K-means clustering for optimal domain decomposition of parallel hierarchical N-body simulations

Youssef Marzouk
Sandia National Laboratories
Livermore, CA 94551 USA
<ymarzou@sandia.gov>

Ahmed Ghoniem
Massachusetts Institute of Technology
Cambridge, MA 02139 USA
<ghoniem@mit.edu>

support from: DOE-MICS, NERSC, Hertz Foundation
Computational $N$-body problems

- gravitational astrophysics
- vortex particle methods for high-Re flow
  + molecular dynamics, SPH, …
Example: vortex particle methods

- Incompressible, inviscid vorticity transport equation:

\[
\frac{D\omega}{Dt} = \omega \cdot \nabla u, \quad \nabla \cdot u = 0; \quad u \cdot \hat{n} \text{ specified on } \partial D
\]

- Represent the vorticity field on a set of particles, each with vector strength \( \alpha = \bar{\omega} dV = \Gamma \delta \chi \):

\[
\omega(x,t) = \sum_{i}^{N} \alpha_i(t) f_\sigma(|x - \chi_i(t)|) \quad f_\sigma(r) = (1/\sigma^3)f(r/\sigma)
\]

Regularization: \( \sigma \) is a smoothing radius, \( f(r) \) is a radially symmetric core function.

- Given \( \omega \), the velocity field can be recovered as follows:

\[
u = u_\omega + u_p \quad \text{(Helmholtz decomposition)}
\]

\[
u_\omega = \int K(x,x') \times \omega(x')dx' \quad K(x,x') = -\frac{1}{4\pi} \frac{x - x'}{|x - x'|^3}
\]

\[
u_p = \nabla \phi
\]

\[
\nabla^2 \phi = 0, \quad n \cdot \nabla \phi = n \cdot u - n \cdot u_\omega \text{ on } \partial D
\]

\{ vortical component, from Biot-Savart law \}

\{ potential component, to satisfy normal-velocity BC \}
Vortex methods

- Particles are Lagrangian computational elements; solve equations of motion via numerical integration for trajectories \( \chi_i(t) \) and weights \( \alpha_i(t) \).
- **Filament methods**: Vorticity remains aligned with material elements \( \delta \chi_i(t) \):
  \[
  \frac{D \delta \chi}{Dt} = \delta \chi \cdot \nabla u \quad \leftrightarrow \quad \frac{D \omega}{Dt} = \omega \cdot \nabla u
  \]
  \[
  \bar{\alpha} = \Gamma \delta \bar{\chi}
  \]
  \[
  \Delta \text{ Solve vorticity transport equation (convection + stretching and tilting) by integrating ODE to advect nodes of each material element.} \Gamma \text{ is conserved.}
  \]

\[
\mathbf{u}(\chi_i(t)) = u(\chi_i)
\]
\[
\mathbf{u}_\omega(x) = \sum_i K_\sigma(x - \chi_i) \times \alpha_i
\]
\[
\text{here:} \quad -4\pi K_\sigma (\bar{x} - \bar{y}) = \frac{\bar{x} - \bar{y}}{\left( |\bar{x} - \bar{y}|^2 + \sigma^2 \right)^{3/2}}
\]

\[
\text{vortex elements described by discrete nodes \{\chi_i\}}
\]
\[
u \text{ obtained from Biot-Savart law + potential flow}
\]
Vortex methods

- Properties of 3-D vortex methods:
  - Convergence to smooth solutions of the Euler equations as $\sigma \to 0$, with the constraint $h/\sigma < 1$.
    $\Rightarrow$ Maintain core overlap! Insert new elements in response to flow strain.
  - Grid-free: adaptive distribution of computational points, only over the support of the vorticity.
  - Advection without numerical dissipation.
  - Vortex filament schemes: automatically conserve circulation, linear impulse, angular impulse, and kinetic energy.
Vortex filaments in an evolving transverse jet. 
\( r=7; \ t=0 \) to 2.52

Contour indicates velocity magnitude.

**Features:**
periodic shear layer rollups; near field matures; upright trajectory; transition to small scales.
Computational challenges

- Physics: High resolution, long-time runs require \( N = O(10^6) - O(10^7) \)
- N-body problems: with direct summation, computational cost scales as \( O(N^2) \)
- Irregular particle distributions, non-uniform particle density — all dynamically varying! New particles (remeshing)!
- Issues apply to N-body simulations in a variety of contexts (vortex methods, astrophysics, molecular dynamics, SPH)

How to meet these challenges on parallel architectures?
Tools for fast, scalable particle simulation...
Treecodes for N-body problems

- **Hierarchical** methods for velocity (potential, force) evaluation—replace *particle-particle* interactions with *particle-cluster* interactions [Appel, Barnes-Hut]—cost $O(N \log N)$

  $$u(x) = \sum_{i=1}^{N} w_i K(x, y_i)$$

- What kernel $K(x,y)$?
- What kind of approximation (spherical harmonics, Taylor expansion, etc)? To what order $p$?
- How to construct hierarchy? (Oct-tree: leaf node size? adaptivity?)

- Cluster-cluster interactions: FMM [Greengard-Rokhlin], cost $O(N)$
• Serial treecode [Lindsay and Krasny 2001]
  - Velocity from Rosenhead-Moore kernel (algebraic smoothing $\sigma$):
    \[
    \ddot{u}(\bar{x}_j) = \sum_{i=1}^{N} K_{\sigma}(\bar{x}_j, \bar{y}_i) \times \ddot{w}_i
    \]
    \[
    K_{\sigma}(x,y) = -\frac{1}{4\pi} \frac{x - y}{(|x - y|^2 + \sigma^2)^{3/2}}
    \]
  - Constructs an adaptive oct-tree:
  
  ⇒ use Taylor expansion of kernel in Cartesian coordinates:
  \[
  \ddot{u}(\bar{x}_{tar}) = \sum_{i=1}^{N_c} \sum_{p} \frac{1}{p!} D_y^p K_{\sigma}\left(\bar{x}_{tar}, \bar{y}_i; \bar{y}_i - \bar{x}_{tar}\right) p \times \ddot{w}_i
  \]
  Taylor coefficients, computed with recurrence relation
  Cell moments, stored for re-use
Treecodes for N-body problems

- LK Cartesian treecode (cont.)
  - Recurrence relation for Taylor coefficients allows high-order expansions at low cost.
  - Cell moments computed as needed, then stored for re-use.
  - Leaf size \( N_0 \approx 512 \)

- Error criterion (on vector potential) used to determine order of expansion \( p \) and level of hierarchy:

\[
\frac{M_p(c)}{4\pi R^{p+1}} \leq \varepsilon \quad \text{where} \quad M_p(c) = \sum_{i}^{N_c} \|x_i - y_c\|^p |w_i|
\]
Domain decomposition

- Initial attempts at parallelization had poor efficiency. (How bad? I’ll show you later…)
- Issue = good partition geometry. Why?

Example — “parallelize over targets”
red = proc 1; blue = proc 2
BAD PARTITIONING ⇒
duplicate computations
(locally essential trees overlap)

GOOD TARGET
PARTITION
Domain decomposition

Example—”parallelize over sources”
red = proc 1; blue = proc 2

BAD PARTITIONING ⇒ each target particle participates in more particle-cluster interactions @ higher \( p \)

BEETTER SOURCE PARTITIONING

- What is the “best” partition and how can we compute it?
Clustering Algorithms

- **K-means clustering** [MacQueen 1967]
  - Used frequently in machine learning, multivariate statistics
  - Optimally assigns N particles to k clusters:
    \[
    J = \sum_{c}^{k} \sum_{i}^{N_c} \| x_i - y_c \|^2 |w_i| \quad \text{(finds a local minimum)}
    \]
  - Proceeds iteratively:
    1. **Assign** each particle \( x_i \) to the nearest cluster centroid \( y_c \)
    2. **Update** centroid locations:
      \[
      y_c = \frac{\sum_{i}^{N_c} w_i x_i}{\sum_{i}^{N_c} w_i}
      \]
  - Compare cost function \( J \) to treecode error criterion…
    \[
    \frac{M_p (c)}{4\pi R^{p+1}} \leq \varepsilon \quad \text{where} \quad M_p (c) = \sum_{i}^{N_c} \| x_i - y_c \|^p |w_i|
    \]
    (satisfy at lower \( p \), higher level of hierarchy)
  - Implicitly computes Voronoi partition, convex cluster boundaries.
  - *Many* variations; fully parallelizable [Dhillon and Modha 2001]
Domain decomposition

- Global set of N source particles
- Partitioned into k clusters
- Local oct-tree constructed within each cluster
Domain decomposition

Cluster partition
N=157297, k=128
Domain decomposition

cluster partition
N=157297, k=128
Domain decomposition

- Alternative partitioning algorithms for parallel N-body problems:
  - ORB (orthogonal recursive bisection) [Warren 1990]: Domains are leaf nodes of a binary tree; may not have optimal geometry (e.g., long and narrow…)
  - HOT (hashed oct-tree) [Warren & Salmon 1995] and “costzones” [Singh 1995]: Sort and partition bodies of a global oct-tree; restrict tree adaptivity; domains may span spatial discontinuities…
Load-balanced clustering

- Load balance in vortex treecodes — why is this a difficult problem?
  - Irregular particle distribution, non-uniform density: equipartition doesn’t ensure load balance
  - “Per-particle-cost” not well-defined
  - NEW particles are introduced every timestep

- Define “load imbalance” as \( \left( \max_k t_k \right) / \bar{t} \)
Load-balanced clustering

- Heuristics for dynamically load-balancing clusters:
  - Adaptive **scaling** of cluster distances by $s_k$:
    \[
    J = \sum_{i} \min_{k} \left( s_k \|x_i - y_k\|^2 |w_i| \right)
    \]
  - General scheme: update $s_k$ based on previous timestep’s times, scalings:
    \[
    s_k^{n+1} = f(s_k^n, t_k^n; k = 1..K)
    \]
  - Increase $s_k$ for high-cost clusters and vice-versa
    \[
    s_k^{n+1} = s_k^n \left( 1 + \alpha \tanh \left( \beta \frac{t_k^n - \bar{t}_i^n}{\bar{t}_i^n} \right) \right) \quad (\alpha=0.1, \beta=2)
    \]
  - **Split/merge**: use local k-means (with $k=2$) to split highest-cost cluster centroid at every timestep.
  - Safety: re-seed centroids for k-means iteration when imbalance persists above a threshold (e.g., 1.5).
Results: timings

N=1164184 speedup = serial time/parallel time
Results: timings

N=1164184

parallel efficiency = speedup/k
Results

Load imbalance, $N=1164184$

Note that $\frac{1}{imbal} < pareff$
Results

- K-means clustering performs *better* than load balance would suggest. Why?

- **Clustered*** $k=1024$ case creates a different partition of space than the serial case (i.e., a global oct-tree)

- Clustering—Influence of source particles evaluated at lower orders of expansion for same (or better) accuracy!
Error control

- Test N=261481 (from evolving transverse jet); for block and k-means partition.

- Global accuracy parameter $\varepsilon=10^{-2}$, $\varepsilon=10^{-4}$:

- Convert to $\varepsilon_k$ for each cluster:
  \[ \varepsilon_k = \frac{M_0(k)}{M_0} \varepsilon \]

- Absolute error:
  \[ \equiv \max_i \left( \| \tilde{u}_i - \bar{u}_i^{DS} \|_2 \right) \]
Results: k-means time

- Serial $k$-means scales as $O(Nkd)$. Parallelize on $k$ processors $\rightarrow O(N)$

\[
\text{time per k-means iteration}
\]

VERY large $N$: Could use accelerated $k$-means [Pelleg & Moore 99] for better scaling...
Results: dynamic load balancing

- Transverse jet simulation, $k=1024$; $r=7$. Two cases:
  - k-means clustering with dynamic load balancing
  - plain k-means, reseeding centroids at each step

Averaged normalized load distribution, $k=1024$; $10^6 < N < 2.5 \times 10^6$

![Graphs showing load distribution comparison](image)

plain k-means with dynamic load balancing
Results: dynamic load balancing
Results
Conclusions

- Scalable particle methods and N-body problems:
  - Hierarchical N-body interactions – *partition geometry* is critical; $k$-means clustering yields optimal geometric properties.
  - Scaling and other heuristics for dynamic load balancing.
  - Demonstrated with vortex particle methods for $k$ up to 1024; tools extend to N-body problems in many fields.

- Further developments:
  - **Hierarchical** clustering (to replace oct-tree construction)
    $\Rightarrow$ $k$-means not just for parallel partitioning!
  - New approaches to load balance: solve constrained problem by adding a penalty term to the $k$-means cost function
  - Clustering with respect to additional element properties (e.g., core size)