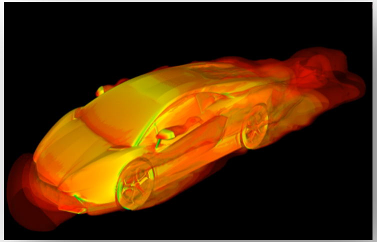
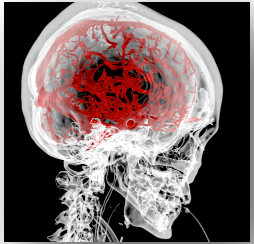
Optimizing the ESPRESO solver based on the hybrid FETI method for MIC architectures

Vít Vondrák

Lubomír Říha, Michal Merta, Ondřej Meca, Alexandros Markopoulos, Tomáš Brzobohatý





IT4Innovations
national01$#&0
supercomputing
center@#01%101

Outline

- Introduction to IT4Innovations computing infrastructure
- Total FETI and Hybrid Total FETI method
  - Intel Parallel Computing center at IT4Innovations (ESPRESO solver)
  - Hybrid FETI method
  - Acceleration of FETI method on Intel MIC
- Performance and scalability of ESPRESO
- BEM4I – Boundary element method code on Intel MIC

IT4Innovations infrastructure history

1. Anselm
94 TFLOPs system
June 2013

2. IT4I building
July 2014

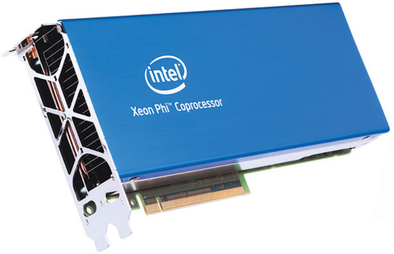
3. Salomon
2 PFLOPs system
July 2015

IT4Innovations
national01$#&0
supercomputing
center@#01%101

Anselm in numbers

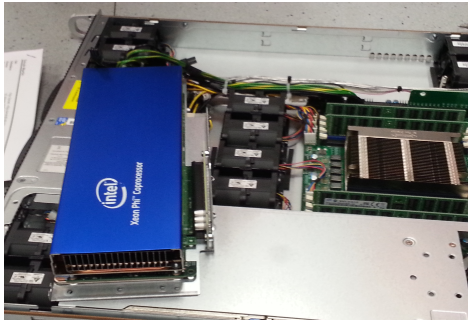
- 209 compute nodes
- 3344 Intel Sandy bridge cores
- 15136 GB RAM (64, 96, 512)
- 24 nVidia Tesla K20
- 4 Intel Xeon Phi 5110P (240 cores)

- Rpeak 94TFlop/s (94*10^12 flops per sec)
- Rmax 73TFlop/s (LINPACK)



Salomon in numbers

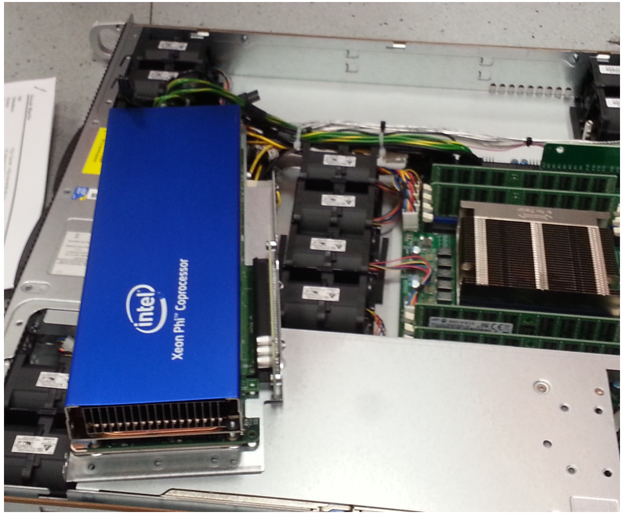
- 1008 compute nodes
- 24192 Intel Haswell cores
- 129024 GB RAM (128)
- 864 Intel Xeon Phi 7120P (52704 cores)

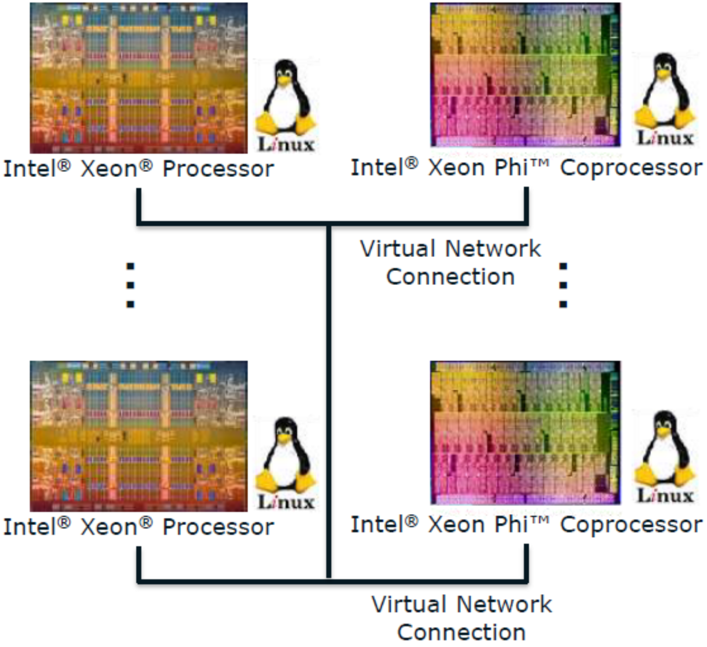


- Rpeak 2PFlop/s (2*10^15 flops per sec)
- Rmax 1.5Flop/s (LINPACK)

- **#55 in top500.org (July 2015)**
- **#20 in Europe (June 2015)**
- **(#1 Intel Xeon Phi in Europe)**

Intel Xeon Phi coprocessors in Salomon





Xeon Phi 7120P

x86 architecture, **1.2TF**

864x Intel Xeon Phi 7120P

61(244) cores, 512 bit FMA

16 GB RAM

Intel KNC Architecture

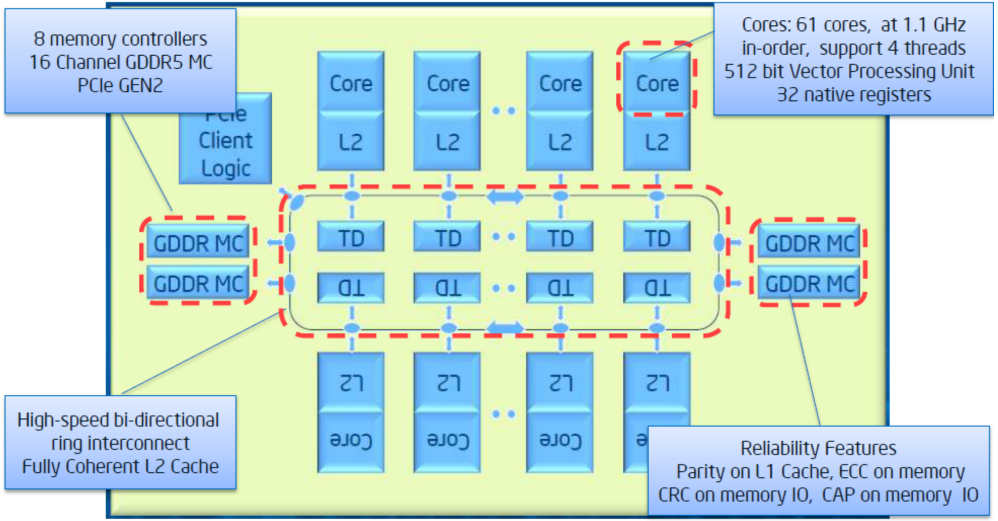
Up to 61 Cores, 244 Threads
512-bit SIMD instructions
>1TFLOPS DP-F.P. peak
Up to 16GB GDDR5 Memory

- 352 GB/s peak, but ~170 GB/s measured

PCIe 2.0 x16 - 5.0 GT/s, 16-bit
Data Cache

- L1 32KB/core
- L2 512KB/core, 30.5 MB/chip

Up to 300W TDP (card)

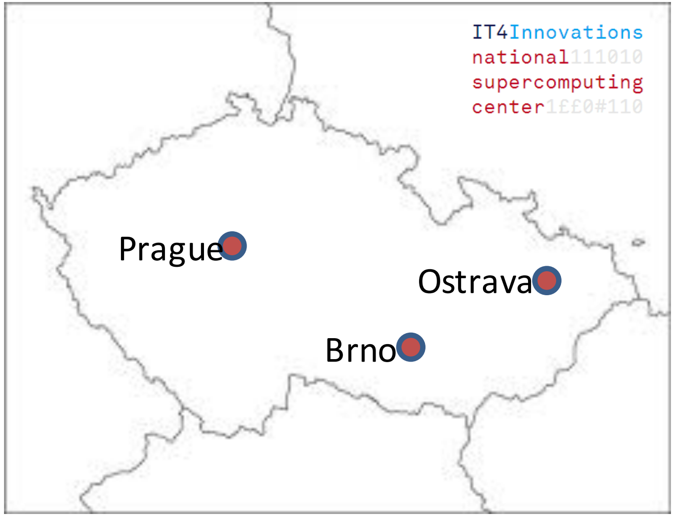


Linux* operating system

- IP addressable - coprocessor becomes a network node
- Common x86/IA
- Programming Models and SW-Tools

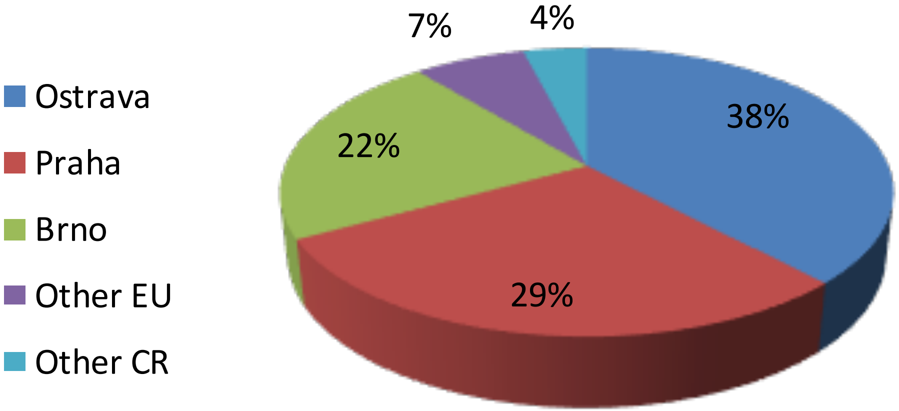
IT4Innovations
national01$#&0
supercomputing
center@#01%101

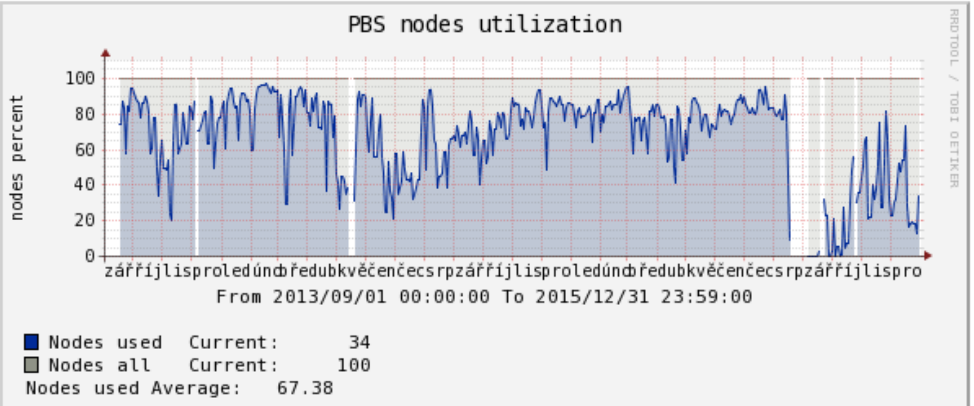
Users of IT4Innovations infrastructure



142 mil corehours distributed
584 users in 234 projects
avg. util. 70%, max util. 98%

Allocated IT4I resources since 6/2013





IT4Innovations
national
supercomputing
center

Intel® Parallel Computing Center

Chinese Academy of Science – Computer Network Information Center
University of Bristol
C-DAC
COSMOS – University of Cambridge
CSC-Finland
Dana Farber Cancer Institute – Harvard Medical School
Georgia Institute of Technology
Hartree Centre STFC (Daresbury)
Harvard - Boston Children's Hospital
Johns Hopkins University
KAUST
KISTI

CERFACS
CERN
University of Chicago
Durham University
University of Edinburgh – EPCC
University of Edinburgh
ICHEC
Imperial College London
Indiana University
Lawrence Berkeley National Laboratory
LRZ-TUM
Lomonosov Moscow State University

CINECA
University of Colorado, Boulder & NCAR
COPPETEC
ETH Zurich
Freie Universität Berlin
Georgia Institute of Technology
Indian Institute of Science
Iowa State University
IT4Innovations
Louisiana State University
Northwestern University
University of Oklahoma

ZIH - Universität Dresden
University of Oregon
Pennsylvania State University
Princeton University
Texas Advanced Computing Center (TACC)
Texas Advanced Computing Center (TACC) / Kitware
University of Tennessee

ZIB
Purdue University
Pacific Northwest National Laboratory
RWTH Aachen University
University of Tennessee - ICL
University of Tokyo - CISS
University of Tokyo - ITC

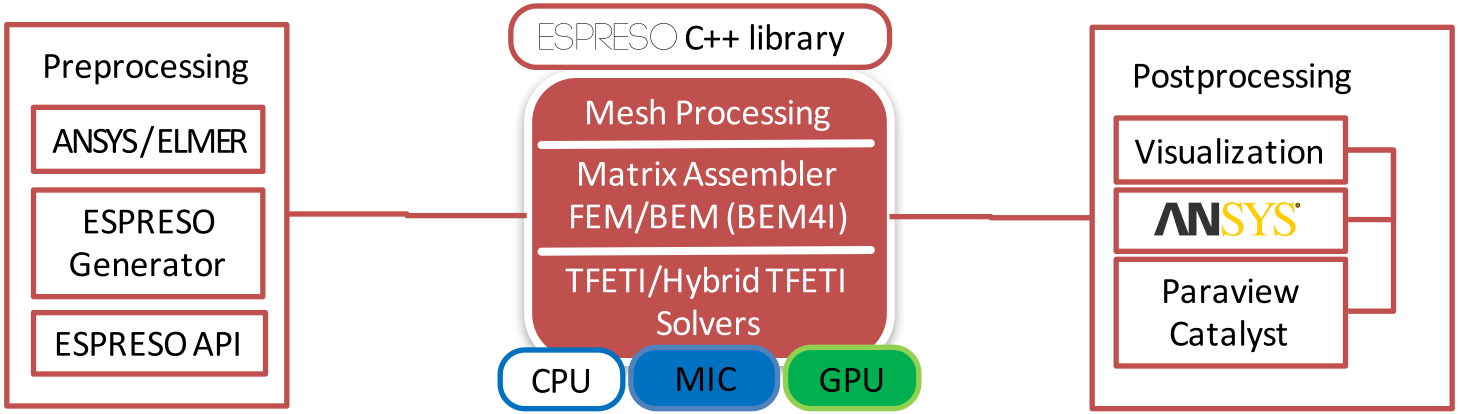
San Diego Supercomputer Center - University of California, San Diego
Sao Paulo State University (UNESP)
Stanford University
University of Tsukuba
University of Utah
University of Wisconsin - Madison

Intel® Parallel Computing Center

- WP1: Development of highly parallel algorithms and libraries
  - Algorithm development & implementation - ESPRESO
  - Algorithm optimization
- WP2: Development and support of HPC community codes
  - Development of the API
  - Plug-ins for selected community codes – OpenFOAM, ELMER

ESPRESO Parallel solver

Various input format including API
Mesh processing and Matrix Assemblers
Multi-level domain decomposition method
Support for modern multi and many-core accelerators



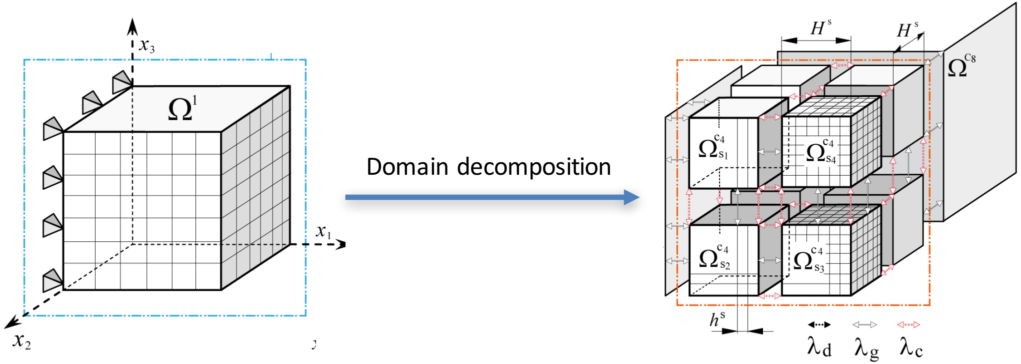
IT4Innovations national supercomputing center

E-xa-S-cale P-a-R-allel f-E-ti SO-lver

Total FETI method

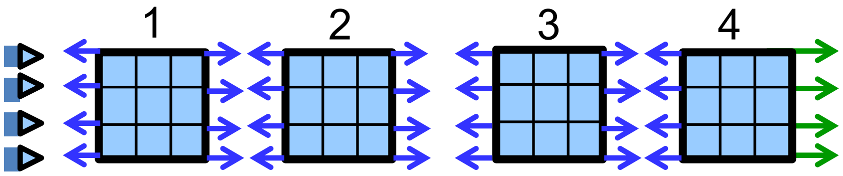
Total FETI

- Non-overlapping domain decomposition method
- Mutual continuity of primal variables between neighboring subdomains is enforced by dual variables, i.e., Lagrange multipliers obtained iteratively by the Krylov subspace methods

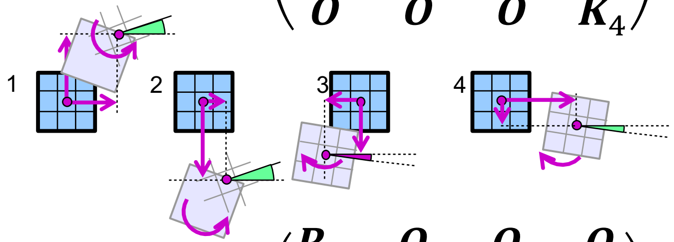


A METHOD OF FINITE-ELEMENT TEARING AND INTERCONNECTING AND ITS PARALLEL SOLUTION ALGORITHM
By: FARHAT, C; ROUX, FX; *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, Volume: 32, Issue: 6, 1991
Highly scalable parallel domain decomposition methods with an application to biomechanics
By: Klawonn, Axel; Rheinbach, Oliver; *ZAMM-ZEITSCHRIFT FUR ANGEWANDTE MATHEMATIK UND MECHANIK*, Volume: 90, Issue: 1, 2010
Total FETI domain decomposition method and its massively parallel implementation
By: Kozubek, T.; Vondrak, V.; Mensik, M.; et al.; *ADVANCES IN ENGINEERING SOFTWARE* Volume: 60-61, 2013

Total FETI method



$$K = \begin{pmatrix} K_1 & O & O & O \\ O & K_2 & O & O \\ O & O & K_3 & O \\ O & O & O & K_4 \end{pmatrix}$$



$$R = \begin{pmatrix} R_1 & O & O & O \\ O & R_2 & O & O \\ O & O & R_3 & O \\ O & O & O & R_4 \end{pmatrix}$$

$$\min \frac{1}{2} \boldsymbol{u}^T \boldsymbol{K} \boldsymbol{u} - \boldsymbol{u}^T \boldsymbol{f} \;\; s.t. \;\; \boldsymbol{B}\boldsymbol{u} = \boldsymbol{c}$$

$$\min \frac{1}{2} \lambda^T \boldsymbol{F} \lambda - \lambda^T \boldsymbol{d} \;\; s.t. \;\; \boldsymbol{G}\lambda = \boldsymbol{o}$$

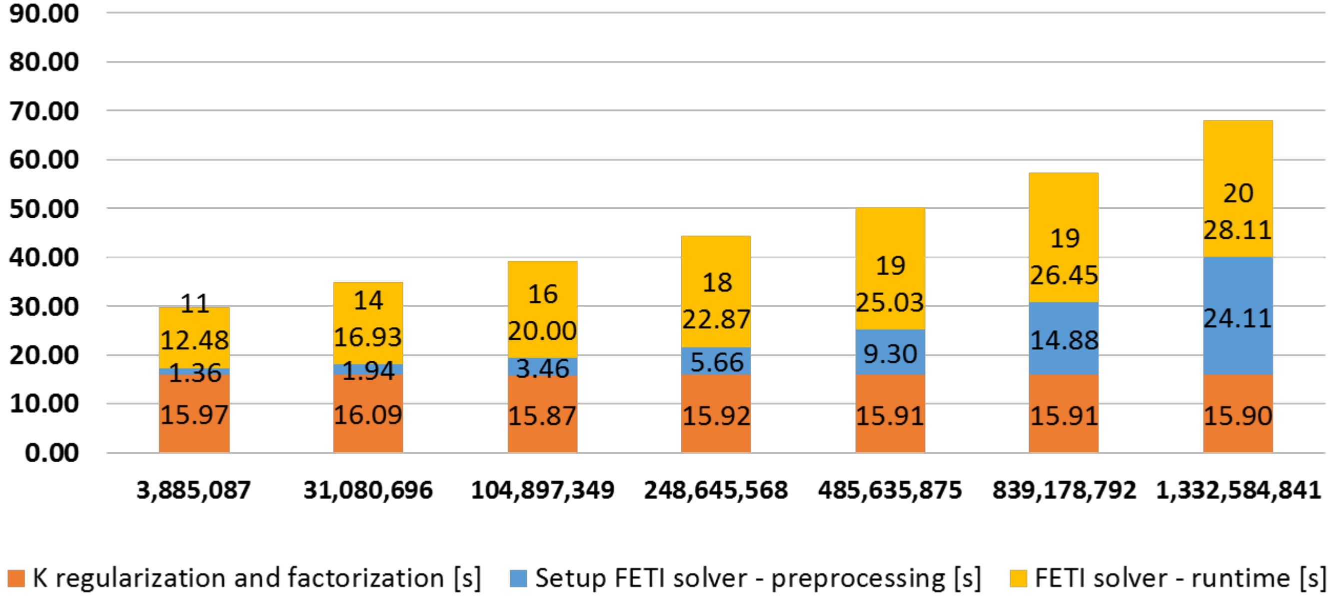
$$\boldsymbol{F} = \boldsymbol{B}\boldsymbol{K}^{+}\boldsymbol{B}^{\mathrm{T}}, \boldsymbol{G}^{T} = -\boldsymbol{B}\boldsymbol{R},$$
$$\boldsymbol{d} = \boldsymbol{B}\boldsymbol{K}^{+}\boldsymbol{f} - \boldsymbol{c}$$

Projected Conjugate Gradient method

$$\boldsymbol{g}_{k+1} = \boldsymbol{g}_k + \alpha_k \boldsymbol{F}\boldsymbol{p}_k \qquad \boldsymbol{g}_{k+1}^{proj} = \boldsymbol{P}\boldsymbol{g}_{k+1}$$

$$\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{G}^{T}(\boldsymbol{G}\boldsymbol{G}^{T})^{-1}\boldsymbol{G}$$

Total FETI: Weak scalability



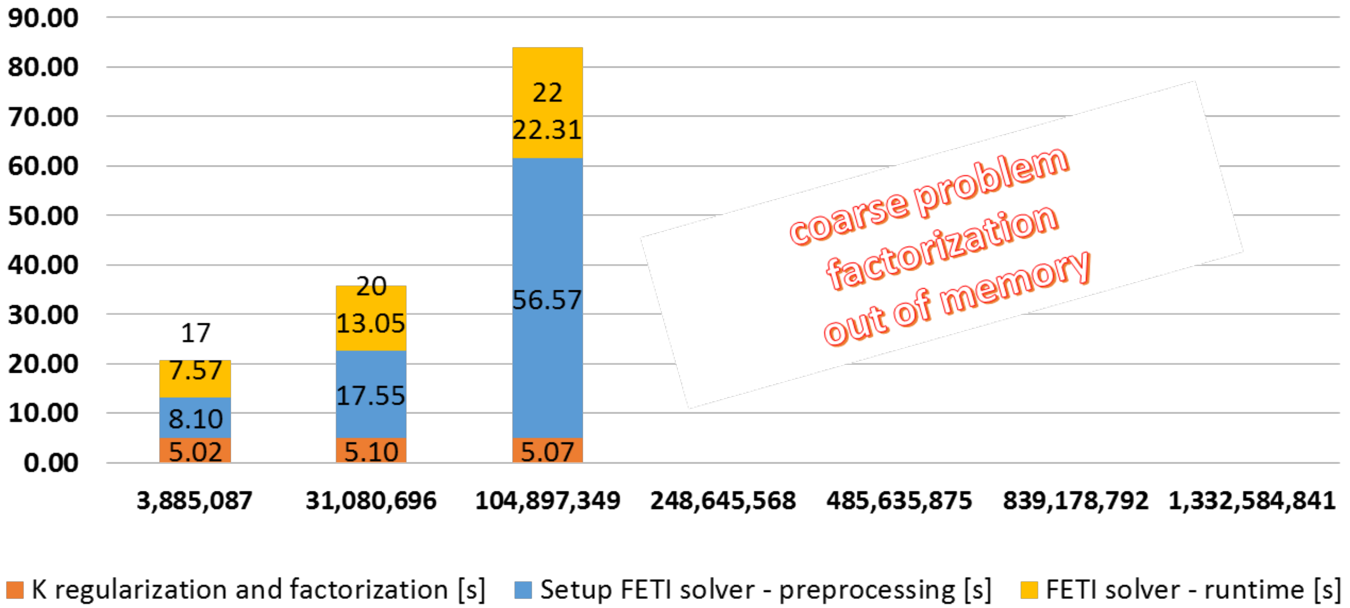
24^3 - domain size

4^3 subdomains per node – processed in parallel on 24 cores using Cilk++

Test ran on : 1, 8, 27, 64, 125, 216 and 343 nodes (1^3 … 7^3)– each 24 cores

IT4Innovations
national01$#&0
supercomputing
center@#01%101

Reaching Total FETI limits



12^3 - domain size

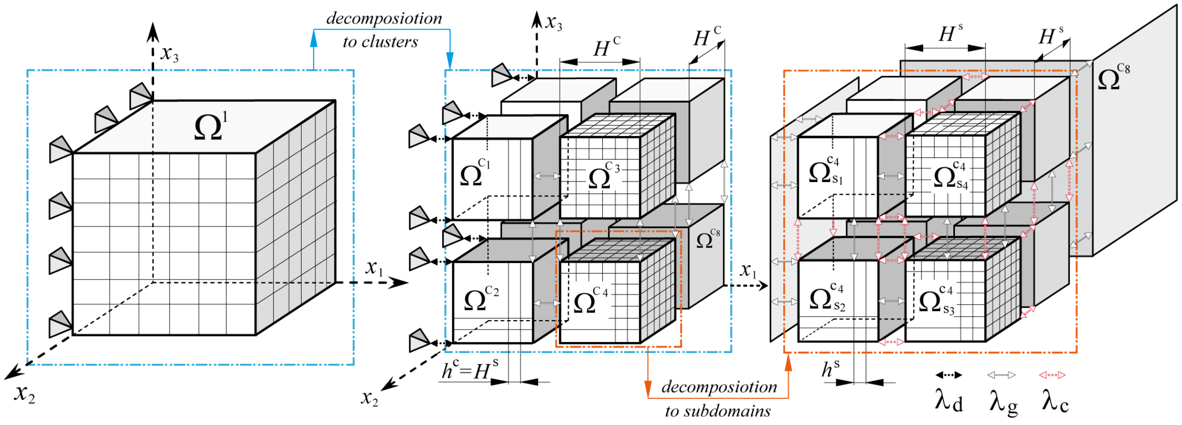
9^3 subdomains per node – processed in parallel on 24 cores using Cilk++

Test ran on : 1, 8, 27 nodes – each 24 cores

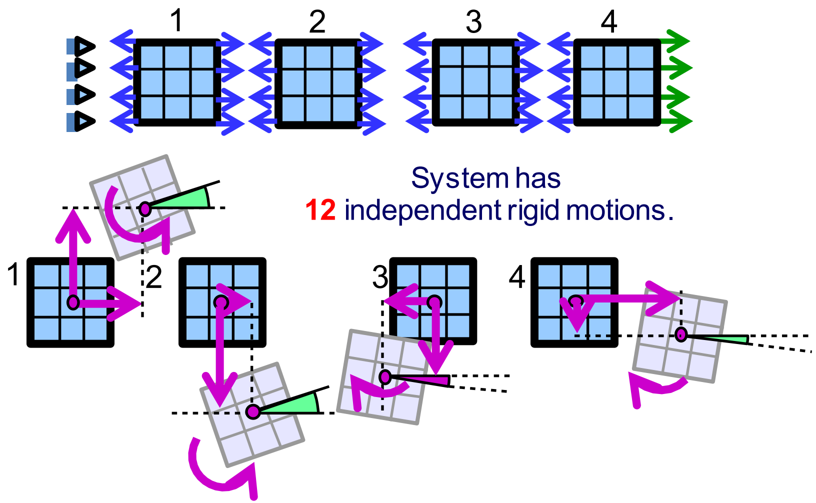
Total FETI and Hybrid Total FETI Methods

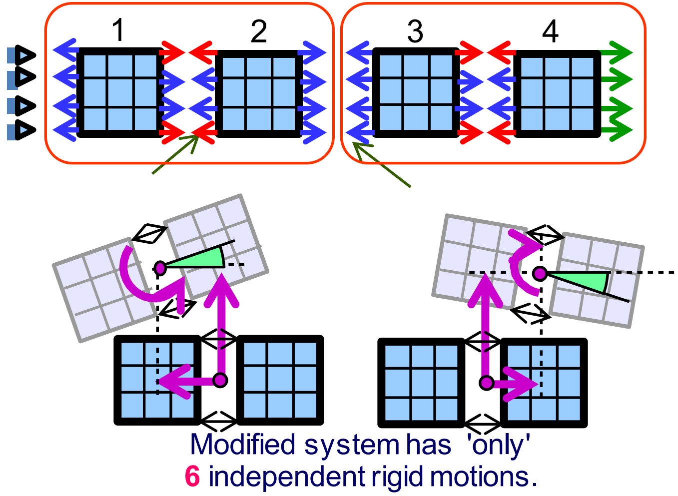
Hybrid Total FETI

- Non-overlapping domain decomposition method
- Subdomains grouped into non-overlapping clusters
- Mutual continuity of primal variables between neighboring subdomains is enforced by dual variables, i.e., Lagrange multipliers obtained iteratively by the Krylov subspace methods



Hybrid Total FETI solver – Why is it scalable?





Total FETI (2D case)
Problem decomposed into 4 subdomains generates **coarse problem matrix (GG^T)** with dimension:

3 *(number of SUBDOMAINS) = 12

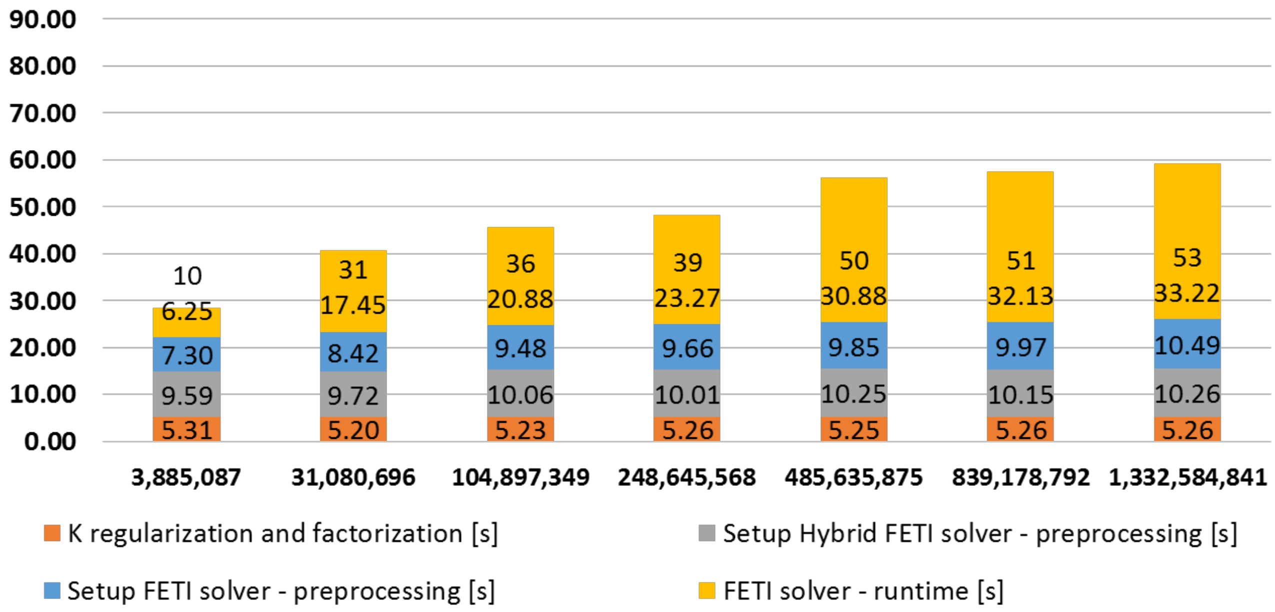
Hybrid Total FETI (2D case)
Beam decomposed into 2 clusters (*each consists of N subdomains*) generates **coarse problem matrix (GG^T)** with dimension

3 *(number of CLUSTERS) = 6

Number of clusters = number of nodes

IT4Innovations
national01$#&0
supercomputing
center@#01%101

HTFETI: Weak scalability



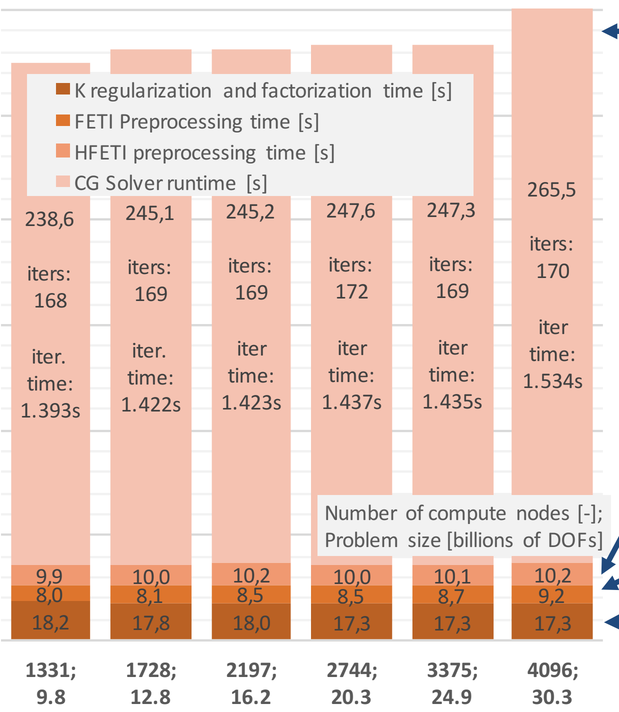
12^3 - domain size

9^3 subdomains per cluster - HFETI corners in corners – no corners on edges

Test ran on : 1, 8, 27, 64, 125, 216 and 343 nodes (1^3 … 7^3) – each 24 cores

IT4Innovations national01$#&0 supercomputing center@#01%101

Main blocks of the Hybrid Total FETI solver



Iterative solver step (*CG solver runtime*)
- the most time consuming part
- Runtime depends on number of iterations = how well the problem is conditioned
- Essential kernels will be discussed later

Preprocessing step – all operations executed one time
- Coarse problem matrix assembling and calculation of its inverse (*FETI preprocessing*)
  - Assembling – **nearest neighbor MPI** communication required
  - Inverse matrix calculation – **1 MPI broadcast** – **local** factorization and solve (PARDISO)
- Additional preprocessing inside clusters (*HTFETI preprocessing*)
  - **Several local** factorizations and solves using PARDISO
- Stiffness matrix factorization (*K factorization*)
  - **Local** factorizations and solve using PARDISO

IT4Innovations
national01$#&0
supercomputing
center@#01%101

Computing kernels of FETI method

Projected Conjugate Gradient in FETI

$r_0 := b - Ax_0;\ u_0 := M^{-1} r_0;\ p_0 := u_0$
for $i = 0, \ldots, m-1$ **do**
  $s := Ap_i$
  $\alpha := \langle r_i, u_i \rangle \,/\, \langle s, p_i \rangle$
  $x_{i+1} := x_i + \alpha p_i$
  $r_{i+1} := r_i - \alpha s$
  $u_{i+1} := M^{-1} r_{i+1}$
  $\beta := \langle r_{i+1}, u_{i+1} \rangle \,/\, \langle r_i, u_i \rangle$
  $p_{i+1} := u_{i+1} + \beta p_i$
end for

90 – 95% of runtime spent in Ap_i

Pre-processing – K factorization

1.) $x = B_1^T \cdot \lambda$ - **SpMV**

2.) $y = K^{-1} \cdot x$ - solve on CPU

3.) $\lambda = B_1 \cdot y$ - **SpMV**

4.) stencil data exchange in λ

- MPI – Send and Recv
- OpenMP – shared mem. vec

Sparse Matrix-Vector product

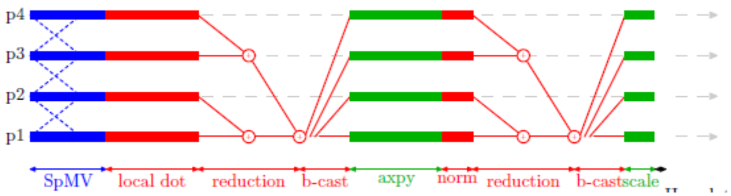
- Only communication with neighbors
- Good scaling

Dot-product

- Global communication
- Scales as $\log(P)$

Scalar vector multiplication, vector-vector addition

- No communication



IT4Innovations
national01$#&0
supercomputing
center@#01%101

How to Accelerate FETI methods with Xeon Phi
Approach 1 – Using Sparse Matrices

90 – 95% of runtime spent in Ap_i

Projected Conjugate Gradient in FETI

$r_0 := b - Ax_0;\ u_0 := M^{-1}r_0;\ p_0 := u_0$
for $i = 0, \ldots, m-1$ **do**
  $s := Ap_i$
  $\alpha := \langle r_i, u_i \rangle \,/\, \langle s, p_i \rangle$
  $x_{i+1} := x_i + \alpha p_i$
  $r_{i+1} := r_i - \alpha s$
  $u_{i+1} := M^{-1} r_{i+1}$
  $\beta := \langle r_{i+1}, u_{i+1} \rangle \,/\, \langle r_i, u_i \rangle$
  $p_{i+1} := u_{i+1} + \beta p_i$
end for

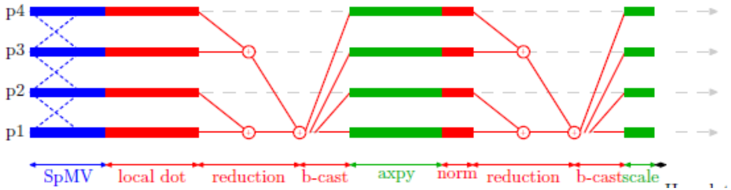
Pre-processing - K→MIC - K factorization on MIC

1.) $x = B_1^T \cdot \lambda$ - **SpMV on CPU**
2.) $x \rightarrow$ MIC - **PCIe transfer from CPU**
3.) $y = K^{-1} \cdot x$ - solve on MIC
4.) $y \leftarrow$ MIC - **PCIe transfer to CPU**
5.) $\lambda = B_1 \cdot y$ - **SpMV on CPU**
6.) stencil data exchange in λ
- MPI – Send and Recv
- OpenMP – shared mem. vec

Sparse Matrix-Vector product
- Only communication with neighbors
- Good scaling

Dot-product
- Global communication
- Scales as $\log(P)$

Scalar vector multiplication, vector-vector addition
- No communication



How to Accelerate FETI methods with Xeon Phi Approach 1 – Using Sparse Matrices

	Haswell 1x E5-2680v3 CPU		MIC 1x Intel Xeon Phi 7120p accelerator					
	12 cores		60 threads		120 threads		240 threads	
dim.	Fact. [s]	Solve [s]	Fact. [s]	Solve [s]	Fact. [s]	Solve [s]	Fact. [s]	Solve [s]
2187 x 2187 – 2800 domains	***6.9***	**17.0**	***44.8***	46.9	57.5	28.3	68.4	**18.9**
6591 x 6591 – 700 domains	***7.2***	**18.7**	***24.5***	41.4	22.9	26.2	27.5	**17.1**
10125 x 10125 – 360 domains	***6.9***	**17.4**	20.2	36.9	***19.3***	21.7	21.9	**17.3**
12288 x 12288 – 300 domains	***7.8***	**18.5**	22.2	40.0	***19.8***	26.1	27.3	**20.5**

Factorization (preprocessing) - **7120p Xeon Phi is approximately 3-4x slower than CPU**

Solver (iterative solver) - **7120p Xeon Phi is as fast as CPU**

With this approach preprocessing time remains the same, but iterative solver processing time is reduced by 50% by Intel Xeon Phi.

How to Accelerate FETI methods with Xeon Phi
Approach 2 – Using Dense Matrices (Schur Complement)

Projected Conjugate Gradient in FETI

```
r_0 := b − Ax_0; u_0 := M^{-1} r_0; p_0 := u_0
for i = 0, ..., m − 1 do
    s := Ap_i
    α := ⟨r_i, u_i⟩ / ⟨s, p_i⟩
    x_{i+1} := x_i + α p_i
    r_{i+1} := r_i − α s
    u_{i+1} := M^{-1} r_{i+1}
    β := ⟨r_{i+1}, u_{i+1}⟩ / ⟨r_i, u_i⟩
    p_{i+1} := u_{i+1} + β p_i
end for
```

90 – 95% of runtime spent in Ap_i

Pre-processing - $S_c = B_1 K^{-1} B_1^T \rightarrow$ MIC

1.) $\lambda \rightarrow$ MIC - **PCIe transfer from CPU**

2.) $\lambda = S_c \cdot \lambda$ - **DGEMV, DSYMV on MIC**

3.) $\lambda \leftarrow$ MIC - **PCIe transfer to CPU**

4.) stencil data exchange in λ

- MPI – Send and Recv
- OpenMP – shared mem. vec

Requires algorithmic changes in the FETI solver in both preprocessing and iterative solver steps

Key features this approach:

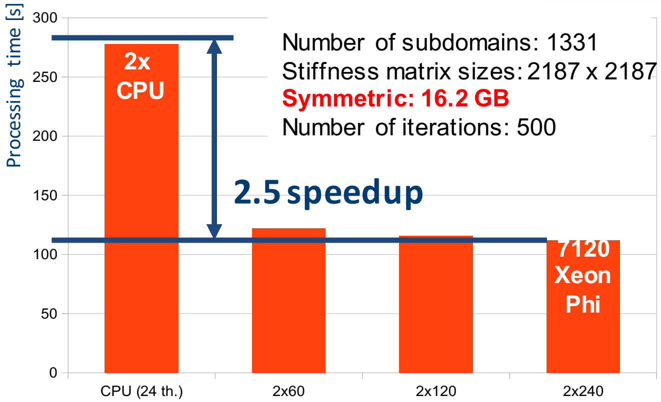
- Increases preprocessing time – Schur Complement (SC) computation for each subdomain
- Reduce iterative solver time – single iteration time is reduced

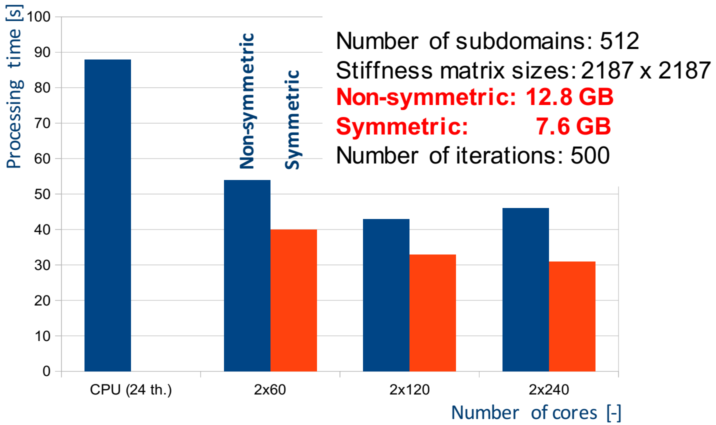
IT4Innovations national supercomputing center

Schur Complement Computation on CPU and Xeon Phi

Comparison of SC computation using PARDISO SC and MKL

	Haswell 1x E5-2680v3 CPU	MIC 1x Intel Xeon Phi 7120p accelerator	
	12 cores	60x3 threads	60x4 threads
Domain size x number of domains	SC [s]	SC [s]	SC [s]
2187 x 2187 - 1500 subdomains	**26.7**	**70.4**	71.1
6591 x 6591 - 250 subdomains	**28.5**	80.5	**78.4**
12288 x 10125 - x 60 subdomains	**29.5**	**59.5**	88.5





Key features of this approach:

- Increases preprocessing time – Schur Complement (SC) computation for each subdomain
- Reduce iterative solver time – single iteration time is reduced

ESPRESO on Salomon

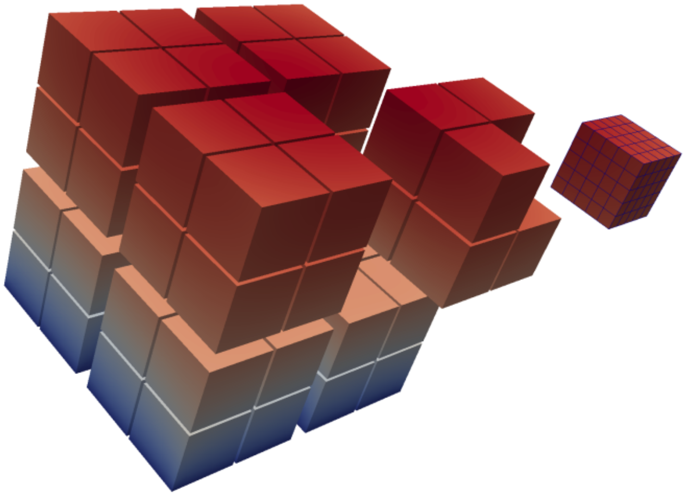
2,016	Intel Xeon E5-2680v3, 2.5GHz, 12cores
864	Intel Xeon Phi 7120P, 61cores, 16GB RAM

ESPRESO Problem Generator

Massively parallel Benchmark Generator
- designed to perform large scale tests – tested up to 120 billion unknowns
- problem with all matrix objects generated in seconds

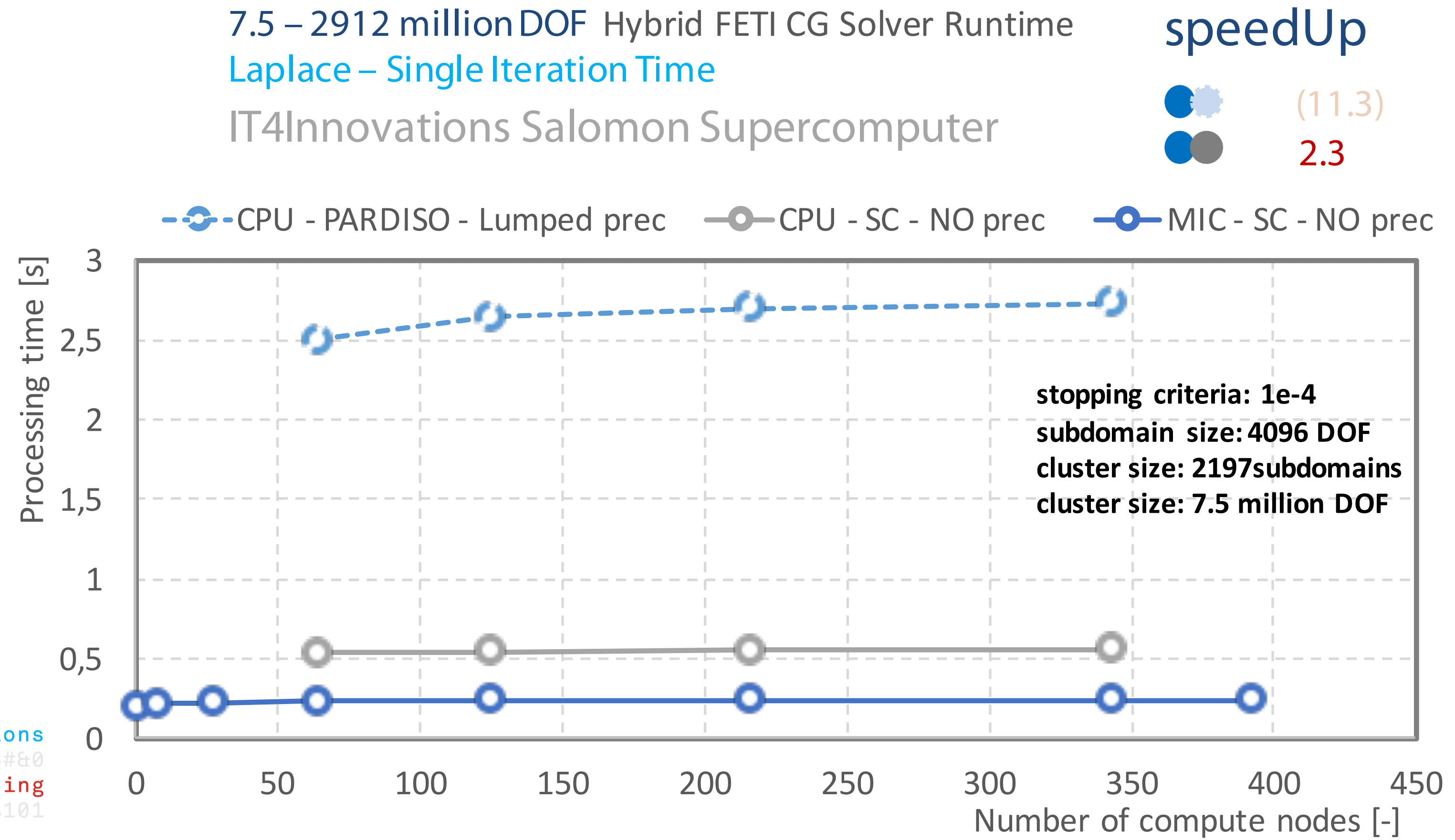
Model problems
- Cube
- Sphere

Physics
- Laplace equation
- Linear Elasticity



CPU vs MIC solver

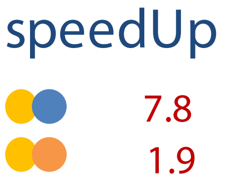
7.5 – 2912 million DOF Hybrid FETI CG Solver Runtime
Laplace – Single Iteration Time
IT4Innovations Salomon Supercomputer

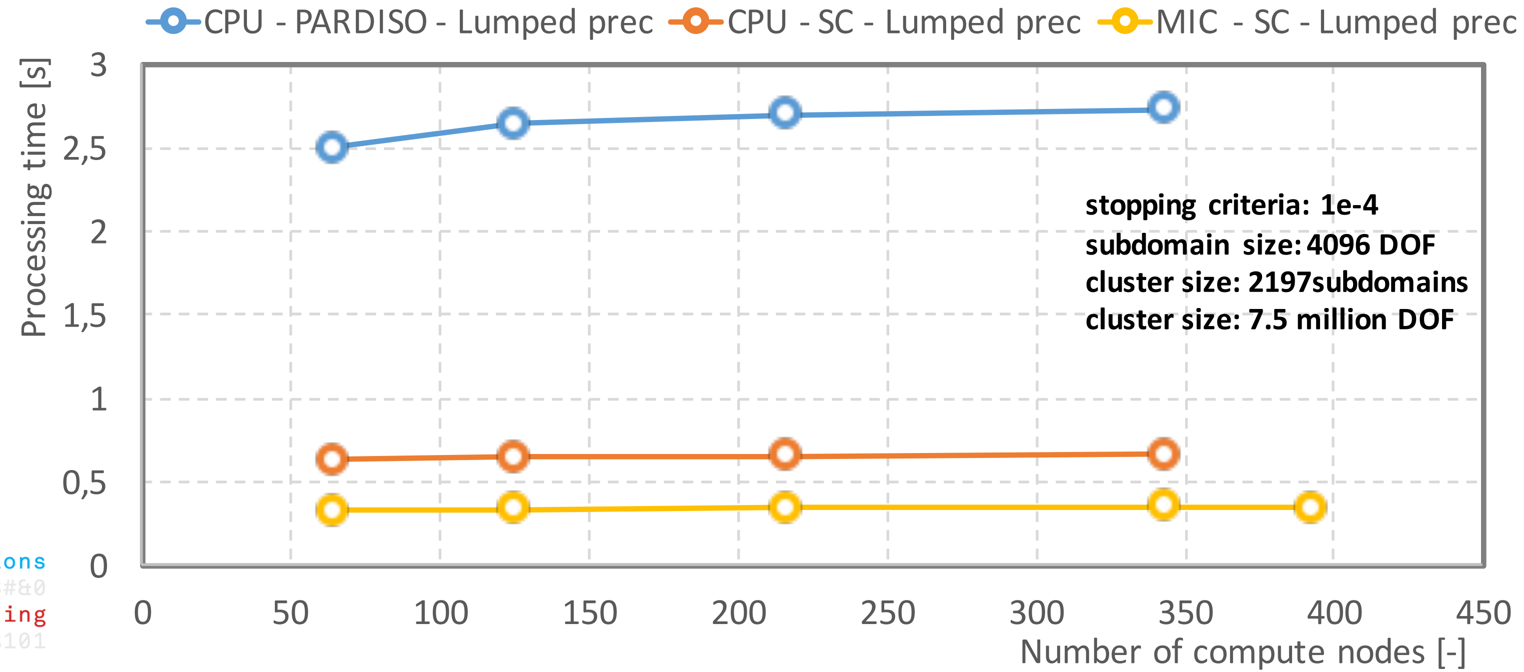


IT4Innovations
national01$#&0
supercomputing
center@#01%101

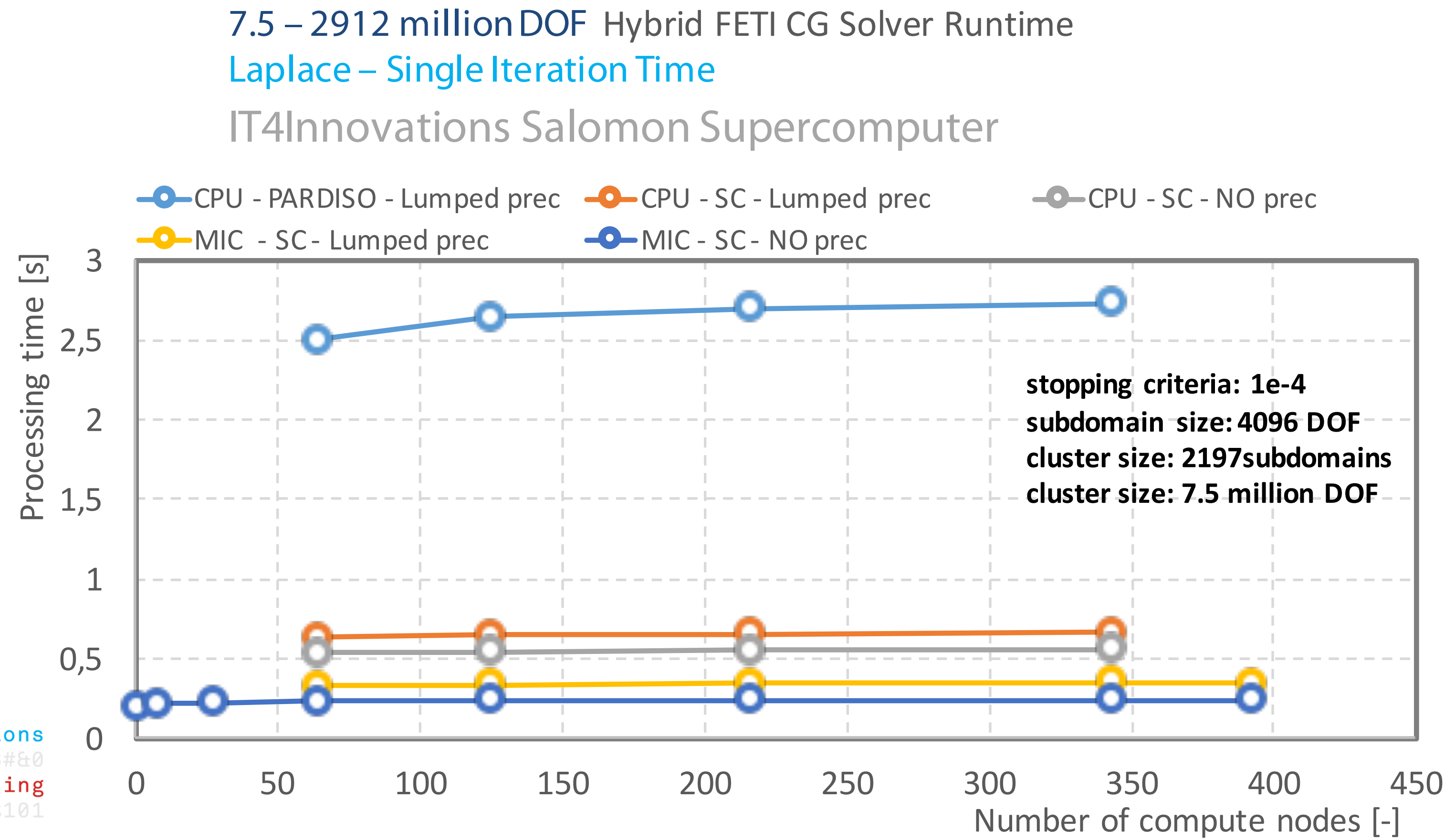
CPU vs MIC solver

7.5 – 2912 million DOF Hybrid FETI CG Solver Runtime
Laplace – Single Iteration Time
IT4Innovations Salomon Supercomputer





CPU vs MIC solver

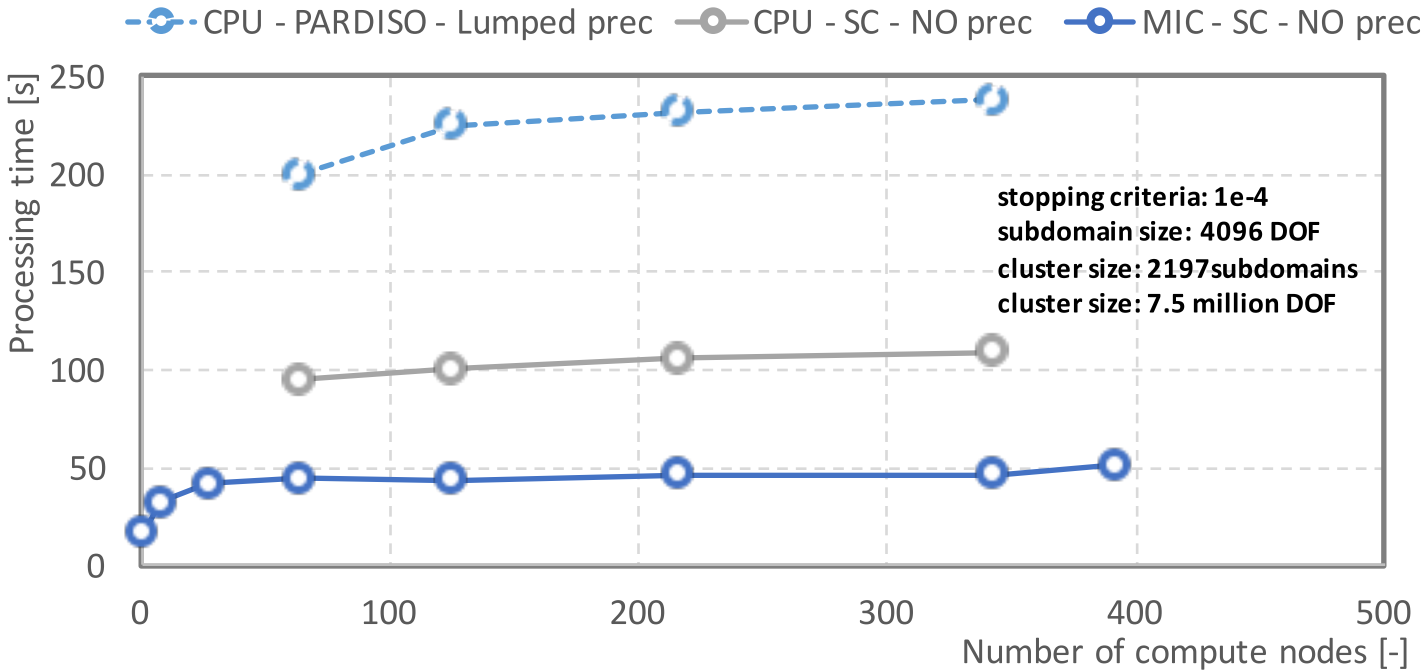


IT4Innovations
national01$#&0
supercomputing
center@#01%101

CPU vs MIC solver

7.5 – 2912 million DOF Hybrid FETI CG Solver Runtime
Laplace – CG Solver Runtime w/o Precon.
IT4Innovations Salomon Supercomputer

speedUp
(11.3)
2.3



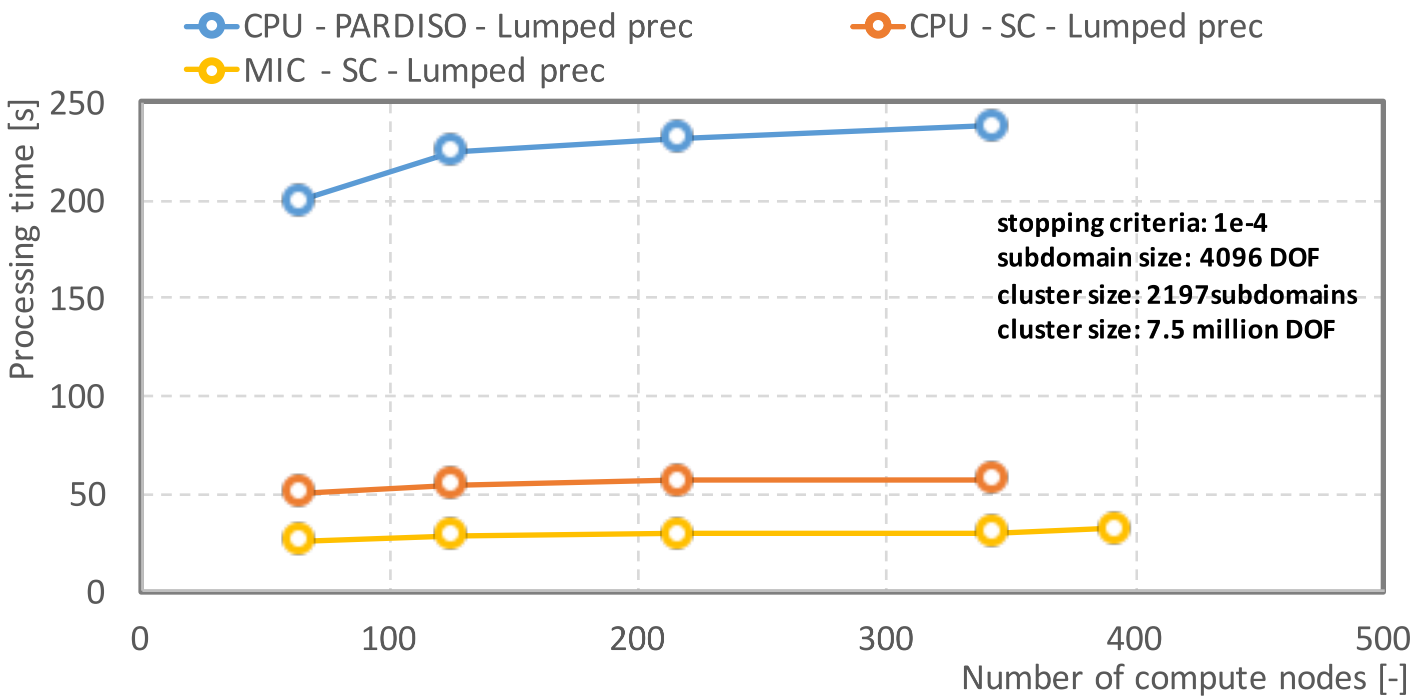
CPU vs MIC solver

7.5 – 2912 million DOF Hybrid FETI CG Solver Runtime
Laplace – CG Solver Runtime w. Lumped Precon.
IT4Innovations Salomon Supercomputer

speedUp

7.8

1.9



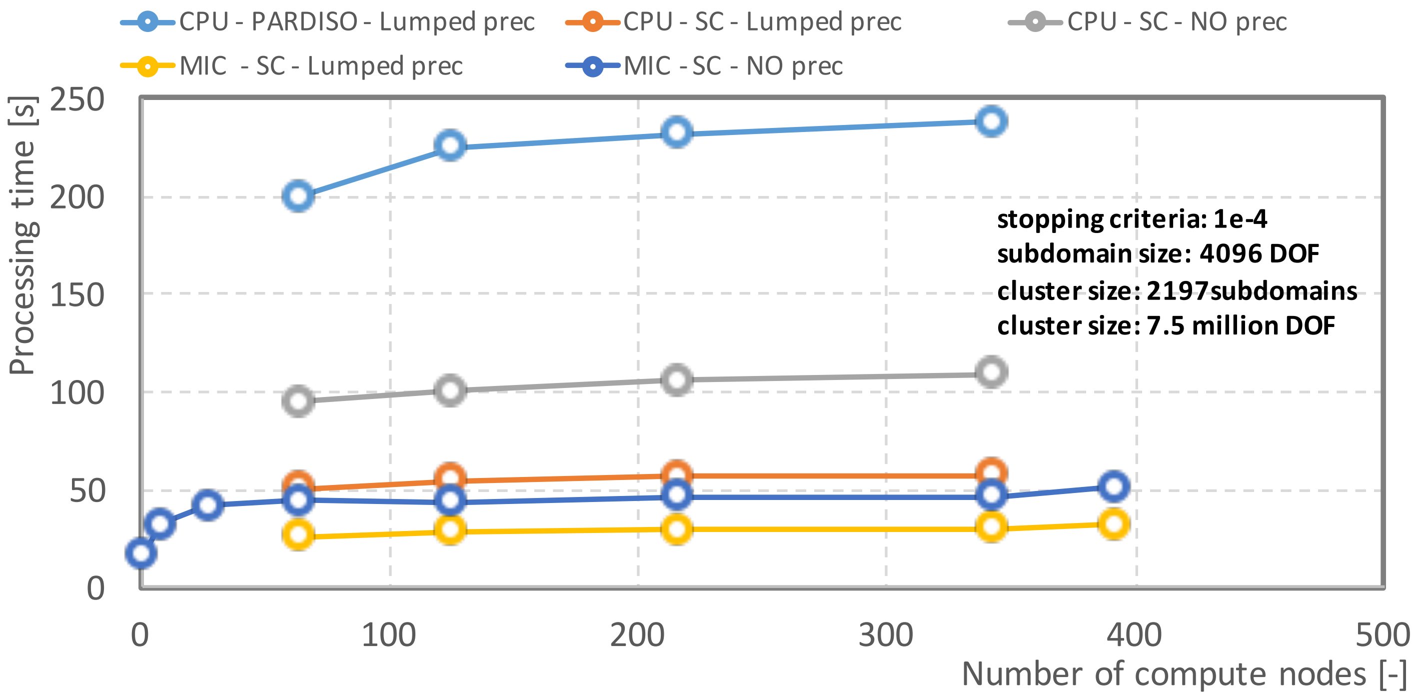
IT4Innovations
national01$#&0
supercomputing
center@#01%101

CPU vs MIC solver

7.5 – 2912 million DOF Hybrid FETI CG Solver Runtime

Laplace – CG Solver Runtime w/o Preprocessing

IT4Innovations Salomon Supercomputer



IT4Innovations
national01$#&0
supercomputing
center@#01%101

ESPRESO on TITAN

OAK RIDGE National Laboratory

TITAN 3rd in TOP500 LIST

18,688 AMD Opteron 6274 16-core CPUs
18,688 Nvidia Tesla K20X GPUs

2.7 million core hours dedicated to:

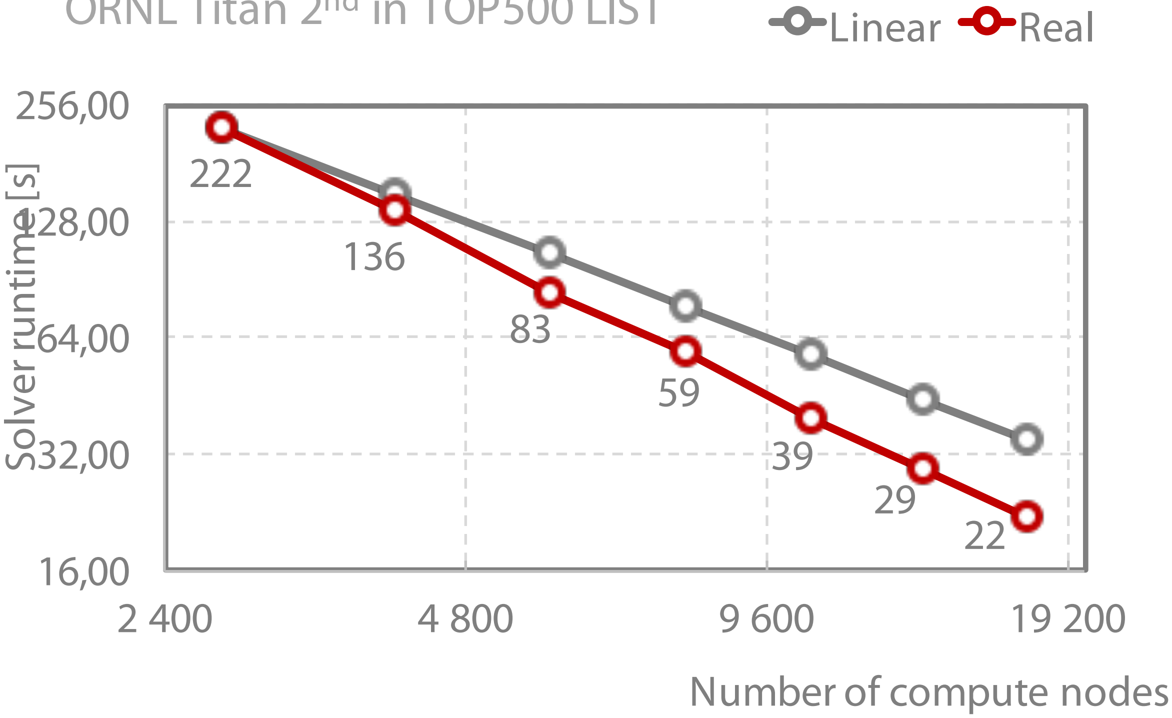
- scalability optimization of ESPRESO
- optimization of GPU accelerated version for large scale problems

Strong Scalability Test

20 billion DOF on up to 17 576 Compute Nodes (281 216 cores)

Heat transfer (Laplace equation)

ORNL Titan 2nd in TOP500 LIST



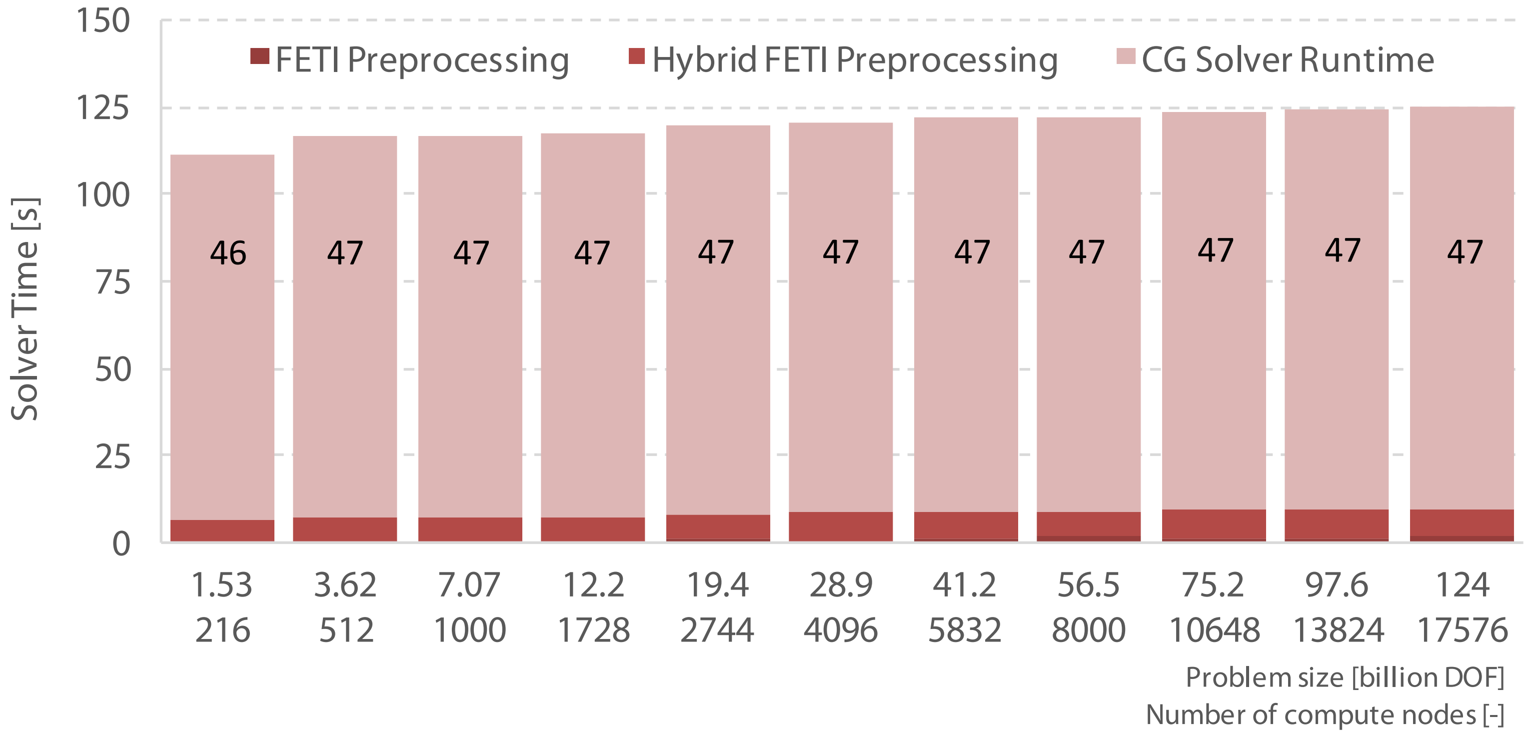
IT4Innovations
national01$#&0
supercomputing
center@#01%101

Weak Scalability Test

Up to 124 billion DOF on 17576 Compute Nodes (281 216 cores)

Heat transfer (Laplace equation)

ORNL Titan 2nd in TOP500 LIST



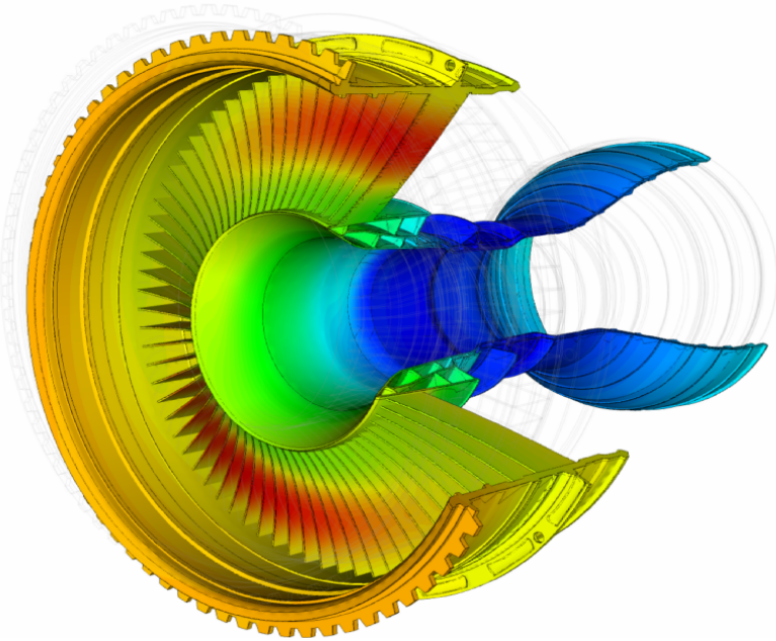
IT4Innovations national supercomputing center

Strong Scalability Test

300 million unknown - ANSYS Workbench real world problem

Linear elasticity

IT4Innovations – SALOMON Supercomputer

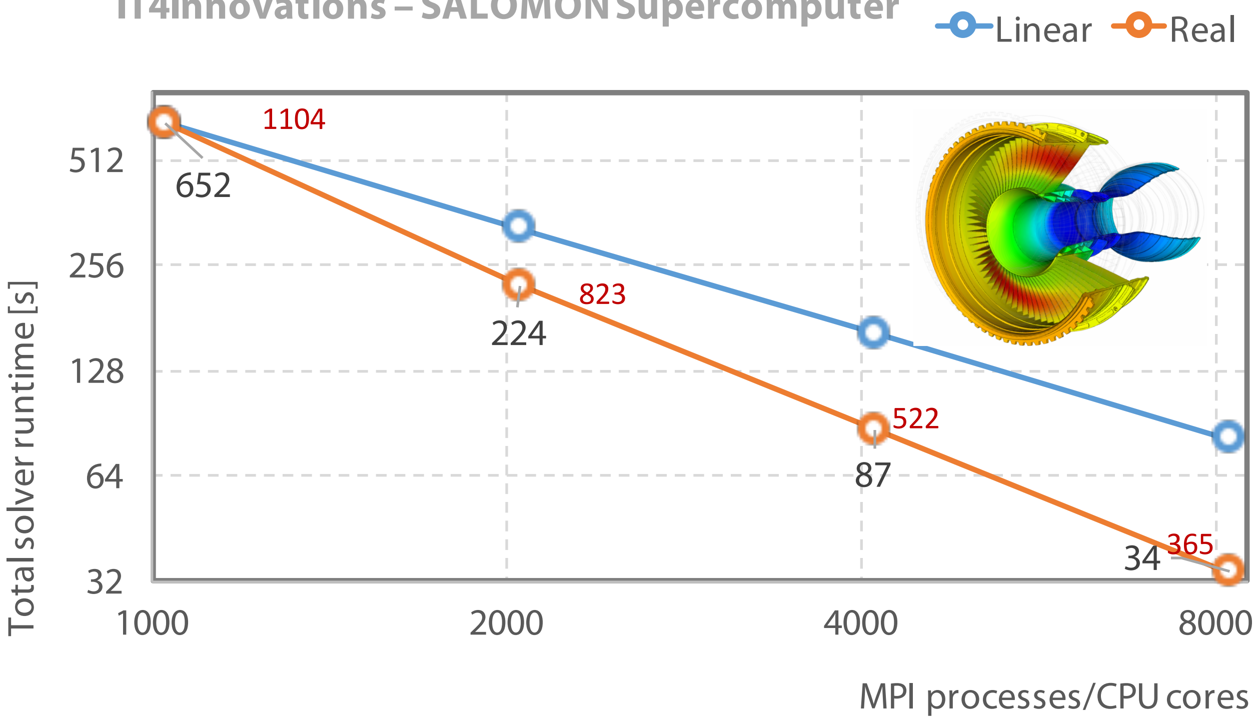


Strong Scalability Test

300 million unknown - ANSYS Workbench

Linear elasticity

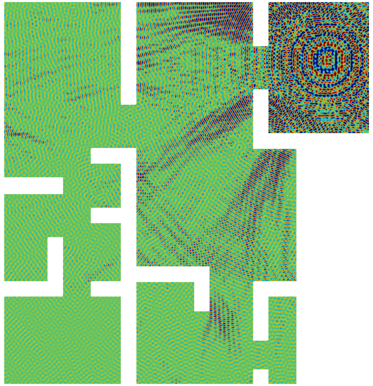
IT4Innovations – SALOMON Supercomputer

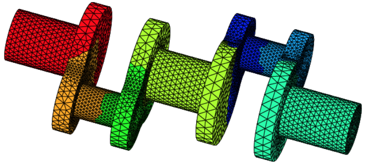


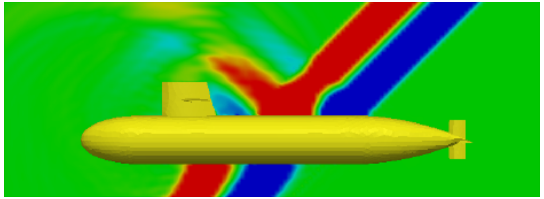
IT4Innovations
national01$#&0
supercomputing
center@#01%101

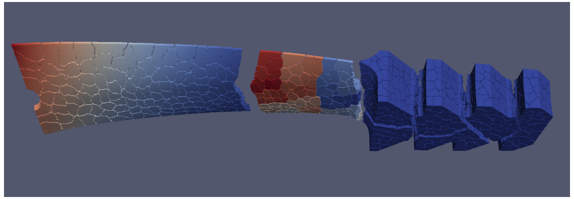
Boundary element library: BEM4I

- Developed at IT4Innovations NSC
- Reduces problem to the boundary of a computational domain
- Suitable mainly for problems on unbounded domains or shape optimization
- Heat transfer, wave scattering, linear elasticity



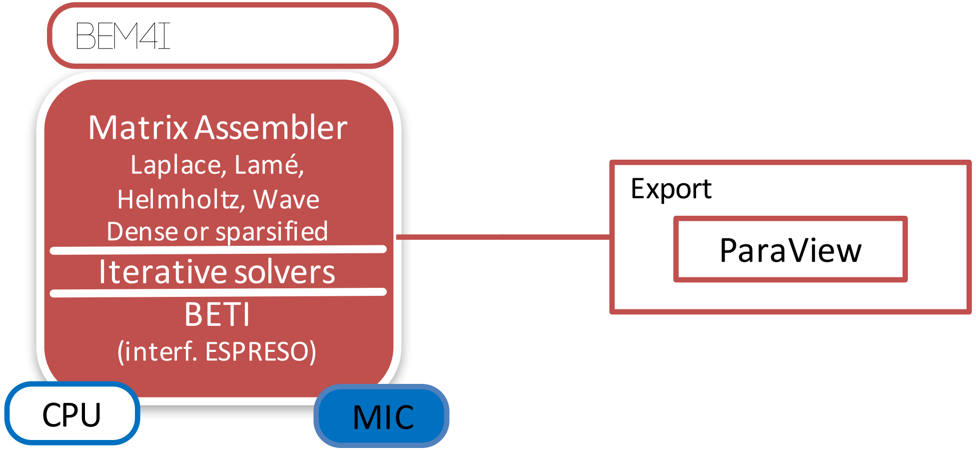






Boundary element library: BEM4I

- Templated C++ library
- SIMD vectorization (using Intel's pragmas or Vc library)
- OpenMP and MPI parallelization
- Intel Xeon Phi acceleration



IT4Innovations
national01$#&0
supercomputing
center@#01%101

Discretization

- Galerkin method for discretization of the boundary integral equation
- Matrix formulation

$$\begin{cases} \text{Find } \boldsymbol{t}_h \in \mathbb{R}^N \text{ such that} \\ V\boldsymbol{t}_h = (\frac{1}{2}M_h + K_h)\boldsymbol{g}_h. \end{cases}$$

- System matrices (single layer and double layer matrix)

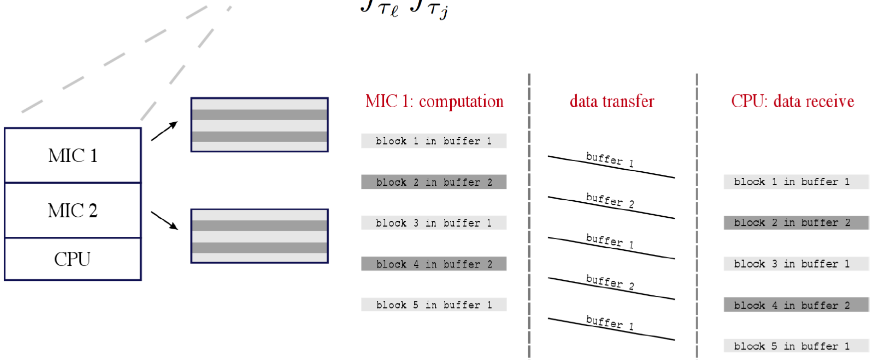
$$V_h[i,j] := \frac{1}{4\pi}\int_{\tau_i}\int_{\tau_j}\frac{1}{\|\boldsymbol{x}-\boldsymbol{y}\|}\,\mathrm{d}s_{\boldsymbol{y}}\,\mathrm{d}s_{\boldsymbol{x}} \qquad K_h[k,l] := \frac{1}{4\pi}\int_{\tau_k}\int_{\Gamma}\frac{(\boldsymbol{x}-\boldsymbol{y},\boldsymbol{n}_{\boldsymbol{y}})}{\|\boldsymbol{x}-\boldsymbol{y}\|^3}\varphi_l(\boldsymbol{y})\,\mathrm{d}s_{\boldsymbol{y}}\,\mathrm{d}s_{\boldsymbol{x}}$$

- System matrix assembly - quadratic complexity
  - Computationally most demanding part of BEM
  - Parallelized by OpenMP, MPI
  - Accelerated by Intel Xeon Phi coprocessor
  - Possible utilization of Fast BEM methods (sparsification)

Acceleration of the system matrix assembly

- Matrix split into parts for host CPUs and coprocessors
  - Load balanced according to theoretical perfomance of host CPUs and MICs
- Coprocessor parts further split into smaller submatrices
  - To fit into coprocessor memory
  - To overlap communication by computation (double buffering)
- Computation accelerated using offload mode of the coprocessor
- On the coprocessor the code is parallelized using ordinary OpenMP pragmas
- Vectorization of the code (e.g. using #pragma simd) and scalability up to hundreds of threads necessary to obtain good performance on the coprocessor

$$\mathsf{V}_h[\ell, j] := \int_{\tau_\ell} \int_{\tau_j} v(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{y}} \, \mathrm{d}\boldsymbol{s}_{\boldsymbol{x}}$$



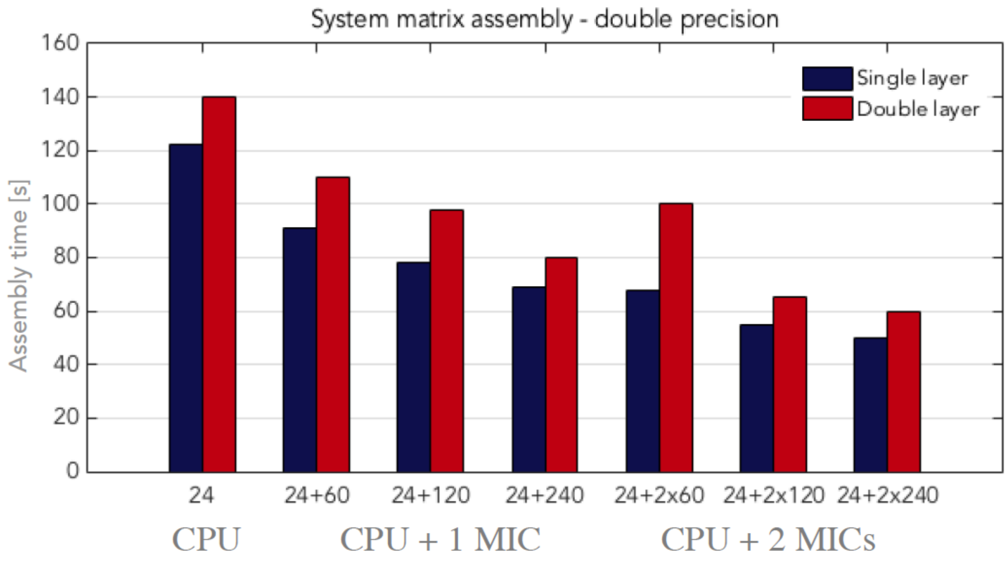
matrix distribution among CPU and MICs

double buffering to overlap communication with computation

Acceleration of the system matrix assembly

Laplace equation

- 81920 surface elements
- Maximum speedup using two cards approx. 2.5



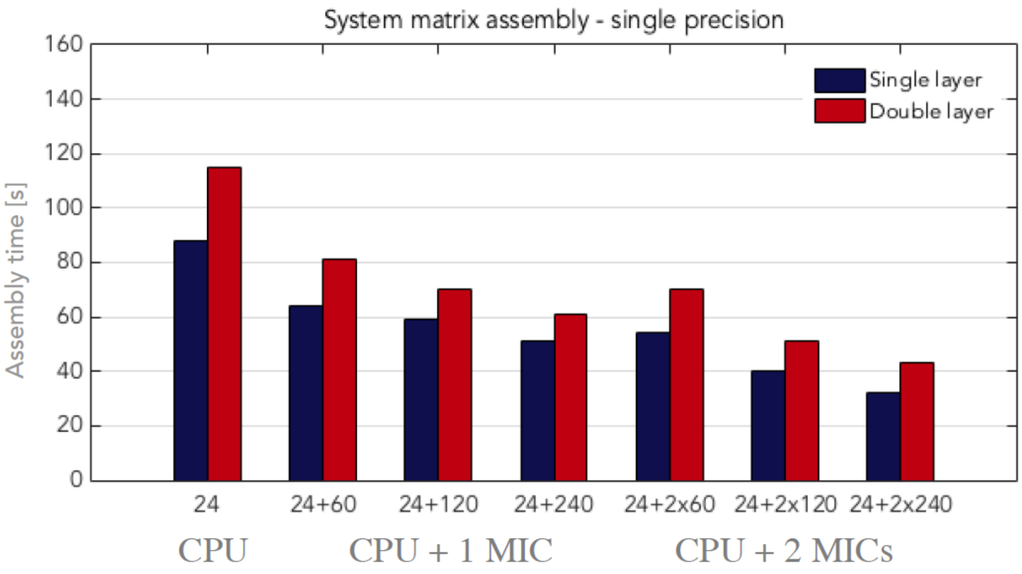
double precision

IT4Innovations
national01$#&0
supercomputing
center@#01%101

Acceleration of the system matrix assembly

Laplace equation

- 81920 surface elements
- Maximum speedup using two cards approx. 2.5

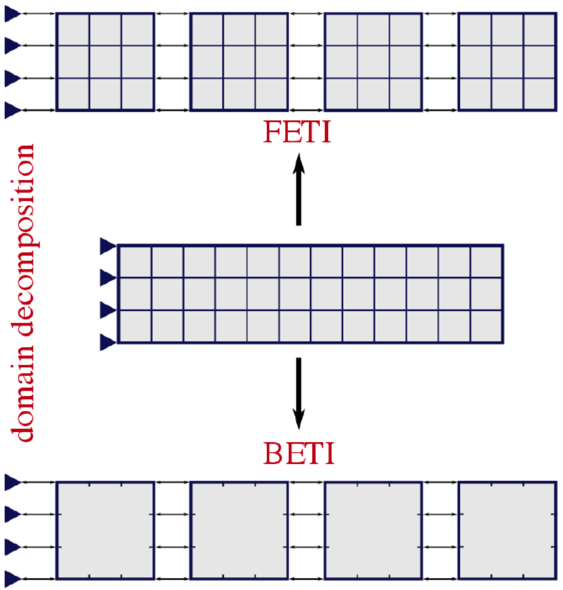


single precision

BETI for linear elasticity

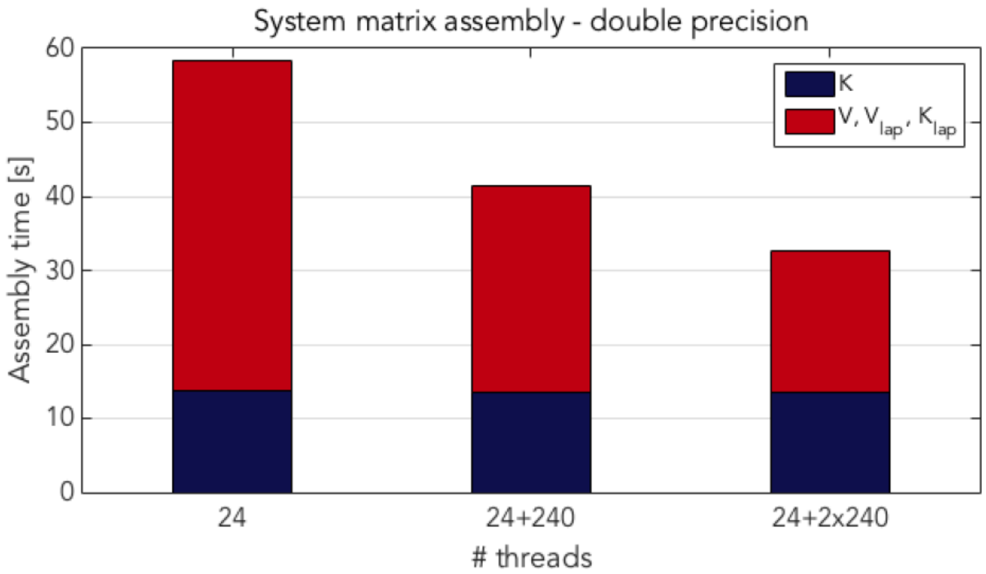
- Accelerating BETI (Boundary Element Tearing and Interconnecting) domain decomposition method for the linear elasticity problems
- Interface to the ESPRESO domain decomposition library
- BEM4I generates the local Dirichlet-to-Neumann map for each subdomain (Steklov-Poincaré operator)

$$S_h = (1/2M_h + K_h^\top)V_h^{-1}(1/2M_h + K_h)$$



BETI for linear elasticity

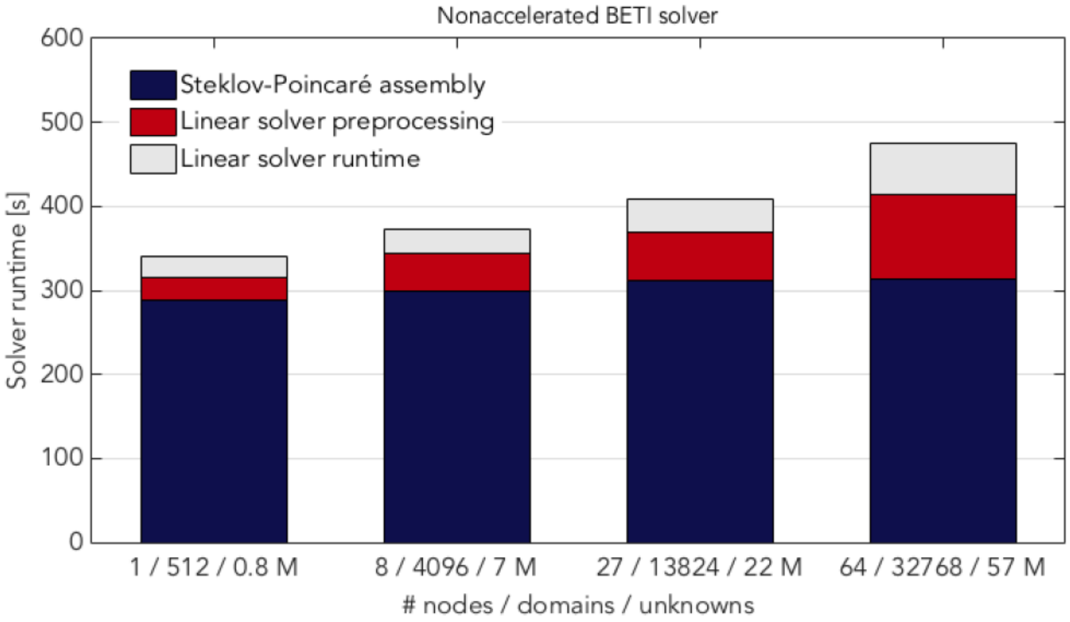
- 20480 surface elements
- Maximum speedup using two cards
  - V, Vlap, Klap (in red): 2.5
  - Total: 1.8



Single node acceleration

BETI for linear elasticity

Non-accelerated BETI up to 64 nodes/57 mil. surface unknowns



- Currently working on reducing the time for Steklov-Poincaré assembly using accelerators

Conclusions

- Hybrid FETI implementation shows very high scalability
  - Numerical and parallel
  - Strong and weak
- Many core architectures bring higher performance
  - But this is not for free
  - Only certain parts of the code may be accelerated
- Codes with full matrices like BEM4I benefit from MIC architecture