Parallel Multipole Methods

Paul Beckman, Mariya Savinov

NYU Courant

May 2, 2022

Motivating Problem

Motivating Problem

- Consider a collection of interacting particles with a potential

Motivating Problem

- Consider a collection of interacting particles with a potential
- **Total potential** at a point x due to **particles** x_j with **charges** q_j is

$$u(x) = \sum_{j=1}^{n} \frac{q_j}{|x - x_j|} = \sum_{j=1}^{n} q_j \phi(x - x_j)$$

Motivating Problem

- Consider a collection of interacting particles with a potential
- **Total potential** at a point x due to **particles** x_j with **charges** q_j is

$$u(x) = \sum_{j=1}^{n} \frac{q_j}{|x - x_j|} = \sum_{j=1}^{n} q_j \phi(x - x_j)$$

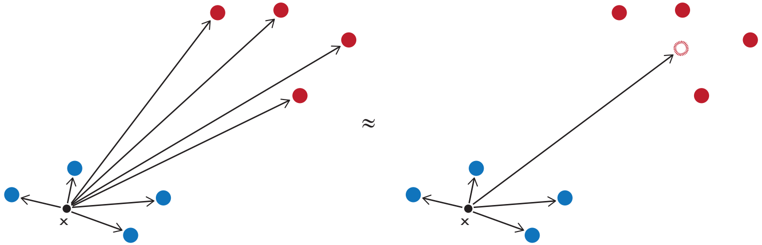
- Separate sum into **near-field** and **far-field** contributions

Motivating Problem

- Consider a collection of interacting particles with a potential
- **Total potential** at a point x due to **particles** x_j with **charges** q_j is

$$u(x) = \sum_{j=1}^{n} \frac{q_j}{|x - x_j|} = \sum_{j=1}^{n} q_j \phi(x - x_j)$$

- Separate sum into **near-field** and **far-field** contributions
  - Take *aggregate* effect of far-field charges via **Taylor series expansion**



Barnes-Hut Tree Code: Taylor Expansion

Barnes-Hut Tree Code: Taylor Expansion

- Use **Taylor series expansion** of ϕ for small $\delta = \dfrac{x_j - x^*}{x^* - y}$:

$$\begin{aligned}
\phi(x_j - y) &= \phi(x^* - y)\phi(1 + \delta) \\
&\approx \phi(x^* - y)\left[\sum_{m=0}^{p} \frac{\phi^{(m)}(1)}{m!}\delta^m + O(\delta^{p+1})\right] \\
&= \sum_{m=0}^{p} a_m(x_j - x^*)S_m(x^* - y) + O(\delta^{p+1})
\end{aligned}$$

Barnes-Hut Tree Code: Taylor Expansion

- Use **Taylor series expansion** of ϕ for small $\delta = \dfrac{x_j - x^*}{x^* - y}$:

$$\begin{aligned}\phi(x_j - y) &= \phi(x^* - y)\phi(1+\delta) \\ &\approx \phi(x^* - y)\left[\sum_{m=0}^{p} \frac{\phi^{(m)}(1)}{m!}\delta^m + O(\delta^{p+1})\right] \\ &= \sum_{m=0}^{p} a_m(x_j - x^*)S_m(x^* - y) + O(\delta^{p+1})\end{aligned}$$

- For a set of points x_j, potential contribution can be approximated:

$$\sum_{j\in\text{far-field}} q_j\phi(x_j - y) \approx \sum_{m=0}^{p} \underbrace{\left[\sum_{j\in\text{far-field}} q_j a_m(x_j - x^*)\right]}_{\text{weight } w} S_m(x^* - y) + O(\delta^{p+1})$$

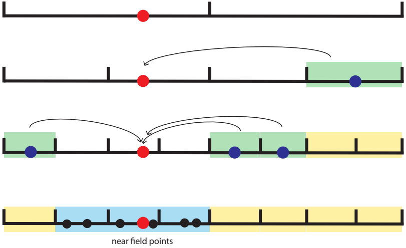
Barnes-Hut Tree Code: Interaction List

Barnes-Hut Tree Code: Interaction List

- compute terms involving far-field cells *not included at higher levels*

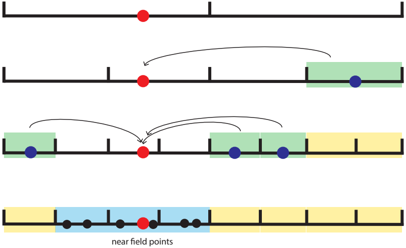
Barnes-Hut Tree Code: Interaction List

- compute terms involving far-field cells *not included at higher levels*



Barnes-Hut Tree Code: Interaction List

- compute terms involving far-field cells *not included at higher levels*

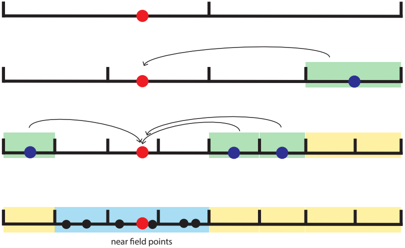


- Then for a point x, the potential is approximately

$$u(x) = \sum_{\ell=1}^{O(\log N)} \sum_{m=0}^{p} w_{\ell,k(\ell),m} S_m \left(x^*_{\ell,k(\ell)} - x \right) + O\left(2^{-p}\right)$$

Barnes-Hut Tree Code: Interaction List

- compute terms involving far-field cells *not included at higher levels*



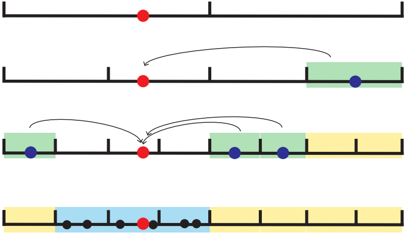
- Then for a point x, the potential is approximately

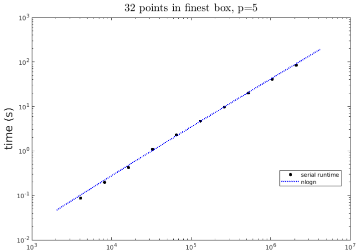
$$u(x) = \sum_{\ell=1}^{O(\log N)} \sum_{m=0}^{p} w_{\ell,k(\ell),m} S_m \left(x^*_{\ell,k(\ell)} - x \right) + O\left(2^{-p}\right)$$

- For each box, compute weights $w_{\ell,k,m} \longrightarrow$ total cost $O(N \log N)$

Barnes-Hut Tree Code: Interaction List

- compute terms involving far-field cells *not included at higher levels*





- Then for a point x, the potential is approximately

$$u(x) = \sum_{\ell=1}^{O(\log N)} \sum_{m=0}^{p} w_{\ell,k(\ell),m} S_m \left(x^*_{\ell,k(\ell)} - x \right) + O\left(2^{-p}\right)$$

- For each box, compute weights $w_{\ell,k,m} \longrightarrow$ total cost $O(N \log N)$

Parallelism

Parallelism

- Parallelizing with OpenMP

Parallelism

- Parallelizing with OpenMP
  - *Challenge:*

Parallelism

- Parallelizing with OpenMP
  - *Challenge:* Recursive tree code may prevent significant speedup

Parallelism

- Parallelizing with OpenMP
  - *Challenge:* Recursive tree code may prevent significant speedup
- Two options of how to parallelize

Parallelism

- Parallelizing with OpenMP
  - *Challenge:* Recursive tree code may prevent significant speedup
- Two options of how to parallelize
- **OPTION 1:** Parallelize the loops in the code

Parallelism

- Parallelizing with OpenMP
  - *Challenge:* Recursive tree code may prevent significant speedup
- Two options of how to parallelize
- **OPTION 1:** Parallelize the loops in the code
  - computing weights, adding near-field terms, and adding far-field terms

Parallelism

- Parallelizing with OpenMP
  - *Challenge:* Recursive tree code may prevent significant speedup
- Two options of how to parallelize
- **OPTION 1:** Parallelize the loops in the code
  - computing weights, adding near-field terms, and adding far-field terms
  - *Potential Problem:* Loops of varying sizes, possibly large overhead

Parallelism

- Parallelizing with OpenMP
  - *Challenge:* Recursive tree code may prevent significant speedup
- Two options of how to parallelize
- **OPTION 1:** Parallelize the loops in the code
  - computing weights, adding near-field terms, and adding far-field terms
  - *Potential Problem:* Loops of varying sizes, possibly large overhead
- **OPTION 2:** Distribute the work in the tree among threads

Parallelism

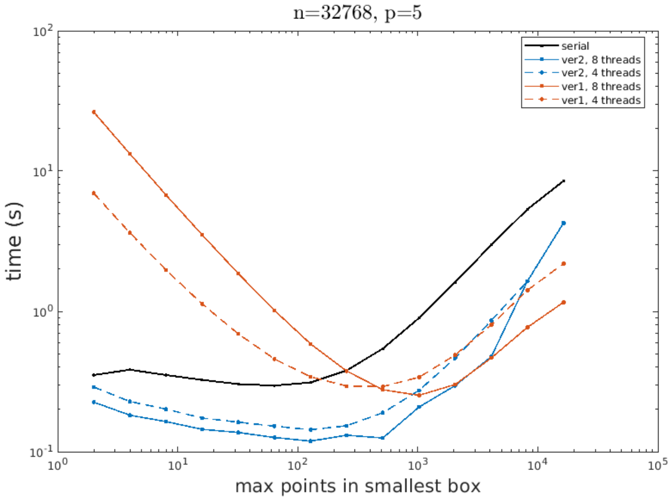
- Parallelizing with OpenMP
  - *Challenge:* Recursive tree code may prevent significant speedup
- Two options of how to parallelize
- **OPTION 1:** Parallelize the loops in the code
  - computing weights, adding near-field terms, and adding far-field terms
  - *Potential Problem:* Loops of varying sizes, possibly large overhead
- **OPTION 2:** Distribute the work in the tree among threads
  - At tree levels $\ell < O(\log r)$ for r threads, open a new section

Parallelism

- Parallelizing with OpenMP
  - *Challenge:* Recursive tree code may prevent significant speedup
- Two options of how to parallelize
- **OPTION 1:** Parallelize the loops in the code
  - computing weights, adding near-field terms, and adding far-field terms
  - *Potential Problem:* Loops of varying sizes, possibly large overhead
- **OPTION 2:** Distribute the work in the tree among threads
  - At tree levels $\ell < O(\log r)$ for r threads, open a new section
  - *Potential Problem:* Nested parallelism

Results: Comparing Parallel Options

Results: Comparing Parallel Options

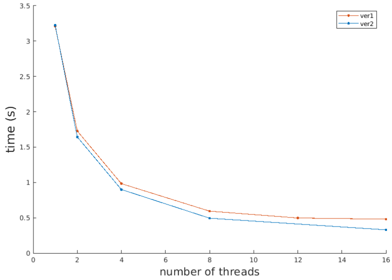


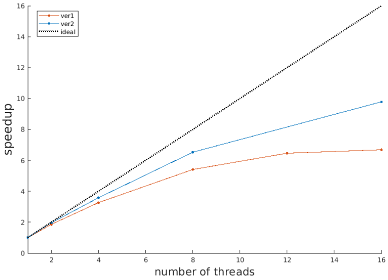
`crackle1`: Intel Xeon E5630 (2.53 GHz)

Results: Strong Scaling

Results: Strong Scaling

$n = 32768$, max points in box 2048, and $p = 5$



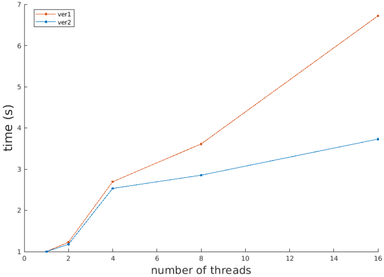


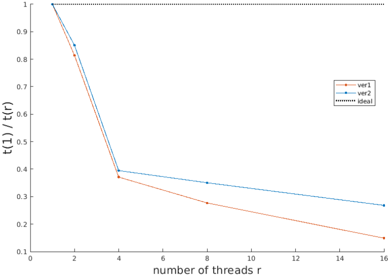
`crunchy1`: AMD Opteron 6272 (2.1 GHz)

Results: Weak Scaling

Results: Weak Scaling

$n = 8192 \cdot r$, max points in box $= 2048$, and $p = 5$





`crunchy1`: AMD Opteron 6272 (2.1 GHz)

References

Josh Barnes and Piet Hut, *A hierarchical O(NlogN) force-calculation algorithm*, Nature **324** (1986), no. 6096, 446–449.

Long Chen, *Introduction to fast multipole methods*.

Leslie Greengard and Vladimir Rokhlin, *A fast algorithm for particle simulations*, Journal of computational physics **73** (1987), no. 2, 325–348.