Excalibur SLE
Exascale Computing for System-Level Engineering
Workshop on Software Engineering 14-15 July 2020

ExaFMM—10+ years, 7 re-writes

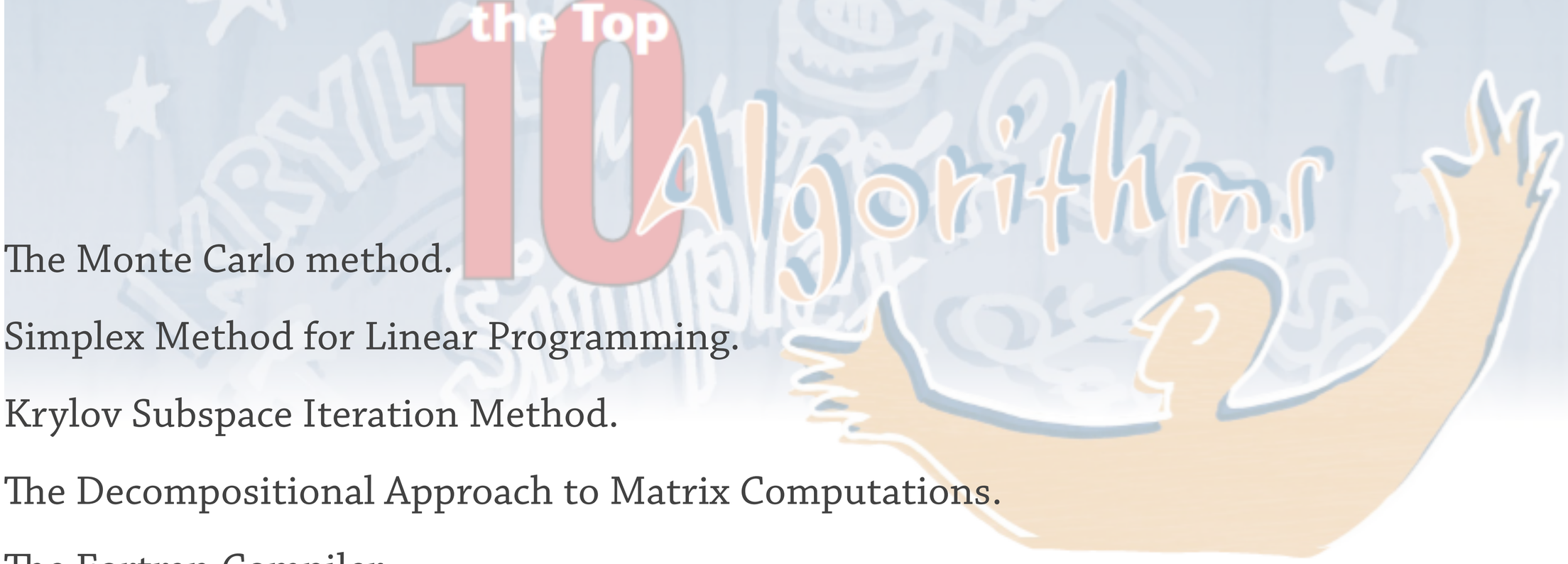
The tortuous progress of computational research

@LorenaABarba

ExaFMM

- Library implementing the fast multipole method
- High-performance: C++ / MPI / OpenMP
- 7th FMM implementation written in the group!

https://github.com/exafmm

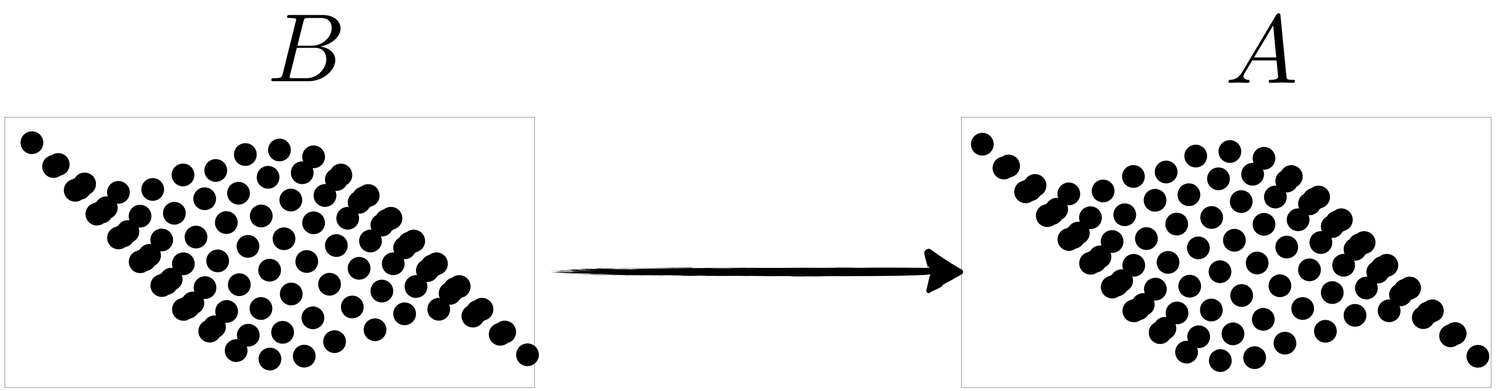


- 1946 — The Monte Carlo method.
- 1947 — Simplex Method for Linear Programming.
- 1950 — Krylov Subspace Iteration Method.
- 1951 — The Decompositional Approach to Matrix Computations.
- 1957 — The Fortran Compiler.
- 1959 — QR Algorithm for Computing Eigenvalues.
- 1962 — Quicksort Algorithms for Sorting.
- 1965 — Fast Fourier Transform.
- 1977 — Integer Relation Detection.
- 1987 — **Fast Multipole Method**

Dongarra& Sullivan, IEEE Comput. Sci. Eng., Vol. 2(1):22-- 23 (2000)

Gravitational potential

$$\phi_i = \sum_{j=0}^{N} m_j \cdot \mathbb{K}(\mathbf{x}_i, \mathbf{y}_j)$$

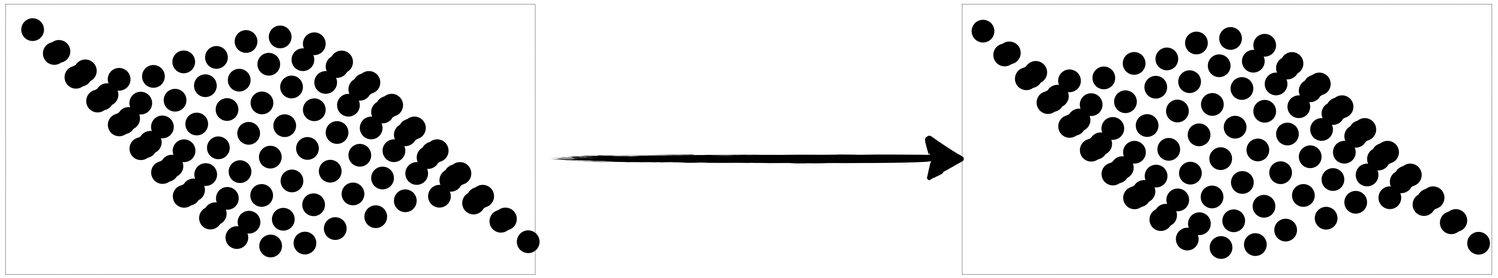


M31 Andromeda

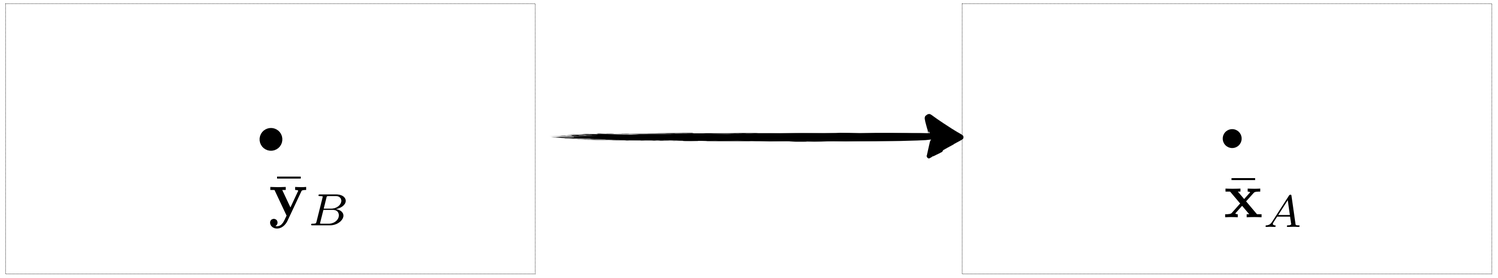
Milky Way

$$\phi_i = \sum_{j=0}^{N} m_j \cdot \mathbb{K}(\mathbf{x}_i, \mathbf{y}_j)$$

B A

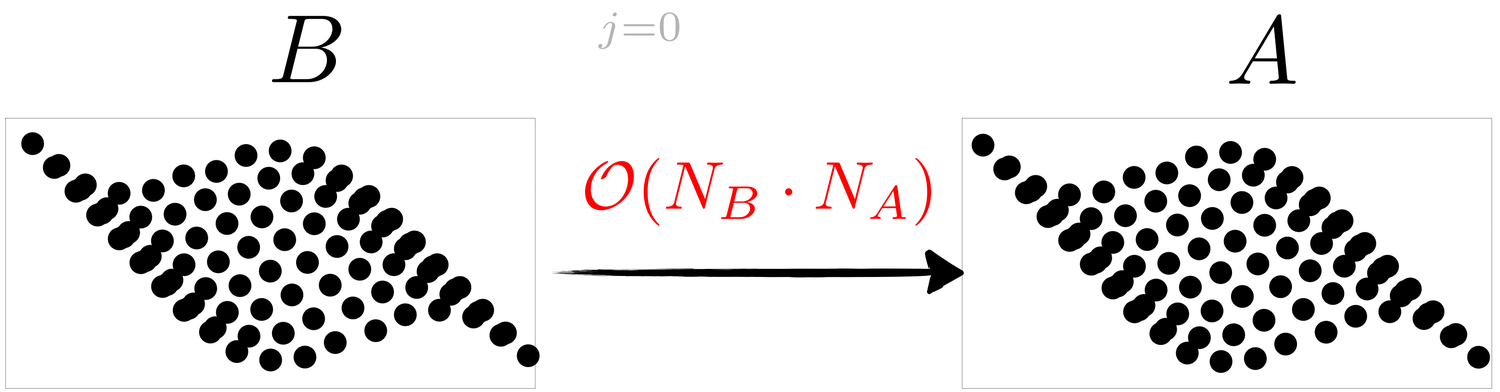


$$m_B = \sum_{j=0}^{N} m_j$$

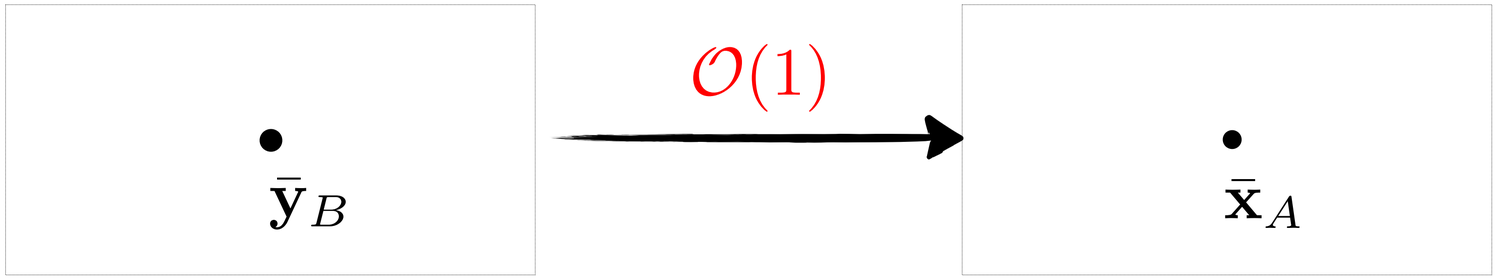


$$\phi_A = m_B \cdot \mathbb{K}(\bar{\mathbf{x}}_A, \bar{\mathbf{y}}_B)$$

$$\phi_i = \sum_{j=0}^{N} m_j \cdot \mathbb{K}(\mathbf{x}_i, \mathbf{y}_j)$$

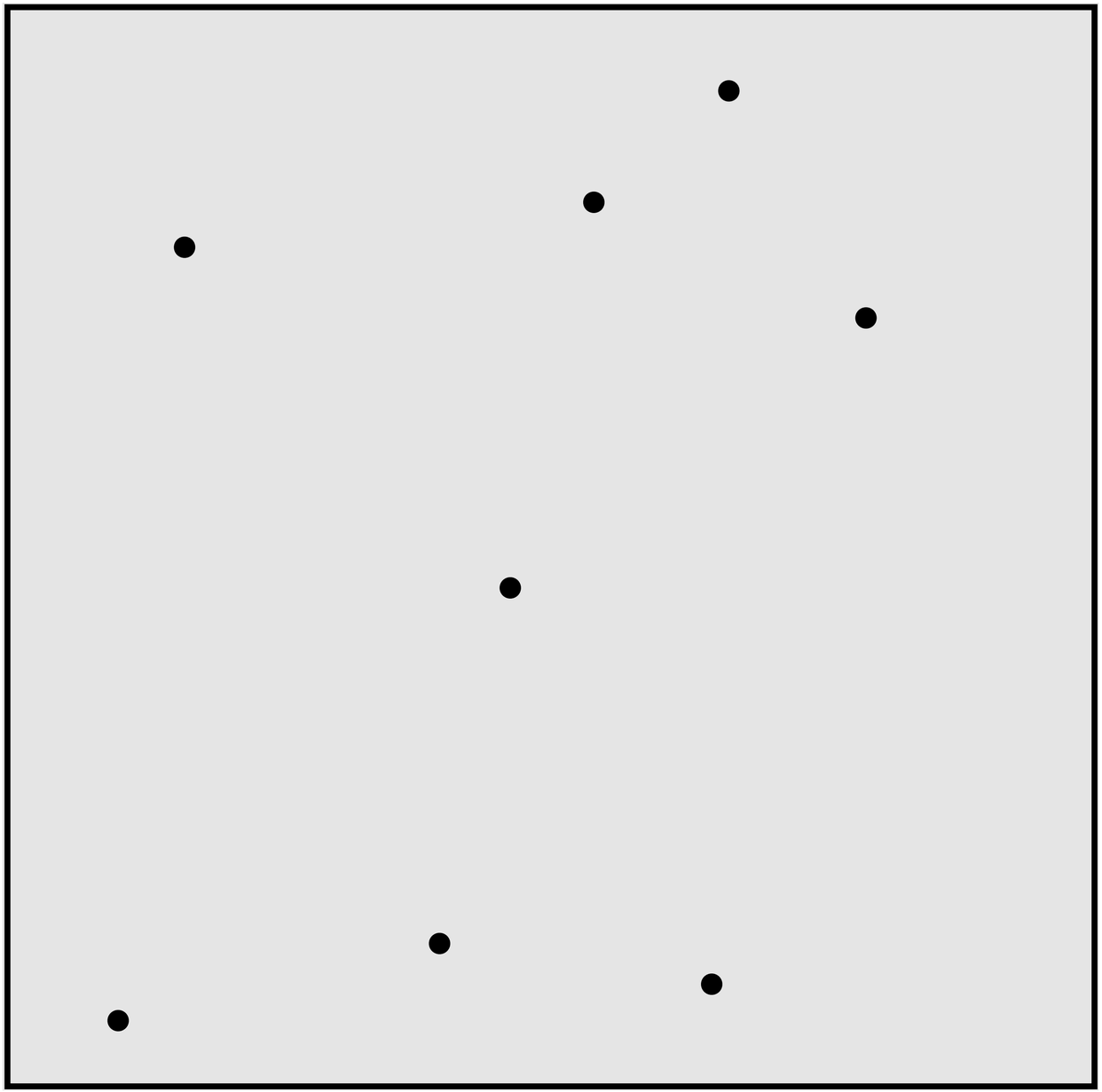


$$m_B = \sum_{j=0}^{N} m_j \quad \mathcal{O}(N_B) \qquad \mathcal{O}(N_A)$$

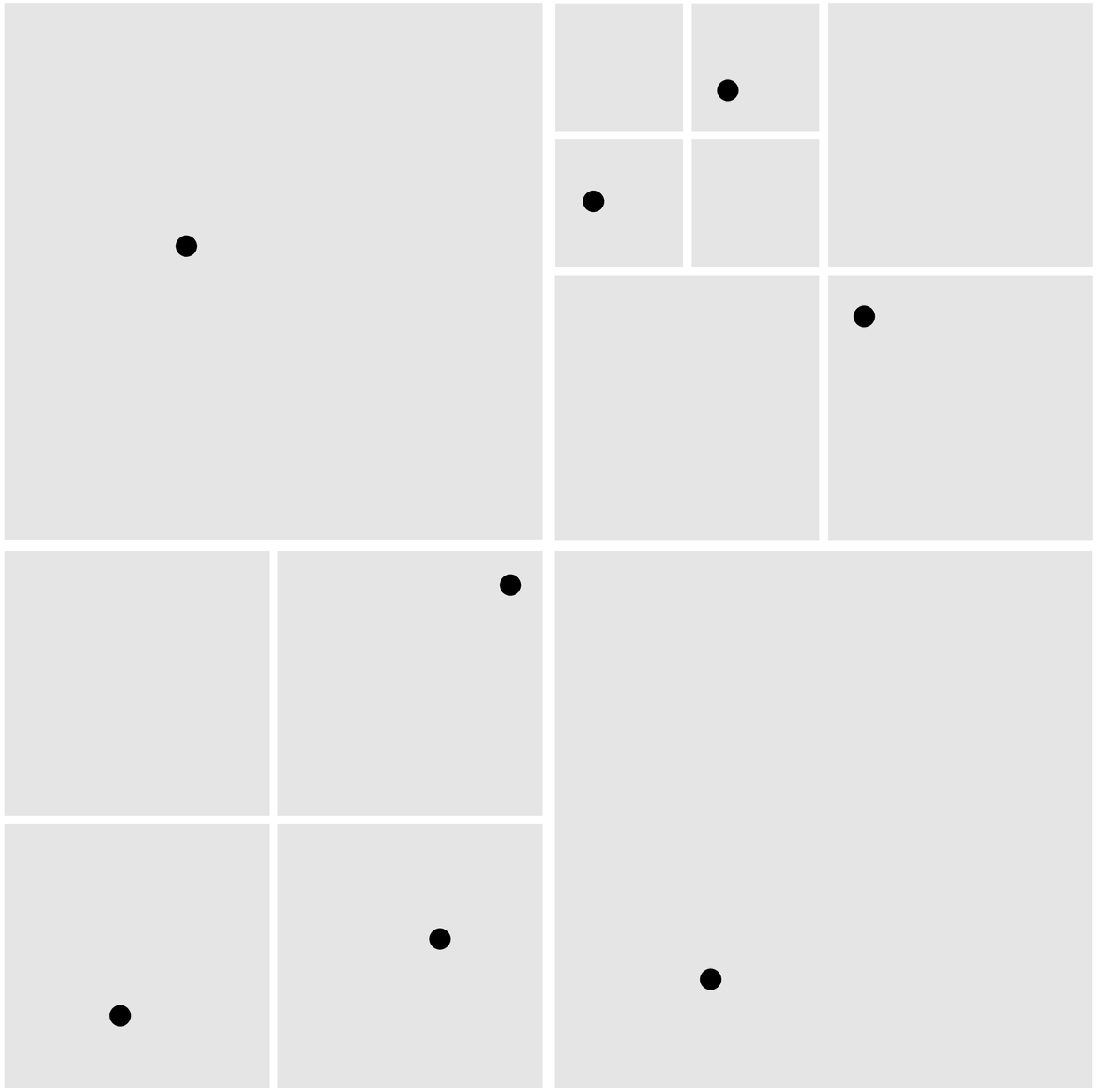


$$\phi_A = m_B \cdot \mathbb{K}(\bar{\mathbf{x}}_A, \bar{\mathbf{y}}_B)$$

Level 0

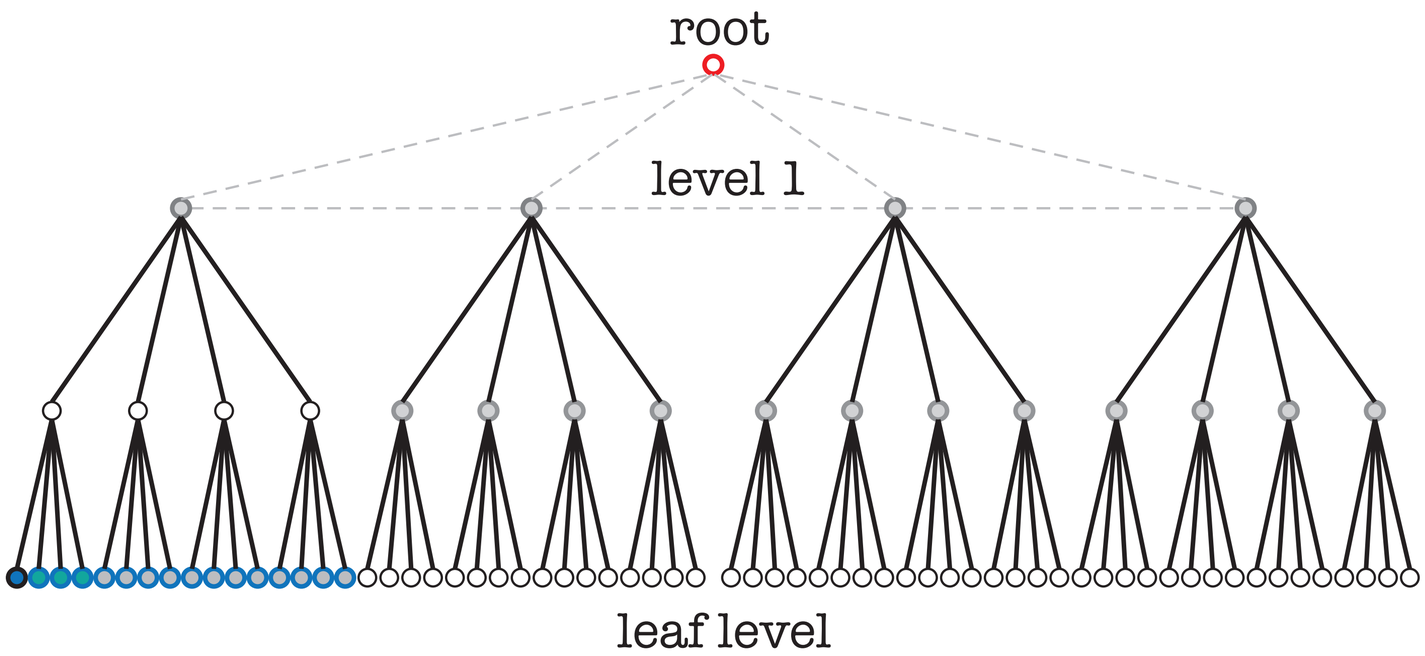


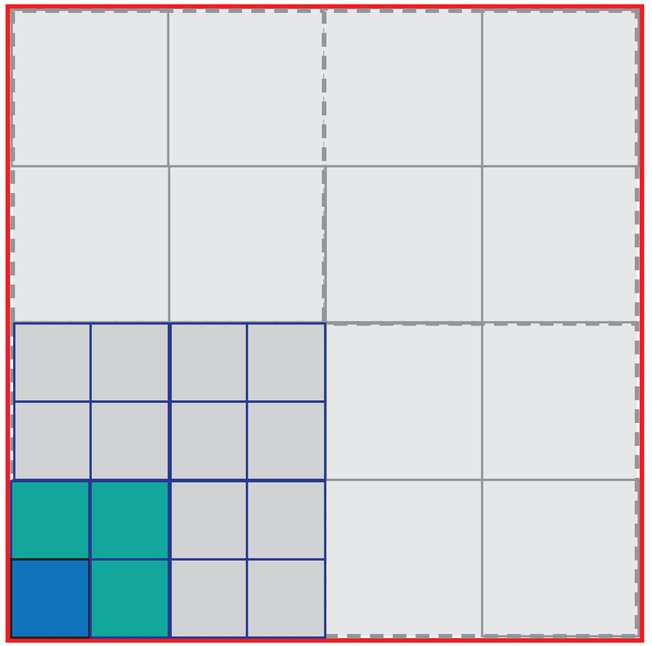
Source: http://arborjs.org/docs/barnes-hut

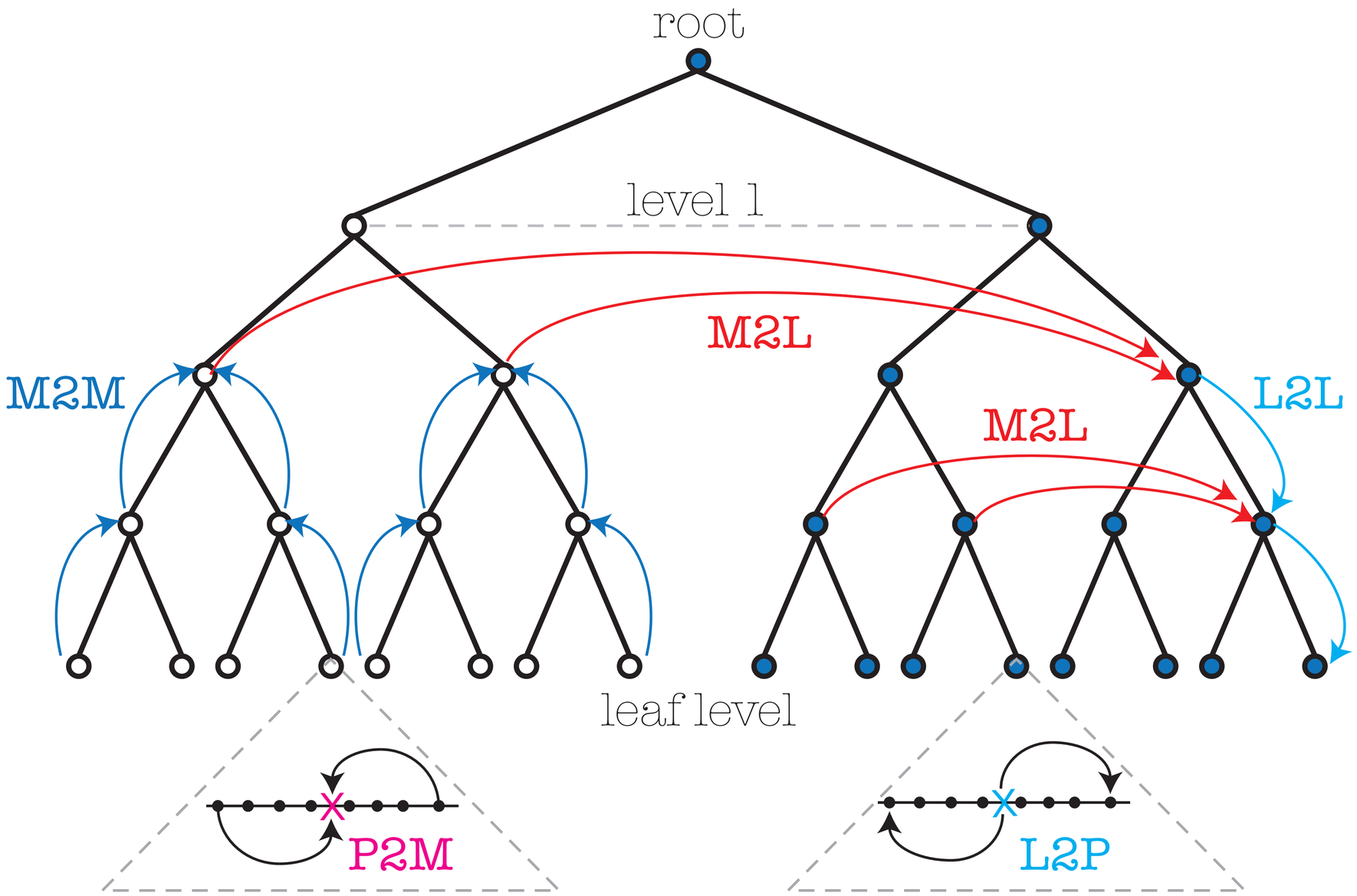


Source: http://arborjs.org/docs/barnes-hut

Well-separated regions



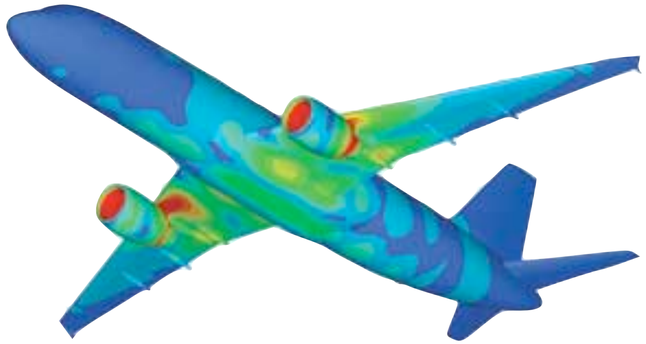




"A tuned and scalable fast multipole method as a preeminent algorithm for exascale systems", Rio Yokota, L A Barba. *Int. J. High-perf. Comput*. 26(4):337–346 (November 2012)

Turn the PDE to an integral equation:

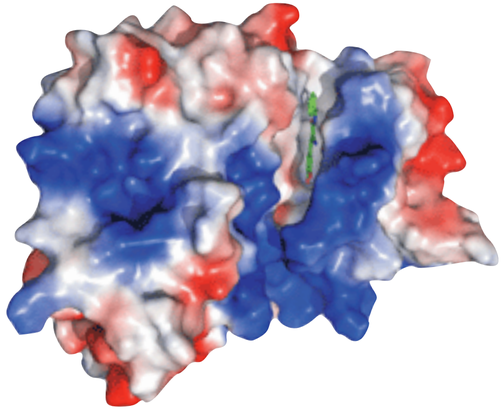
$$\nabla^2 u = f \quad \downarrow \quad u = \int_\Omega G f d\Omega$$



- Poisson

$$\nabla^2 u = -f$$

Astrophysics
Electrostatics
Fluid mechanics



- Helmholtz

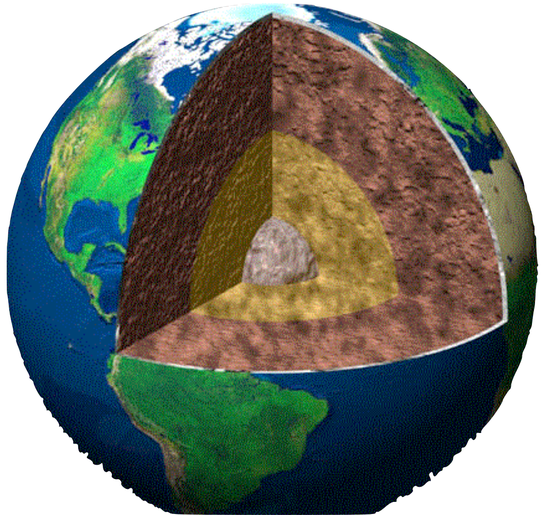
$$\nabla^2 u + k^2 u = -f$$

Acoustics
Electromagnetics

- Poisson-Boltzmann

$$\nabla \cdot (\epsilon \nabla u) + k^2 u = -f$$

Geophysics
Biophysics



Fast mat-vec:

accelerate iterations of Krylov solvers

speed-up Boundary Element Method (BEM) solvers

Lab-grown codes:

1. PyFMM (2008)—Python, 2D prototype, serial

   `https://github.com/barbagroup/pyfmm`

2. PetFMM (2009)—PETSc-based, 2D/3D, parallel

   `https://bitbucket.org/petfmm/`

3. GemsFMM (2010)—C CUDA, 3D

   `https://github.com/barbagroup/gemsfmm`

4. ExaFMM v.0.1 alpha (2011)—C++/MPI/CUDA

   `https://github.com/exafmm/exafmm-alpha`

5. FMMTL (2012)—C++/CUDA

   `https://github.com/ccecka/fmmtl`

ExaFMM v0.1 (alpha):

1. Inter-node parallelism with MPI
2. Intra-node multithreading with OpenMP
3. GPU-enabled using CUDA
4. strong scaling :10^8 particles on 2048 Cray procs.,93% parallel efficiency for the non-SIMD code
5. First release late 2011, continued dev to 2016...

Tree parallelization technique:

- “Local essential tree” (LET), from Salmon & Warren, 1993: *“A parallel hashed Oct-Tree N-body algorithm”*

https://doi.org/10.1145/169627.169640

SC18 Names Mike Warren & John Salmon Test of Time Award Winners for Their Paper "A Parallel Hashed Oct-Tree N-Body Algorithm"

June 27, 2018

by SC Insider

The SC Test of Time Award (ToTA) Committee announced the selection of "A Parallel Hashed Oct-Tree N-Body Algorithm" by John Salmon and Mike Warren as the SC18 ToTA winner.

But ExaFMM (v.0.1) is...

- over-engineered with many layers of class inheritance, to implement many features
- long and complex: ~12,000 Lines of Code (LoC)
- hard to maintain!

ExaFMM v.1 — 8 months, year-3 PhD student

1. Same parallel strategy using LET & analytical kernels
2. Added (module-level) tests & regression tests
3. Simplified everywhere, with good naming (matching mathematical equations), improved interfaces
4. Modularized: tree part totally separate from kernels
5. ~70% fewer LoC
6. Sixth implementation in the group!

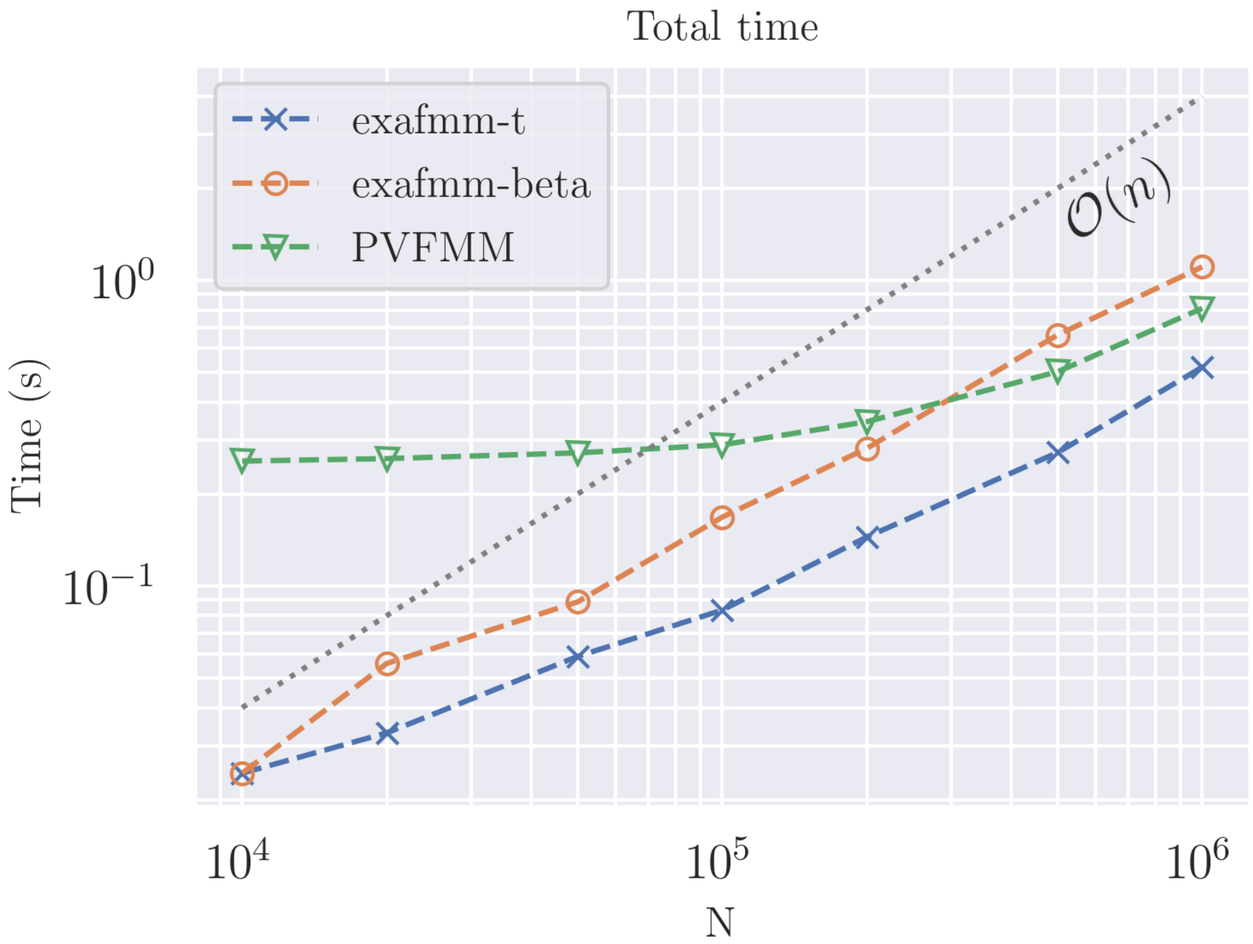
`https://github.com/exafmm/exafmm`

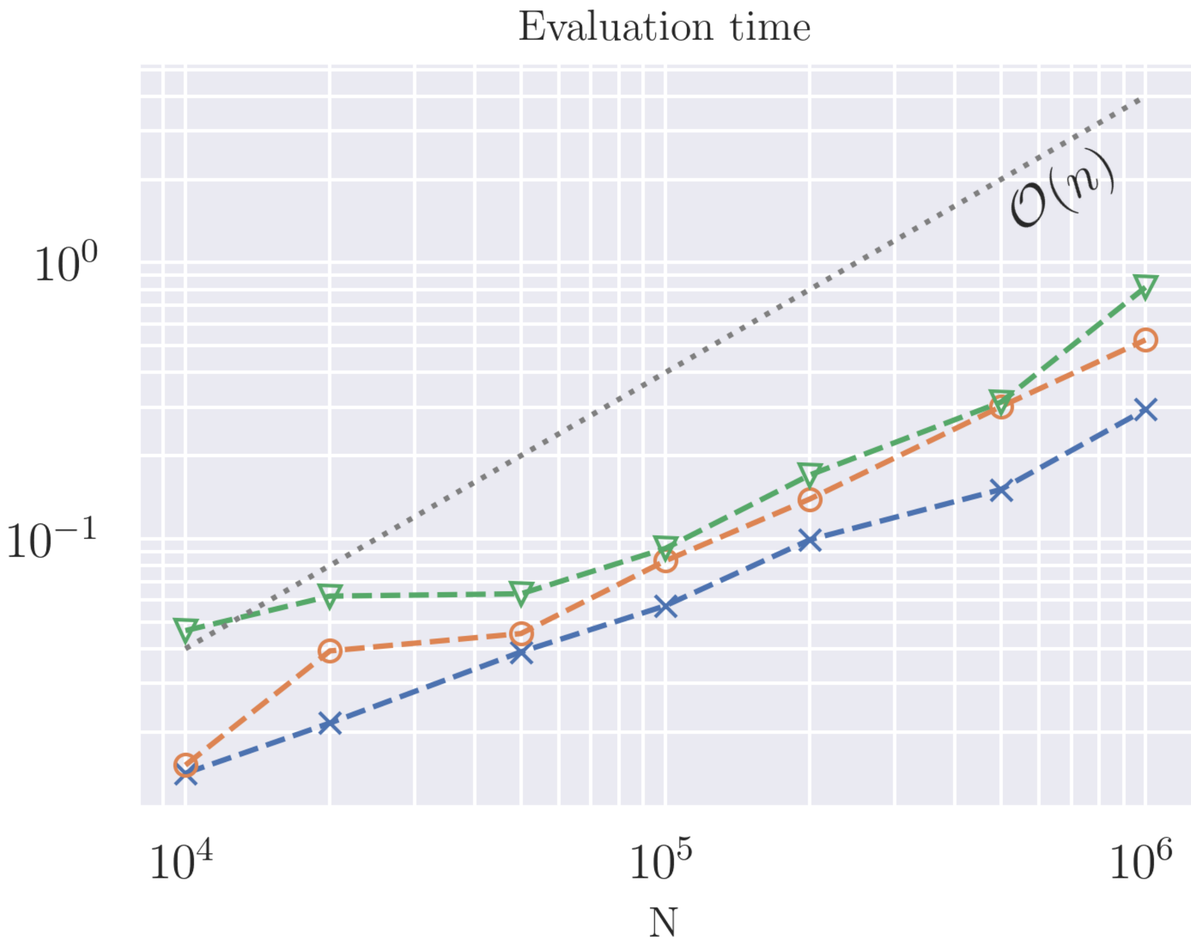
ExaFMM-t: same student, 2 more years!

1. Kernel-independent version of the algorithm
2. Goal: reusable, standard code; C++ fancy stuff: *in moderation!*
3. Shallow inheritance, conservative use of classes, clean function interfaces
4. Easy to extend
5. Faster or competitive with state-of-the art

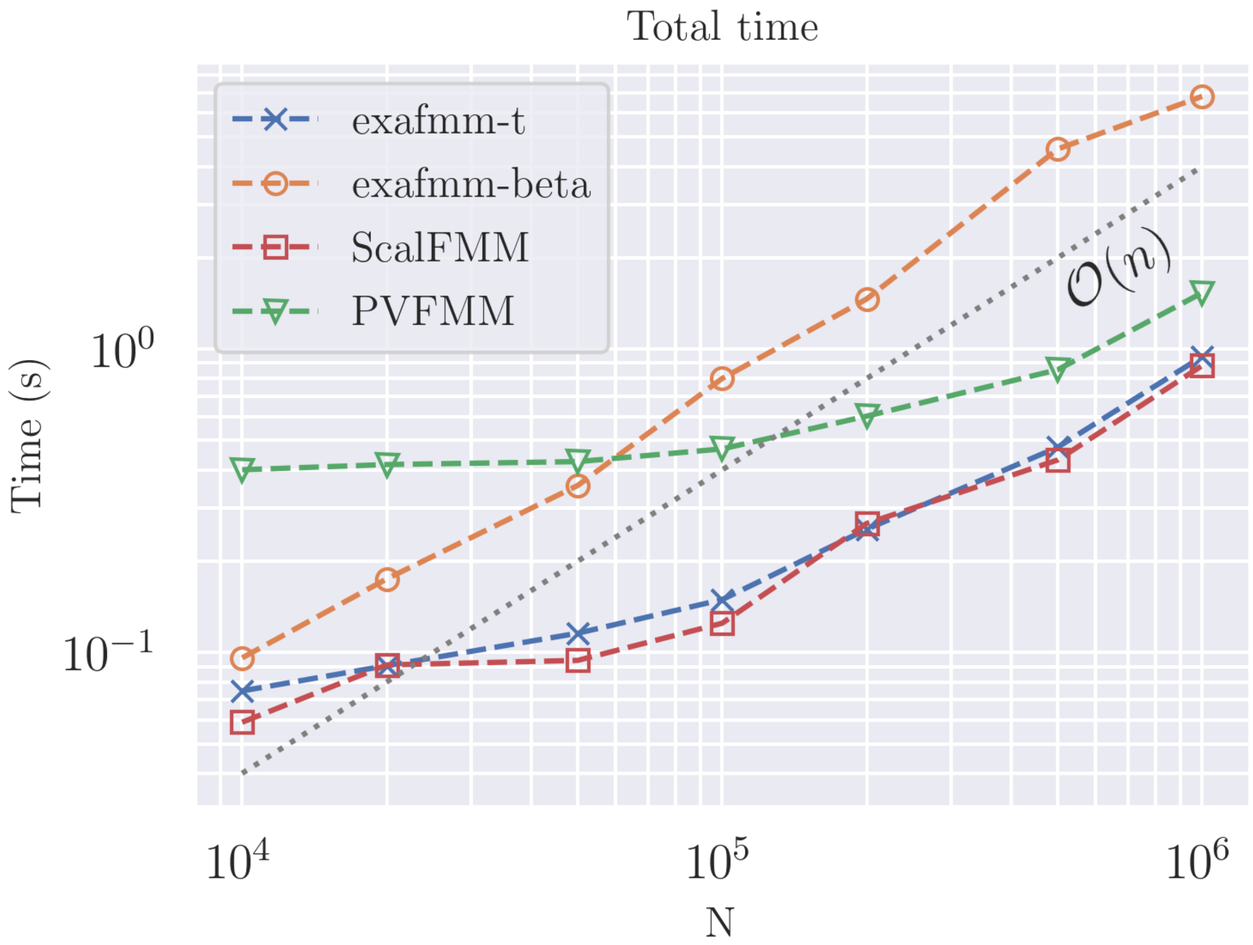
`https://github.com/exafmm/exafmm-t`

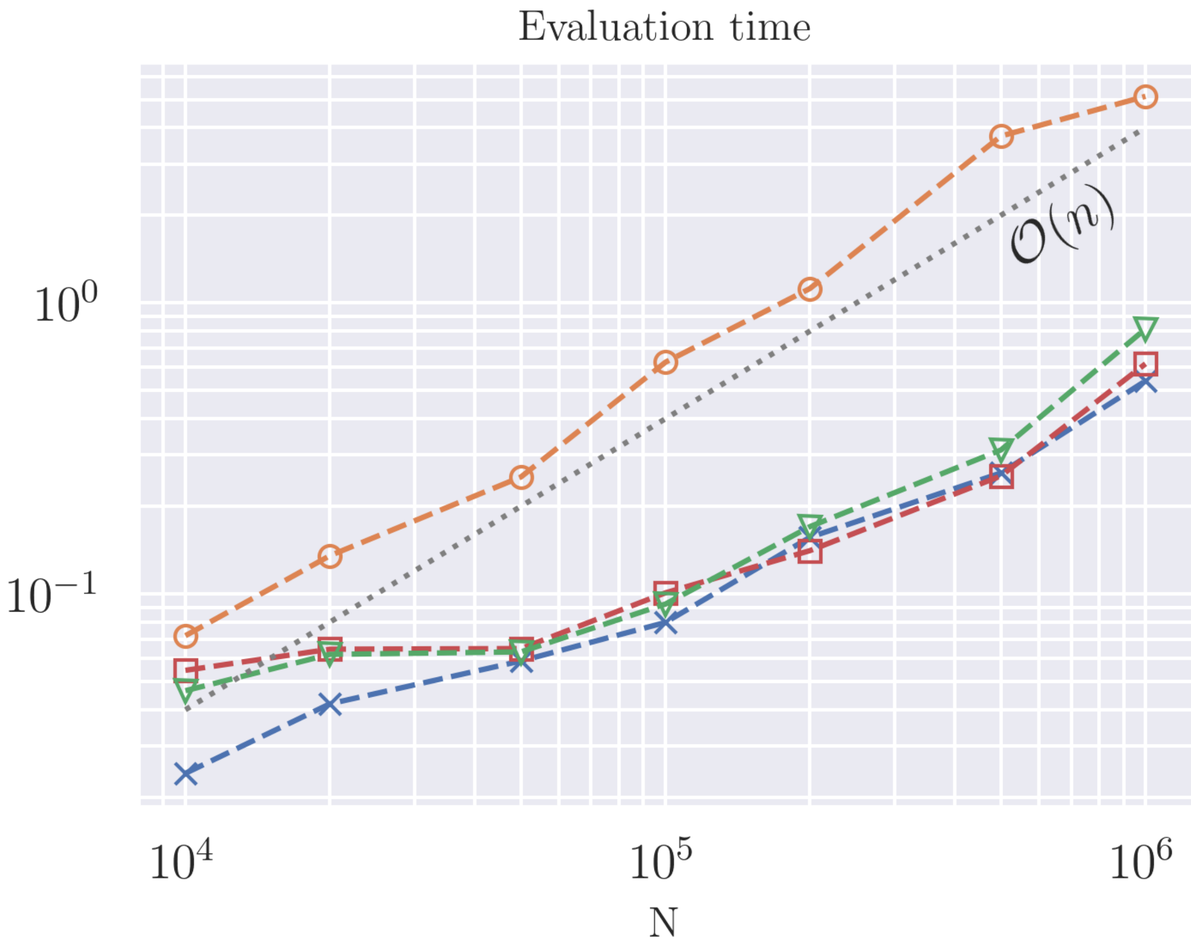
ExaFMM-t performance, single prec, 14-core i9



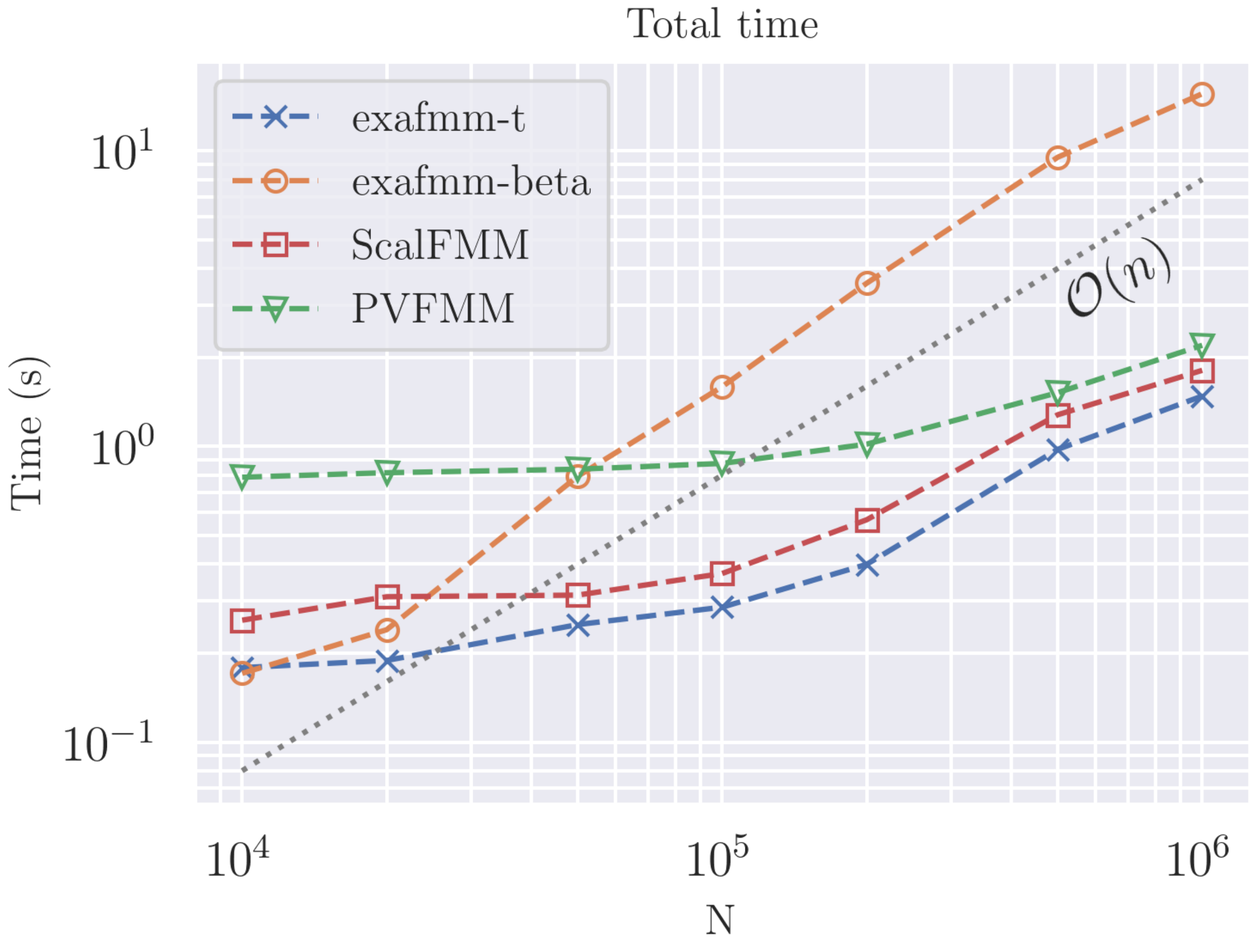


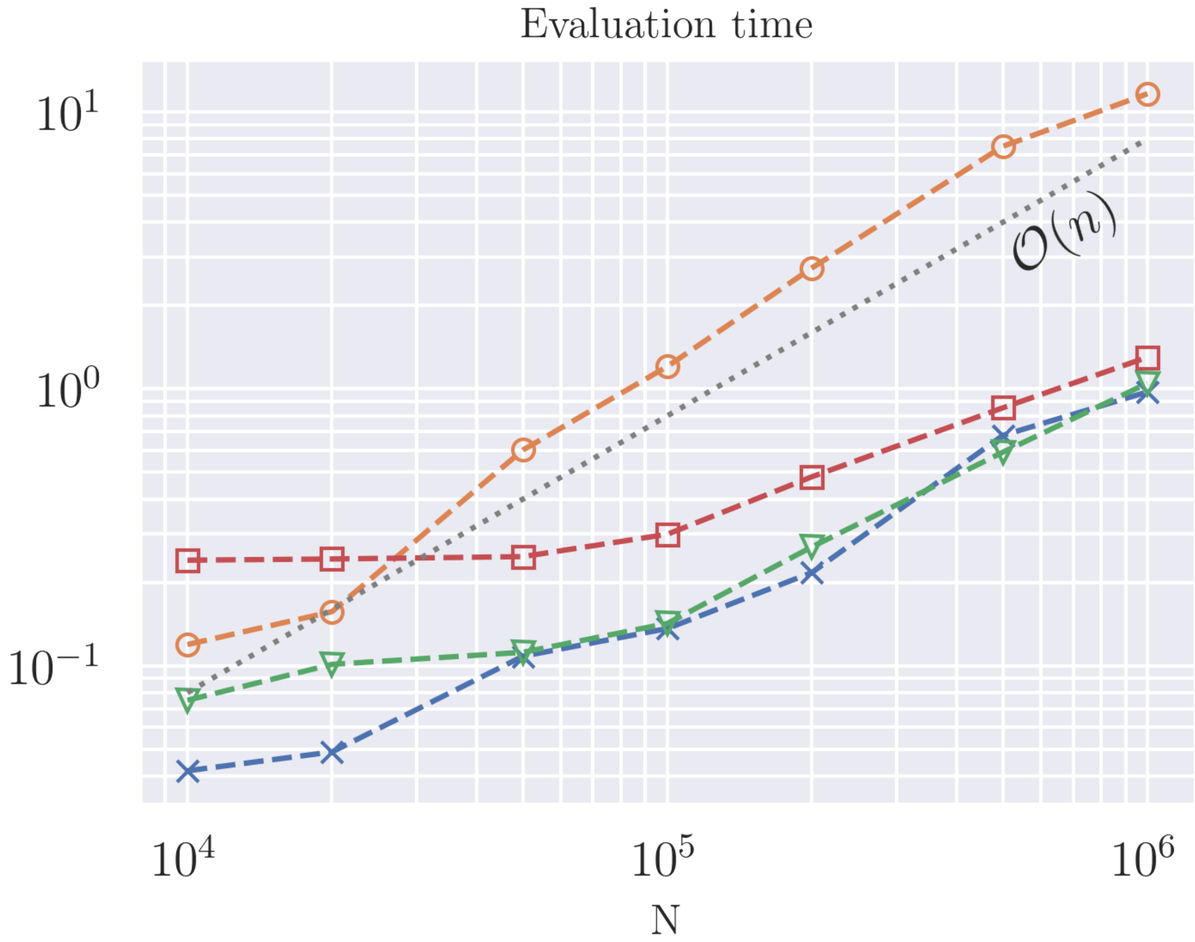
ExaFMM-t performance, double prec, 14-core i9





ExaFMM-t performance, high-accuracy, 14-core i9





ExaFMM-Python interface

1. pybind11 — to create Python bindings for C++ code
2. less than 400 lines to create the Python interface!
3. allows using exaFMM as a package, with NumPy arrays for the sources and target
4. same performance
5. pip install it (setup.py will compile exaFMM locally)

https://github.com/exafmm/exafmm-t

ExaFMM in a Jupyter notebook!

```
In [1]: import numpy as np
        import exafmm.laplace as laplace

In [2]: laplace.__doc__

Out[2]: "exafmm's submodule for Laplace kernel"
```

https://github.com/exafmm/exafmm-t/blob/master/examples/laplace_example.ipynb

Lessons, challenges

Important and complex algorithm, but no mature software library available? *Let's fix that! 10 years later…*

1. Challenge: in research setting, string of novice developers

2. Lesson: novices tend to over-engineer, and skip essentials (documentation, modularization, robust design)

➡ over-engineered code is the worst for research computing!

Lessons, challenges

Important and complex algorithm, but no mature software library available? *Let's fix that! 10 years later*...

1. Challenge: testing is *hard* in research computing: test on hardware with different SIMD capability, test different compilers, test edge cases...
2. Lesson: implement code review as regular practice
3. Goal: code simplicity + high-performance

Cost?

1. Faculty start-up fund + NSF CAREER award … $1 million?
2. 10+ years
3. few papers (and rejections: “not novel enough”)

https://lorenabarba.com/publications/

Benefit to lab members and science-tech world

1. PhD student 1: staff scientist at Swiss National Supercomputing Center
2. Postdoc: faculty at Tokyo Institute of Technology
3. PhD student 2: dev-tech engineer at NVIDIA
4. Collaborating post-doc: research engineer at NVIDIA
5. PhD student 3: graduating soon, hire him!

Lorena A. **Barba group**

RESEARCH PUBLICATIONS CODE EVENTS BLOG PEOPLE

RESEARCH

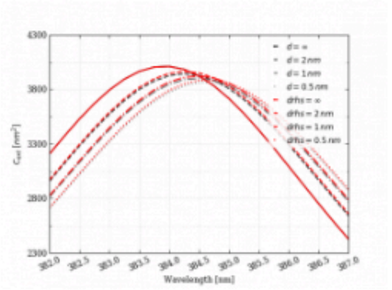
CODE

Python workflow for high-fidelity modeling of overland hydrocarbon flows with GeoClaw and cloud computing

Poster presented at the SciPy 2019 Conference, Austin, TX

How repro-packs can save your future-self

Natalia Clementi



The story of a bug-fix after the research paper was published. Back in December 2019 we published a paper along with its reproducibility packages. These repro-packs, as we call them, consist of all the files necessary to reproduce the results in our paper (data and plots) and we deposit them in Zenodo and Figshare archives... CONTINUE »

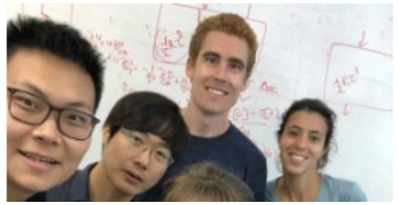
BLOG DISCUSS 06.1.2020

PEOPLE

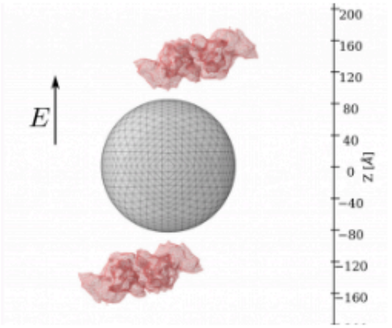
BLOG

EVENTS

Fulbright Scholar joins the group



Computational nanoplasmonics in the quasistatic limit for biosensing applications



Preprint: arXiv 1812.10722 Dec. 31, 2018. Submitted: Dec. 31, 2018; Jan. 20, 2019; Mar. 20, 2019. Accepted: Nov. 1. Published: Dec. 16, 2019. DOI: 10.1103/PhysRevE.100.063305 Abstract This work uses the long-wavelength limit to compute LSPR response of biosensors, expanding the open-source PyGBe code to compute the extinction cross-section of metallic nanoparticles in the presence of any target... CONTINUE »

NEWS DISCUSS 11.1.2019

http://lorenabarba.com https://github.com/barbagroup