Particle transport in turbulence simulations on Frontera

P.K. Yeung (Georgia Tech)
E-mail: pk.yeung@ae.gatech.edu

K. Ravikumar (HPE), R. Uma-Vaideswaran (Georgia Tech)
S. Subramaniam (Iowa State Univ)

TACC Frontera Users Meeting, Austin, TX
August 4-5, 2022

NSF CDS&E Program, Grants 1953186 (GT), 1953298 (ISU)
LRAC Allocation and new CSA project for LCCF at TACC

Thanks: Lars Koesterke, John Cazes
Eulerian view: $u(x, t)$ and the Navier Stokes equations
- Direct numerical simulation (DNS) on $N^3$ grid ($N$ up to 12288 on Frontera)
- Simplified geometry: Fourier pseudo-spectral methods (3D FFT)
- Flow as a continuous medium, or viewpoint of a fixed observer

Lagrangian view: follow motion of fluid particles
- Observer moving with instantaneous, disorderly flow. Difficult in experiments.
- Examples: where is the smoke going? or where did the disease agent come from?
- Extensions to multi-phase problems: particle inertia, two-way interactions

Focus in this talk: particle-tracking algorithms on Frontera
- How do we do it accurately and efficiently, with minimal cost dependence on particle count, while making best use of the machine hardware or software?
Particle Tracking Algorithms

Equation of motion (with specified initial positions)

\[ \frac{dx^+}{dt} = u^+(t) = u(x^+(t), t) \]

- Velocity is obtained by interpolation from neighboring grid points
- Cubic splines (Yeung & Pope 1988): 4th order accurate and twice differentiable (helpful if acceleration to be computed by differentiating velocity signal)

Overall view of essential operations at each time step:
- Form \((N+3)^3\) cubic spline coefficients from \(N^3\) instantaneous velocity field
- Get localized coordinates (in grid spacings) for particles
- Obtain interpolated velocity using spline coeffs. in each particle’s neighborhood.
- Time advance by same Runge-Kutta scheme as in Eulerian DNS
Considerations for Massive Parallelism

(1) Minimize communication costs; and
(2) make them independent of particle count

Spline coefficients: cost is substantial, but independent of particles
- Like FFTs, operate one direction at a time (tensor products)
- \((N + 3)^3\) coefficients distributed among \(P\) MPI processes (first 3 carry more data)
- Transposes between pencils in different directions: Alltoall or Alltoallv

Particles: depends on whether the 64 spline coeffs. are available locally
- NO if each MPI process tracking same particles at any time.
- YES if control for each particle is mapped dynamically to MPI processes at each time step, and coefficients from neighbors are available (using ghost layers).
Partitioning an $N^3$ 3D domain among $P$ MPI processes

- 1D (slabs): limited to $P \leq N$, if CPU memory per core is very large. One transpose e.g. $x - z$ slabs to $x - y$ slabs, using MPI_COMM_WORLD
- 2D (pencils): allows larger $P$, greater flexibility for shorter run times if needed. Two transposes, using row or column communicators $P = P_r \times P_c$.
- Pack and unpack before and after A2A’s with contiguous messages

Choice of 2D processor grid, based on communication performance

- At larger $N$ and $P$, best to let each row communicator occupy its own node, since on-node communication is inherently faster
- Frontera: let $P_r = 48$, even if 8 cores out of 56 are left idle

Splines: start with pencils in $x$, array shape $(NX)(NZ/P_r)(NY/P_c)$
Generation of cubic spline coefficients

\[
\text{SPL}(x) \rightarrow P \rightarrow \text{A2A (row)} \rightarrow \text{UP} \rightarrow \text{SPL}(z) \rightarrow P \rightarrow \text{A2A (col)} \rightarrow \text{UP} \rightarrow \text{SPL}(y)
\]

- 1D splines are based on a tridiagonal system in inner loops (set by problem size)
  - recurrence in algebra reduces efficiency and tends to limit benefit from threads
  - yet strong scaling (outer loops) is likely to be good

- Communication across the nodes dominates the cost, with complication that last dimension of \((N+3)\) has to be divided (non-uniformly) among \(P_c\) processes
  - Alltoall: larger messages of uniform size, that include some null entries; or
  - Alltoallv: messages of non-uniform size, more book-keeping

- Predictor-corrector time advance scheme: spline coefficients need to be generated just once (but 3 velocity components) per DNS time step
Spline coefficients: performance \((P_r = 48, \ P_c = \#N\text{nodes})\)

Timings measured using 3D splines kernel on 7/24-25/2022 (Texascale Days)

<table>
<thead>
<tr>
<th>(N)</th>
<th>Nodes</th>
<th>1D-Splines</th>
<th>(P+\text{UP})</th>
<th>A2A\text{(row)}</th>
<th>A2A\text{(col)}</th>
<th>A2AV\text{(col)}</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>3072</td>
<td>32</td>
<td>0.908</td>
<td>0.655</td>
<td>0.225</td>
<td>1.213</td>
<td>-</td>
<td>3.013</td>
</tr>
<tr>
<td>3072</td>
<td>64</td>
<td>0.456</td>
<td>0.319</td>
<td>0.128</td>
<td>1.033</td>
<td>-</td>
<td>1.940</td>
</tr>
<tr>
<td>6144</td>
<td>256</td>
<td>1.201</td>
<td>0.534</td>
<td>0.335</td>
<td>2.805</td>
<td>-</td>
<td>4.929</td>
</tr>
<tr>
<td>6144</td>
<td>256</td>
<td>1.203</td>
<td>0.529</td>
<td>0.332</td>
<td>-</td>
<td>2.233</td>
<td>4.335</td>
</tr>
<tr>
<td>6144</td>
<td>512</td>
<td>0.616</td>
<td>0.252</td>
<td>0.184</td>
<td>4.572</td>
<td>-</td>
<td>5.645</td>
</tr>
<tr>
<td>6144</td>
<td>512</td>
<td>0.615</td>
<td>0.249</td>
<td>0.183</td>
<td>-</td>
<td>1.385</td>
<td>2.453</td>
</tr>
<tr>
<td>12288</td>
<td>1024</td>
<td>3.013</td>
<td>1.258</td>
<td>1.047</td>
<td>-</td>
<td>9.511</td>
<td>15.500</td>
</tr>
<tr>
<td>12288</td>
<td>2048</td>
<td>1.720</td>
<td>0.598</td>
<td>0.809</td>
<td>-</td>
<td>4.260</td>
<td>7.752</td>
</tr>
<tr>
<td>12288</td>
<td>4096</td>
<td>0.968</td>
<td>0.272</td>
<td>0.527</td>
<td>-</td>
<td>3.426</td>
<td>5.601</td>
</tr>
</tbody>
</table>

- When \(N/P_c\) decreases, A2AV (col) becomes more competitive than A2A
- But as node count increases, benefit may be compromised by congestion.
Ghost layers for spline coefficients

Averting potentially massive communication costs

- Particles wander around: identities of sub-domains holding info on grid points in each particle's neighborhood constantly changing (somewhat randomly)
- Interpolation formula for particles near sub-domain boundaries will require information (spline coefficients) from multiple (neighboring) MPI processes
- Remote direct memory addressing via Fortran co-arrays worked well on Blue Waters (Buaria & Yeung CPC 2017), but not general

A Ghost-layer approach

- Since splines are cubic, ghost layers need to be 4 - 1 = 3 grid spacings deep.
- Over one time step no particle can travel more than 1 grid spacing. Needs info only from immediate neighbors ($3^d - 1$ in a $d$-dimensional decomposition).
- Use one-sided MPI (GET) within 2D decomposition of spline coefficients
Ghost layers and One-sided MPI

- **MPI_WIN_CREATE**: time-consuming, call just once at beginning of code
- **Three buffers**: E-W (row comm), N-S (col comm), (all 4) **Corners** (world)
- **Call MPI_WIN_FENCE (....)**
- **Call MPI_GET (....)**
- **Call MPI_WIN_FENCE (....)**

Message sizes dependent on $P_r \times P_c$. Some packing/unpacking required (less so for $N - S$, which however are likely to have largest messages)

Ex: lower half of $4 \times 16$ decomposition in $x - z$ directions, with full pencil in $y$

When ratio of $P_c/P_r$ is large, N-S comm. between the nodes is dominant.
The actual interpolation

- Interpolation formula using 3D spline coefficients $S_{pqr}$ where $1 \leq p, q, r \leq N + 3$

$$f^{+}(x^{+}) = \sum_{k=1}^{4} \sum_{j=1}^{4} \sum_{i=1}^{4} b_{i}(x') c_{j}(y') d_{k}(z') S_{pqr}$$

- $x', y', z'$ are normalized local particle coordinates (between 0 and 1)
- $b_{i}(x'), c_{j}(y'), d_{k}(z')$ are pre-computed spline coefficients (in each direction: only 4 are nonzero at any given location)
- Cost is proportional to number of particles tracked

- But straightforward, at very low cost if free of communication (which is true in present code)
Migration of particles between adjacent sub-domains

After each position update, identify particles migrating
- out to (on it from) one of 8 neighboring sub-domains
- get total counts, exchange data by MPI_SEND and MPI_RECV on ID, position and velocity of particles involved
- local shuffling: squeeze out “empty slots” resulting from departing particles in the list of particles that each MPI process is keeping dynamically.

Cost depends on fraction of particles migrating
- only some of those which are within 1 grid spacing (in $x$ or $z$ directions) of sub-domain boundaries: typically, well under 1%
- homogeneous turbulence: sustained local accumulations are unlikely; — thus no strong departure from load-balancing
July Texascale data appear to be consistent with “large” queue results in late Dec 2021:

<table>
<thead>
<tr>
<th>Grid ($N^3$)</th>
<th># nodes</th>
<th>$N_p$ (10^6)</th>
<th>Wall time per step</th>
<th>% Weak Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>1536³</td>
<td>4</td>
<td>22</td>
<td>11.6 3.06 0.29 0.26 0.02 15.1</td>
<td>- -</td>
</tr>
<tr>
<td>3072³</td>
<td>32</td>
<td>88</td>
<td>12.4 3.20 0.48 0.14 0.01 16.2</td>
<td>93.5 93.2</td>
</tr>
<tr>
<td>6144³</td>
<td>256</td>
<td>352</td>
<td>15.5 4.52 0.59 0.07 0.03 20.7</td>
<td>80.0 78.3</td>
</tr>
<tr>
<td>12288³</td>
<td>2048</td>
<td>1057</td>
<td>17.5 8.69 1.61 0.04 0.11 28.0</td>
<td>88.6 73.9</td>
</tr>
</tbody>
</table>

- Eulerian part (mostly 3D FFT) shows approx 90% weak scaling for each 2X in $N$
- Encountered variability associated with inter-node communication.
- Particle tracking does pose further challenges (due to interpolation), but present implementation allows high particle counts with little extra cost.
Inertial Particles

Added mass (buoyancy) effect and “Stokes drag” (for small particles of diameter $d_p$)

$$\frac{dv}{dt} = \beta \frac{du(x, t)}{dt} - \frac{1}{\tau_p} [v - u(x, t)] .$$

$\beta = 3 \rho_f / (\rho_f + 2 \rho_p)$, $\tau_p = d_p^2 / (3 \beta \nu)$, $St = \tau_p / \tau_\eta$; $u$ is fluid velocity at particle position.

- If $\beta \to 0$ an integrating factor solution to equation of motion is

$$v(t_{n+1}) = \exp\left(-\frac{\Delta t}{\tau_p}\right) \left[ v(t_n) + \frac{1}{\tau_p} \int_{t_n}^{t_{n+1}} \exp\left(\frac{t - t_n}{\tau_p}\right) u(x, t) \, dt \right]$$

- $\beta \neq 0$: $du/dt$ along Stokes particle trajectories not easy to obtain, but an integration by parts gives, with $\alpha, \gamma$ being simple functions of $\tau_p/\Delta t$.

$$v(t_{n+1}) = \exp\left(-\frac{\Delta t}{\tau_p}\right) v(t_n) + \alpha \, u(x, t_n) + \gamma \, u(x', t_{n+1})$$

No increase in computational cost if two-way interactions not included.
Summary and Next Steps

Focused on algorithm for particle tracking today

- Scalability of particle tracking using cubic splines is a nontrivial challenge (but worth the effort, from a science point of view)
- (a) Letting each communicator occupy a node, (b) one-sided MPI for ghost layers and (c) dynamic mapping between particles and parallel processes have led to an algorithm capable of tracking over a billion particles on Frontera
- Hierarchical alltoall(v) across the nodes may help: to be examined

Moving on towards more (or better) science: looking to the LCCF

- Multiphase flow: particle equation more complicated (as above) but present algorithm equally applicable, at nearly the same cost
- Magnetohydrodynamic turbulence: two-way coupling between turbulence (of electrically conducting fluid) and magnetic fields.