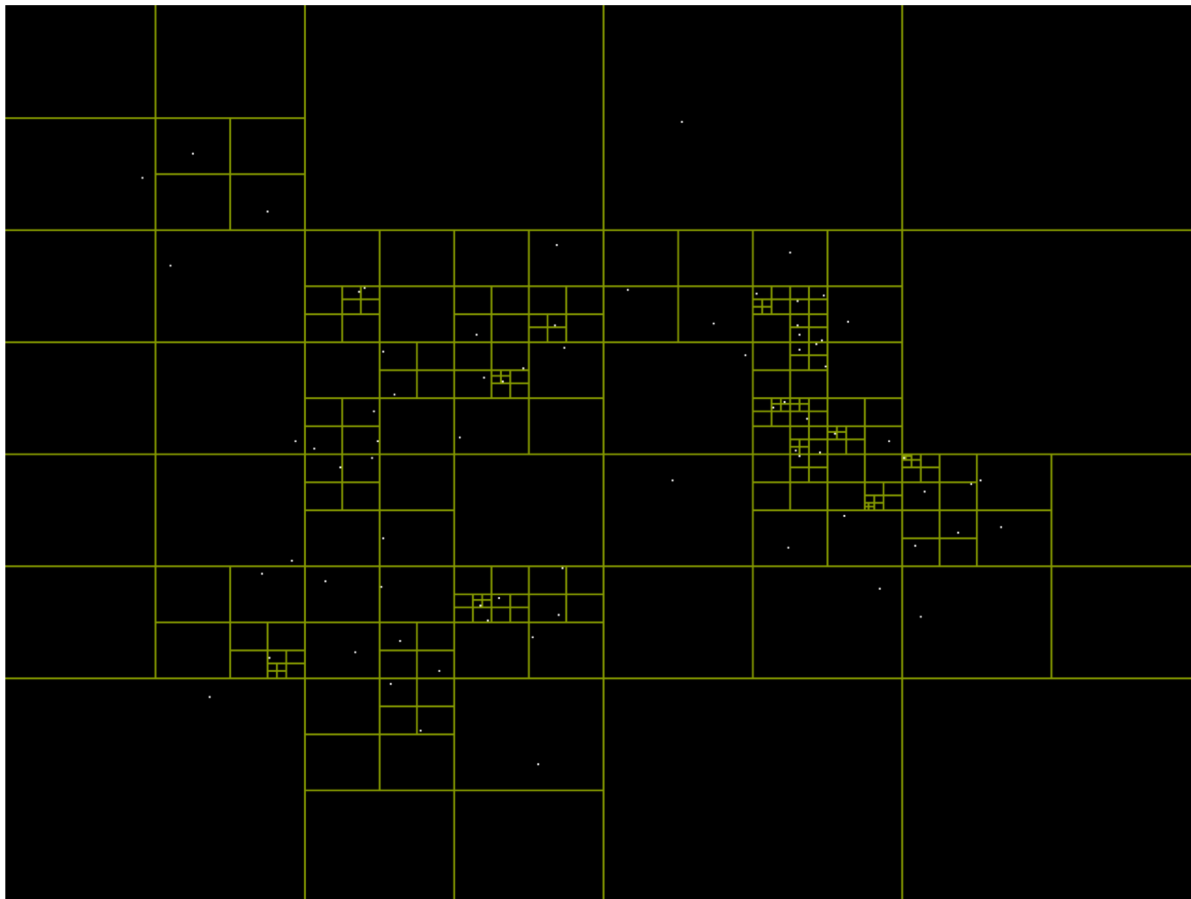



A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green color. They are positioned diagonally, with the blue one partially covering the green one.

Fast Multipole Method (FMM)

Jorge Arellano





Developed in 1987 by Leslie Greengard & Vladimir Rokhlin Jr.,
FMM was originally developed as a fast algorithm for approximating
the N-body problem

$$O(N^2) \quad \text{to} \quad O(N)$$



Motivation

“Complexity Trumps Hardware.”

The current trend in computer architecture is moving towards multi-core processors and many-core processors where the Byte/flop ratio is decreasing with every generation.

(CPU's < GPU's)

Faster algorithms make life easier for lazy people like me



Simulation of Celestial Bodies

<https://youtu.be/W04TzMMpp9A>

Benchmarks

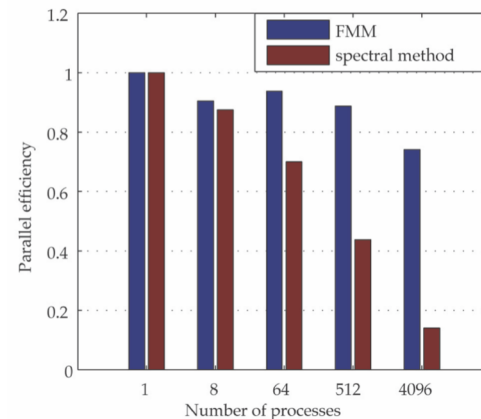
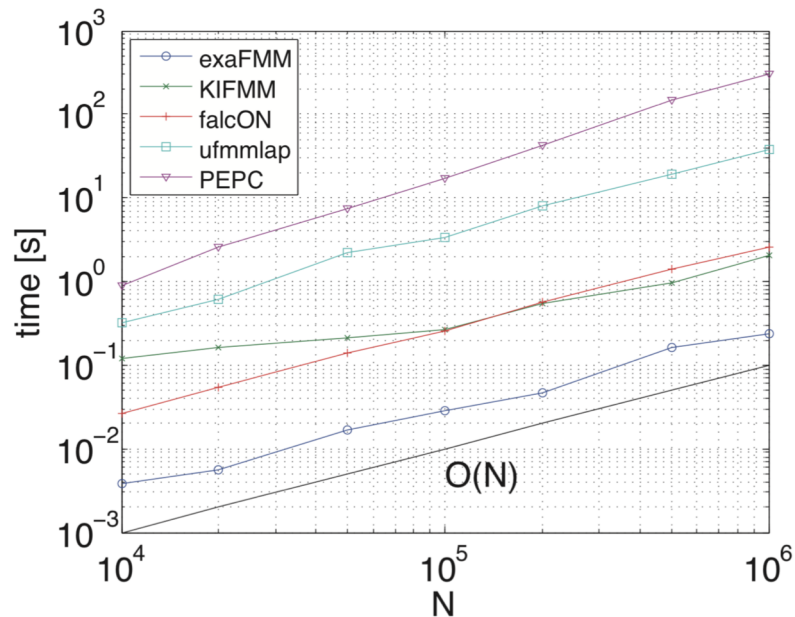


Figure 2. Weak scaling from 1 to 4096 processes of two parallel application codes for fluid turbulence, one using an FMM-based solver on GPUs (one GPU per MPI process), the other an FFT-based solver on CPUs. Figure used under CC-BY license; doi:10.6084/m9.figshare.92425.



Conclusion

FMM works!

But only useful when the problem can be parallelized

Low byte/flop (dense lin-*alg*) tend to have high complexity $O(N)$ and algorithms with low complexity (FFT, Sparse Lin-*alg*) have high Byte/flop. FMM has an impressive combination of $O(N)$ complexity and a Byte/flop that is even lower than matrix-matrix multiplication.

A possible alternative more PDE solvers.



references

https://www.amin.space/blog/quadtree_debug/

<https://sinews.siam.org/Details-Page/how-will-the-fast-multipole-method-fare-in-the-exascale-era>

<https://sites.google.com/site/rioyokota/research/fmm>

http://adl.stanford.edu/cme342/Lecture_Notes_files/lecture13-14_1.pdf

<https://pdfs.semanticscholar.org/360d/20e6f65d43ff1482d553e4249c6334dedd76.pdf>