



Block-preconditioned iterative methods on multicore computer systems and GPU

Performance study

Maya Neytcheva and Ali Dorostkar

together with Dimitar Lukarski and Yvan Notay

PMAA, Lugano, July 2-5, 2014



The concept citing Prof. Dr. Wolfgang Bangerth:

- Solving realistic, large scale applied problems with the most modern numerical methods can be seen as a multidimensional optimization problem, with many levels of complexity that have to be simultaneously taken into account.



The concept citing Prof. Dr. Wolfgang Bangerth:

- Solving realistic, large scale applied problems with the most modern numerical methods can be seen as a multidimensional optimization problem, with many levels of complexity that have to be simultaneously taken into account.
 - The code that enables such large scale computer simulations requires a whole collection of algorithms:
 - ▶ unstructured, adaptive or moving meshes;
 - ▶ time-dependent processes, repeated solution of (non)linear systems;
 - ▶ inner-outer solution procedures, preconditioners utilizing internal algebraic structures,
 - ▶ methods that have a recursive nature.



The concept citing Prof. Dr. Wolfgang Bangerth:

- Solving realistic, large scale applied problems with the most modern numerical methods can be seen as a multidimensional optimization problem, with many levels of complexity that have to be simultaneously taken into account.
 - The code that enables such large scale computer simulations requires a whole collection of algorithms:
 - All this has to work efficiently on modern computer architectures.



The concept citing Prof. Dr. Wolfgang Bangerth:

- Solving realistic, large scale applied problems with the most modern numerical methods can be seen as a multidimensional optimization problem, with many levels of complexity that have to be simultaneously taken into account.
 - The code that enables such large scale computer simulations requires a whole collection of algorithms:
 - All this has to work efficiently on modern computer architectures.
 - Codes at this level of complexity can no longer be written from scratch.



Outline of the talk

Performance study of a complex numerical solution procedure on CPU and GPU

- ▶ Target problem from Geophysics
- ▶ The algebraic problem
- ▶ Numerical solution procedure - preconditioned inner-outer iterations, implemented using several packages for scientific computations
- ▶ Performance results - CPU&GPU
- ▶ ... if time permits: describe a high-quality Schur complement approximation
- ▶ Conclusions, to-do list



Faults in Scandinavia



Pärvie fault, Sweden, a wave in the Earth

"...Pärvie is an old Lappish word meaning roughly *wave in the ground*. ... a 150-km-long slash across the land, which at its maximum resembles a tsunami crest frozen in the rock.

How did it get there?"

Arch C. Johnston, seismologist at the University of Memphis, *Science*, Vol 274, 1996



The applied problem - Glacial isostatic adjustment (GIA)

[Click here to view the movie](#)



The mathematical model

$$\underbrace{\nabla \cdot \boldsymbol{\sigma}}_{(A)} - \underbrace{\nabla(\rho_0 \mathbf{u} \cdot \nabla \Phi_0)}_{(B)} - \underbrace{\rho_1 \nabla \Phi_0}_{(C)} - \underbrace{\rho_0 \nabla \Phi_1}_{(D)} = \mathbf{f} \quad \text{in } \Omega \subset \mathbb{R}^3$$

$$\nabla \cdot (\nabla \Phi_1) - 4\pi G \rho_1 = 0,$$

$$\rho_1 + \rho_0 \nabla \cdot \mathbf{u} + \mathbf{u} \cdot \frac{\partial \rho_0}{\partial \mathbf{r}} = 0.$$

(A) - force due to spatial gradients in stress

(B) - so-called *advection of pre-stress* - crucial for the model

(C) - internal buoyancy

(D) - self-gravitation effects

The density ρ_1 is determined by the linearized continuity equation.



The mathematical model, cont.

In addition, we need appropriate constitutive relations, describing stress-strain-displacement relations:

$$\sigma(\mathbf{x}, t) = \sigma_E(\mathbf{x}) - \sigma_I(\mathbf{x}, t),$$

where $\sigma_E(\mathbf{x}) = C(\mathbf{x}, 0)\varepsilon_E$ is the instantaneous stress due to elastic (reversible) response to load and $\sigma_I(\mathbf{x}, t) = C(\mathbf{x}, t)\varepsilon_I$ is the contribution due to inelastic response (viscoelastic).

$$\sigma(\mathbf{x}, t) = C(\mathbf{x}, 0)\varepsilon_E - \int_0^t \dot{\gamma} C(\mathbf{x}, t) R d\tau.$$



The simplified problem

2D model, often used in the geophysics community.

Self-gravitation is excluded and due to the latter, the Earth is modelled as incompressible.

$$\begin{aligned} -\nabla \cdot (2\mu \varepsilon(\mathbf{u})) - \nabla(\mathbf{u} \cdot \nabla p_0) + (\nabla \cdot \mathbf{u}) \nabla p_0 - \mu \nabla p &= \mathbf{f} \text{ in } \Omega \\ \mu \nabla \cdot \mathbf{u} - \frac{\mu^2}{\lambda} p &= 0 \text{ in } \Omega \end{aligned}$$

μ , λ - the Lamé coefficients, ε is the strain tensor,

\mathbf{u} is the displacement vector,

p_0 is the so-called *pre-stress*,

$p = \frac{\lambda}{\mu} \nabla \cdot \mathbf{u}$ is the so-called kinematic pressure, introduced in order to be able to simulate fully incompressible materials, i.e., $\lambda = \infty$.



Weak form and discretization

In this talk: ONLY elastic response!

$$\begin{aligned} a(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p) &= (\mathbf{v}_h, \mathbf{g}) + (\mathbf{v}_h, \gamma) \\ b(\mathbf{u}_h, q_h) - \frac{\mu^2}{\lambda} m(p_h, q_h) &= 0 \end{aligned}$$

The model is discretized using a stable finite element pair of spaces, Taylor-Hood Q2-Q1 on quadrilateral mesh.

It gives rise to algebraic systems of equations that are **large**, **sparse**, **nonsymmetric** and **indefinite**, with a saddle point structure,

$$\begin{bmatrix} F & B^T \\ B & -M \end{bmatrix}$$

The pivot block F is nonsymmetric.



*The efficiency of solving one system of the above type is crucial as it is to be embedded in a **time-evolution procedure**, where systems with matrices with similar characteristics have to be solved repeatedly.*



Description of the numerical algorithm - very classical

The matrix

$$\begin{bmatrix} F & B^T \\ B & -M \end{bmatrix},$$

is preconditioned by

$$\begin{bmatrix} [F] & 0 \\ B & -[\tilde{S}] \end{bmatrix},$$

where

- \tilde{S} is an approximation of the (negative) Schur complement matrix $S = M + BF^{-1}B^T$,
- $[*]$ means a (preconditioned) inner solver with a given matrix.



Numerical efficiency

What numerical efficiency to expect from the preconditioner

$$\begin{bmatrix} [F] & 0 \\ B & -[\tilde{S}] \end{bmatrix} ?$$

The theory says: very few (**outer**) iterations if

→ we solve accurately with F ,

→ \tilde{S} is a very good approximation for S .

Computational efficiency

The inner solvers with F and \tilde{S} must be very efficient.



Software used - commercial and publicly available

- ▶ **ABAQUS (ABAQUS FEA)** - software suite for finite element analysis and computer-aided engineering, released in 1978.

General-purpose FE analysis program, most suited for numerical modelling of structural (static and dynamic) response.

- Ease of use on complex problems
- Simple input language
- Comprehensive data checking
- Wide range of preprocessing and post-processing options
- MANY man-years efforts providing, among the many features, efficient numerical solvers.



ABAQUS- 'however's'

- enhanced numerical simulations of GIA problems are not straightforwardly performed with ABAQUS since important terms in the continuous model, such as prestress advection, cannot be added directly, leading to the necessity to modify the model in order to be able to use the package.
- ABAQUS cannot handle purely incompressible materials - Poisson's ratio ν cannot be set to 0.5 but to some closer value, such as 0.4999, for instance.
- ABAQUS is a commercial software.



Software used - commercial and publicly available

- ▶ ABAQUS (ABAQUS FEA)
- ▶ Deal.ii (Differential Equations Analysis Library)
 - ▶ First public version in 2000
 - ▶ Our main toolbox
 - ▶ We utilize the interfaces to other packages, in particular, Trilinos.



Software used - commercial and publicly available

- ▶ ABAQUS (ABAQUS FEA)
- ▶ Deal.ii (Differential Equations Analysis Library)
- ▶ Trilinos
 - ▶ Vast collection of algorithms
 - ▶ Object oriented
 - ▶ Used through deal.ii wrappers. Packages used:
 - ▶ Epetra for sparse matrix and vector storage,
 - ▶ Teuchos for passing parameters to solver and preconditioner,
 - ▶ ML for multigrid preconditioning,
 - ▶ AZTEC for the iterative solver (GMRES, FGMRES).



Software used - commercial and publicly available

- ▶ ABAQUS (ABAQUS FEA)
- ▶ Deal.ii (Differential Equations Analysis Library)
- ▶ Trilinos
- ▶ PARALUTION (D. Lukarski) - sparse linear algebra library
 - ▶ First public version in 2013 - <http://www.paralution.com>
 - ▶ Focus on fine-grained parallelism (multicore CPU and GPU)
 - ▶ **Goal:** to provide a portable library containing iterative methods and preconditioners for linear systems with sparse matrices, to be run on state of the art hardware.
 - ▶ **Run-time decision where to run the application - on CPU or GPU.**
 - ▶ build-in plug-in to deal.ii
 - ▶ The plug-in exports and imports data from deal.ii to PARALUTION.



Software used - commercial and publicly available

- ▶ ABAQUS (ABAQUS FEA)
- ▶ Deal.ii (Differential Equations Analysis Library)
- ▶ Trilinos
- ▶ PARALUTION (D. Lukarski)
- ▶ AGMG (Y. Notay, A. Napov)
 - ▶ implements an aggregation-based algebraic multigrid method.
 - ▶ <http://homepages.ulb.ac.be/~ynotay/AGMG/>
 - ▶ solve algebraic systems of linear equations
 - ▶ expected to be efficient for large systems arising from the discretization of scalar second order elliptic PDEs.
 - ▶ purely algebraic
 - ▶ written in FORTRAN 90



Three different AMG implementations

- ▶ Trilinos/AMG

Deal.ii configures the algebraic Multigrid (AMG) preconditioner from Trilinos using the following default settings:

- ▶ Chebyshev smoother with two pre- and post-smoothing steps.
- ▶ Uncoupled aggregation with threshold of 0.02
- ▶ One Multigrid cycle



Three different AMG implementations

- ▶ PARALUTION/AMG
 - ▶ Coarse grid size - 2000
 - ▶ Coupling strength - 0.001
 - ▶ Coarsening type - smoothed aggregation
 - ▶ Multi-colored Gauss-Seidel smoother with relaxation parameter set to 1.3
 - ▶ Pre-smoothing steps - 1
 - ▶ Post-smoothing steps - 2
 - ▶ One multigrid cycle
 - ▶ Smoother matrix format - as in ELLPACK (ELL)
 - ▶ Operator matrix format - compressed sparse row (CSR)



Three different AMG implementations, cont.

- ▶ AGMG

An aggregation-based algebraic multigrid method. Uses double pairwise aggregation algorithm which makes the coarsening faster. It uses 'pure' aggregation-based prolongation and K-cycle multigrid to circumvent the relatively bad scalability of the standard V-cycle.



Description of the computer facilities

(C1) CPU: Intel(R) Xeon(R) 1.6GHz 12 cores

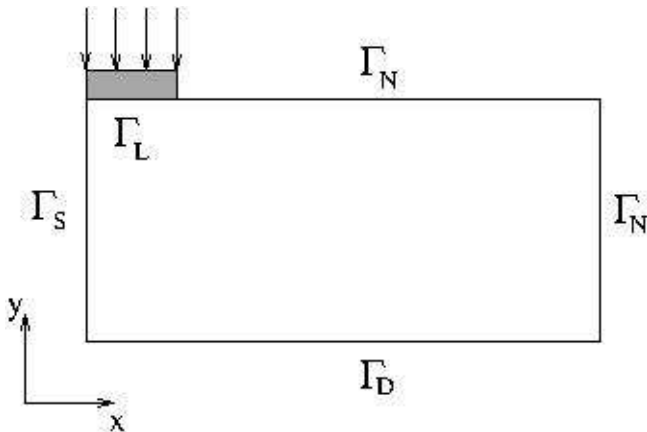
(C2) ▶ CPU: Intel(R) Core(TM) i5-3550 CPU 3.30GHz 4 cores
 ▶ GPU: NVIDIA K40, 12G, 2880 cores

Performance results for ABAQUS and deal.ii/Trilinos- on (C1)
PARALUTION CPU version of the solver - on both (C1) and (C2)
PARALUTION GPU - on (C2)

Parallelism - via built-in functionality of the packages to use OpenMp.
The maximum number of threads equals the number of cores available.
No hyperthreading.



Numerical tests:



The geometry of the problem



Numerical tests:

$$\begin{bmatrix} F & B^T \\ B & -M \end{bmatrix}, \quad \begin{bmatrix} [F] & 0 \\ B & -[\tilde{S}] \end{bmatrix}$$

Inner iterations, AMG-preconditioned FGMRES



Deal.ii/Trilinos/AMG vs ABAQUS/sparse-direct on (C1)

No. of thr.	Deal.II/Trilinos				ABAQUS		
	DOF	Iterations	Setup	Solve (2/3)	DOFs	Setup	Solve
1	1 479 043	19(5, 1)	5.63	51.7 (34.46)	986 626	7.44	59
4		19(5, 1)	5.42	35.2 (23.46)		7.49	33
8		19(5, 1)	5.41	31.9 (21.26)		7.51	28
1	5 907 203	19(5, 1)	23.2	243 (162)	3 939 330	29.72	269
4		19(5, 1)	22.3	157 (104)		29.93	145
8		19(5, 1)	22.1	136 (90)		29.94	122



Deal.ii/Trilinos/AMG vs ABAQUS/sparse-direct on (C1)

No. of thr.	Deal.II/Trilinos				ABAQUS		
	DOF	Iterations	Setup	Solve (2/3)	DOFs	Setup	Solve
1	1 479 043	19(5, 1)	5.63	51.7 (34.46)	986 626	7.44	59
4		19(5, 1)	5.42	35.2 (23.46)		7.49	38
8		19(5, 1)	5.41	31.9 (21.26)		7.51	28
1	5 907 203	19(5, 1)	23.2	243 (162)	3 939 330	29.72	269
4		19(5, 1)	22.3	157 (104)		29.93	145
8		19(5, 1)	22.1	136 (90)		29.94	122

Factor 4!



Deal.ii/Trilinos/AMG vs ABAQUS/sparse-direct on (C1)

No. of thr.	Deal.II/Trilinos				ABAQUS		
	DOF	Iterations	Setup	Solve (2/3)	DOFs	Setup	Solve
1	1 479 043	19(5, 1)	5.63	51.7 (34.46)	986 626	7.44	59
4		19(5, 1)	5.42	35.2 (23.46)		7.49	33
8		19(5, 1)	5.41	31.9 (21.26)		7.51	28
1	5 907 203	19(5, 1)	23.2	243 (162)	3 939 330	29.72	269
4		19(5, 1)	22.3	157 (104)		29.93	145
8		19(5, 1)	22.1	136 (90)		29.94	122

Both do not scale across threads.



Trilinos vs PARALUTION on (C1), CPU only

Threads	DOFS	Trilinos			PARALUTION		
		Iterations	Setup	Solve	Iterations	Setup	Solve
1	370 883	19(4, 1)	1.41	11.83	25(5, 5)	1.710	8.777
4		19(4, 1)	1.35	8.243	25(5, 5)	1.031	3.624
8		19(4, 1)	1.35	7.637	25(5, 5)	0.917	3.743
12		19(4, 1)	1.34	6.893	25(5, 5)	0.893	3.299
1	1 479 043	19(5, 1)	5.63	51.77	25(5, 5)	6.958	35.376
4		19(5, 1)	5.42	35.17	25(5, 5)	4.071	14.640
8		19(5, 1)	5.41	31.97	25(5, 5)	3.629	15.724
12		19(5, 1)	5.37	29.67	25(5, 5)	3.465	13.013
1	5 907 203	19(5, 1)	23.2	243	26(5, 5)	27.913	153.34
4		19(5, 1)	22.26	158	26(5, 5)	16.478	67.242
8		19(5, 1)	22.16	139	26(5, 5)	14.287	66.428
12		19(5, 1)	21.93	127	26(5, 5)	13.680	60.180



PARALUTION: CPU vs GPU on (C2)

DOF	Outer iter.	Time on CPU (s)		Time on GPU (s)	
		Setup	Solve	Setup	Solve
23 603	24	0.04	0.39	0.16	2.8
93 283	24	0.18	1.14	0.27	2.0
370 883	25	0.75	4.88	0.9	4.0
1 479 043	25	2.98	20.86	3.7	5.8
5 907 203	26	15.83	92.85	out of memory	



Time distribution for Trilinos/AMG

DOF	Solution with M	Solution with \tilde{S}	Total solution time
23 603	0.625 (90%)	0.0286 (10%)	0.69
93 283	3.49 (93%)	0.0983 (7%)	3.72
370 883	15.3 (93%)	0.441 (7%)	16.3
1 479 043	77.2 (94%)	2.22 (6%)	81.8
5 907 203	350 (94%)	10.3 (6%)	370

$$\begin{bmatrix} M & 0 \\ B & -C \end{bmatrix},$$



Trilinos/AMG vs AGMG on (C1), CPU, serial mode

DOFs	Trilinos/AMG			AGMG		
	Itr.	Setup	Solve	Itr.	Setup	Solve
23 603	18(3, 1)	0.0986	0.504	19(3, 1)	0.029	0.394
93 283	19(4, 1)	0.349	2.81	19(3, 1)	0.128	1.68
370 883	19(4, 1)	1.4	11.8	19(4, 1)	0.515	7.42
1 479 043	19(5, 1)	5.63	51.7	18(4, 1)	2.09	30.9
5 907 203	19(5, 1)	23.2	243	19(5, 1)	8.69	147

Inner stopping criterion: relative 0.1, for both blocks.



Trilinos/AMG vs AGMG on (C1), CPU, serial mode

DOFs	Trilinos/AMG			AGMG		
	Itr.	Setup	Solve	Itr.	Setup	Solve
23 603	18(3, 1)	0.0986	0.504	19(3, 1)	0.029	0.394
93 283	19(4, 1)	0.349	2.81	19(3, 1)	0.128	1.68
370 883	19(4, 1)	1.4	11.8	19(4, 1)	0.515	7.42
1 479 043	19(5, 1)	5.63	51.7	18(4, 1)	2.09	30.9
5 907 203	19(5, 1)	23.2	243	19(5, 1)	8.69	147



The quality of the outer preconditioner $\gg > 0$

High quality Schur approximation $\gg > S$



Conclusions:



Conclusions:

The major outcomes of the performance study are as follows:

- (i) Large-scale coupled problems can be successfully implemented using publicly available numerical linear algebra software. Compared with highly specialized and optimized commercial software, the open source libraries, included in this study, allow to enhance the mathematical model and make it more realistic, adding features that are not straightforwardly incorporated when using commercial software.



Conclusions:

The major outcomes of the performance study are as follows:

- (i) Large-scale coupled problems can be successfully implemented using publicly available numerical linear algebra software. Compared with highly specialized and optimized commercial software, the open source libraries, included in this study, allow to enhance the mathematical model and make it more realistic, adding features that are not straightforwardly incorporated when using commercial software.
- (ii) For large enough problem sizes that fit into the memory of the GPU, the PARALUTION-GPU implementation performs noticeably faster than the other tested CPU implementations.



Conclusions, cont.

The major outcomes of the performance study are as follows:

- (iii) Open source numerical libraries successfully compete with highly efficient commercial packages in terms of overall simulation time and show better price-performance ration.



Conclusions, cont.

The major outcomes of the performance study are as follows:

- (iii) Open source numerical libraries successfully compete with highly efficient commercial packages in terms of overall simulation time and show better price-performance ration.
- (iv) None of the tested OpenMp-based CPU implementations shows satisfactory scalability. This makes it necessary to extend the performance tests using MPI, which is a subject of future work.



Thank you for your attention!
Looking forward to your comments and questions.



The quality of the Schur complement approximation - numerical observation

Text text text

«< conclude



The Schur complement approximation

\tilde{S} – the so-called *element-by-element* approach:

We notice, that for any finite element pair of spaces, chosen to approximate \mathbf{u} and p , the system matrix \mathcal{A} can be assembled from element matrices that are also of saddle point form. Namely,

$$\mathcal{A} = \sum_{i=1}^m R_i^T \mathcal{A}_i^{(e)} R_i, \quad \text{where} \quad \mathcal{A}_i^{(e)} = \begin{bmatrix} F_i^{(e)} & (B_i^{(e)})^T \\ B_i^{(e)} & -M_i^{(e)} \end{bmatrix},$$

R_i are Boolean matrices that define local-to-global mapping of the degrees of freedom and m is the number of the finite elements in the discretization mesh.

«< conclude



The Schur complement approximation, cont.

$$S_i^{(e)} = M_i^{(e)} + B_i^{(e)} \left(F_i^{(e)} + h^2 I^{(e)} \right)^{-1} (B_i^{(e)})^T,$$

$i = 1, 2, \dots, m$, where h is the characteristic size of the spatial mesh.

$$\tilde{S} = \sum_{i=1}^m \tilde{R}_i^T S_i^{(e)} \tilde{R}_i$$

«< conclude



Explaining the convergence of the inner solvers

We can show the following estimates (for the case when no convection terms are included)

$$S \leq \tilde{S} \leq \beta S$$

where β does not depend on h . «< conclude