

# Distributed fast boundary element method

Michal Merta, Jan Zapletal, Michal Kravčenko

michal.merta@vsb.cz, jan.zapletal@vsb.cz, michal.kravcenko@vsb.cz

## BOUNDARY ELEMENT METHOD

The idea of the boundary element method (BEM) is to reformulate the volume PDE as an equivalent boundary integral equation.

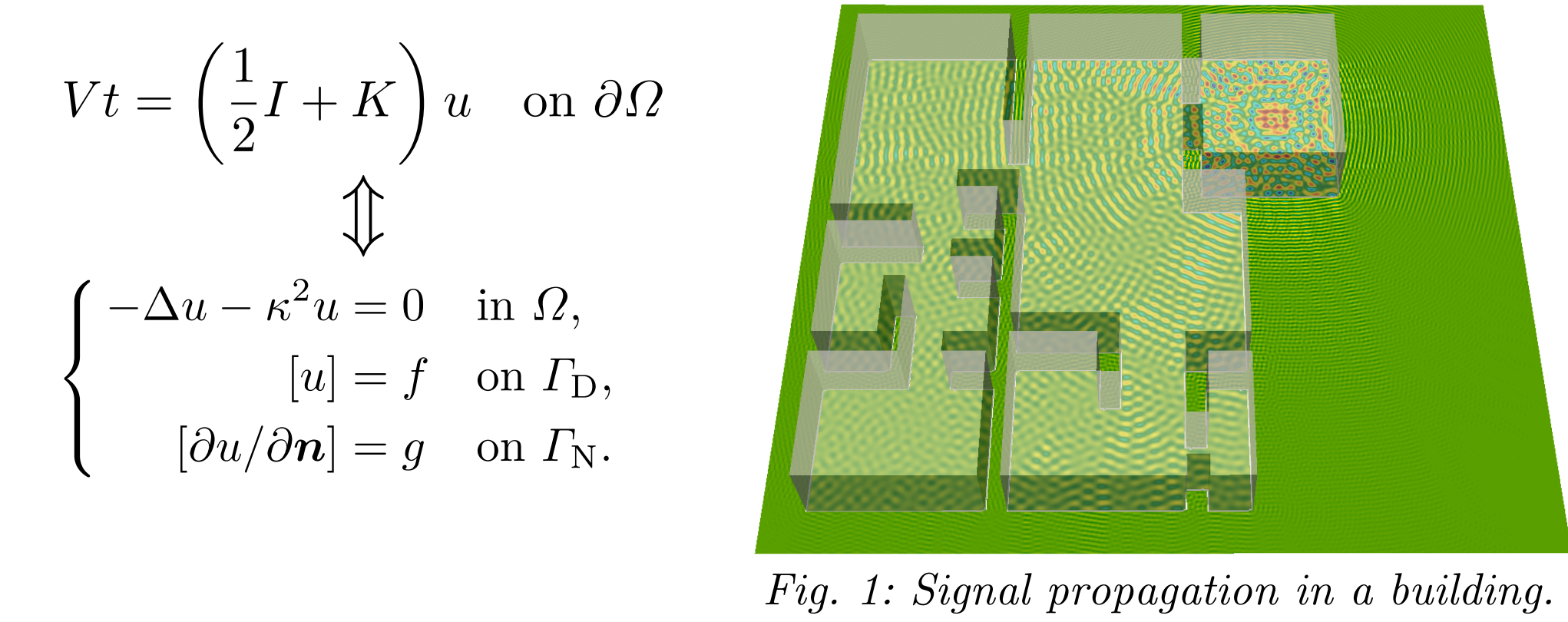


Fig. 1: Signal propagation in a building.

Reduction of a problem to the boundary of the domain makes the method well suited for problems on unbounded domains (e.g., **sound** or **electromagnetic wave scattering**) or **shape optimization problems**.

## THE BEM4I LIBRARY

BEM4I is a library of parallel BEM solvers developed at IT4Innovations. It supports solution of the **Laplace**, **Lamé**, **Helmholtz**, and **wave** equations (see Fig. 2). To enable solution of large scale problems the **adaptive cross approximation** (ACA) is used.

- To utilize modern HPC hardware we employ
- **OpenMP SIMD vectorization** for evaluation of singular integrals,
  - **OpenMP threading** for local element contributions,
  - **MPI-distributed ACA**,
  - **MPI** for BETI (with the domain decomposition ESPRESO lib. [5]),
  - **offload** to Intel Xeon Phi coprocessors [1].

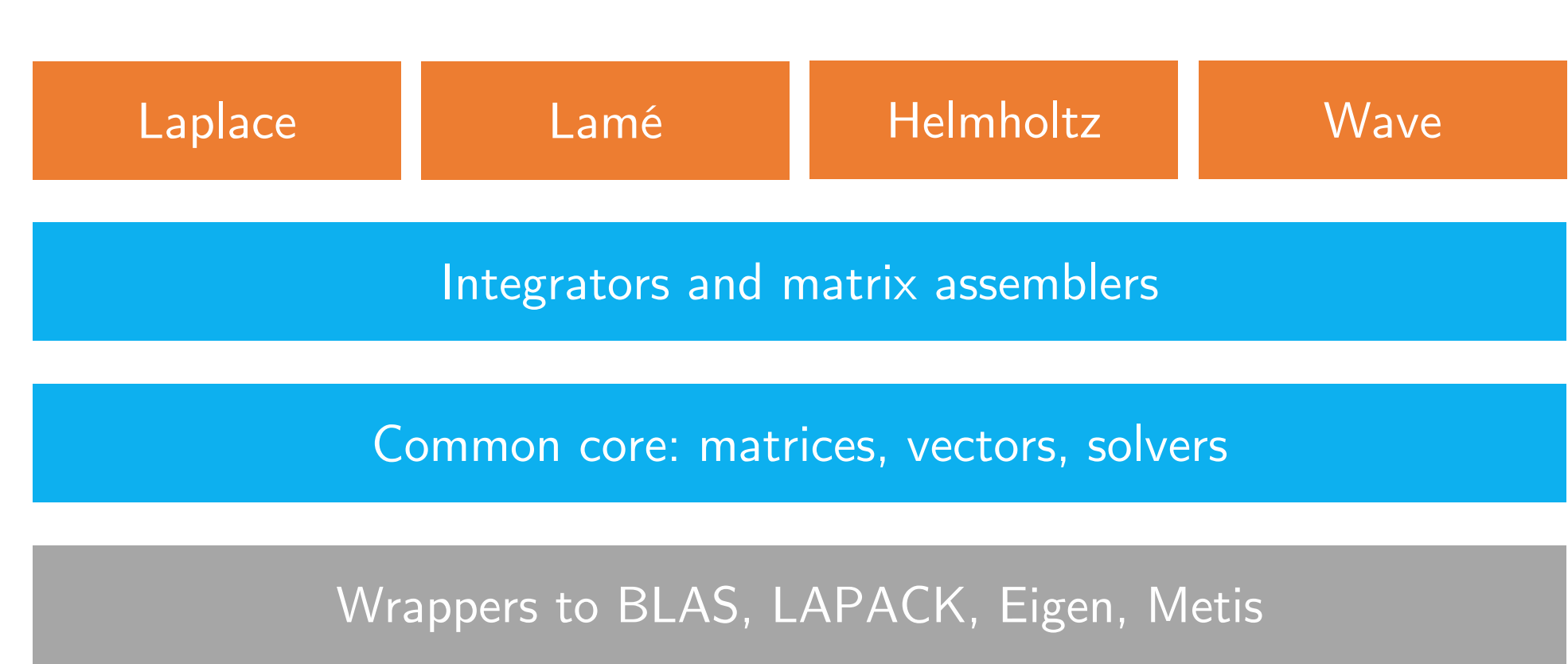
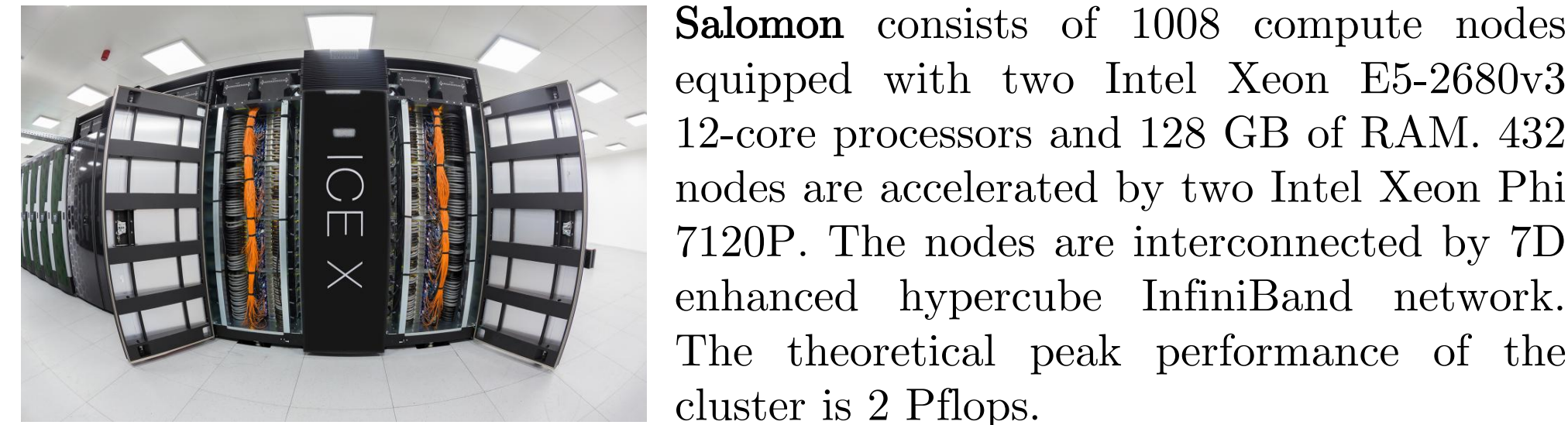


Fig. 2: Structure of the BEM4I library.

## ENVIRONMENT

The numerical experiments were performed on the Salomon supercomputer at IT4Innovations, Marconi at Cineca and Taurus at TU Dresden.



**Marconi** (Cineca) is a Tier-0 system based on the Lenovo NeXtScale platform. Its A2 partition consists of 3600 nodes equipped with 68-core Intel Xeon Phi 7250 Knights Landing CPUs (in total 11 Pflops).

**Taurus** (TU Dresden) contains 32 nodes with 64-core Intel Xeon Phi 7210.

## ADAPTIVE CROSS APPROXIMATION

To reduce computational complexity and memory requirements, the **fully populated** BEM system matrices can be **approximated** using, e.g., the adaptive cross approximation method (ACA) [6].

```
k = 1; i_k = 1; U = {}
repeat
  if k > 1 then
    i_k := arg max_{i ∈ Z} |u_{k-1}(i)|
  end if
  ṽ := A(i_k, :)
  for ℓ := 1, ..., k - 1 do
    ṽ_k := ṽ_k - u_ℓ(i_k) v_ℓ
  end for
  Z := Z ∪ {i_k}
  if ṽ does not vanish then
    j_k := arg max_j |ṽ_k(j)|
    v_k := ṽ_k(j_k)^-1 ṽ_k
    u_k := A(:, j_k)
    for ℓ := 1, ..., k - 1 do
      u_k := u_k - v_ℓ(j_k) u_ℓ
    end for
    k := k + 1
  end if
until stopping criterion fulfilled
```

Alg. 1. The ACA algorithm

Alg. 1 generates sequence of crosses of rows and columns and approximates the input matrix by the matrix

$$S_k := \sum_{\ell=1}^k \mathbf{u}_\ell \mathbf{v}_\ell^\top = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k][\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k]^\top$$

## MODIFIED ACA ALGORITHM

The original **ACA algorithm** **fails** when the input matrix contains **zero blocks**. Partial pivoting only selects pivots from one non-zero block. In BEM, this can happen when clusters lie on the same plane (see Fig. 3).

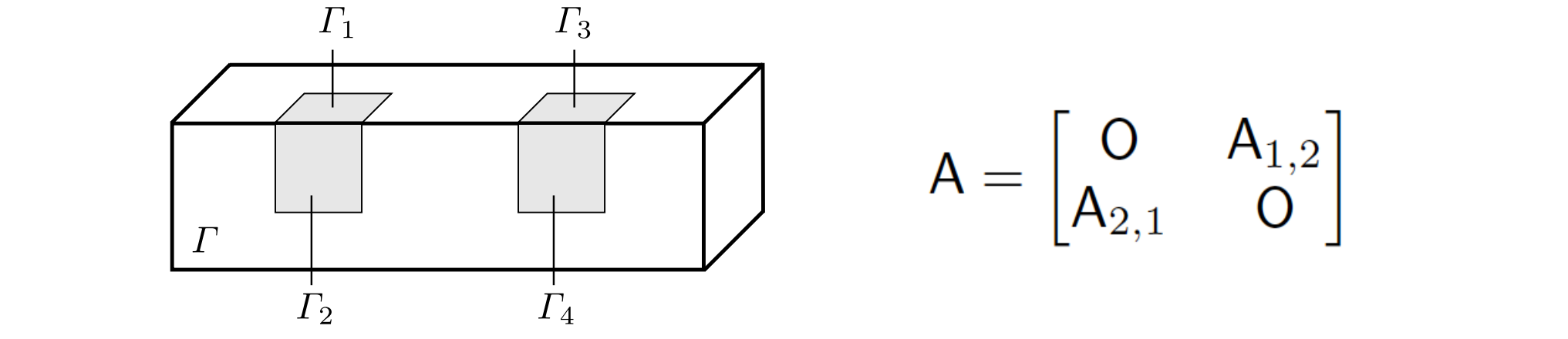


Fig. 3. Problematic pair of clusters and the corresponding block matrix.

Our modification of the ACA algorithm identifies the problematic blocks in advance, assembles corresponding reference rows and columns, and introduces outer loop with stopping criterion based on their residuals [7].

|           |               | Original ACA |                      | Modified ACA |                      |
|-----------|---------------|--------------|----------------------|--------------|----------------------|
| $N$       | $\mathcal{E}$ | $K_h$ [%]    | Error                | $K_h$ [%]    | Error                |
| 10,722    | $10^{-4}$     | 52.3         | $2.51 \cdot 10^{-2}$ | 52.9         | $2.51 \cdot 10^{-2}$ |
| 42,888    | $10^{-4}$     | 19.7         | $1.35 \cdot 10^{-2}$ | 20.1         | $1.17 \cdot 10^{-2}$ |
| 171,552   | $10^{-5}$     | 7.6          | $1.12 \cdot 10^{-2}$ | 7.8          | $5.62 \cdot 10^{-3}$ |
| 686,208   | $10^{-6}$     | 4.0          | $4.66 \cdot 10^{-2}$ | 4.1          | $2.75 \cdot 10^{-3}$ |
| 2,744,832 | $10^{-7}$     | 2.7          | $8.96 \cdot 10^{-2}$ | 2.8          | $1.41 \cdot 10^{-3}$ |

Tab. 1. Compression and accuracy of the original and modified ACA tested using the interior Helmholtz problem and the geometry from Fig. 5.

|                           | Original ACA | Modified ACA |
|---------------------------|--------------|--------------|
| Planes identification [s] | –            | 0.12         |
| $V_h$ assembly [s]        | 84.9         | 87.3         |
| $K_h$ assembly [s]        | 111.8        | 118.0        |

Tab. 2. Assembly times using the original and modified ACA for a problem with 686 208 surface elements.

## VECTORIZATION AND SHARED MEMORY PARALLELIZATION

To generate row and columns for Alg. 1 one has to deal with evaluation of singular integrals of the type

$$V_h[\ell, k] := \frac{1}{4\pi} \int_{\tau_\ell} \int_{\tau_k} \frac{1}{\|x - y\|} ds_y ds_x.$$

BEM4I employs various techniques to efficiently utilize wide SIMD registers of modern CPUs [2,3]:

- **OpenMP SIMD pragmas**,
- **data alignment and padding**,
- **AoS to SoA** transition,
- **unit-strided** memory access,
- **collapsing** loops,
- **data duplication** to ensure unit-strided memory access.

Moreover, assembly of individual blocks is distributed among available **OpenMP threads**. Every block is assembled by a single OpenMP thread to avoid data races.

Scalability of the SIMD vectorization and shared memory threading is shown in Fig. 4.

## MASSIVELY PARALLEL ACA BEM

In [4, 7] we present a distributed version of ACA distributing matrix blocks among nodes using **cyclic graph decompositions** (see Fig. 5 and 6).

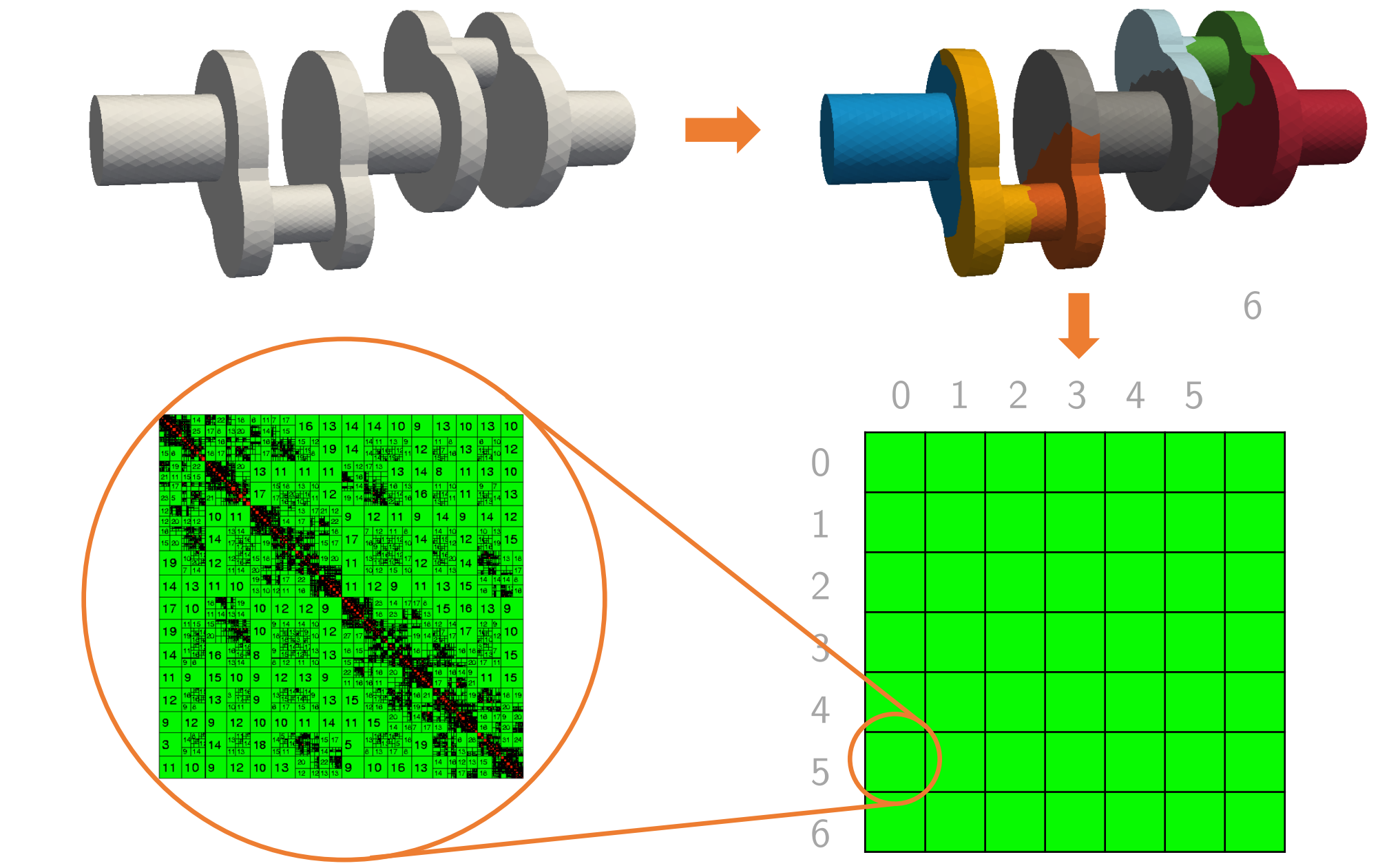


Fig. 5. Decomposition of the original mesh induces block structure of the system matrix. Each block is approximated using low-rank matrix.

Matrix blocks are distributed among MPI ranks such that the number of submeshes per rank is minimized and each rank owns one diagonal block.

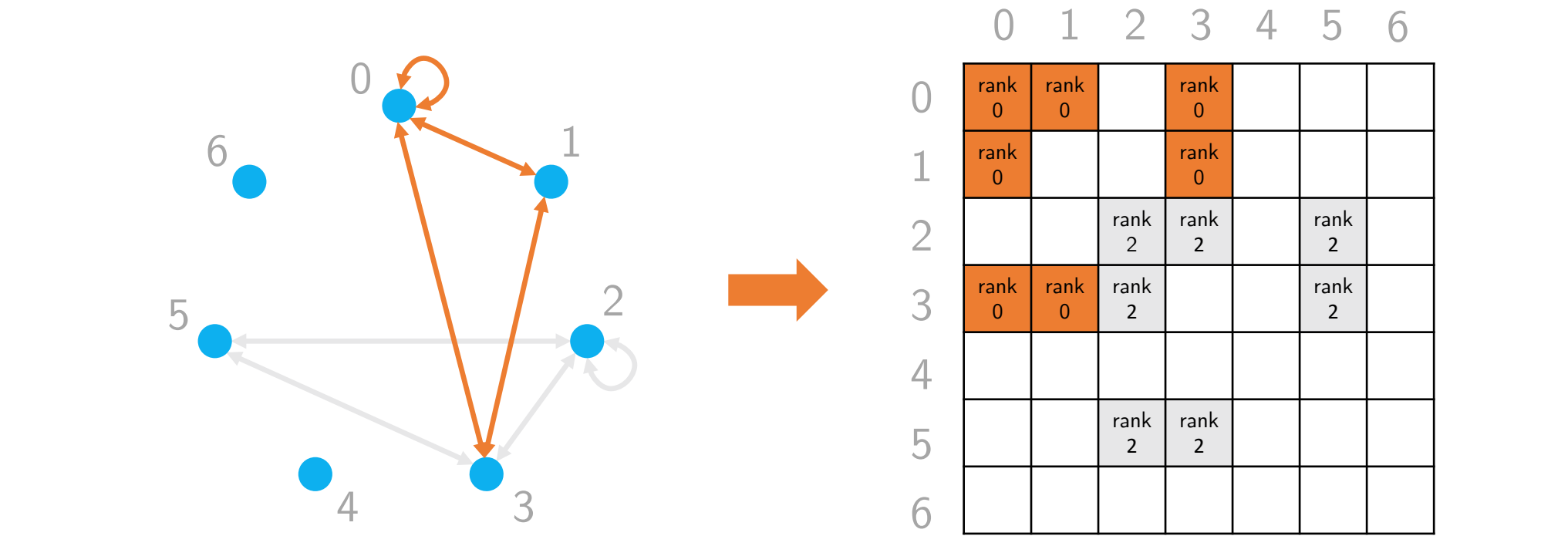


Fig. 6. Distribution of the blocks based on the cyclic graph decomposition.

## SCALABILITY FOR THE HELMHOLTZ EQUATION

We solve exterior Dirichlet problem for the Helmholtz equation corresponding to **sound scattering off a perforated plate** with dimensions  $6 \times 5 \times 0.15$  m (see Fig. 7). A point source is located one meter away and the wave-number is  $\kappa = 16$  (which corresponds to 873 Hz).

Strong scalability was tested on up to 256 nodes of the Salomon cluster (see Fig. 8).

Fig. 7. Sound scattering off a perforated plate.

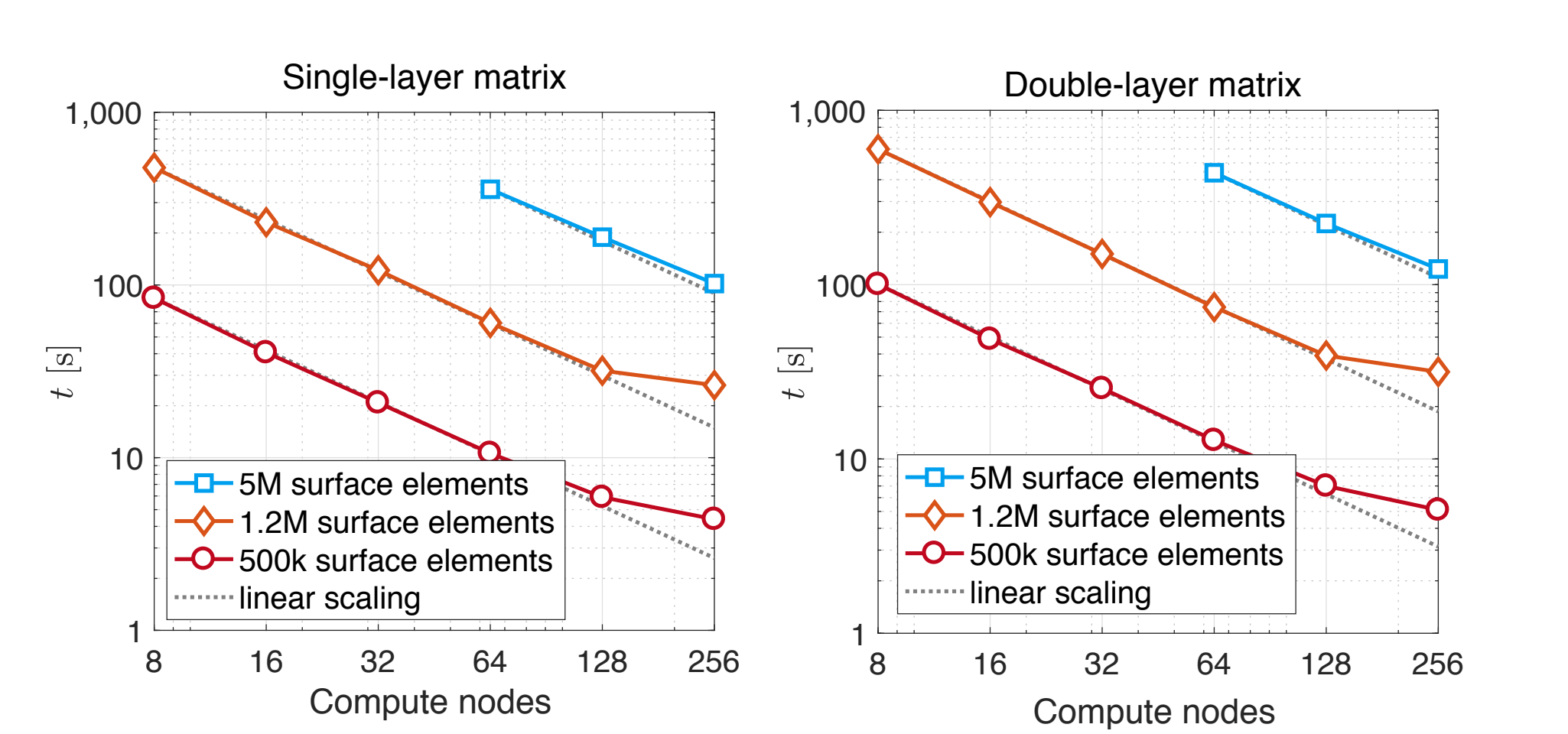


Fig. 8. Strong scalability of the distributed assembly of the system matrices on the Salomon cluster.

## HALF-SPACE SOUND SCATTERING PROBLEM

We test the performance on the **half-space sound scattering problem** simulating the influence of a **sound barrier** (see Fig. 9).

The geometry is discretized into **5 259 264 surface elements**. The size of the building is  $30 \times 8 \times 15$  m. The barrier has the radius 50 m and height 4 m. A point source is located 51 m from the building, producing time-harmonic spherical waves with  $\kappa = 4$ , i.e.,  $f = 218$  Hz. See Tab. 3.

| # compute nodes / MPI ranks | 256 / 512   |
|-----------------------------|-------------|
| $V_h$ assembly [s]          | 116.23      |
| $K_h$ assembly [s]          | 507.00      |
| $V_h$ memory [% / TB]       | 1.37 / 6.03 |
| $K_h$ memory [% / TB]       | 2.00 / 4.38 |
| time per iteration [s]      | 0.86        |

Tab. 3. Distributed solution of the half-space sound scattering problem.



Fig. 9. Sound scattering off a barrier in a half-space.

## CONCLUSION

The presented distribution scheme proves to be scalable during assembly of the system matrices, but also reduces communication necessary during the matrix-vector multiplication. Therefore, in combination with a communication hiding Krylov solver and a suitable parallel preconditioner, it could be applied to massively parallel solution of problems with tens of millions surface elements.

### References

- [1] Merta, M.; Zapletal, J.; Jaros, J. *Many Core Acceleration of the Boundary Element Method*. LNCS 9611, 2016.
- [2] Zapletal, J.; Merta, M.; Malý, L. *Boundary Element Quadrature Schemes for Multi- and Many-Core Architectures*. Comput. Math. Appl. 74, 2017.
- [3] Zapletal, J.; Of, G., Merta, M. *Parallel and vectorized implementation of boundary integral operators*. Submitted.
- [4] Lukas, D. et al. *A parallel fast boundary element method using cyclic graph decompositions*. Numer. Alg. 70, 2015.

- [5] Riha, L. et al. *Massively Parallel Hybrid Total FETI (HTFETI) Solver*. PASC16, ACM, 2016.
- [6] Bebendorf, M. *Approximation of boundary element matrices*. Numer. Math. 86, 2000.
- [7] Kravcenko, M.; Merta, M.; Zapletal, J. *Distributed Fast Boundary Element Methods for Helmholtz Problems*. Submitted.



### Acknowledgments

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project 'IT4Innovations excellence in science – LQ1602', from the Large Infrastructures for Research, Experimental Development and Innovations project 'IT4Innovations National Supercomputing Center – LM2015070', and by VSB-TU Ostrava under the grant SP2017/165.