ESPRESO - ExaScale PaRallel FETI Solver

Hybrid FETI Solver
Report

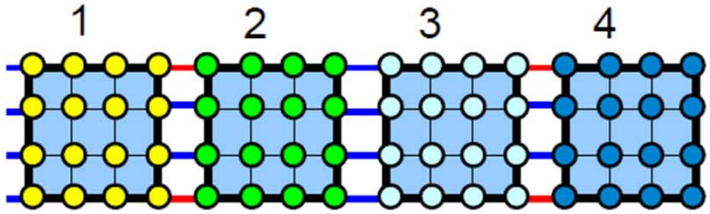
Lubomir Riha, Tomas Brzobohaty

IT4Innovations

Outline

- HFETI theory
  - from FETI to HFETI
  - communication hiding and avoiding techniques
- our new HFETI Solver
  - implementation description
  - performance and scalability
- conclusions
  - work done during stay in FRG

From FETI to HFETI



$$\begin{pmatrix} K & B^T \\ B & O \end{pmatrix} \begin{pmatrix} u \\ \lambda \end{pmatrix} = \begin{pmatrix} f \\ c \end{pmatrix} \qquad K = \begin{pmatrix} K_1 & O & O & O \\ O & K_2 & O & O \\ O & O & K_3 & O \\ O & O & O & K_4 \end{pmatrix}$$

matrix B per subdomains

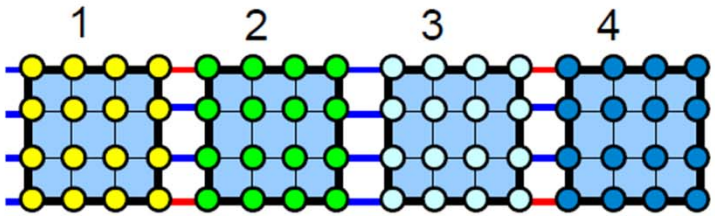
$$\begin{pmatrix} K_1 & O & O & O & B_1^T \\ O & K_2 & O & O & B_2^T \\ O & O & K_3 & O & B_3^T \\ O & O & O & K_4 & B_4^T \\ B_1 & B_2 & B_3 & B_4 & O \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \lambda \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ c \end{pmatrix}$$

$$\begin{pmatrix} K_1 & O & O & O & B_{0,1}^T & O & B_{1,1}^T \\ O & K_2 & O & O & B_{0,2}^T & O & B_{1,2}^T \\ O & O & K_3 & O & O & B_{0,3}^T & B_{1,3}^T \\ O & O & O & K_4 & O & B_{0,4}^T & B_{1,4}^T \\ B_{0,1} & B_{0,2} & O & O & O & O & O \\ O & O & B_{0,3} & B_{0,4} & O & O & O \\ B_{1,1} & B_{1,2} & B_{1,3} & B_{1,4} & O & O & O \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \lambda_{0,12} \\ \lambda_{0,34} \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ c_{0,12} \\ c_{0,34} \\ c_1 \end{pmatrix}$$

Both KKT systems are identical, only matrix *B* on right side is reordered and divided per rows (with matrix *P*)

$$PB_i = \begin{pmatrix} B_{0,i} \\ B_{1,i} \end{pmatrix}$$

From FETI to HFETI



$$\begin{pmatrix} K & B^T \\ B & O \end{pmatrix}\begin{pmatrix} u \\ \lambda \end{pmatrix}=\begin{pmatrix} f \\ c \end{pmatrix} \qquad K=\begin{pmatrix} K_1 & O & O & O \\ O & K_2 & O & O \\ O & O & K_3 & O \\ O & O & O & K_4 \end{pmatrix}$$

matrix B per subdomains

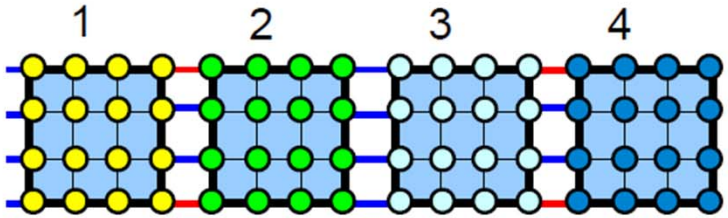
$$\begin{pmatrix} K_1 & O & O & O & B_1^T \\ O & K_2 & O & O & B_2^T \\ O & O & K_3 & O & B_3^T \\ O & O & O & K_4 & B_4^T \\ B_1 & B_2 & B_3 & B_4 & O \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \lambda \end{pmatrix}=\begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ c \end{pmatrix}$$

$$\begin{pmatrix} K_1 & O & B_{0,1}^T & O & O & O & B_{1,1}^T \\ O & K_2 & B_{0,2}^T & O & O & O & B_{1,2}^T \\ B_{0,1} & B_{0,2} & O & O & O & O & O \\ O & O & O & K_3 & O & B_{0,3}^T & B_{1,3}^T \\ O & O & O & O & K_4 & B_{0,4}^T & B_{1,4}^T \\ O & O & O & B_{0,3} & B_{0,4} & O & O \\ B_{1,1} & B_{1,2} & O & B_{1,3} & B_{1,4} & O & O \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \lambda_{0,12} \\ u_3 \\ u_4 \\ \lambda_{0,34} \\ \lambda_1 \end{pmatrix}=\begin{pmatrix} f_1 \\ f_2 \\ c_{0,12} \\ f_3 \\ f_4 \\ c_{0,34} \\ c_1 \end{pmatrix}$$

Both KKT systems are identical, only matrix *B* on right side is reordered and divided per rows (with matrix *P*)

$$PB_i=\begin{pmatrix} B_{0,i} \\ B_{1,i} \end{pmatrix}$$

From FETI to HFETI



$$\begin{pmatrix} \tilde{K}_{12} & O & \tilde{B}_{12}^T \\ O & \tilde{K}_{34} & \tilde{B}_{34}^T \\ \tilde{B}_{12} & \tilde{B}_{34}^T & O \end{pmatrix} \begin{pmatrix} \tilde{u}_{12} \\ \tilde{u}_{34} \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} \tilde{f}_{12} \\ \tilde{f}_{34} \\ \tilde{c} \end{pmatrix}$$

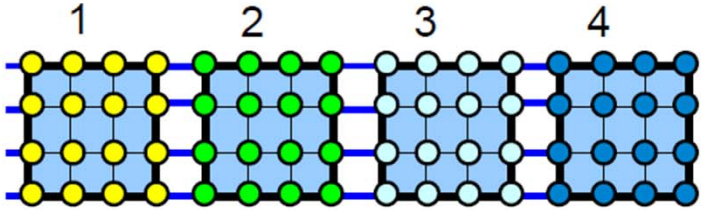
substitutions

$$\tilde{K}_{12} = \begin{pmatrix} K_1 & O & B_{0,1}^T \\ O & K_2 & B_{0,2}^T \\ B_{0,1} & B_{0,2} & O \end{pmatrix} \quad \tilde{B}_{12} = \begin{pmatrix} B_{1,1} & B_{1,2} & O \end{pmatrix} \quad \tilde{u}_{12} = \begin{pmatrix} u_1 \\ u_2 \\ \lambda_{0,12} \end{pmatrix} \quad \tilde{f}_{12} = \begin{pmatrix} f_1 \\ f_2 \\ c_{12} \end{pmatrix}$$

$$\tilde{K}_{34} = \begin{pmatrix} K_3 & O & B_{0,3}^T \\ O & K_4 & B_{0,4}^T \\ B_{0,3} & B_{0,4} & O \end{pmatrix} \quad \tilde{B}_{34} = \begin{pmatrix} B_{1,3} & B_{1,4} & O \end{pmatrix} \quad \tilde{u}_{34} = \begin{pmatrix} u_3 \\ u_4 \\ \lambda_{0,34} \end{pmatrix} \quad \tilde{f}_{34} = \begin{pmatrix} f_3 \\ f_4 \\ f_{34} \end{pmatrix}$$

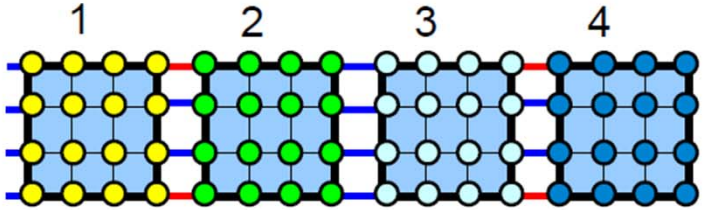
$$\tilde{\lambda} = \lambda_1 \qquad \tilde{c} = c_1$$

From FETI to HFETI

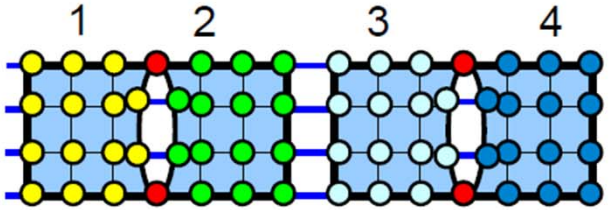


4 independently floating subdomains

$$\begin{pmatrix} K_1 & O & O & O & B_1^T \\ O & K_2 & O & O & B_2^T \\ O & O & K_3 & O & B_3^T \\ O & O & O & K_4 & B_4^T \\ B_1 & B_2 & B_3 & B_4 & O \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \lambda \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ c \end{pmatrix}$$



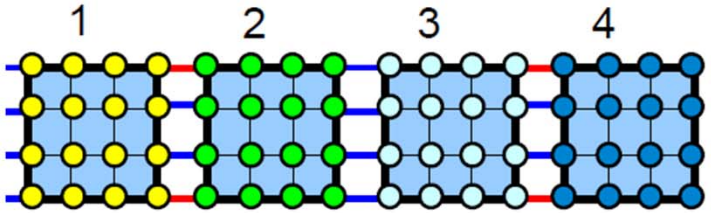
=



- 2 independently floating subdomains
- equivalent with FETI-DP approach, subdomains 1 and 2 (3 and 4) are connected per corners nodes)

$$\begin{pmatrix} \tilde{K}_{12} & O & \tilde{B}_{12}^T \\ O & \tilde{K}_{34} & \tilde{B}_{34}^T \\ \tilde{B}_{12} & \tilde{B}_{34}^T & O \end{pmatrix} \begin{pmatrix} \tilde{u}_{12} \\ \tilde{u}_{34} \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} \tilde{f}_{12} \\ \tilde{f}_{34} \\ \tilde{c} \end{pmatrix}$$

From FETI to HFETI



Kernels of augmented stiffness matrices

$$\begin{pmatrix} \tilde{K}_{12} & O & \tilde{B}_{12}^T \\ O & \tilde{K}_{34} & \tilde{B}_{34}^T \\ \tilde{B}_{12} & \tilde{B}_{34}^T & O \end{pmatrix} \begin{pmatrix} \tilde{u}_{12} \\ \tilde{u}_{34} \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} \tilde{f}_{12} \\ \tilde{f}_{34} \\ \tilde{c} \end{pmatrix} \qquad \tilde{R} = \begin{pmatrix} \tilde{R}_{12} & O \\ O & \tilde{R}_{34} \end{pmatrix}$$

$$\tilde{K}_{12} = \begin{pmatrix} K_1 & O & B_{0,1}^T \\ O & K_2 & B_{0,2}^T \\ B_{0,1} & B_{0,2} & O \end{pmatrix} \qquad \tilde{R}_{12} = \begin{pmatrix} R_1 \\ R_2 \\ O \end{pmatrix}$$

$$\tilde{K}_{34} = \begin{pmatrix} K_3 & O & B_{0,3}^T \\ O & K_4 & B_{0,4}^T \\ B_{0,3} & B_{0,4} & O \end{pmatrix} \qquad \tilde{R}_{34} = \begin{pmatrix} R_3 \\ R_4 \\ O \end{pmatrix}$$

Kernel of one cluster is assembled from kernels of its all subdomains. All subdomains' kernels have to be written from one coordinate system (requirement of the compatibility).

HFETI Matrix G (and GG^T)

$$\tilde{G}^T = -\tilde{B}\tilde{R} \qquad \tilde{B} = \begin{pmatrix} B_1 & B_2 & O & B_3 & B_4 & O \end{pmatrix} \qquad \tilde{R} = \begin{pmatrix} R_1 & O \\ R_2 & O \\ O & O \\ O & R_3 \\ O & R_4 \\ O & O \end{pmatrix}$$

$$G^T = -\begin{pmatrix} B_1 R_1 + B_2 R_2 & B_3 R_3 + B_4 R_4 \end{pmatrix}$$

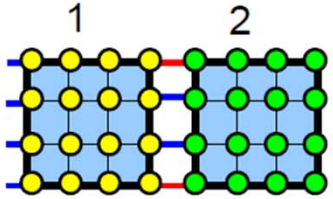
Size of GG^T is given by defect of global stiffness matrix

$$GG^T = \begin{pmatrix} R_1^T B_1^T + R_2^T B_2^T \\ R_3^T B_3^T + R_4^T B_4^T \end{pmatrix} \begin{pmatrix} B_1 R_1 + B_2 R_2 & B_3 R_3 + B_4 R_4 \end{pmatrix}$$

Compare GG^T assembled for classical TFETI

$$GG^T = \begin{pmatrix} R_1^T B_1^T \\ R_2^T B_2^T \\ R_3^T B_3^T \\ R_4^T B_4^T \end{pmatrix} \begin{pmatrix} B_1 R_1 & B_2 R_2 & B_3 R_3 & B_4 R_4 \end{pmatrix}$$

HFETI $\tilde{x} = \tilde{K}^{+}\tilde{b}$ for i-th cluster



$$\begin{pmatrix} K_1 & O & B_{0,1}^T \\ O & K_2 & B_{0,2}^T \\ B_{0,1} & B_{0,2} & O \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \mu \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ d \end{pmatrix} \qquad \begin{pmatrix} K & B_0^T \\ B_0 & O \end{pmatrix} \begin{pmatrix} x \\ \mu \end{pmatrix} = \begin{pmatrix} b \\ d \end{pmatrix}$$

$$\begin{pmatrix} F_0 & G_0^T \\ G_0 & O \end{pmatrix} \begin{pmatrix} \mu \\ \beta \end{pmatrix} = \begin{pmatrix} g_0 \\ e_0 \end{pmatrix}$$

$$F_0 = B_0 K^{+} B_0^T$$

$$G_0 = -R_0^T B_0^T$$

$$g_0 = B_0 K^{+} f_0 - c_0$$

$$e_0 = -R_0^T f_0^T$$

$$\beta_0 = \left(G_0 F_0^{-1} G_0^T\right)^{+} \left(G_0 F_0^{-1} g_0 - e_0\right)$$

$$\mu = F_0^{-1}\left(g_0 - G_0^T \beta_0\right)$$

$$x = K^{+}\left(b - B_0^T \mu\right) + R_0 \beta$$

Our new TFETI and HFETI Solver

- Implementation in C++
  - based on Intel MKL
    - sparse and dense BLAS routines
    - PARDISO direct sparse solver
  - compiled with Intel Compiler and Intel MPI

- Parallelization tools and strategies
  - hybrid parallelization for multi-socket, multicore compute nodes
    - enables over subscription of CPU cores – each core can process multiple sub-domains
  - distributed memory parallelization – MPI
    - use of MPI 3.0 standard features (Intel MPI 5.0 Beta)
      - non-blocking global operations (MPI_Iallredeuce, …) for global DOT product
  - shared memory parallelization – using Intel Cilk+
    - enables parallel reduction using custom reduce operations with C++ objects
    - conversion to OpenMP is planned using OpenMP standard 4.0

HFETI Solver – development stages

- Stage 1
  - optimization of global communication of FETI solver
    - using communication hiding and avoiding techniques
      - pipelined CG solvers
        - » single global reduction
      - solving coarse problem using distributed inverse matrix
        - » single global AllGather operation (instead of MPI_Gather and MPI_Scatter)
        - » parallel solve – each subdomain has only 6 dense rows of $(GG^T)^{-1}$
      - nearest neighbor communication optimization – multiplication with gluing matrix
        - » B for FETI and B1 for HFETI (next slide)
        - » fully scalable with groving number of compute nodes

- Stage 2
  - optimization of inter cluster processing
  - efficient solution of inner coarse problem

- Stage 3
  - combine both together

Reduction of global communication in FETI and HFETI

Hiding latencies in Krylov solvers

Preconditioned Conjugate Gradient

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

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

Chronopoulos/Gear CG

Only one global reduction each iteration.

1: $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $w_0 := Au_0$
2: $\alpha_0 := \langle r_0, u_0 \rangle / \langle w_0, u_0 \rangle$; $\beta := 0$; $\gamma_0 := \langle r_0, u_0 \rangle$
3: **for** $i = 0, \ldots, m-1$ **do**
4: $\quad p_i := u_i + \beta_i p_{i-1}$
5: $\quad s_i := w_i + \beta_i s_{i-1}$
6: $\quad x_{i+1} := x_i + \alpha p_i$
7: $\quad r_{i+1} := r_i - \alpha s_i$
8: $\quad u_{i+1} := M^{-1} r_{i+1}$
9: $\quad w_{i+1} := Au_{i+1}$
10: $\quad \gamma_{i+1} := \langle r_{i+1}, u_{i+1} \rangle$
11: $\quad \delta := \langle w_{i+1}, u_{i+1} \rangle$
12: $\quad \beta_{i+1} := \gamma_{i+1} / \gamma_i$
13: $\quad \alpha_{i+1} := \gamma_{i+1} / (\delta - \beta_{i+1}\gamma_{i+1}/\alpha_i)$
14: **end for**

Hiding latencies in Krylov solvers

Chronopoulos/Gear CG

Only one global reduction each iteration.

1: $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $w_0 := Au_0$
2: $\alpha_0 := \langle r_0, u_0\rangle \,/\, \langle w_0, u_0\rangle$; $\beta := 0$; $\gamma_0 := \langle r_0, u_0\rangle$
3: **for** $i = 0, \ldots, m-1$ **do**
4: $\quad p_i := u_i + \beta_i p_{i-1}$
5: $\quad s_i := w_i + \beta_i s_{i-1}$
6: $\quad x_{i+1} := x_i + \alpha p_i$
7: $\quad r_{i+1} := r_i - \alpha s_i$
8: $\quad u_{i+1} := M^{-1} r_{i+1}$
9: $\quad w_{i+1} := Au_{i+1}$
10: $\quad \gamma_{i+1} := \langle r_{i+1}, u_{i+1}\rangle$
11: $\quad \delta := \langle w_{i+1}, u_{i+1}\rangle$
12: $\quad \beta_{i+1} := \gamma_{i+1}/\gamma_i$
13: $\quad \alpha_{i+1} := \gamma_{i+1}/(\delta - \beta_{i+1}\gamma_{i+1}/\alpha_i)$
14: **end for**

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

Preconditioned pipelined CG

1: $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $w_0 := Au_0$
2: **for** $i = 0, \ldots$ **do**
3: $\quad \gamma_i := \langle r_i, u_i\rangle$
4: $\quad \delta := \langle w_i, u_i\rangle$
5: $\quad m_i := M^{-1} w_i$
6: $\quad n_i := Am_i$
7: $\quad$ **if** $i > 0$ **then**
8: $\quad\quad \beta_i := \gamma_i/\gamma_{i-1}$; $\alpha_i := \gamma_i/(\delta - \beta_i\gamma_i/\alpha_{i-1})$
9: $\quad$ **else**
10: $\quad\quad \beta_i := 0$; $\alpha_i := \gamma_i/\delta$
11: $\quad$ **end if**
12: $\quad z_i := n_i + \beta_i z_{i-1}$
13: $\quad q_i := m_i + \beta_i q_{i-1}$
14: $\quad s_i := w_i + \beta_i s_{i-1}$
15: $\quad p_i := u_i + \beta_i p_{i-1}$
16: $\quad x_{i+1} := x_i + \alpha_i p_i$
17: $\quad r_{i+1} := r_i - \alpha_i s_i$
18: $\quad u_{i+1} := u_i - \alpha_i q_i$
19: $\quad w_{i+1} := w_i - \alpha_i z_i$
20: **end for**

Hiding latencies in Krylov solvers: Cost model

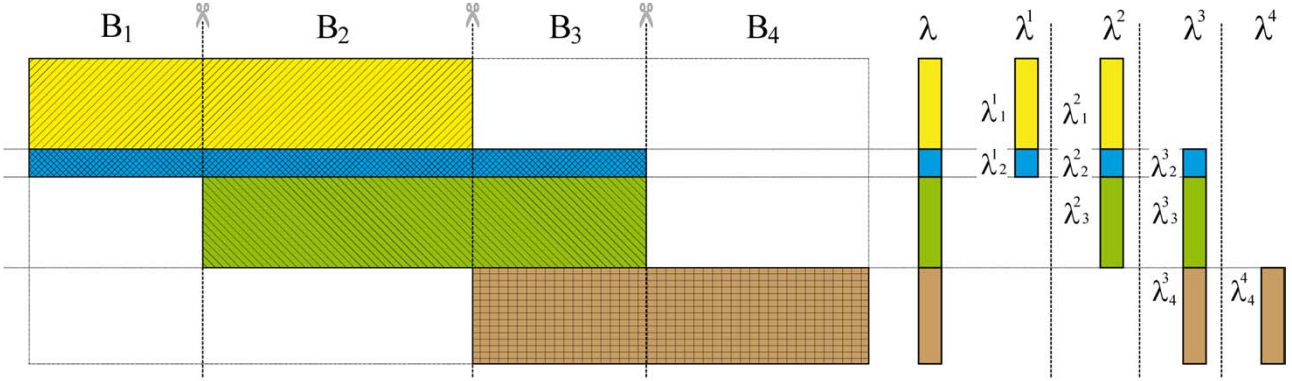
- G := time for a global reduction
- SpMV := time for a sparse-matrix vector product
- PC := time for preconditioner application
- local work such as AXPY is neglected

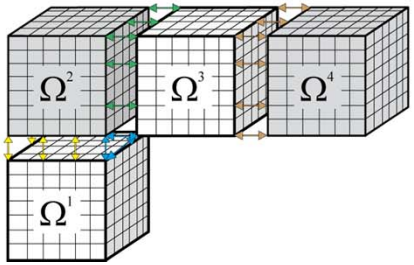
	flops	time (excl, AXPYs, DOTs)	#glob syncs	memory
CG	10	$2G + SpMV + PC$	2	4
Chro/Gea	12	$G + SpMV + PC$	1	5
pipe-CG	20	$\max(G, SpMV + PC)$	1	9

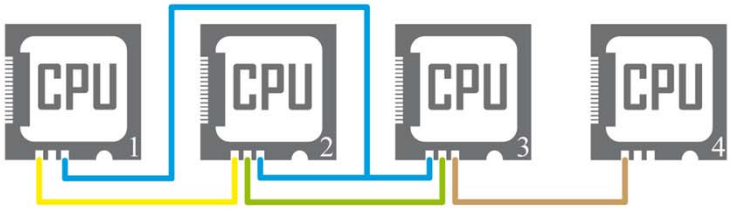
P. Ghysels and W. Vanroose, *in Press*, Parallel Computing, 2013

Reduction of global communication in FETI and HFETI

- MPI processes iterate only over λs “they need” for local processing
  - in case of FETI – the λs required for multiplication with B matrix of a subdomain
  - in case of HFETI – the λs required for multiplication with B1 matrices of all subdomains per cluster
- global update of vector λ
  - becomes only nearest neighbor type of communication
  - scales with number of MPI processes (subdomains)

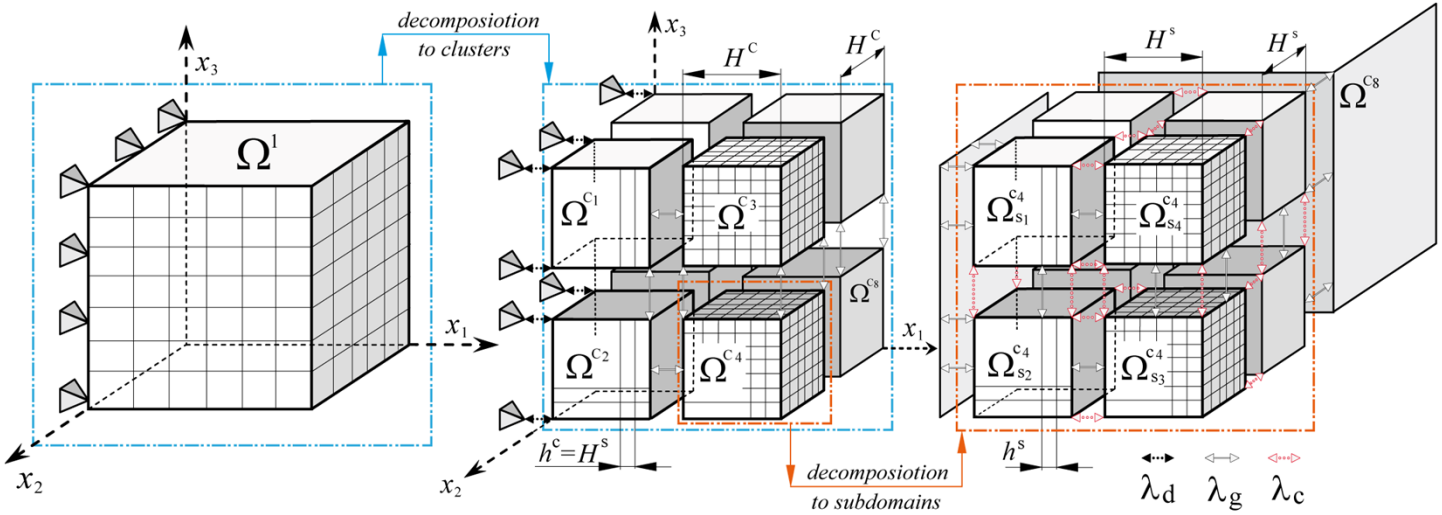




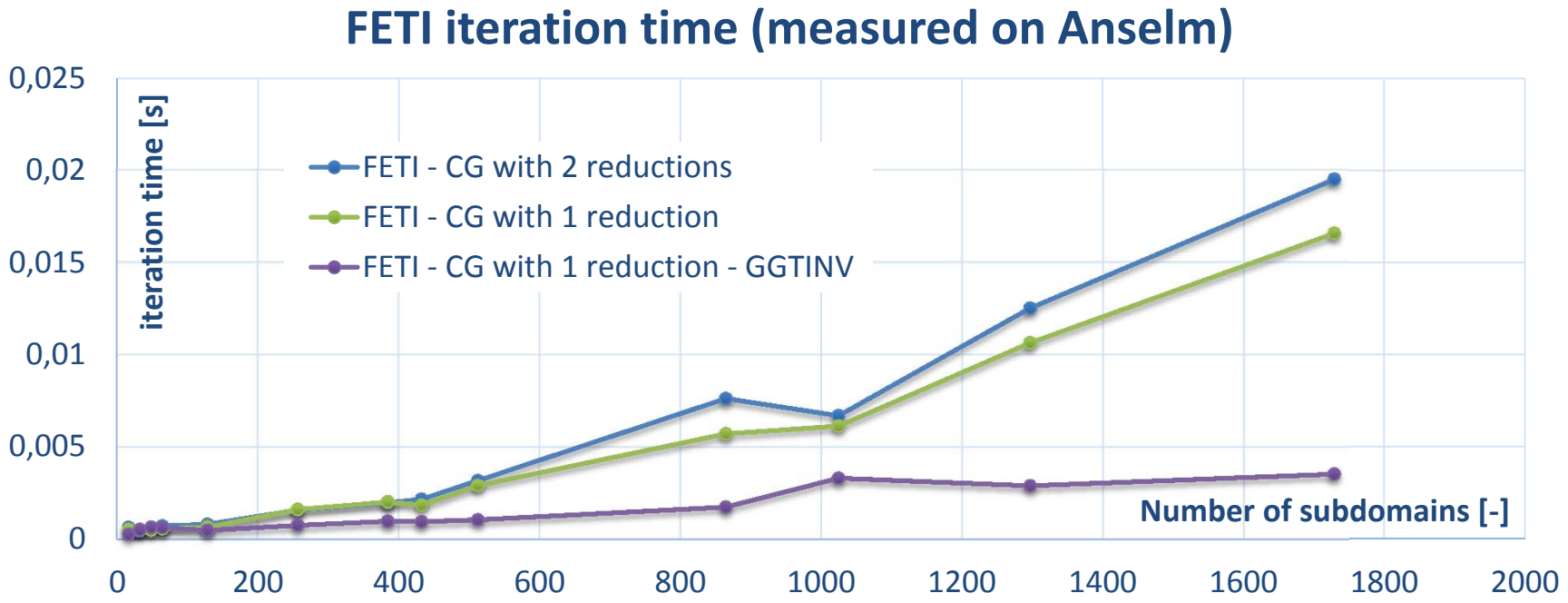


HFETI Solver - Benchmark

- 3D CUBE - elasticity
  - FETI
    - H^C - FETI decomposition into sub-domains
    - $(h^C)^3$ – number of elements per subdomain
    -
  - HFETI
    - H^C – decomposition into clusters
    - H^S – cluster decomposition into sub-domains
      - $(h^S)^3$ – number of elements per sub-domain



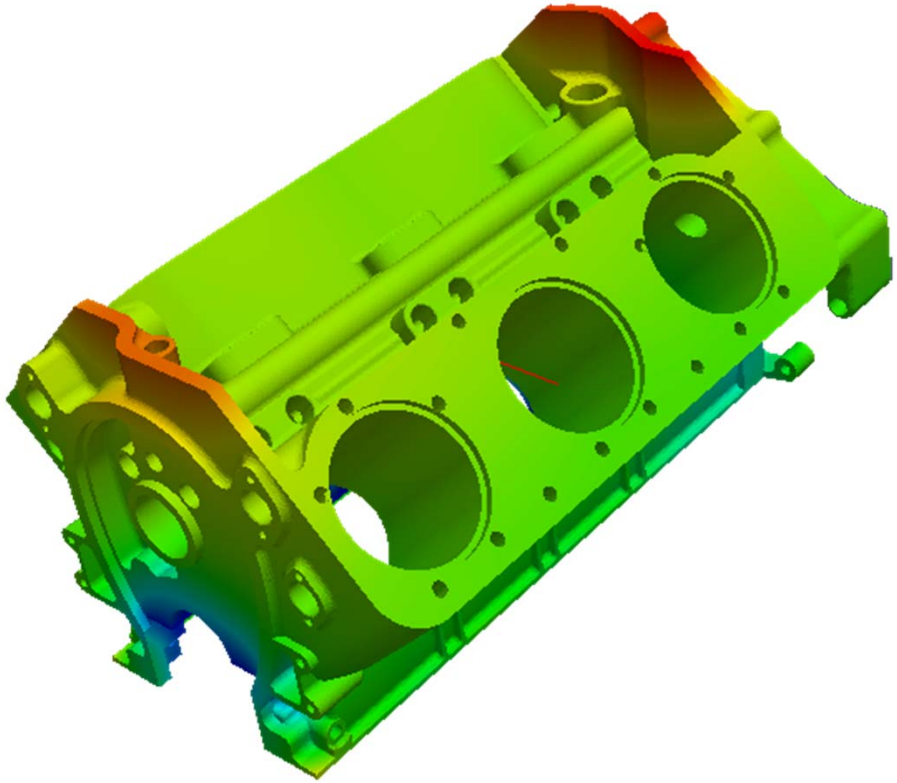
Stage 1: optimization of global communication for FETI solver



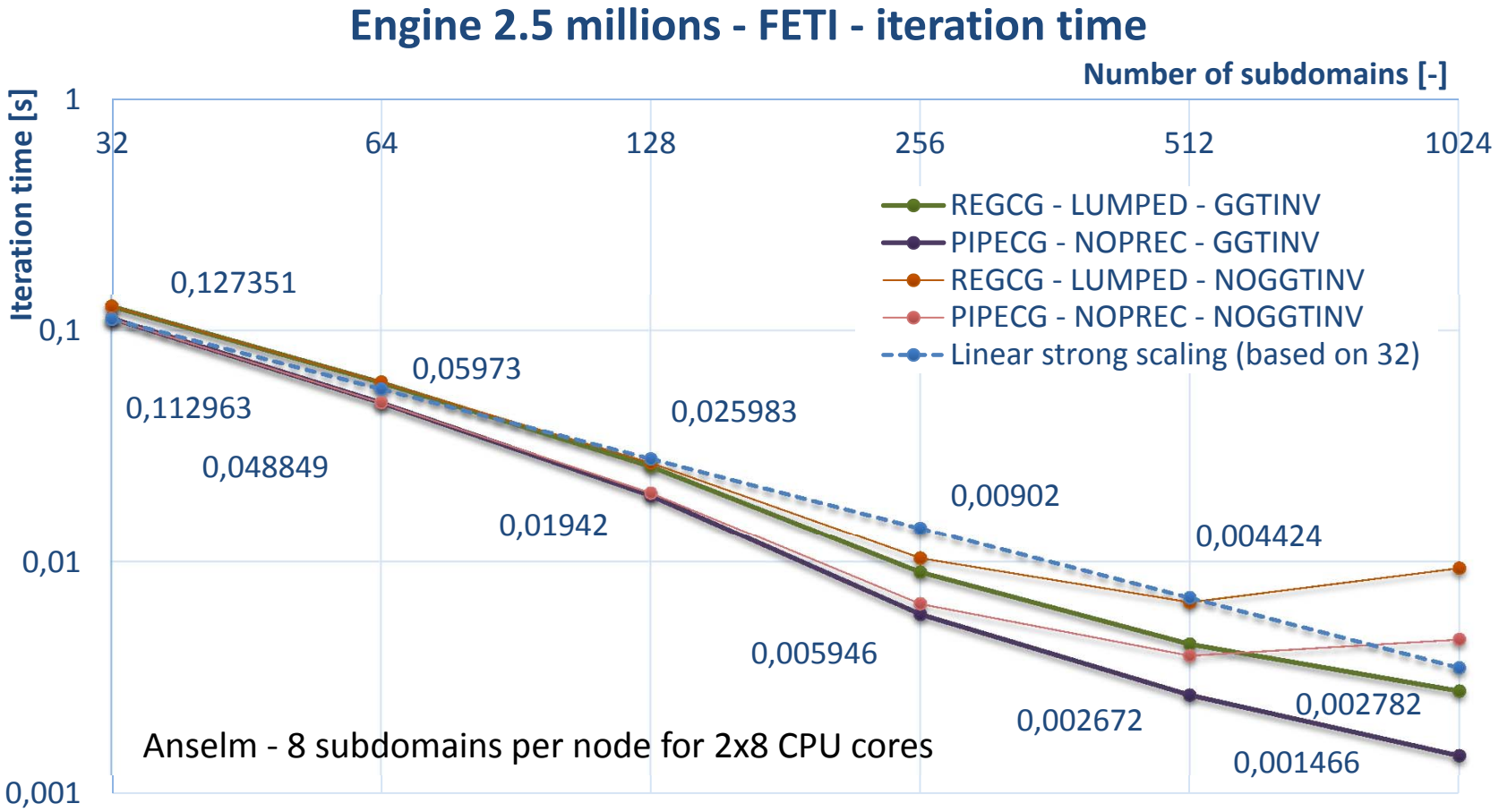
- **CG algorithm with 2 reductions** – is a general version of the CG algorithm
- **CG algorithm with 1 reduction** – is based on Preconditioned Pipelined CG algorithm, where projector is used in place of preconditioner (this algorithm is ready to use non-blocking global reduction for further performance improvements (comes in Intel MPI 5.0))
- **GGTINV** – parallelizes the solve of coarse problem plus merges two Gather and Scatter global operations into single AllGather
- $5^3 = 3*(5+1)^3 = 648$ DOFs – small domain size is chosen to identify all communication bottlenecks of the solver

Engine Benchmark – Superlinear Strong Scaling with FETI

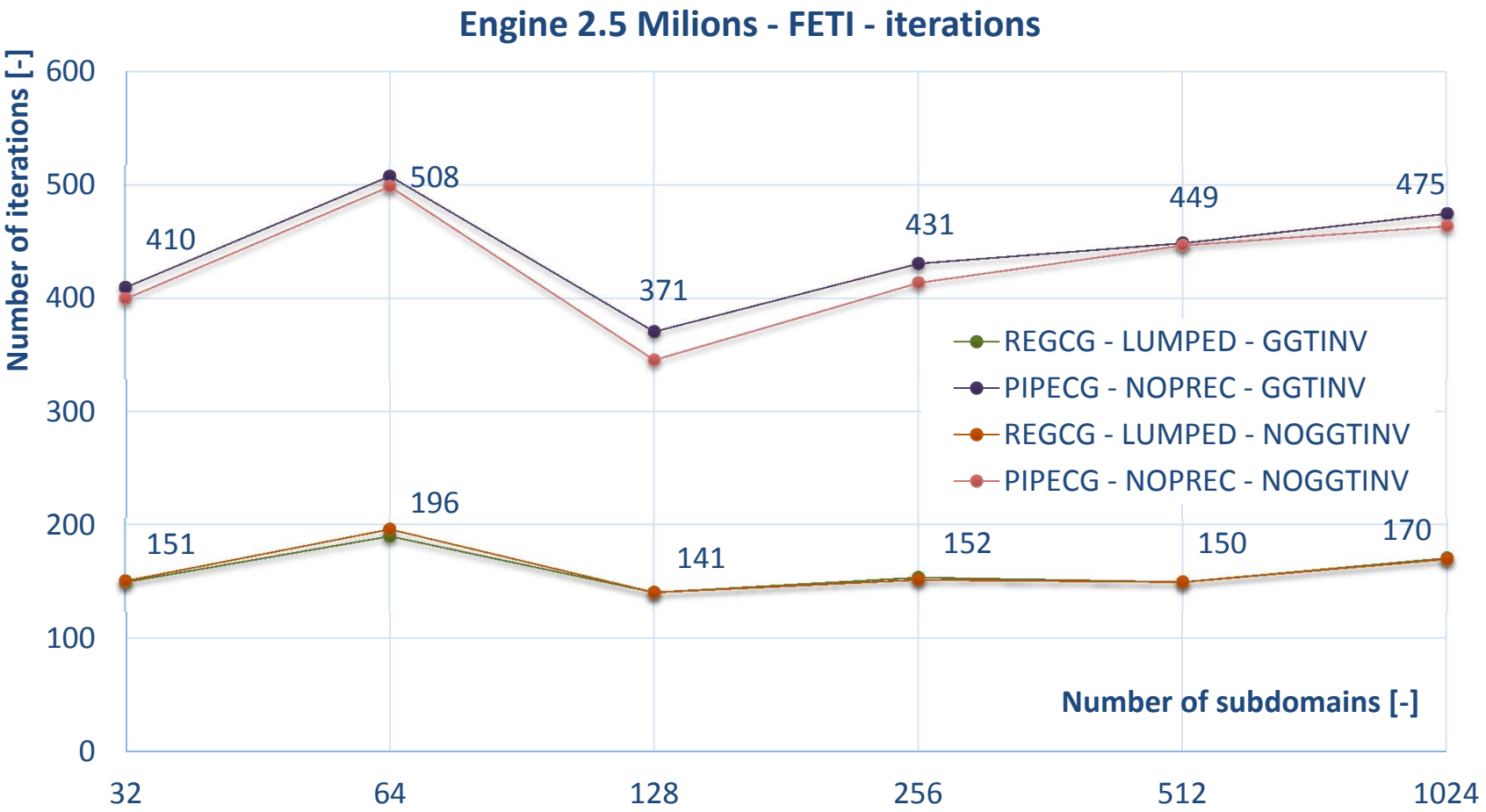
- 2.5 millions of DOFs
- decomposed into: 34, 64, 128, 256, 512 and 1024 subdomains
- FETI solver
- measured on Anselm
  - 128 nodes with 16 CPU cores (2x8)
  - 8 subdomains per node
- ..



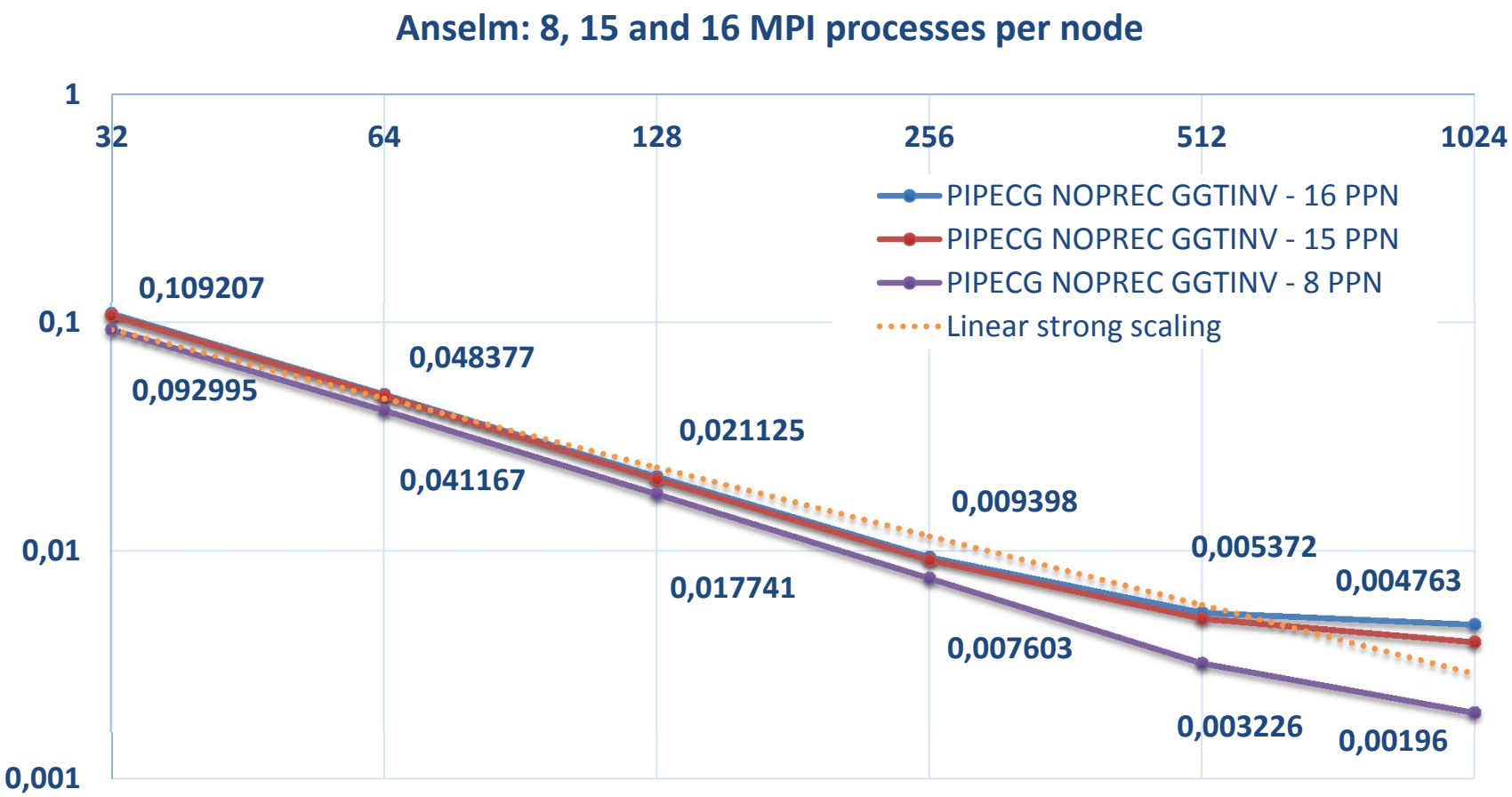
Engine Benchmark – Superlinear Strong Scaling with FETI



Engine Benchmark – Strong Scaling



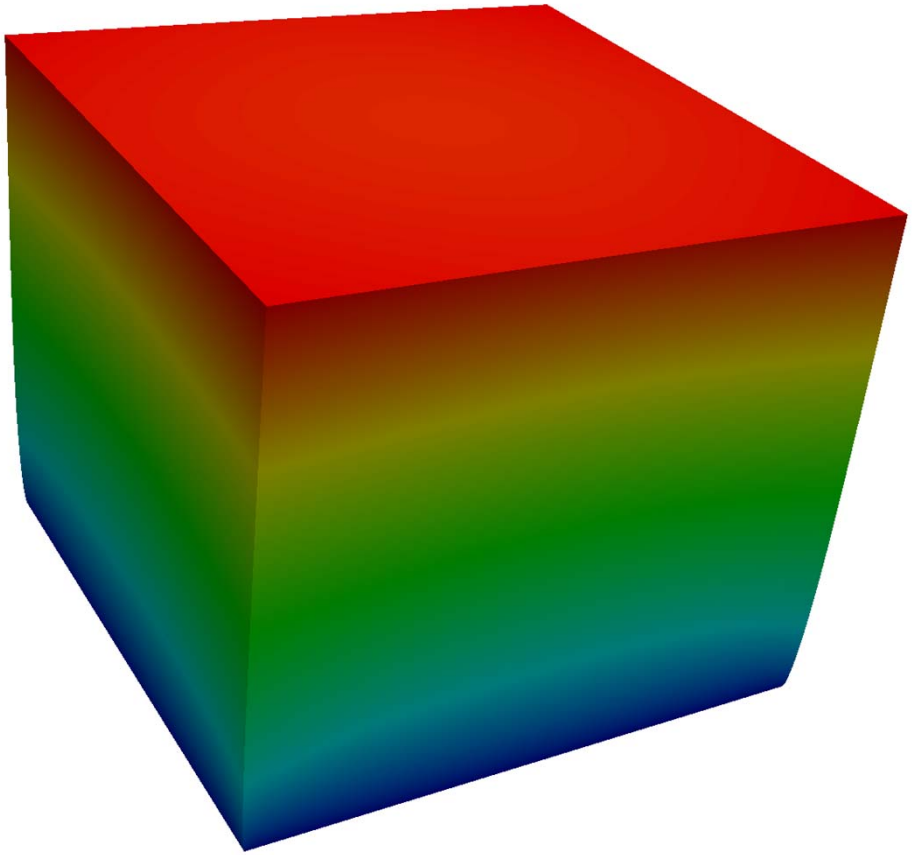
FETI - time per iteration for various utilization of compute node:



Since solver is memory bounded using 8 CPU cores per node gives each process faster access to the memory and significantly improves the performance.

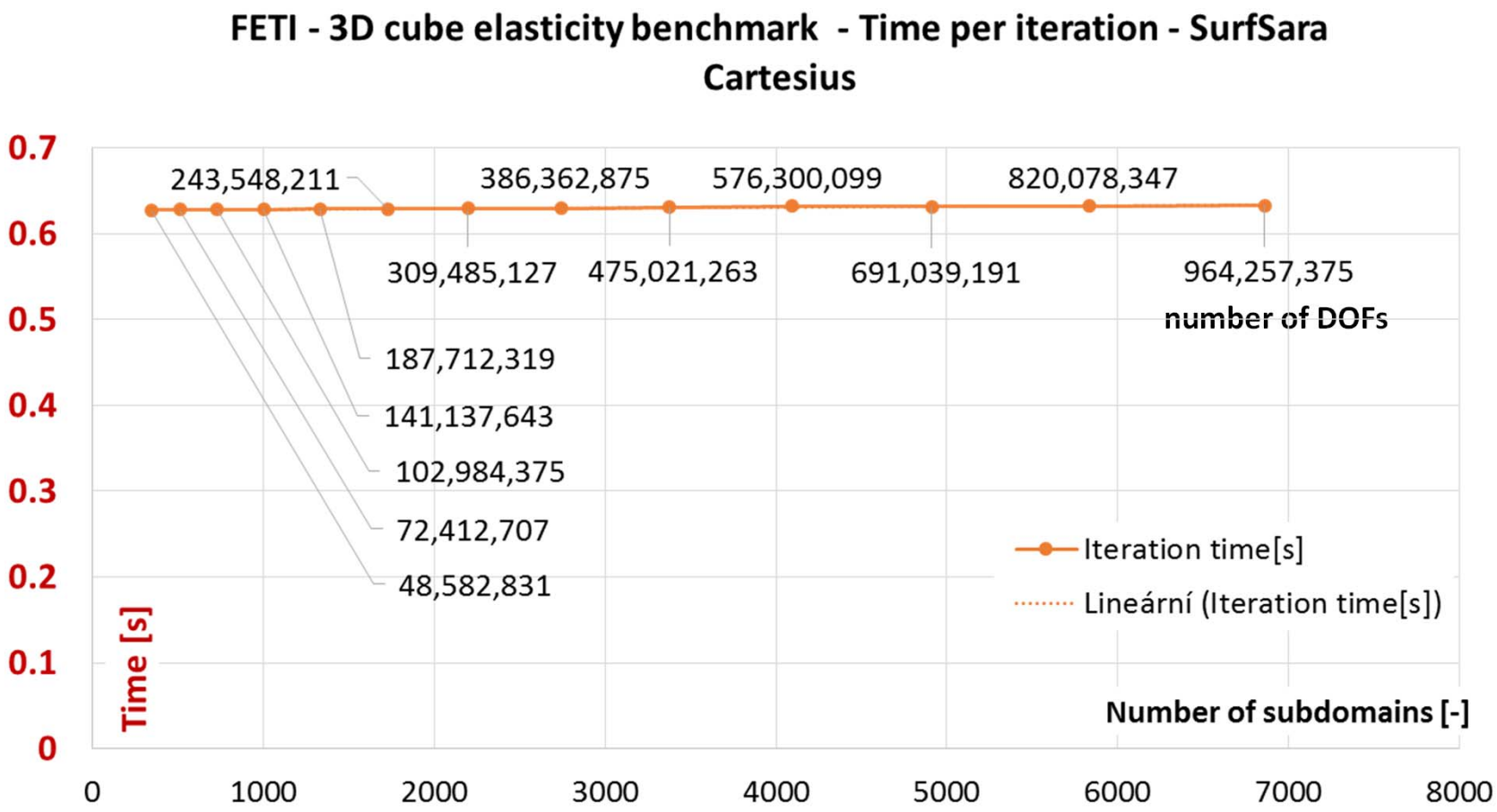
Large Scale Benchmarks

- 3D cube - elasticity benchmark
  - highly parallel benchmark generator scales up to thousands of sub-domains
  - large sub-domains 120,000 to 160,000 DOFs
    - sparse direct solver takes most of the memory and processing time (over 99%)
- Tested on Total FETI method only
  - shows the bottleneck of the method – the coarse problem that will be reduced by using total Hybrid FETI method
- No preconditioner
  - will be implemented in near future

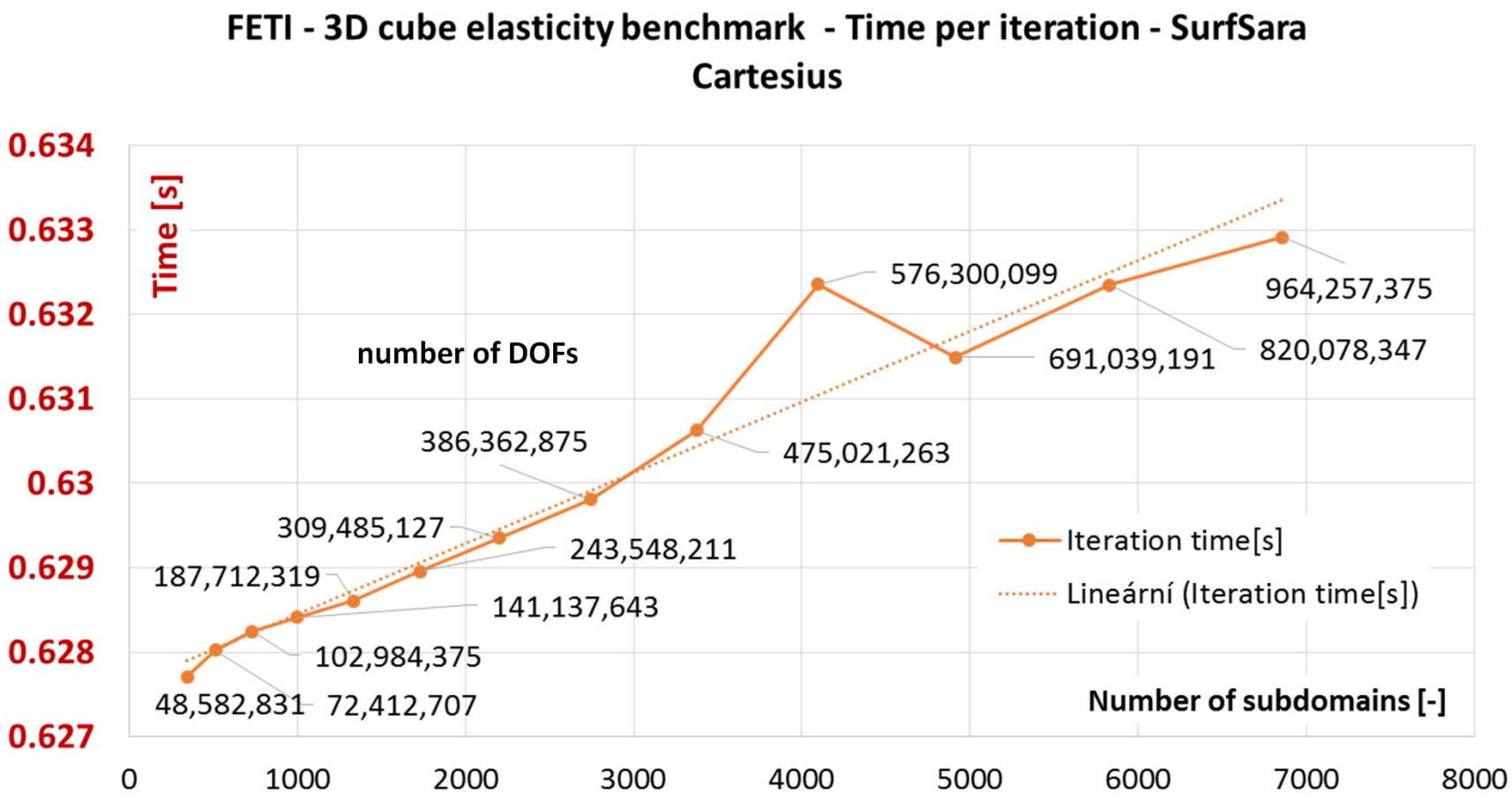


Visualization of displacement

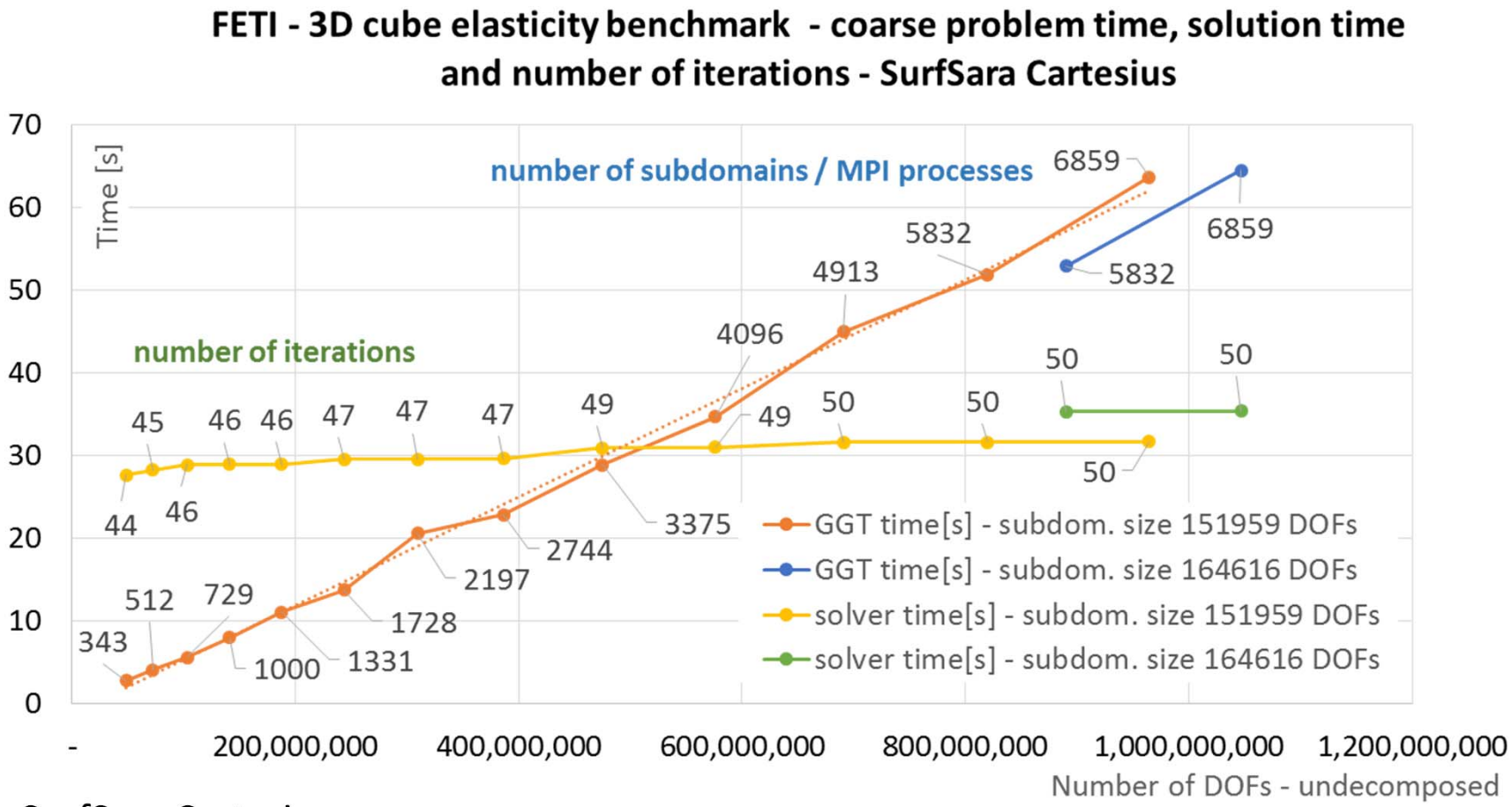
TFETI - Scalability of iterative solver



TFETI - Scalability of iterative solver



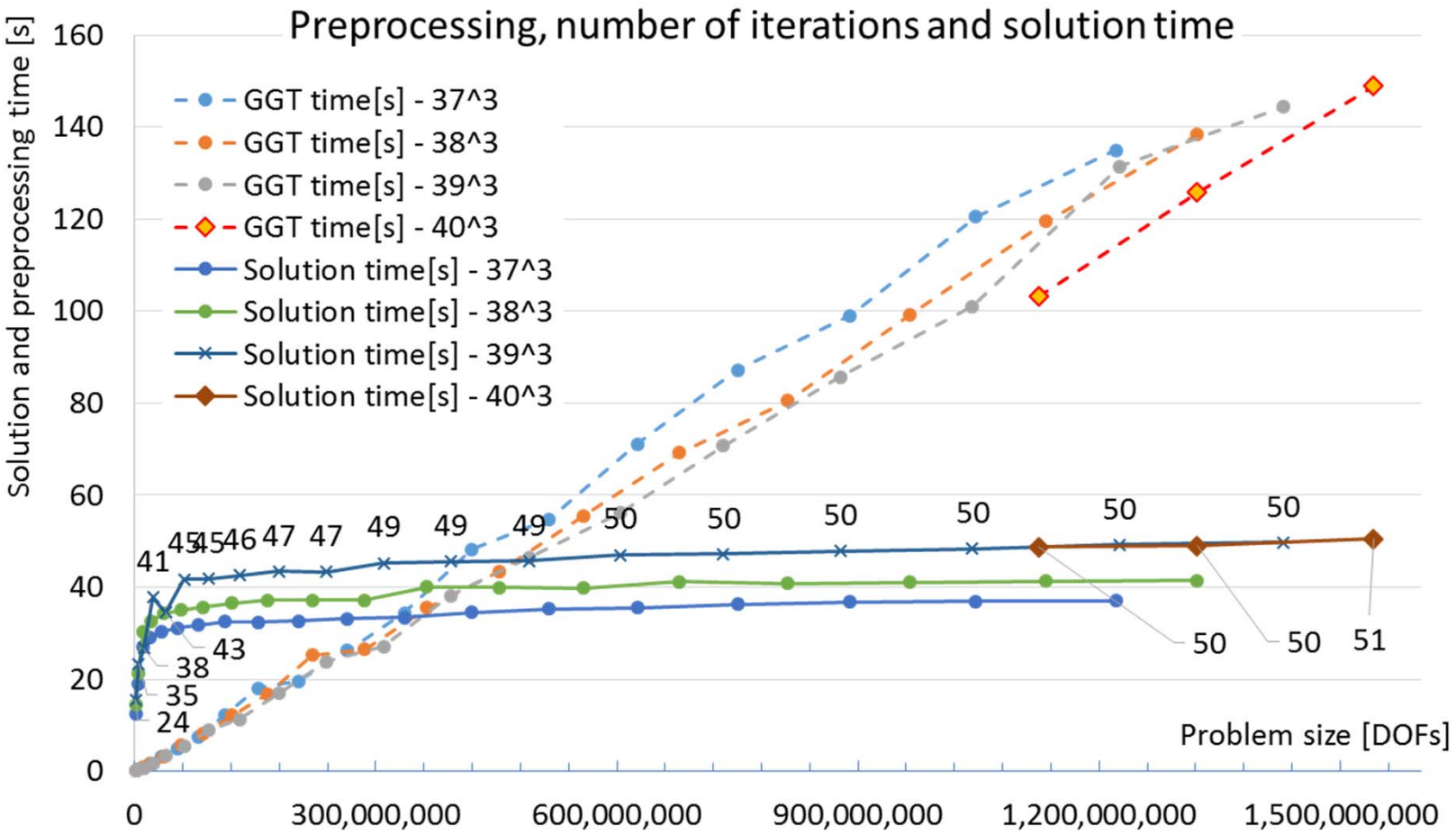
TFETI - Large Scale Benchmarks



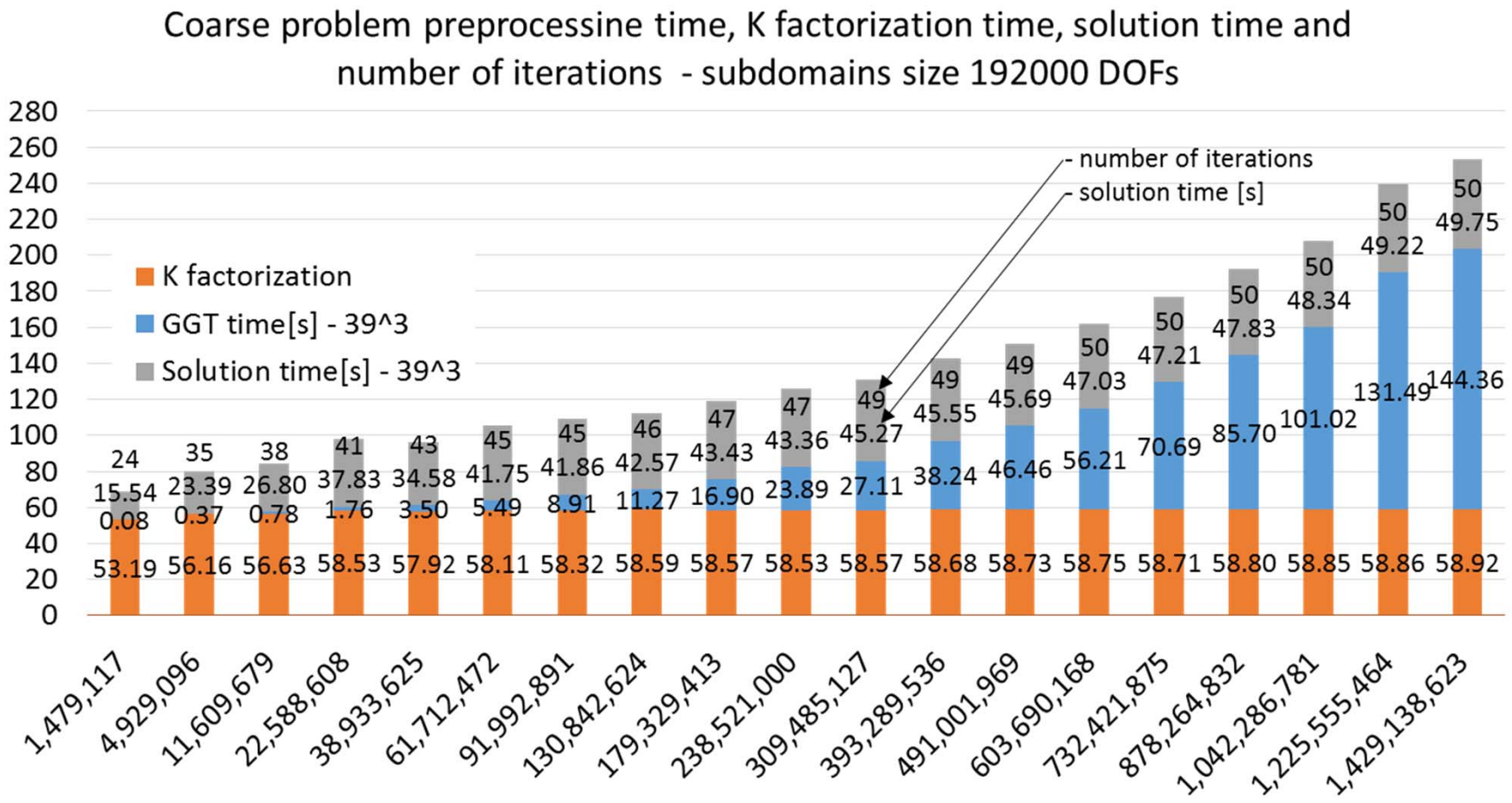
SurfSara Cartesius

- BULLx system with 64 GB of RAM per node and 24 CPU cores (2 x 12 cores)
- benchmark used all 24 cores per node – 2.6 GB of RAM per subdomain

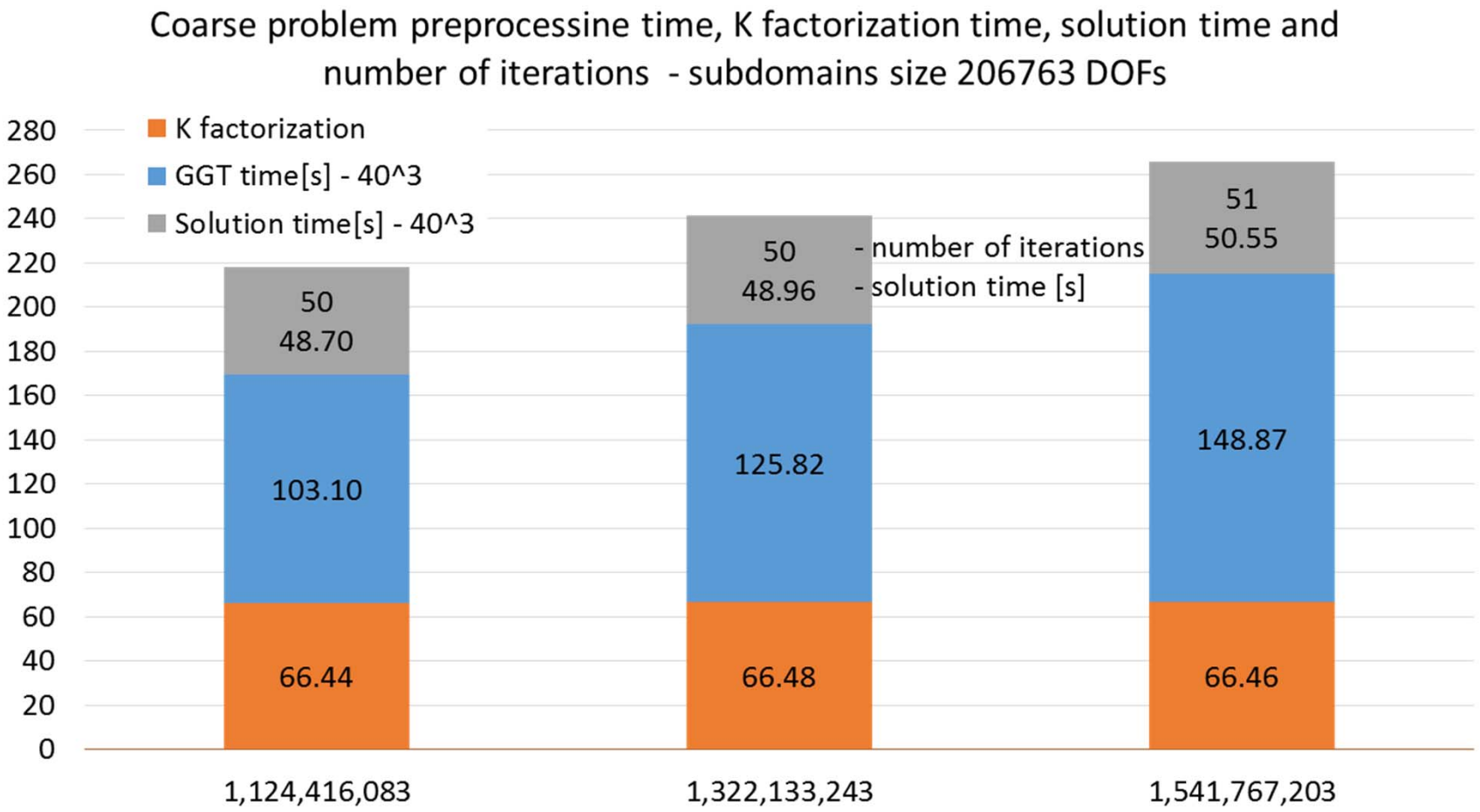
TFETI - Large Scale Benchmarks



TFETI - Large Scale Benchmarks



TFETI - Large Scale Benchmarks



HFETI Solver – development stages

- Stage 1
  - optimization of global communication of FETI solver
    - using communication hiding and avoiding techniques
      - pipelined CG solvers
        - » single global reduction
      - solving coarse problem using distributed inverse matrix
        - » single global AllGather operation (instead of MPI_Gather and MPI_Scatter)
        - » parallel solve – each subdomain has only 6 dense rows of $(GG^T)^{-1}$
      - nearest neighbor communication optimization – multiplication with gluing matrix
        - » B for FETI and B1 for HFETI (next slide)
        - » fully scalable with groving number of compute nodes

- Stage 2
  - optimization of inter cluster processing
  - efficient solution of inner coarse problem

- Stage 3
  - combine both together

Stage 2: Inter Cluster Processing

- It is important to select correct cluster decomposition
  - affects both preprocessing and solution time

domain size	subdomains	size	avg	sum	prec	iter
4	343	128625	0.123941	5.949182	75.503219	48
5	216	139968	0.072914	3.718598	37.732576	51
6	125	128625	0.040325	2.217873	16.381135	55
8	64	139968	0.030154	1.718762	10.04497	57
11	27	139968	0.034425	1.893399	7.490033	55
17	8	139968	0.064372	3.347334	7.876874	52

Anselm
Problem size:
~120 000

Optimal decomposition:
dom. size: $3*(8+1)^3 = 2187$
cluster size: $4^3 = 64$

domain size	subdomains	size	avg	sum	prec	iter
5	512	331776	0.309891	17.973677	338.035839	58
6	343	352947	0.198848	12.328554	180.041968	62
7	216	331776	0.118965	7.613787	90.402162	64
11	64	331776	0.080177	5.13133	27.456335	64
15	27	331776	0.095755	6.03256	19.761467	63
23	8	331776	0.156146	8.431877	29.455252	54

Anselm
Problem size:
~330 000

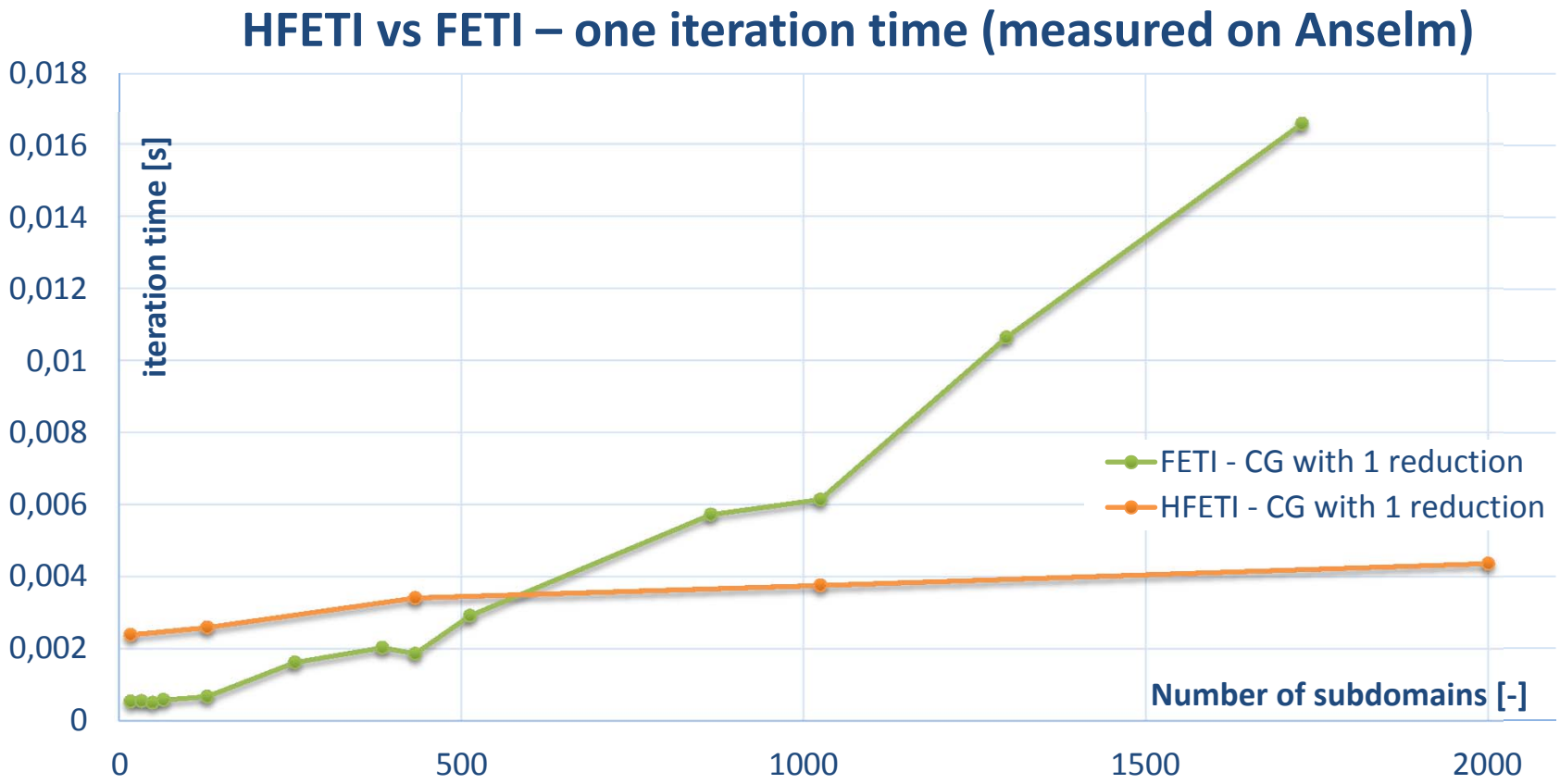
Optimal decomposition:
dom. size: $3*(11+1)^3 = 5184$
cluster size: $4^3 = 64$

Note: domains size 'N' is in number of elements per edge of the cube – real size is equal to $3*(N+1)^3$
Note: avg – average iteration time [s]; sum – total time of all iterations = solution time; prec – cluster preprocessing time; iter – number of iterations;

HFETI Solver – development stages

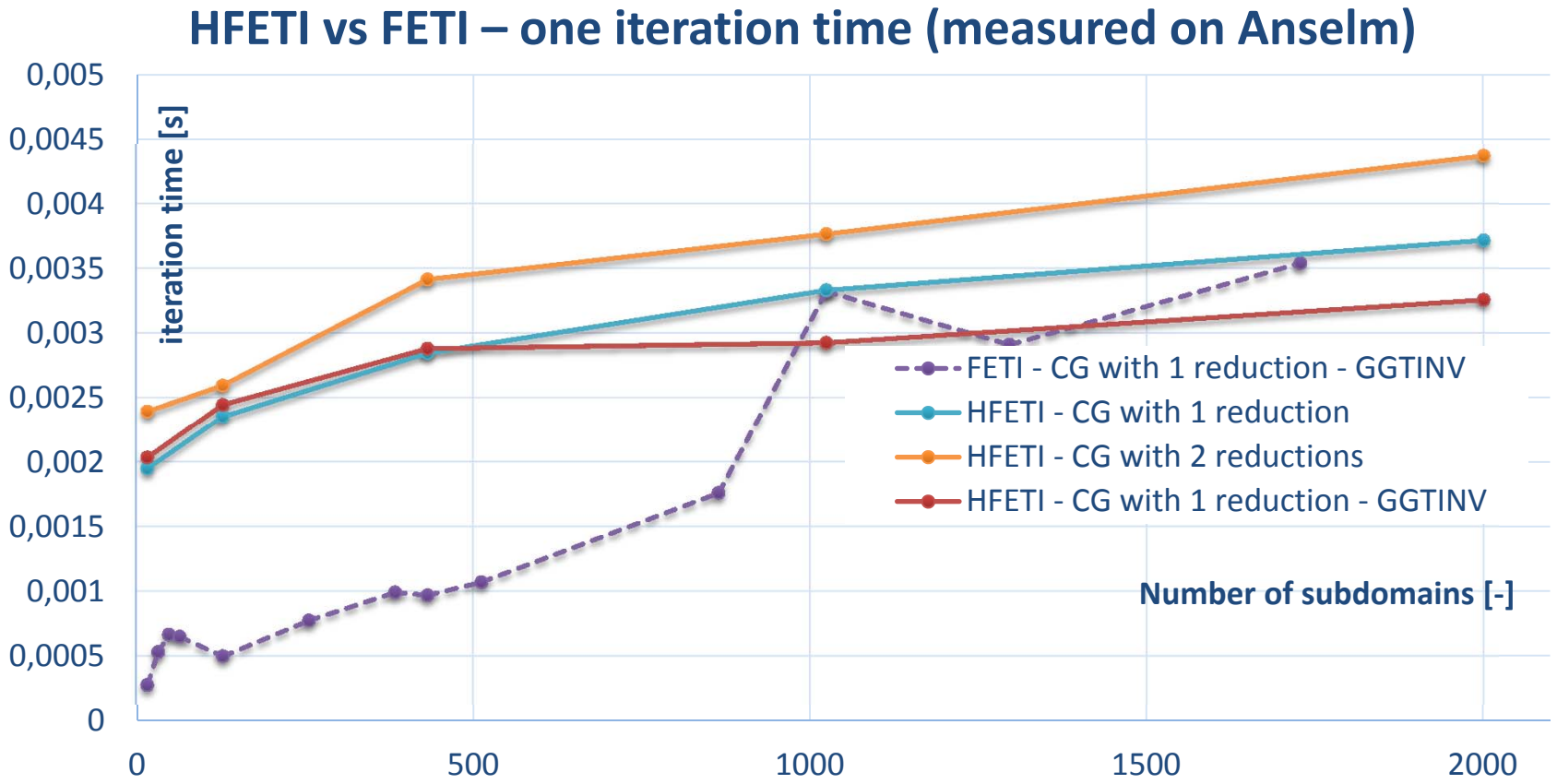
- Stage 1
  - optimization of global communication of FETI solver
    - using communication hiding and avoiding techniques
      - pipelined CG solvers
        - » single global reduction
      - solving coarse problem using distributed inverse matrix
        - » single global AllGather operation (instead of MPI_Gather and MPI_Scatter)
        - » parallel solve – each subdomain has only 6 dense rows of $(GG^T)^{-1}$
      - nearest neighbor communication optimization – multiplication with gluing matrix
        - » B for FETI and B1 for HFETI (next slide)
        - » fully scalable with groving number of compute nodes
- Stage 2
  - optimization of inter cluster processing
  - efficient solution of inner coarse problem
- Stage 3
  - combine both together

Stage 3 – HFETI vs FETI



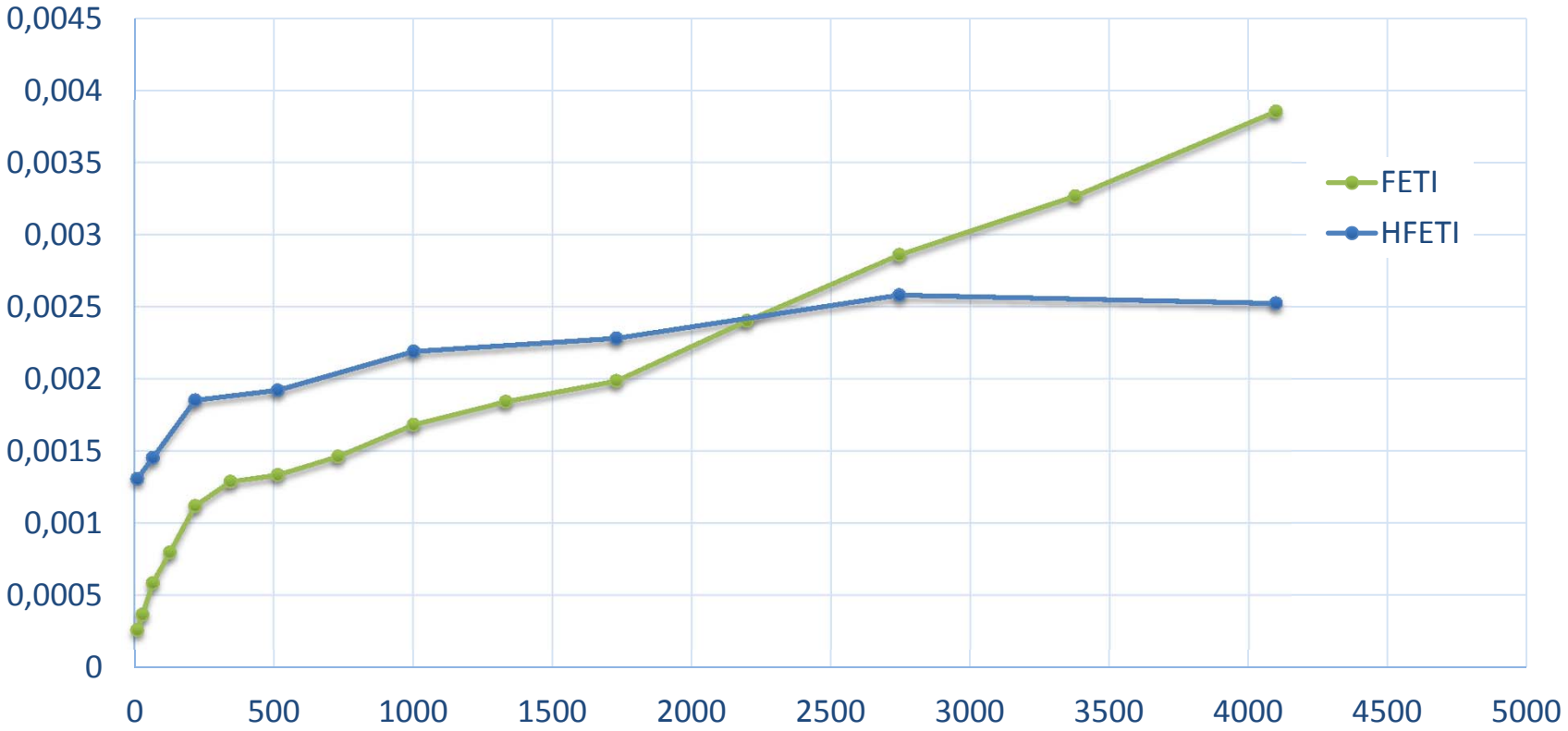
Weak scaling – domain size is fixed; cluster size is fixed; number of domains goes from 1 to 2000
- domain size $5^3 = 3*(5+1)^3 = 648$
- cluster size is 16 domains (1 cluster per node; each node has 16 cores - two 8-core CPUs)

HFETI vs FETI - Performance



Weak scaling – domain size is fixed; cluster size is fixed; number of domains goes from 1 to 2000
- domain size $5^3 = 3*(5+1)^3 = 648$
- cluster size is 16 (1 cluster per node; each node has 16 cores - two 8-core CPUs)
- efficient communication algorithms helps both FETI and HTFETI
  - **CG with 1 reduction + coarse problem solved using distributed inverse matrix (GGTINV)**

HFETI vs FETI - Performance



- cluster size is 8 – (3 clusters per node; each node has 24 cores - two 12-core CPUs)
- comparison of (1) FETI and (2) HFETI, where both FETI a HFETI uses CG with 1 reduction and coarse problem is solved using distributed inverse matrix of GG^T

Conclusions

The solver has the following main features:

- support for Total FETI and Hybrid FETI domain decomposition iterative methods
- support for discretization of static and dynamic linear elasticity problems
  - assembly of the stiffness/mass matrices of unstructured meshes using tetra4, prism6, pyramid5, brick8, tetra10, prism13,pyramid15, and brick20 elements
- lumped local subdomain pre-conditioner
- Implementation of the main iterative solver using the following Krylov solvers
  - regular Conjugate gradient method (CG) with or without the lumped pre-conditioner
  - pipelined CG with or without the lumped pre-conditioner
- coarse problem processing
  - solved sequentially on a master node using the LU decomposition
  - solved in parallel using a distributed inverse matrix
- support for Intel MIC many-core architecture
  - entire solver can run on a (1) single accelerator or a (2) cluster of accelerators using Intel MPI