E3SM All Hands: NGD: Efficient Communication Pattern for Interpolation Semi-Lagrangian Tracer Transport

Andrew M. Bradley
COMPOSE Team: Peter A. Bosler, Andrew M. Bradley, Oksana Guba, Mark A. Taylor
Outline

1 Motivation

2 SL MPI
Outline

1 Motivation

2 SL MPI
Problem: Tracer transport is expensive.

Solution: Semi-Lagrangian (SL) transport. Long time steps; less communication.

Problem: Transport requires property preservation, and SL makes that harder.
  ▶ (mass conservation, shape preservation, tracer consistency, linear correlation preservation)

Solution: CEDR: Property preservation in exactly 1 all-to-all reduction equivalent.\(^1\)

Opportunity: CEDR enables using the fastest, lowest-communication SL method there is: Interpolation SL (ISL) with compact stencil.

Problem: ISL based on high-order compact stencil (GLL element, \(n_p \geq 4\)) is unstable.

Solution: Stabilized ISL.

Problem: HOMME’s deterministic halo exchange is suboptimal for SL.

Solution: ISL-specific optimal communication pattern.
  ▶ This project.

---


Software: [github.com/E3SM-Project/COMPOSE](https://github.com/E3SM-Project/COMPOSE)
Strong scaling HOMME: Status for 40 tracers

HOMME v1 1/4 Degree

- preqx dycore is \(>2.1\times\) faster on KNL at 1350 nodes (strong-scaling limit).
- preqx dycore is \(>3.2\times\) faster on Edison at 3600 nodes (strong-scaling limit).
Outline

1. Motivation

2. SL MPI
Concepts

- Communicate only with ranks holding relevant data.
- For Interpolation SL (ISL), send requests for interpolation.
  - Send/receive departure point data to make requests for interpolation.
  - Send/receive interpolation and bounds data, fulfilling these requests.
  - In both rounds, overlap communication and computation.
Prescribed velocity field driver on Mutrino KNL, HSW; time step requires only 1-halo.

Vary number of elements per rank, number of tracers.

Measure end-to-end time using original MPI and new SL MPI communication patterns and report speedup.
2-halo overhead

- Prescribed velocity field driver on Mutrino KNL; time step requires only 1-halo.
- Vary number of elements per rank, number of tracers.
- Measure extra time taken when 2-halo is active and report percent overhead.
- *Except for this overhead*, speedup of 2-halo is exactly the factor time step increase.

---

2-Halo overhead on KNL (90x60, 84x63, 72x64, 90x60) with 5 tracers

---

2-Halo overhead on KNL (54x64) vs. #tracers

---

\(^2\)still possibly to do: more accurate velocity, which will also have overhead
2-halo convergence (correctness)

2-halo convergence study: nondivergent flow
circle: 1-halo; triangle: 2-halo
circle and solid: $\Delta t$; triangle or dashed: $2\Delta t$
solid: correct point; dashed: nearest available

Mesh parameter $n_e$

log$_2$ $l_2$ relative error

Trigonometry
Gaussian
Cosine Bells
Slotted Cyl.
Corr. Cosine Bells
Fidelity study

Transport error with tuned parameters

- Nondivergent flow test case
- “HOMME tuned” data are from Guba et al, *Optimization-based limiters for the spectral element method*, JCP 2014.
- SL MPI is used with 2-halo capability.
To do

- Algorithm: Communicate and compute once per node (currently redundant for edge nodes).
- Algorithm: Expose all available parallelism for GPU implementation; rework metadata and metadata setup as necessary.
- Software: Rewrite several target-cell, local-mesh data structures so bulk data are shared.
  - Bulk data memory footprint independent of halo size.
Thanks!
Semi-structured messages

- Set of departure points in remote elements is nondeterministic.
- \( \Rightarrow \) unstructured messages.

- Requirements:
  - No memory allocation except in initialization.
  - But don’t use more memory than deterministic 1-halo method does.
  - Message size roughly proportional to number of departure points (up to a little metadata).

- Preferences during time stepping (not initialization):
  - Two passes to set up message structure rather than one pass plus a sort.
  - \( O(1) \) accesses with no hash table, just linear arrays.
  - No holes in arrays, i.e., unused index space.

- Strategy:
  - At initialization, do everything necessary to make the above feasible during time stepping.
  - Assemble and parse messages using these initialization-phase metadata.

\[
\begin{align*}
\text{xs:} & \quad (#x\text{-tgt-rank} \quad \text{integer} \quad < \text{send/recv departure points}) \\
& \quad \text{pad} \quad \text{i} \\
& \quad (lid\text{-src-rank} \quad \text{i} \quad \text{packed only when} \quad #x \quad \text{in} \quad lid \quad > \quad 0) \\
& \quad #x\text{-in-lid} \quad \text{i} \quad > \quad 0 \\
& \quad (\text{lev} \quad \text{i} \quad \text{packed only when} \quad #x \quad \text{in} \quad (lid,\text{lev}) \quad > \quad 0) \\
& \quad #x \quad \text{i} \quad > \quad 0 \\
& \quad x \quad 3 \quad \text{real} \\
& \quad *#x) \quad *#\text{lev}) \quad *#\text{lid}) \quad *#\text{rank} \\
\text{qs:} & \quad (q\text{-extrema} \quad 2 \quad q\text{size} \quad r \quad (\text{min, max}) \quad \text{packed together} \\
& \quad q \quad \text{qsize} \quad r \\
& \quad *#x) \quad *#\text{lev} \quad *#\text{lid} \quad *#\text{rank} \quad < \text{send/recv q data}
\end{align*}
\]
Top-level algorithm at each time step

```c
/* xs: (#x-tgt-rank int
   pad i
   (lid-src-rank i only packed if #x in lid > 0
   #x-in-lid i > 0
   (lev i only packed if #x in (lid,lev) > 0
   #x i > 0
   x 3 real
   *#x) *#lev) *#lid) *#rank */

setup_irecv(cm); // Set up to receive departure point requests.

// Determine where my departure points are. Set up requests to
// remotes as well as to myself to fulfill these. Fill metadata.
// In parallel over my elements, determine element containing
// departure point. Count:
// # points in an element,
// # points in an (element, level).
analyze_dep_points(cm, nets, nete, dep_points);
// In parallel over remote ranks, calculate offsets and fill in
// xs metadata.
pack_dep_points_sendbuf_pass1(cm);
// In parallel over elements having remotes in halo, fill x
// bulk data and record some offsets.
pack_dep_points_sendbuf_pass2(cm, dep_points);

isend(cm); // Send requests.
```
Top-level algorithm at each time step

// While waiting, compute q extrema in each of my elements.
calc_q_extrema<np>(cm, nets, nete);

// Wait for the departure point requests.
recv_and_wait_on_send(cm);
Top-level algorithm at each time step

// Compute the requested q for departure points from remotes.
calc_rmt_q<np>(cm);

// Send q data, skipping messages to ranks who made 0 requests.
isend(cm, false /* want_req */, true /* skip_if_empty */);
Top-level algorithm at each time step

```c
// Set up to receive q for each of my departure point requests
// sent to remotes.
setup_irecv(cm, true /* skip_if_empty */);

// While waiting to get my data from remotes, compute q for
// departure points that have remained in my elements.
calc_own_q<np>(cm, nets, nete, dep_points, q_min, q_max);

// Receive remote q data.
recv(cm, true /* skip_if_empty */);
// Copy these data into my data structures.
copy_q(cm, nets, q_min, q_max);
```
Fidelity study

- Nondivergent flow test case.
- Compare (1) tuned parameters and (2) operational parameters, as in previous slide.
- SL transport is uniformly more accurate.
- For climate results, see Nov 2018 DOE Modeling PI Meeting poster.

---


Resolution: DCMIP2016 Baroclinic Instability

- Configuration: theta-$l$, nonhydrostatic mode, moist, $n_e = 30$, $t_{step} = 300$, $r_{split} \times q_{split} = 6$
- Eulerian at left; SL at right

(a) $q_v$, level 20, day 30

(b) $q_v$, level 30, day 29

(c) Toy chemistry tracer, level 30, day 30

(d) Toy chemistry diagnostic, level 30, day 15