Hierarchical Sparse Hybrid Solvers for Manycores Platforms

Emmanuel AGULLO Luc GIRAUD Abdou GUERMOUCHE
Stojce NAKOV Jean ROMAN

Third Workshop of the CNPq-Inria HOSCAR Project, September 2^{nd}-5^{th}, 2013

Inria Bordeaux - Sud-Ouest Research Center, France

TOTAL

INVENTORS FOR THE DIGITAL WORLD

Outline

1. Introduction

2. Hybrid MPI + threads paralelization

3. Task-based paralelization

4. Conclusion and future work

Motivations

Goal: solving $Ax = b$, where A is sparse

Usual trades off

Direct

- Robust/accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems

Iterative

- Problem dependent efficiency / accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster

Motivations

Goal: solving $Ax = b$, where A is sparse

Usual trades off

Direct

- Robust/accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems

Iterative

- Problem dependent efficiency / accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster

Motivations

Goal: solving $Ax = b$, where A is sparse

Usual trades off

Direct

- Robust/accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems

Iterative

- Problem dependent efficiency / accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster

Motivations

Goal: solving $Ax = b$, where A is sparse

Usual trades off

Direct

- Robust/accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems

Iterative

- Problem dependent efficiency / accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster

Motivations

Goal: solving $Ax = b$, where A is sparse

Usual trades off

Direct

- Robust/accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems

Iterative

- Problem dependent efficiency / accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster

Sparse hybrid (direct/iterative) domain decomposition linear solvers

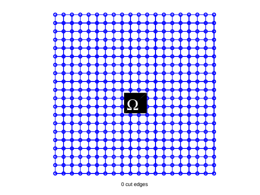
Partitioning the global matrix

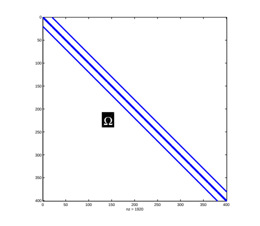
- Global hybrid decomposition:

$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_{\mathcal{I}\mathcal{I}} & \mathcal{A}_{\mathcal{I}\Gamma} \\ \mathcal{A}_{\Gamma\mathcal{I}} & \mathcal{A}_{\Gamma\Gamma} \end{pmatrix}$$

- Global Schur complement:

$$\mathcal{S} = \mathcal{A}_{\Gamma\Gamma} - \mathcal{A}_{\Gamma\mathcal{I}}\mathcal{A}_{\mathcal{I}\mathcal{I}}^{-1}\mathcal{A}_{\mathcal{I}\Gamma}$$





Sparse hybrid (direct/iterative) domain decomposition linear solvers

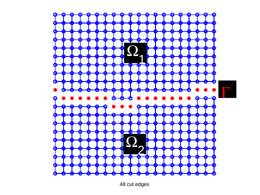
Partitioning the global matrix

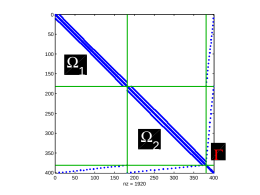
- Global hybrid decomposition:

$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_{\mathcal{I}\mathcal{I}} & \mathcal{A}_{\mathcal{I}\Gamma} \\ \mathcal{A}_{\Gamma\mathcal{I}} & \mathcal{A}_{\Gamma\Gamma} \end{pmatrix}$$

- Global Schur complement:

$$\mathcal{S} = \mathcal{A}_{\Gamma\Gamma} - \mathcal{A}_{\Gamma\mathcal{I}}\mathcal{A}_{\mathcal{I}\mathcal{I}}^{-1}\mathcal{A}_{\mathcal{I}\Gamma}$$





Sparse hybrid (direct/iterative) domain decomposition linear solvers

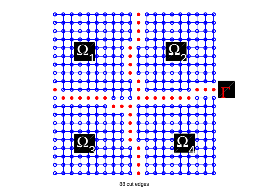
Partitioning the global matrix

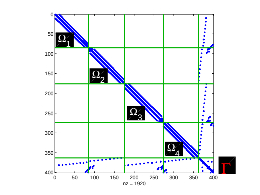
- Global hybrid decomposition:

$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_{\mathcal{I}\mathcal{I}} & \mathcal{A}_{\mathcal{I}\Gamma} \\ \mathcal{A}_{\Gamma\mathcal{I}} & \mathcal{A}_{\Gamma\Gamma} \end{pmatrix}$$

- Global Schur complement:

$$\mathcal{S} = \mathcal{A}_{\Gamma\Gamma} - \mathcal{A}_{\Gamma\mathcal{I}}\mathcal{A}_{\mathcal{I}\mathcal{I}}^{-1}\mathcal{A}_{\mathcal{I}\Gamma}$$





Sparse hybrid (direct/iterative) domain decomposition linear solvers

Global preconditioner based solvers

- ⋆ HIPS (Inria, HiePACS)
- ⋆ PDSLin (LBNL, USA)
- ⋆ ShyLU (Sandia NL)

Local preconditioner based solver

- ⋆ MaPHyS (Inria, HiePACS)

Sparse hybrid (direct/iterative) domain decomposition linear solvers

Global preconditioner based solvers

- ⋆ HIPS (Inria, HiePACS)
- ⋆ PDSLin (LBNL, USA)
- ⋆ ShyLU (Sandia NL)

Local preconditioner based solver

- ⋆ MaPHyS (Inria, HiePACS)

Method used in MaPHyS

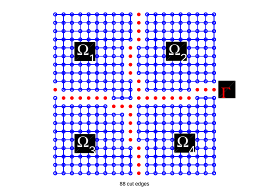
Factorization of local interiors

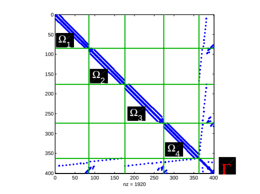
- Global matrix:

$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_{\mathcal{I}\mathcal{I}} & \mathcal{A}_{\mathcal{I}\Gamma} \\ \mathcal{A}_{\Gamma\mathcal{I}} & \mathcal{A}_{\Gamma\Gamma} \end{pmatrix}$$

- Global Schur Complement:

$$\mathcal{S}^{(i)} = \mathcal{A}^{(i)}_{\Gamma\Gamma} - \mathcal{A}_{\Gamma_i\mathcal{I}_i}\mathcal{A}^{-1}_{\mathcal{I}_i\mathcal{I}_i}\mathcal{A}_{\mathcal{I}_i\Gamma_i}$$





Method used in MaPHyS

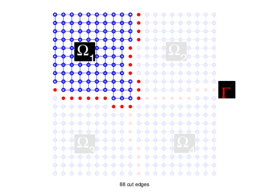
Factorization of local interiors

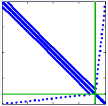
- Local matrix:

$$\mathcal{A}^{(i)} = \begin{pmatrix} \mathcal{A}_{\mathcal{I}_i\mathcal{I}_i} & \mathcal{A}_{\mathcal{I}_i\Gamma_i} \\ \mathcal{A}_{\Gamma_i\mathcal{I}_i} & \mathcal{A}_{\Gamma\Gamma}^{(i)} \end{pmatrix}$$

- Local Schur Complement:

$$\mathcal{S}^{(i)} = \mathcal{A}_{\Gamma\Gamma}^{(i)} - \mathcal{A}_{\Gamma_i\mathcal{I}_i}\mathcal{A}_{\mathcal{I}_i\mathcal{I}_i}^{-1}\mathcal{A}_{\mathcal{I}_i\Gamma_i}$$

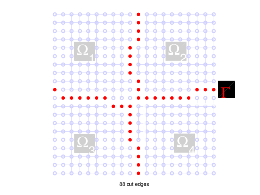


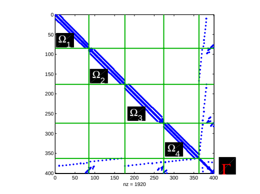


Method used in MaPHyS

Constructing of the preconditioner

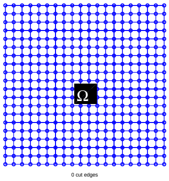
- Local Schur Complement:
  $\mathcal{S}^{(i)} = \mathcal{A}^{(i)}_{\Gamma\Gamma} - \mathcal{A}_{\Gamma_i \mathcal{I}_i} \mathcal{A}^{-1}_{\mathcal{I}_i \mathcal{I}_i} \mathcal{A}_{\mathcal{I}_i \Gamma_i}$
- Algebraic Additive Schwarz Preconditioner
  $\mathcal{M} = \sum_{i=1}^{N} \mathcal{R}^T_{\Gamma_i} (\bar{\mathcal{S}}^{(i)})^{-1} \mathcal{R}_{\Gamma_i}$
  where $\bar{\mathcal{S}}^{(i)}$ is obtained from $\mathcal{S}^{(i)}$ via neighbor to neighbor communications.
- Global Schur complement:
  $\mathcal{S} = \sum_{i=1}^{N} \mathcal{R}^T_{\Gamma_i} (\mathcal{S}^{(i)}) \mathcal{R}_{\Gamma_i}$

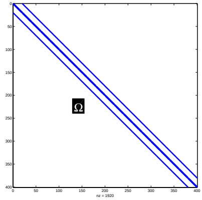




Software used in MAPHYS

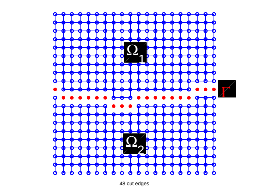
- ⋆ Partitioning the global matrix in several local matrices
  - SCOTCH [Pellegrini and al.]
  - METIS [G. Karypis and V. Kumar]
- ⋆ Local factorization
  - MUMPS [P. Amestoy and al.] (with Schur option)
  - PASTIX [P. Ramet and al.] (with Schur option)
- ⋆ Constructing of the preconditioner
  - MKL library
- ⋆ Solving the reduced system
  - CG/GMRES/FGMRES [V.Fraysse and L.Giraud] using MKL library for the reduced system

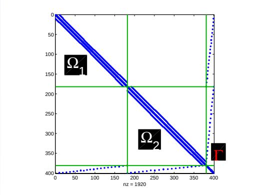




Software used in MaPHyS

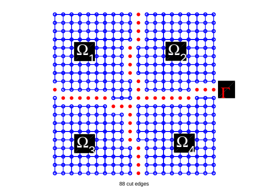
- Partitioning the global matrix in several local matrices
  - Scotch [Pellegrini and al.]
  - Metis [G. Karypis and V. Kumar]
- Local factorization
  - Mumps [P. Amestoy and al.] (with Schur option)
  - PaStiX [P. Ramet and al.] (with Schur option)
- Constructing of the preconditioner
  - Mkl library
- Solving the reduced system
  - Cg/Gmres/FGmres [V.Fraysse and L.Giraud] using Mkl library for the reduced system

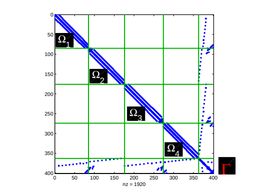




Software used in MaPHyS

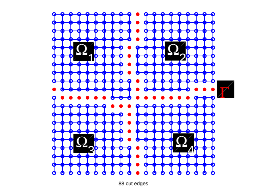
- ⋆ Partitioning the global matrix in several local matrices
  - Scotch [Pellegrini and al.]
  - Metis [G. Karypis and V. Kumar]
- ⋆ Local factorization
  - Mumps [P. Amestoy and al.] (with Schur option)
  - PaStiX [P. Ramet and al.] (with Schur option)
- ⋆ Constructing of the preconditioner
  - Mkl library
- ⋆ Solving the reduced system
  - Cg/Gmres/FGmres [V.Fraysse and L.Giraud] using Mkl library for the reduced system

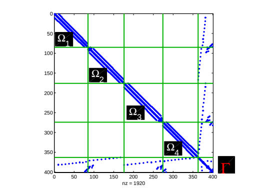




Software used in MaPHyS

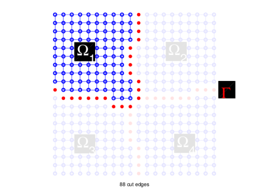
- Partitioning the global matrix in several local matrices
  - Scotch [Pellegrini and al.]
  - Metis [G. Karypis and V. Kumar]
- Local factorization
  - Mumps [P. Amestoy and al.] (with Schur option)
  - PaStiX [P. Ramet and al.] (with Schur option)
- Constructing of the preconditioner
  - Mkl library
- Solving the reduced system
  - CG/GMRES/FGMRES [V.Fraysse and L.Giraud] using Mkl library for the reduced system

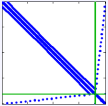




Software used in MAPHYS

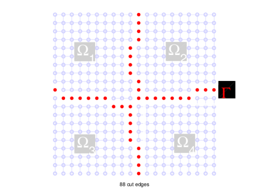
- Partitioning the global matrix in several local matrices
  - SCOTCH [Pellegrini and al.]
  - METIS [G. Karypis and V. Kumar]
- Local factorization
  - MUMPS [P. Amestoy and al.] (with Schur option)
  - PASTIX [P. Ramet and al.] (with Schur option)
- Constructing of the preconditioner
  - MKL library
- Solving the reduced system
  - CG/GMRES/FGMRES [V.Fraysse and L.Giraud] using MKL library for the reduced system

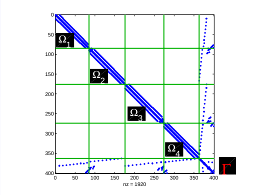




Software used in MAPHYS

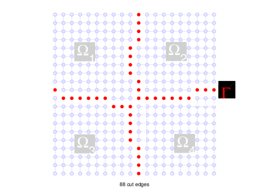
- Partitioning the global matrix in several local matrices
  - SCOTCH [Pellegrini and al.]
  - METIS [G. Karypis and V. Kumar]
- Local factorization
  - MUMPS [P. Amestoy and al.] (with Schur option)
  - PASTIX [P. Ramet and al.] (with Schur option)
- Constructing of the preconditioner
  - MKL library
- Solving the reduced system
  - CG/GMRES/FGMRES [V.Fraysse and L.Giraud] using MKL library for the reduced system

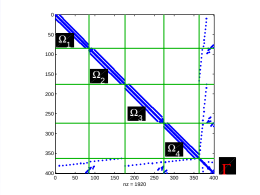




Software used in MaPHyS

- ⋆ Partitioning the global matrix in several local matrices
  - Scotch [Pellegrini and al.]
  - Metis [G. Karypis and V. Kumar]
- ⋆ Local factorization
  - Mumps [P. Amestoy and al.] (with Schur option)
  - PaStiX [P. Ramet and al.] (with Schur option)
- ⋆ Constructing of the preconditioner
  - Mkl library
- ⋆ Solving the reduced system
  - Cg/Gmres/FGmres [V.Fraysse and L.Giraud] using Mkl library for the reduced system





Outline

1. Introduction
2. Hybrid MPI + threads paralelization
3. Task-based paralelization
4. Conclusion and future work

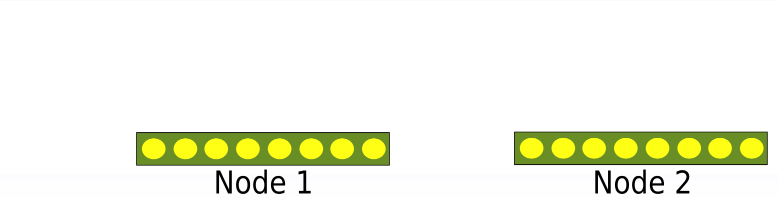
Different type of paralelization of the sparse hybrid linear solvers

Global preconditioner based solvers

- ⋆ HIPS
- ⋆ PDSLin
- ⋆ ShyLU

Local preconditioner based solver

- ⋆ MaPHyS



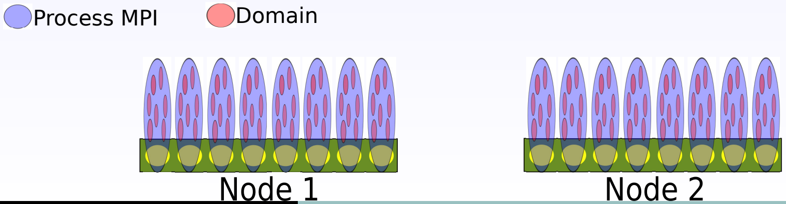
Different type of paralelization of the sparse hybrid linear solvers

Global preconditioner based solvers

- ⋆ HIPS
- ⋆ PDSLIN
- ⋆ SHYLU

Local preconditioner based solver

- ⋆ MAPHYS



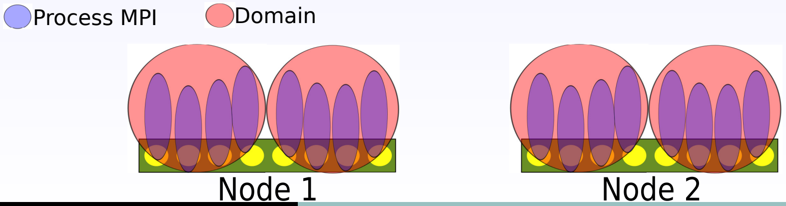
Different type of paralelization of the sparse hybrid linear solvers

Global preconditioner based solvers

- ⋆ HIPS
- ⋆ PDSLIN
- ⋆ SHYLU

Local preconditioner based solver

- ⋆ MAPHYS



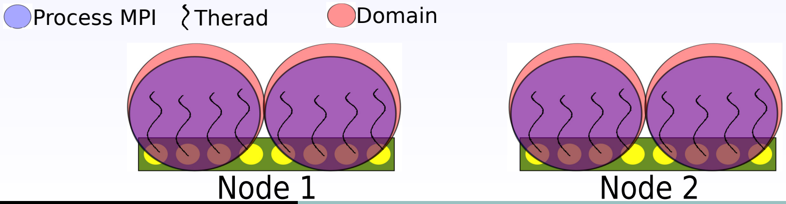
Different type of paralelization of the sparse hybrid linear solvers

Global preconditioner based solvers

- ★ HIPS
- ★ PDSLIN
- ★ SHYLU

Local preconditioner based solver

- ★ MAPHYS



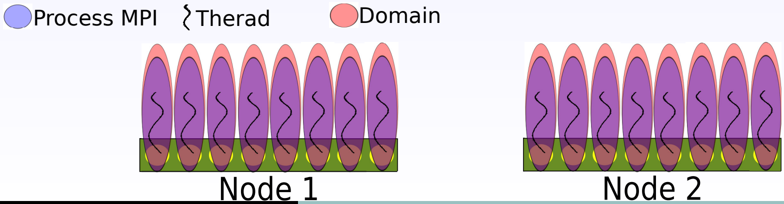
Different type of paralelization of the sparse hybrid linear solvers

Global preconditioner based solvers

- HIPS
- PDSLIN
- SHYLU

Local preconditioner based solver

- MAPHYS (before)



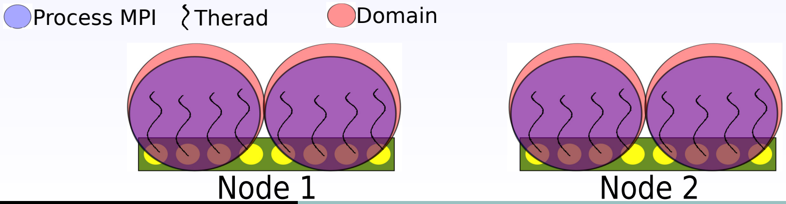
Different type of paralelization of the sparse hybrid linear solvers

Global preconditioner based solvers

- ⋆ HIPS
- ⋆ PDSLIN
- ⋆ SHYLU

Local preconditioner based solver

- ⋆ MAPHYS (this study)



Software used in MAPHYS (before)

Partitioners

- SCOTCH
- METIS

Dense direct solver

- MKL library

Sparse direct solvers

- MUMPS
- PASTIX

Iterative Solvers

- CG/GMRES/FGMRES using MKL library

Software used in MaPHyS (this study)

Dense direct solver

- ⋆ Multi-threaded MKL library

Sparse direct solvers

- ⋆ MUMPS
- ⋆ Multi-threaded PaStiX

Iterative Solvers

- ⋆ CG/GMRES/FGMRES using multi-threaded MKL library

Software used in MAPHYS (this study)

Dense direct solver

- ⋆ Multi-threaded MKL library

Sparse direct solvers

- ⋆ MUMPS
- ⋆ Multi-threaded PASTIX

Iterative Solvers

- ⋆ CG/GMRES/FGMRES using multi-threaded MKL library

Challenge

- ⋆ Composability, Performance

Experimental set up

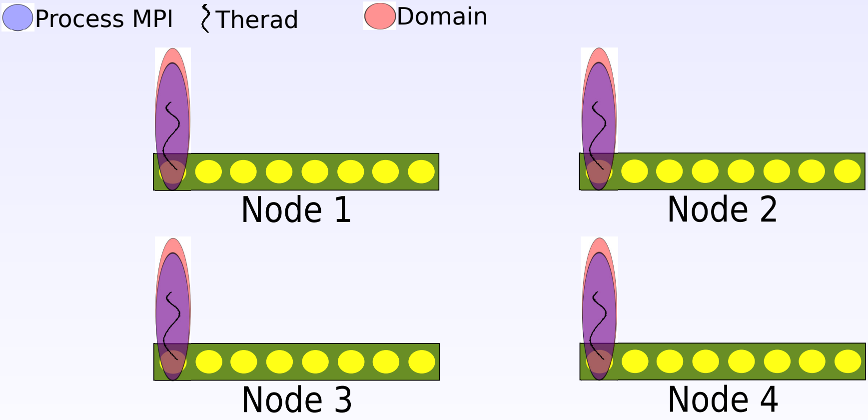
Hardware (on each node)

- Two Quad-core Nehalem Intel® Xeon® X5550
- Memory: 24 GB GDDR3
- Double precision

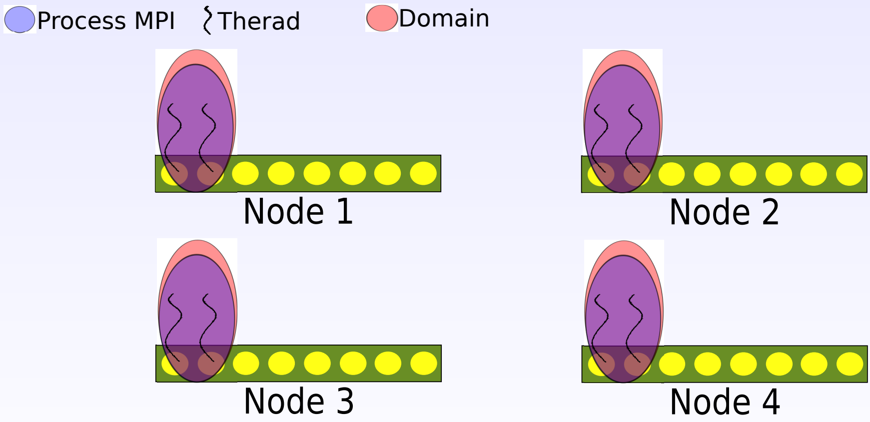
Matrices

	Haltere	Audikw_1	tdr455k
N	1,288,825	943,695	2,738,556
nnz	10,476,775	39,297,771	112,756,352
Precision	Complex	Real	Complex
Description	Electromagnetic problem	3D unstructured structural mechanics	Electromagnetic problem

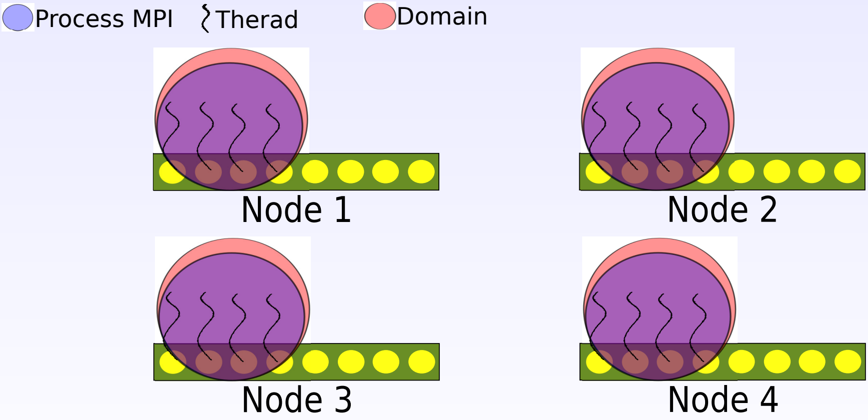
Scalability of MAPHYS on multicore nodes



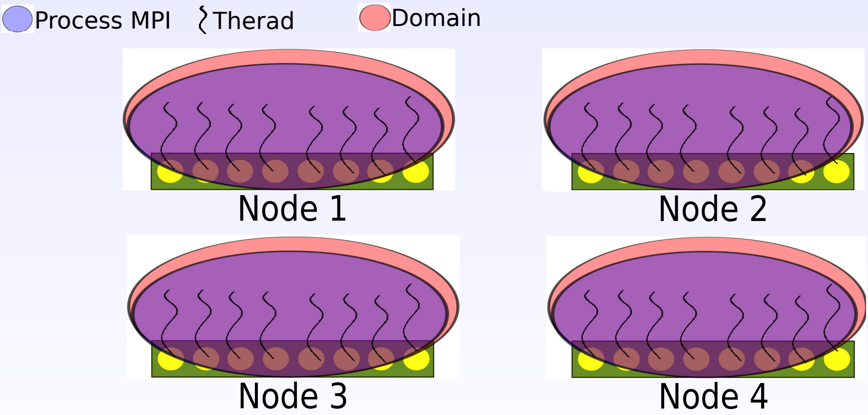
Scalability of MaPHyS on multicore nodes



Scalability of MaPHyS on multicore nodes



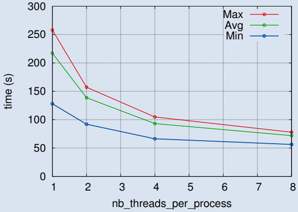
Scalability of MAPHYS on multicore nodes



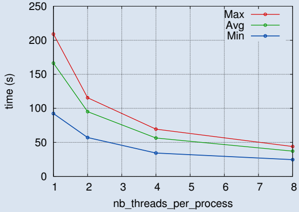
Scalability of MaPHyS on multicore nodes

Achieved performance with the Audi matrix on four nodes

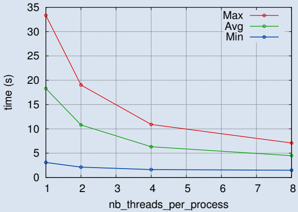
All computational steps



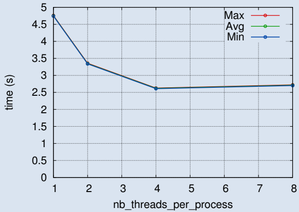
Factorization step



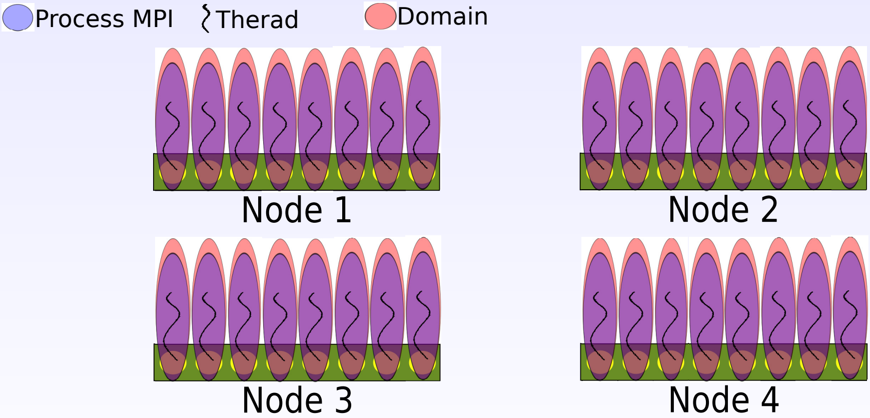
Preconditioning step



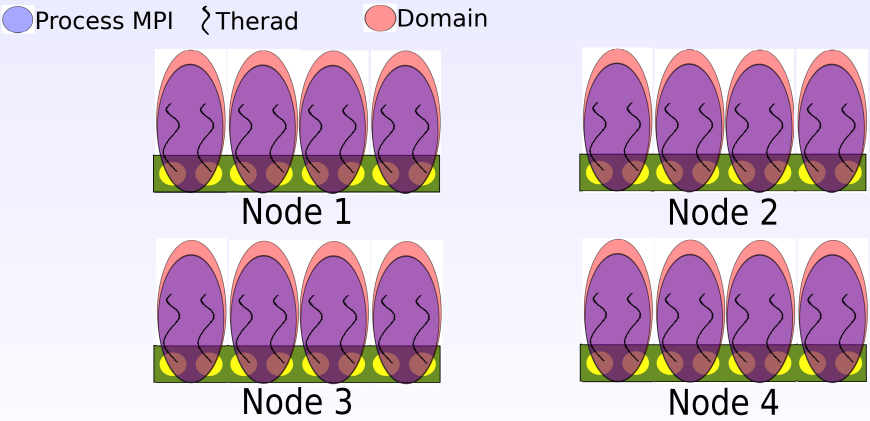
Solve step



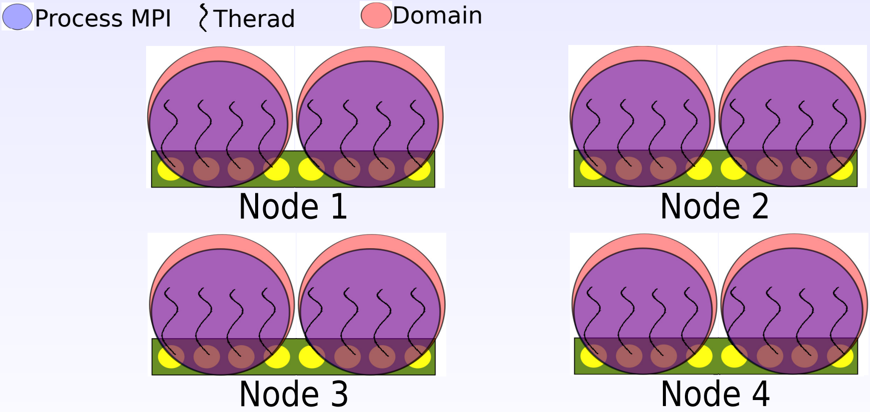
Flexibility to exploit entire multicore nodes



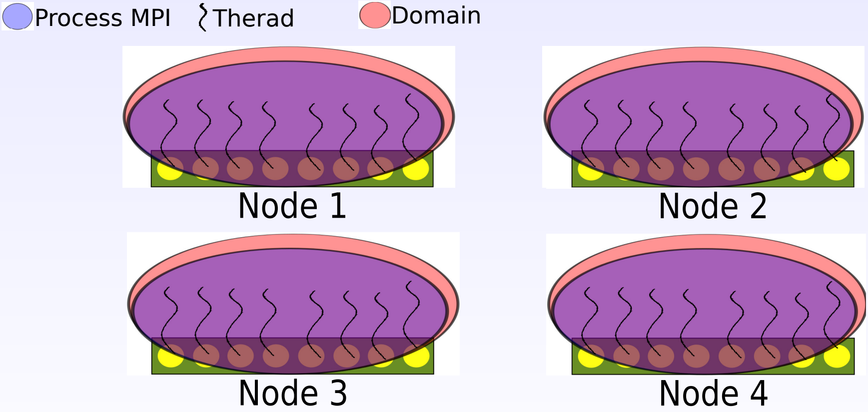
Flexibility to exploit entire multicore nodes



Flexibility to exploit entire multicore nodes



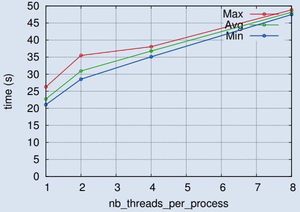
Flexibility to exploit entire multicore nodes



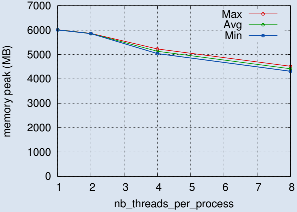
Flexibility to exploit entire multicore nodes

Achieved performance with the Haltere matrix on two nodes

All computational steps



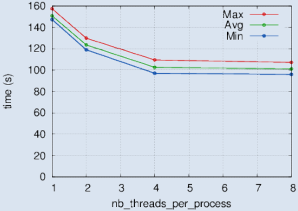
Memory used per node



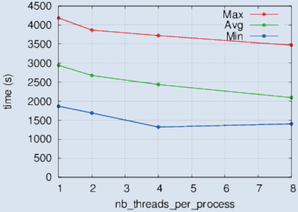
Flexibility to exploit entire multicore nodes

Achieved performance with the tdr455k matrix on sixteen nodes

All computational steps



Memory used per node



Outline

1. Introduction

2. Hybrid MPI + threads paralelization

3. Task-based paralelization

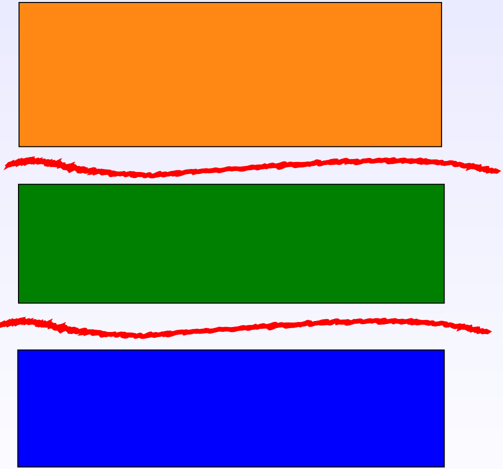
4. Conclusion and future work

Task-based paralelization

Factorization step

Constructing of the preconditioner

Solve

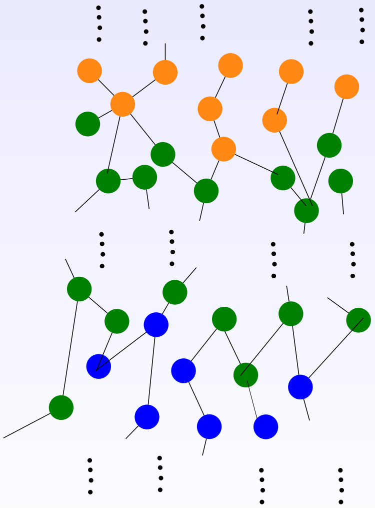


Task-based paralelization

Factorization step

Constructing of the preconditioner

Solve



Approach: Three-layer paradigm

High-level (task-based) algorithm

Runtime (task-based) System

StarPU, PaRSEC, QUARK, SuperMatrix ...

Device kernels

Approach: Three-layer paradigm

High-level (task-based) algorithm

Runtime (task-based) System

StarPU, PaRSEC, QUARK, SuperMatrix ...

Device kernels

Approach: Three-layer paradigm

High-level (task-based) algorithm

Runtime (task-based) System

STARPU, PARSEC, QUARK, SuperMatrix ...

Device kernels

Runtime systems in linear algebra

Dense linear algebra libraries

- PLASMA (QUARK)
- DPLASMA (PaRSEC)
- MAGMA-MORSE (StarPU)
- FLAME (SuperMatrix)

Sparse linear algebra libraries

- PaStiX (StarPU and PaRSEC)
- qr_mumps (StarPU and PaRSEC)

Runtime systems in linear algebra

Dense linear algebra libraries

- PLASMA (QUARK)
- DPLASMA (PARSEC)
- MAGMA-MORSE (STARPU)
- FLAME (SUPERMATRIX)

Sparse linear algebra libraries

- PASTIX (STARPU and PARSEC)
- qr_mumps (STARPU and PARSEC)

Software used in MAPHYS

Dense direct solver

- MAGMA-MORSE library

Sparse direct solvers

- PASTIX

Iterative Solvers

- CG/GMRES/FGMRES

Software used in MaPHyS

Dense direct solver

- ⋆ MAGMA-MORSE library with StarPU

Sparse direct solvers

- ⋆ PaStiX with StarPU

Iterative Solvers

- ⋆ Cg/Gmres/FGmres

Software used in MaPHyS

Dense direct solver

- ⋆ MAGMA-MORSE library with StarPU

Sparse direct solvers

- ⋆ PaStiX with StarPU

Iterative Solvers

- ⋆ Cg/Gmres/FGmres

Tackifying the CG Algorithm

CG Algorithm

$r \leftarrow b$
$r \leftarrow r - Ax$
$p \leftarrow r$
$\delta_{new} \leftarrow dot(r, r)$
$\delta_{old} \leftarrow \delta_{new}$
while $\frac{||b-Ax||}{||b||} \leq eps$ **do**
  $q \leftarrow Ap$
  $\alpha \leftarrow \delta_{new}/dot(p, q)$
  $x \leftarrow x + \alpha p$
  $r \leftarrow r - \alpha q$
  $\delta_{new} \leftarrow dot(r, r)$
  $\beta \leftarrow \delta_{new}/\delta_{old}$
  $\delta_{old} \leftarrow \delta_{new}$
  $p \leftarrow r + \beta p$
end while

Tackifying the CG Algorithm

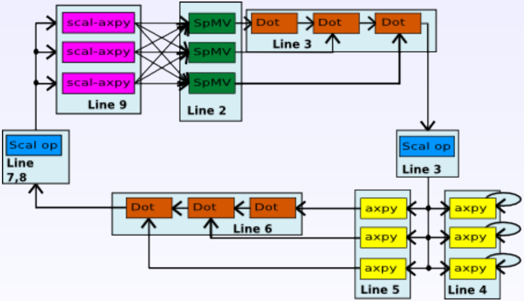
CG Algorithm

$r \leftarrow b$
$r \leftarrow r - Ax$
$p \leftarrow r$
$\delta_{new} \leftarrow dot(r, r)$
$\delta_{old} \leftarrow \delta_{new}$
while $\frac{\|b - Ax\|}{\|b\|} \leq eps$ **do**
    $q \leftarrow Ap$
    $\alpha \leftarrow \delta_{new} / dot(p, q)$
    $x \leftarrow x + \alpha p$
    $r \leftarrow r - \alpha q$
    $\delta_{new} \leftarrow dot(r, r)$
    $\beta \leftarrow \delta_{new} / \delta_{old}$
    $\delta_{old} \leftarrow \delta_{new}$
    $p \leftarrow r + \beta p$
end while

Tackifying the CG Algorithm

Main CG loop

1: **while** $\frac{\|b-Ax\|}{\|b\|} \leq eps$ **do**
2: $q \leftarrow Ap$
3: $\alpha \leftarrow \delta_{new}/dot(p,q)$
4: $x \leftarrow x + \alpha p$
5: $r \leftarrow r - \alpha q$
6: $\delta_{new} \leftarrow dot(r,r)$
7: $\beta \leftarrow \delta_{new}/\delta_{old}$
8: $\delta_{old} \leftarrow \delta_{new}$
9: $p \leftarrow r + \beta p$
10: **end while**



Execution of CG on 3 GPUs

1: **while** $\frac{\|b-Ax\|}{\|b\|} \leq eps$ **do**

2: $\quad q \leftarrow Ap$

3: $\quad \alpha \leftarrow \delta_{new}/dot(p,q)$

4: $\quad x \leftarrow x + \alpha p$

5: $\quad r \leftarrow r - \alpha q$

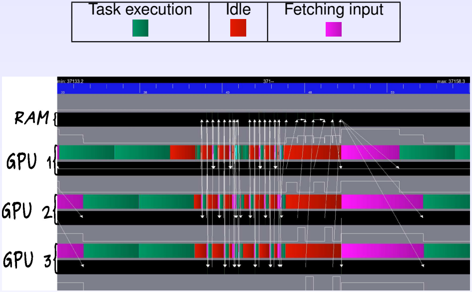
6: $\quad \delta_{new} \leftarrow dot(r,r)$

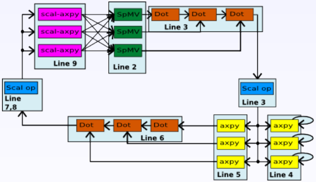
7: $\quad \beta \leftarrow \delta_{new}/\delta_{old}$

8: $\quad \delta_{old} \leftarrow \delta_{new}$

9: $\quad p \leftarrow r + \beta p$

10: **end while**

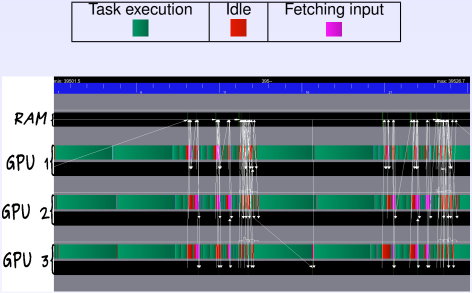




Trace of one iteration of CG with 3 GPUs.

Execution of CG on 3 GPUs

while $\frac{||b-Ax||}{||b||} \leq eps$ **do**
  $q \leftarrow Ap$
  $\alpha \leftarrow \delta_{new}/dot(p,q)$
  $x \leftarrow x + \alpha p$
  $r \leftarrow r - \alpha q$
  $\delta_{new} \leftarrow dot(r,r)$
  $\beta \leftarrow \delta_{new}/\delta_{old}$
  $\delta_{old} \leftarrow \delta_{new}$
  $p \leftarrow r + \beta p$
end while



Trace of one iteration of CG with 3 GPUs after optimizing the task-flow.

Performance

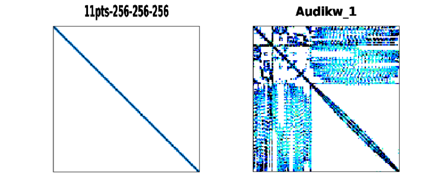
Hardware

NVIDIA Tesla M2070

- Memory: 6 GB GDDR5
- Double precision

Matrices

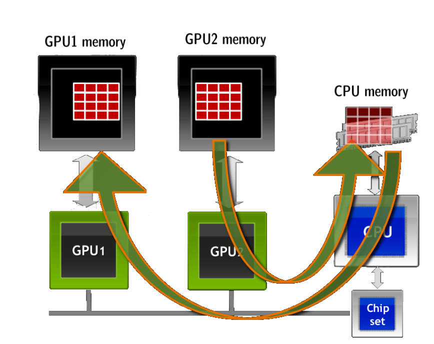
	11pts-256-256-256	Audikw_1
N	16,777,216	943,695
nnz	350,488,576	154,359,999
Description	3D PDE on a Cartesian grid	3D unstructured structural mechanics



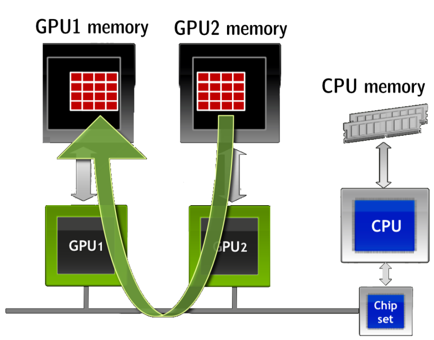
Matrix	Basic CG			Optimized CG		
	1GPU (Gflop/s)	3GPU (Gflop/s)	Speed-up	1GPU (Gflop/s)	3GPU (Gflop/s)	Speed-up
Audikw_1	9.3	12.97	1.39	9.7	16.1	1.7
11ptd-256-256-256	5.24	7.5	1.43	5.7	16.27	2.85

Data transfers between GPUs

GPU-CPU-GPU mechanism

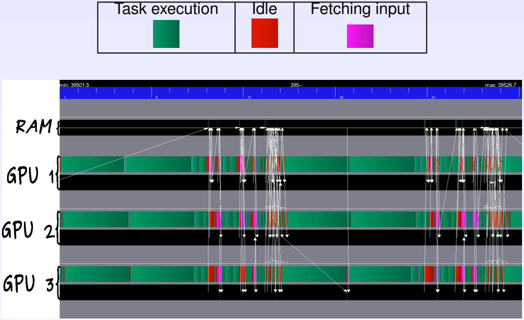


GPU-GPU mechanism



Benefit of using runtime systems

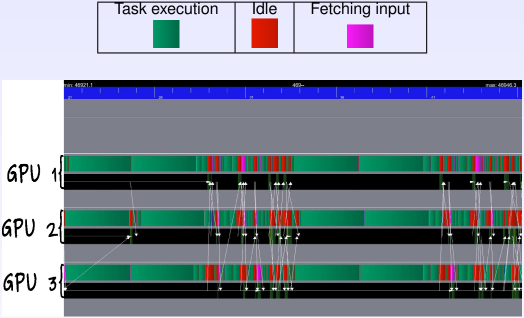
while $\frac{\|b-Ax\|}{\|b\|} \leq eps$ **do**
    $q \leftarrow Ap$
    $\alpha \leftarrow \delta_{new}/dot(p,q)$
    $x \leftarrow x + \alpha p$
    $r \leftarrow r - \alpha q$
    $\delta_{new} \leftarrow dot(r,r)$
    $\beta \leftarrow \delta_{new}/\delta_{old}$
    $\delta_{old} \leftarrow \delta_{new}$
    $p \leftarrow r + \beta p$
end while



GPU-CPU-GPU communication mechanism

Benefit of using runtime systems

while $\frac{||b-Ax||}{||b||} \leq eps$ **do**
$\quad q \leftarrow Ap$
$\quad \alpha \leftarrow \delta_{new}/dot(p,q)$
$\quad x \leftarrow x + \alpha p$
$\quad r \leftarrow r - \alpha q$
$\quad \delta_{new} \leftarrow dot(r,r)$
$\quad \beta \leftarrow \delta_{new}/\delta_{old}$
$\quad \delta_{old} \leftarrow \delta_{new}$
$\quad p \leftarrow r + \beta p$
end while



GPU-GPU communication mechanism

Outline

1. Introduction

2. Hybrid MPI + threads paralelization

3. Task-based paralelization

4. Conclusion and future work

Conclusion and future work

Conclusion

- ⋆ Two-level parallelism efficiently exploits multicore architectures
- ⋆ Synchronizations may be a bottleneck
- ⋆ Investigating task-based paralelization to pipeline all the steps

Future work

- ⋆ Full task-based solver
- ⋆ Further experiments in the collaboration with TOTAL