation on Shared Memory

Parallel Matrix Multiplication works with optimal Speedup

- order multiplications per block
- load-balancing on dense or low rank blocks

For LU factorisation: replace each sequential matrix multiplication with parallel version.

Bisection

Parallelisation on Shared Memory

Parallel Matrix Multiplication works with optimal Speedup

- order multiplications per block
- load-balancing on dense or low rank blocks

For LU factorisation: replace each sequential matrix multiplication with parallel version.

Parallel Complexity of LU Factorisation

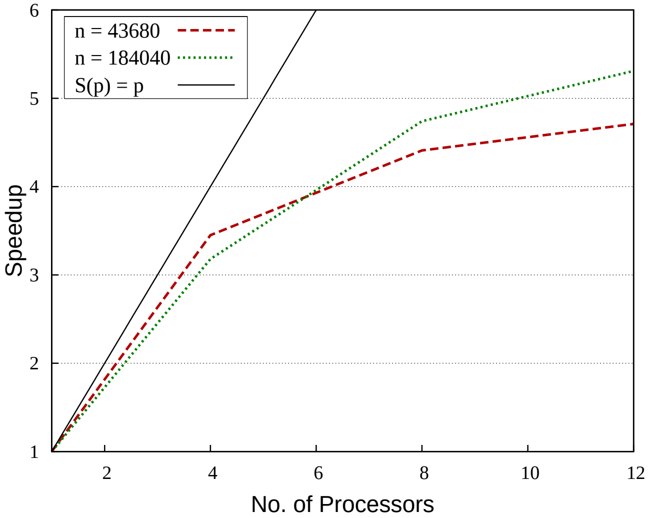
Due to recursion over all matrix blocks:

$$\mathcal{O}\left(\frac{n\log^2 n}{p} + n\log^2 n\right)$$

Only reduction of constant.

Bisection

Numerical Results ($\Omega \subset \mathbb{R}^3$, DLP)



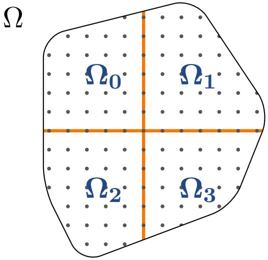
SUN Sunfire 6800, UltraSparcIII+ with 900 MHz

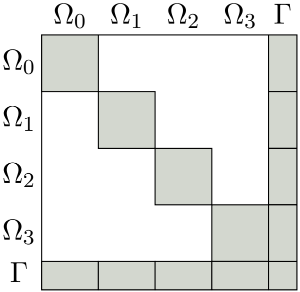
Outline

Direct Domain Decomposition

$\mathcal{H}$-matrices based on Direct Domain Decomposition

Given: decomposition of Ω into p non-overlapping subdomains $\Omega_1, \cdots, \Omega_p$ and global interface Γ:





Assumption: decoupling of the indices by the interface, e.g. local operator and local ansatz functions.

Direct Domain Decomposition

LU Factorisation

$$\begin{pmatrix} A_{11} & & & A_{1\Gamma} \\ & \ddots & & \vdots \\ & & A_{pp} & A_{p\Gamma} \\ A_{\Gamma 1} & \dots & A_{\Gamma p} & A_{\Gamma\Gamma} \end{pmatrix} = \begin{pmatrix} L_{11} & & & \\ & \ddots & & \\ & & L_{pp} & \\ L_{\Gamma 1} & \dots & L_{\Gamma p} & L_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} U_{11} & & & U_{1\Gamma} \\ & \ddots & & \vdots \\ & & U_{pp} & U_{p\Gamma} \\ & & & U_{\Gamma\Gamma} \end{pmatrix}$$

On processor i:

1. factorise $A_{ii} = L_{ii}U_{ii}$, (seq. LU Fac.)
2. solve $A_{i\Gamma} = L_{ii}U_{i\Gamma}$ and $A_{\Gamma i} = L_{\Gamma i}U_{ii}$, (seq. Algo.)
3. compute and exchange $L_{\Gamma i}U_{i\Gamma}$, ($\log p$ steps)
4. update $A_{\Gamma\Gamma} = A_{\Gamma\Gamma} - \sum_i L_{\Gamma i}U_{i\Gamma}$, (seq. Matrix Mult.)
5. factorise $A_{\Gamma\Gamma} = L_{\Gamma\Gamma}L_{\Gamma\Gamma}$ (seq. LU Fac.)

Direct Domain Decomposition

Complexity of LU Factorisation

- equal load of order n/p per subdomain,
- interface of minimal order w.r.t. dimension d:
  - $\mathcal{O}\left(\frac{n}{p}^{(d-1)/d}\right)$ per subdomain and
  - $\mathcal{O}\left(p^{1/d} n^{(d-1)/d}\right)$ for global interface

$$\mathcal{O}\left(\frac{n \log^2 n}{p} + p^{1/d} n^{(d-1)/d} \log^2 n \log p\right)$$

Direct Domain Decomposition

Complexity of LU Factorisation

- equal load of order n/p per subdomain,
- interface of minimal order w.r.t. dimension d:
  - $\mathcal{O}\left(\frac{n}{p}^{(d-1)/d}\right)$ per subdomain and
  - $\mathcal{O}\left(p^{1/d} n^{(d-1)/d}\right)$ for global interface

$$\mathcal{O}\left(\frac{n \log^2 n}{p} + p^{1/d} n^{(d-1)/d} \log^2 n \log p\right)$$

Advantages/Disadvantages

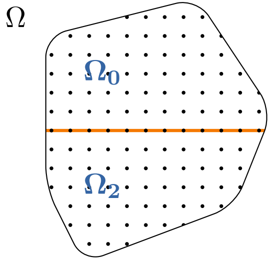
+ small changes to sequential algorithm
− interface can be large ($d = 3$); limits parallel speedup

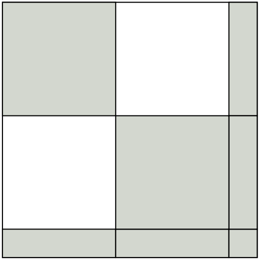
Outline

Nested Dissection

$\mathcal{H}$-matrices based on Nested Dissection

Given: hierarchical decomposition of Ω into 2 non-overlapping subdomains and a local interface:



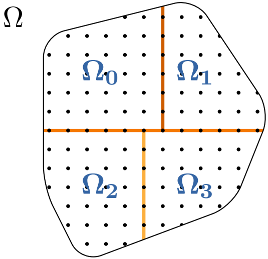


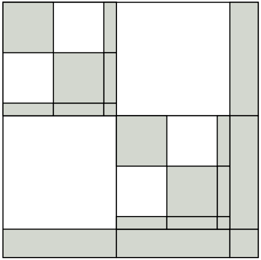
Again assuming decoupling of indices by local interface.

Nested Dissection

$\mathcal{H}$-matrices based on Nested Dissection

Given: hierarchical decomposition of Ω into 2 non-overlapping subdomains and a local interface:



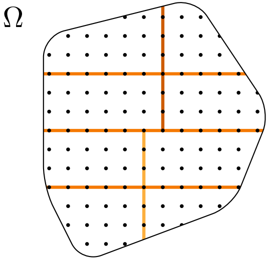


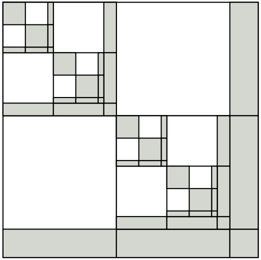
Again assuming decoupling of indices by local interface.

Nested Dissection

$\mathcal{H}$-matrices based on Nested Dissection

Given: hierarchical decomposition of Ω into 2 non-overlapping subdomains and a local interface:



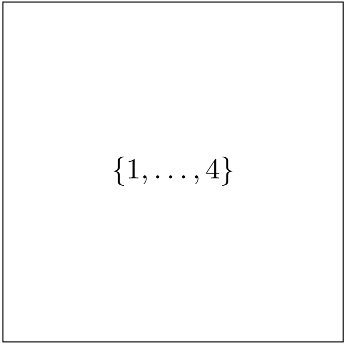


Again assuming decoupling of indices by local interface.

Nested Dissection

Data Distribution

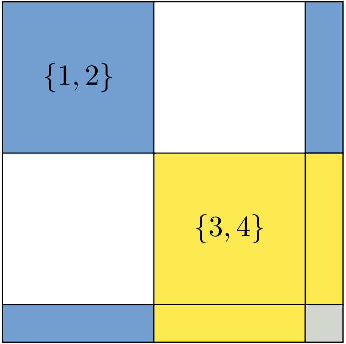
Mapping of processor set $\{1, \ldots, p\}$ onto matrix blocks follows decomposition hierarchy:



Nested Dissection

Data Distribution

Mapping of processor set $\{1,\ldots,p\}$ onto matrix blocks follows decomposition hierarchy:

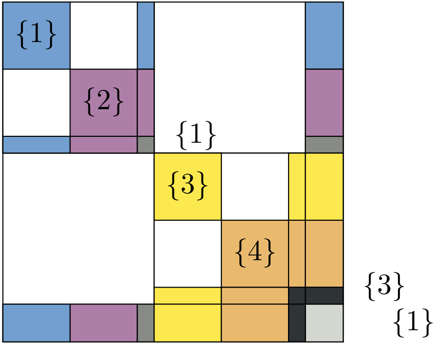


$\{1\}$ (master)

Nested Dissection

Data Distribution

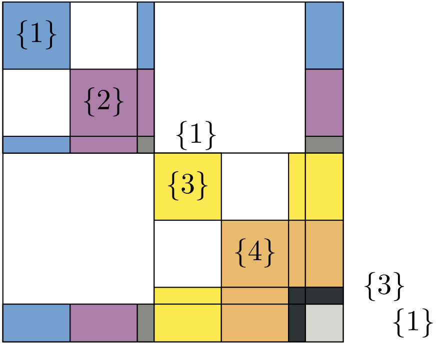
Mapping of processor set $\{1, \ldots, p\}$ onto matrix blocks follows decomposition hierarchy:



Nested Dissection

Data Distribution

Mapping of processor set $\{1, \ldots, p\}$ onto matrix blocks follows decomposition hierarchy:



Difference to Sequential Structure

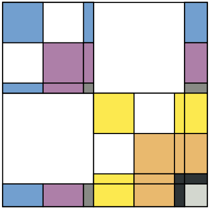
Dense or low rank matrix blocks are not allowed on a level smaller than $\log p$.

Nested Dissection

LU Factorisation

Using algorithm for direct domain decomposition for $p = 2$ but now with recursion.

1. partition local P into P_1, P_2 and choose j such that $i \in P_j$,
2. factorise $A_{jj} = L_{jj} U_{jj}$, (Recursion)
3. solve $A_{j\Gamma} = L_{jj} U_{j\Gamma}$ and $A_{\Gamma j} = L_{\Gamma j} U_{jj}$, (with par. Mult.)
4. compute and exchange $L_{\Gamma j} U_{j\Gamma}$ with local master,
5. on local master: (sequential)
    1. update $A_{\Gamma\Gamma} = A_{\Gamma\Gamma} - \sum_j L_{\Gamma j} U_{j\Gamma}$,
    2. factorise $A_{\Gamma\Gamma} = L_{\Gamma\Gamma} L_{\Gamma\Gamma}$



Nested Dissection

Complexity of LU Factorisation

- equal load per subdomain,
- minimal order w.r.t. d of local interface:

$$\mathcal{O}\left(\frac{n\log^2 n}{p} + n^{(d-1)/d}\log^2 n\log p\right)$$

without $p^{1/d}$ term.

Nested Dissection

Complexity of LU Factorisation

- equal load per subdomain,
- minimal order w.r.t. d of local interface:

$$\mathcal{O}\left(\frac{n \log^2 n}{p} + n^{(d-1)/d} \log^2 n \log p\right)$$

without $p^{1/d}$ term.

Advantages/Disadvantages

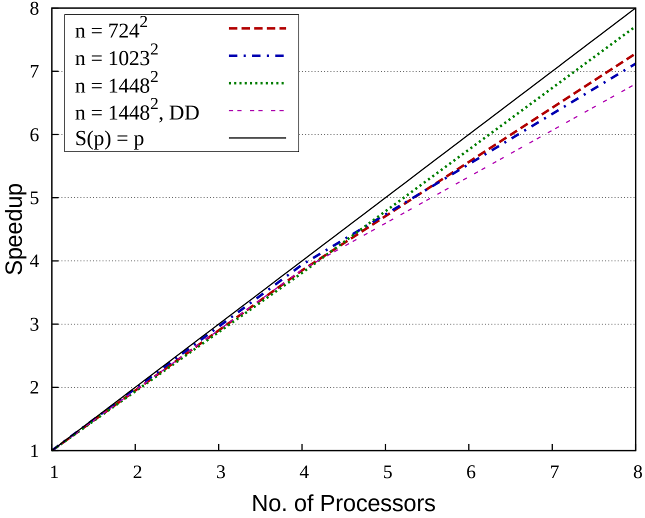
+ partial parallelisation of the interface and therefore lower complexity,
– more complicated algorithm with parallel solve and multiplication

Outline

1. Introduction
2. Bisection
3. Direct Domain Decomposition
4. Nested Dissection
5. Numerical Examples

Numerical Examples

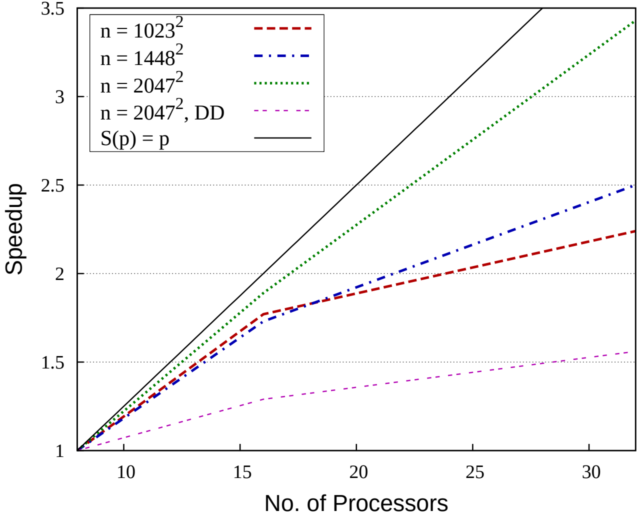
Laplace in $\Omega = [0,1]^2$



AMD Opteron 2.4 GHz, Infiniband Interconnect

Numerical Examples

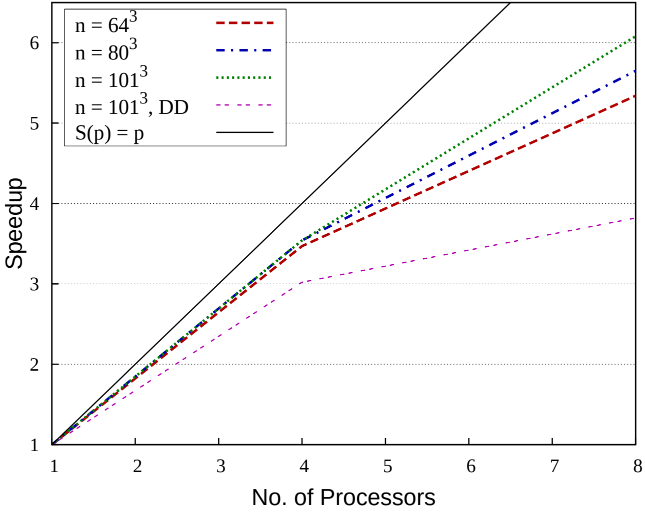
Laplace in $\Omega = [0, 1]^2$



AMD Opteron 2.4 GHz, Infiniband Interconnect

Numerical Examples

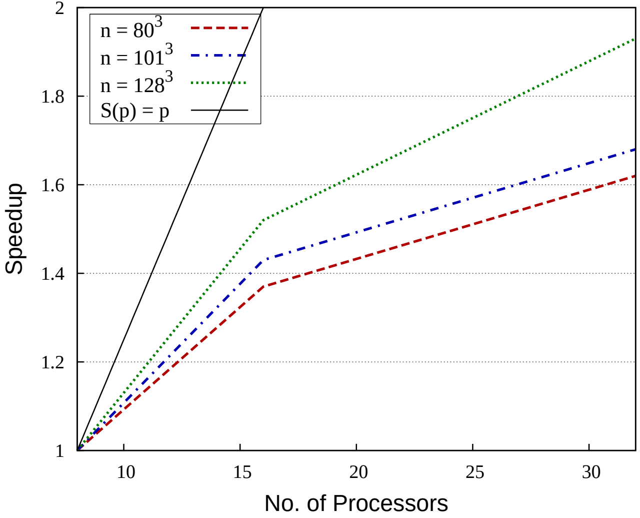
Laplace in $\Omega = [0,1]^3$



AMD Opteron 2.4 GHz, Infiniband Interconnect

Numerical Examples

Laplace in $\Omega = [0,1]^3$



AMD Opteron 2.4 GHz, Infiniband Interconnect