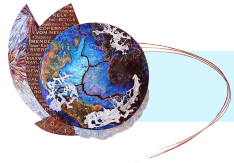
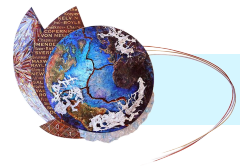


In Search of Lost Runtime

IO Server, Caches, Loops, and Parallelism

Ashley Pera (CIRES, NOAA)
Andy Jacobson (CIRES, NOAA), John Miller (NOAA)

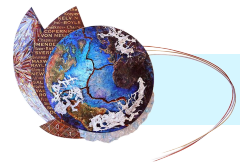




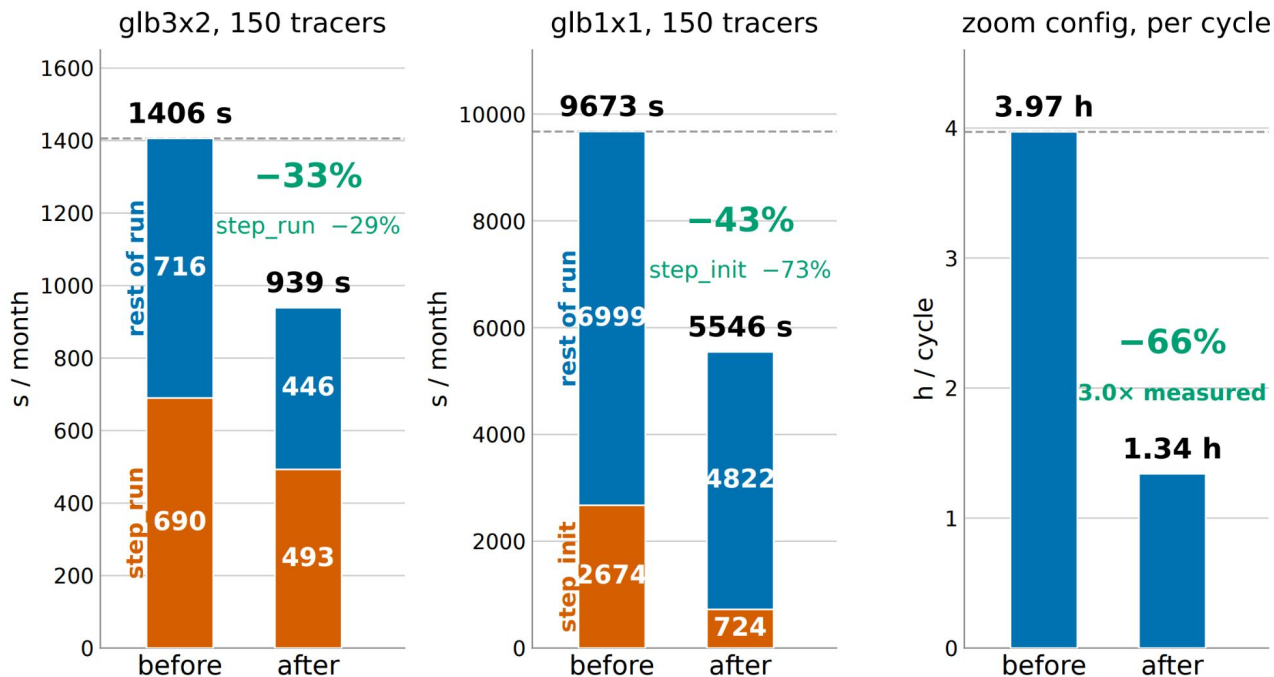
Overview



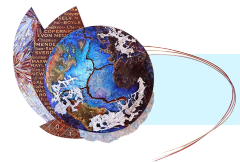
- CarbonTracker – ensemble with many tracers (600+)
 - TM5-zoom
- Iterated many times (EnKF)
- TM5 computes things **twice** and **far from data**
- Everything is bandwidth-bound, not compute
- 3x speedup, verified to 3 ULP



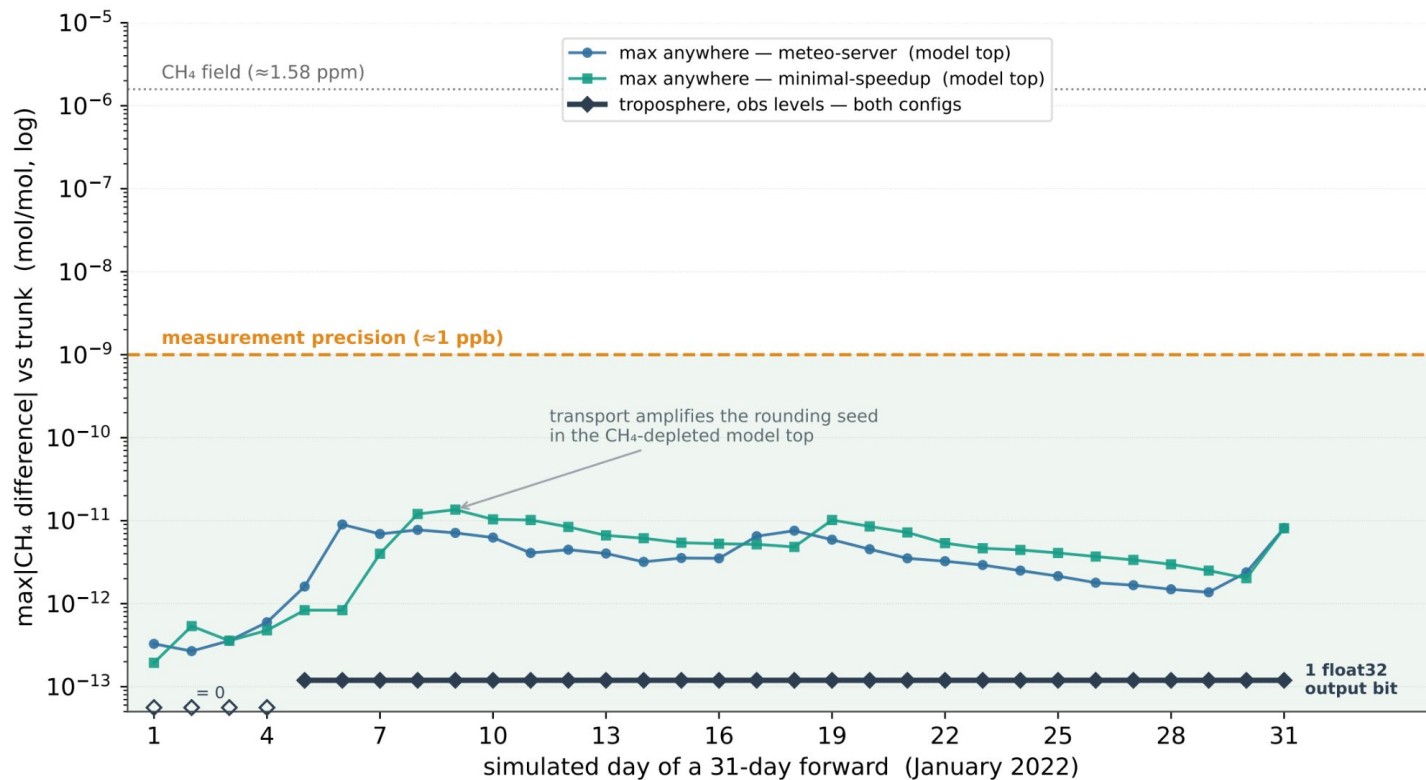
Speedup



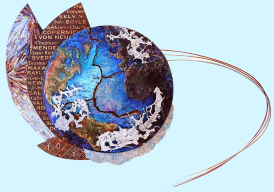
- TM5 computes things **twice** and **far from data**
- Cache or move closer



Verification – Bounded Drift

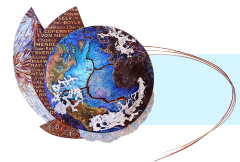


- 31-day glb3×2+nam1×1 forward (CH₄)



Details





Memory Hierarchy

fast and close to the core

move it closer

cache

L3 cache

27.5 MiB per socket
640 GB/s node

↓ 3× less bandwidth

DRAM

192 GB per node
191 GB/s

↓ 27,000× less bandwidth

Lustre (HDF5)

PB-scale, shared
~7 MB/s random access

slow and far

Batched LAPACK, loop order

conv/invert 862 → 22 s (39×)

Kzz on MPI-3 shmem

calc_kzz 94.3 → 16.7 s

Meteo server: prefetch

step_init 872 → 109 s (8×)

Poisson eigenvalue table

64,800 cos() → once (−16 s †)

CFL ndyn cache

check_cfl 85.7 → 0.51 s (−99%)

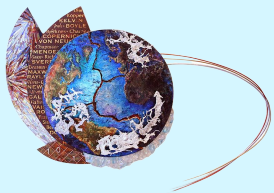
dkg cache + coarsened meteo

reads skipped; ~24× less data

Lustre

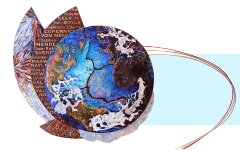
1 minute at the core $\approx$ 2 weeks here

level 3 of 3



Farthest Data-Disk





Meteo Server

without the server — every rank reads, then computes

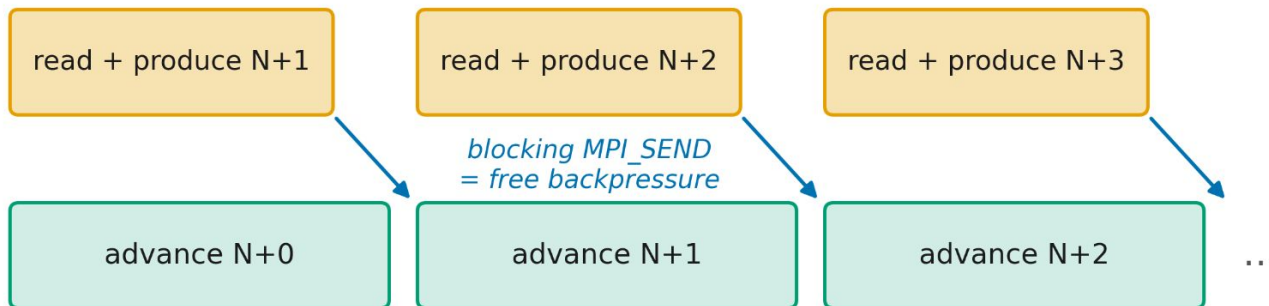
all ~80
ranks



~46% of wall at 1x1

with the server — IO ranks prefetch N+1 while compute advances N

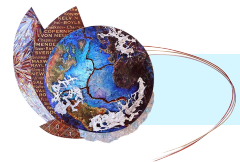
IO ranks
x 2-3



compute
x ~80

io_wait 403 → 214 → 12.9 s (2 IO → 3 IO → 3 IO + coarsened meteo) step_init 872 → 109 s

- We already know ahead of time what we need to read—just read it first

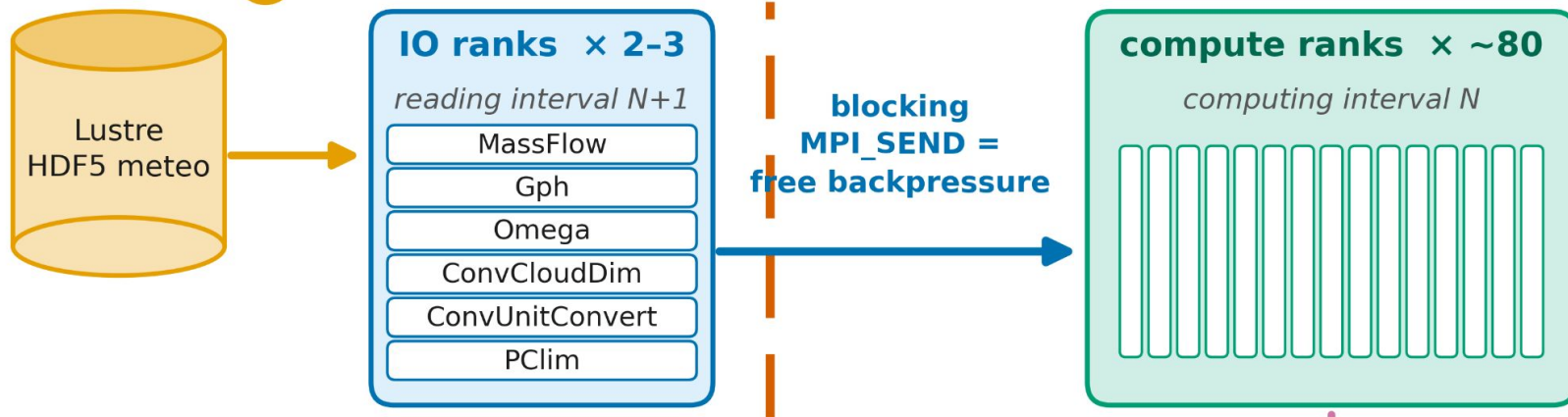


Meteo Server

2026-07-07 contract

the IO side may only produce pure functions of the meteo files + the interval clock

1 ~7 MB/s random access; 46% of wall at 1x1

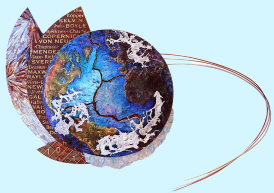


- The simple architecture: have the IO ranks skip compute, just read/send 9

DRAM

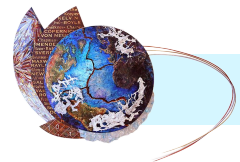
1 minute at the core $\approx$ 11 minutes here

level 2 of 3



Closer Data-DRAM





CFL Ndyn Cache

- The Ndyn is recomputed every step
 - Trial advection to reach CFL
- Only depends on: ndyn_max, max_nloop, grid, regions, and meteo
- A ***single integer***, as much as 30% of the runtime (1x1)

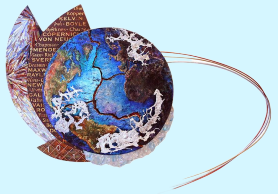
```
root          1037 s
├─ step_init   230 s (22% total)
│   └─ check_cfl    97  ← 42%
│   └─ mass/balance  37  ← 16%
│   └─ everything else  91  ← 42%
└─ step_run     800 s (77% total)
```

1-month, 600 tracers	Computed	Cached
3x2	96.7 s	0.51 s
1x1	1419 s	8.6 s

L3 cache

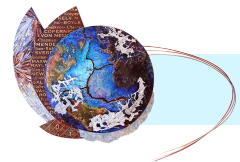
*1 minute at the core $\approx$ 6 minutes here
— nearly awake*

level 1 of 3



Closest Data–On Die

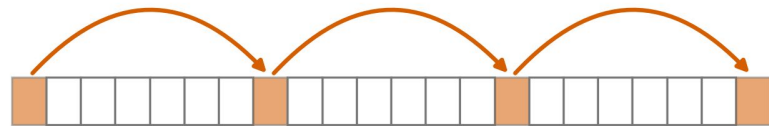




Loop reordering

```
!$omp do schedule(static)
do k_lev = 1, n_levels ! parallel over levels...
  do j_lat = lat_start, lat_end
    do i_lon = lon_start, lon_end
      do i_tracer = 1, nt_loc ! ...but tracer INNERMOST
        rm(i_lon,j_lat,k_lev,i_tracer) = rm(i_lon,j_lat,k_lev,i_tracer) * decay_val
        ! rm(i,j,k,n) is column-major, n slowest-varying:
        ! every i_tracer step jumps  $im*jm*lm*8 = 2.9 \text{ MB}$  -> L1/L2 thrashed
```

tracer innermost

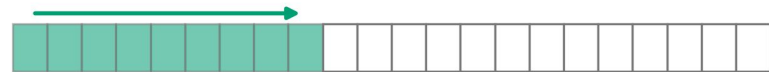


2.9 MB per step

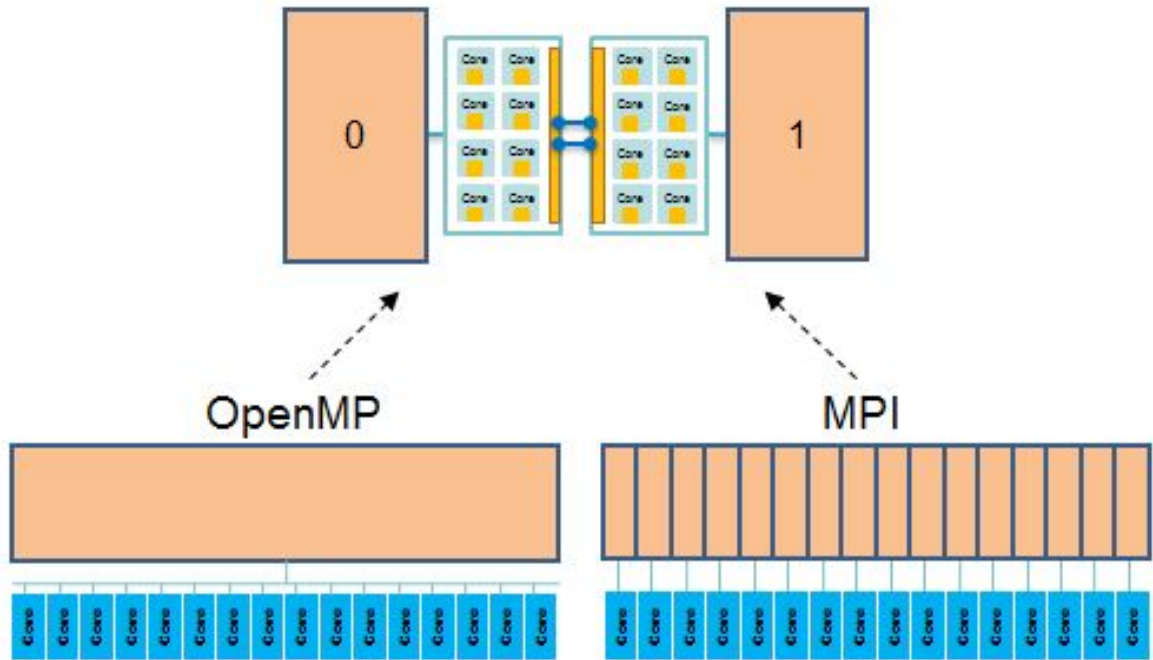
```
! Tracer outermost for cache locality:
! Sweeping one tracer at a time keeps i_lon stride
```

```
do i_tracer = 1, nt_loc
!$omp do schedule(static)
do k_lev = 1, n_levels
  do j_lat = lat_start, lat_end
    do i_lon = lon_start, lon_end
      rm(i_lon,j_lat,k_lev,i_tracer) = rm(i_lon,j_lat,k_lev,i_tracer) * decay_val
```

tracer outermost

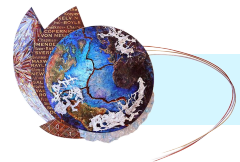


stride-1 → -25%



MPI Shmem/Omp





KZZ Shmem

```
call MPI_Win_allocate_shared(dkg_winsize, 8, MPI_INFO_NULL, &
                             node_comm, dummy_ptr, kzz_dkg_win, ierr )
call c_f_pointer(dkg_baseptr, dkg_shared, (/imr, jmr, lmax_conv/))
if (node_rank == 0) then
    dkg_shared = 0.0 ; blh_shared = 0.0
end if
call MPI_Barrier(node_comm, ierr)
call bldiff(region, phlb, gph, t, q, m, pu, pv,
            dkg_shared, blh_shared, j_mpi_start, j_mpi_end )
call MPI_Barrier(node_comm, ierr)
```

- When using many MPI ranks, can't split KZZ with OMP
- CAN split with MPI shared—much like OMP

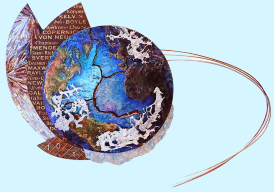
1-month, 150 tracers	Computed	Shared
3x2	94 s	17 s
1x1	857 s	74 s



Conclusions

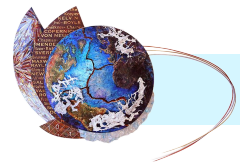


- 3x speedup TM5-zoom
 - Meteo server (disk to ram)
 - ndyn caching (ram to L3)
 - Loop reordering (L3 to L2)
 - Kzz parallel (MPI shmem)
- Save bandwidth up the levels
- Verified to 3ULP



Backup

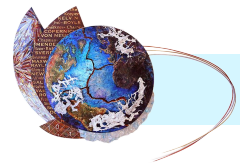




Warmup: Coarsened Meteo (cache)



- We have meteo on a 1x1 grid, but often run 3x2
- Pre-coarsened meteo: 24x less disk traffic
 - (34-level 3x2 instead of 137-level 1x1)
- 25% of the runtime saved

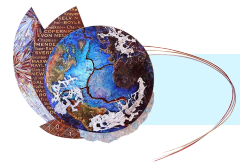


Possion Eigenvalue Cache



- Everything the Poisson solve needs besides the data — the eigenvalues of the discrete Laplacian, the FFTW plans, and for small grids the entire solve operator — depends only on the grid dimensions
- Before: recomputed per call, 64,800 `cos()` evaluations each time `BalanceMassFluxes` runs.
- After: a lazy per-(im,jm) slot table, filled once.

16s saved per month (5%)



Drift matrix

The 3×3 drift matrix — the instrument that caught czeta

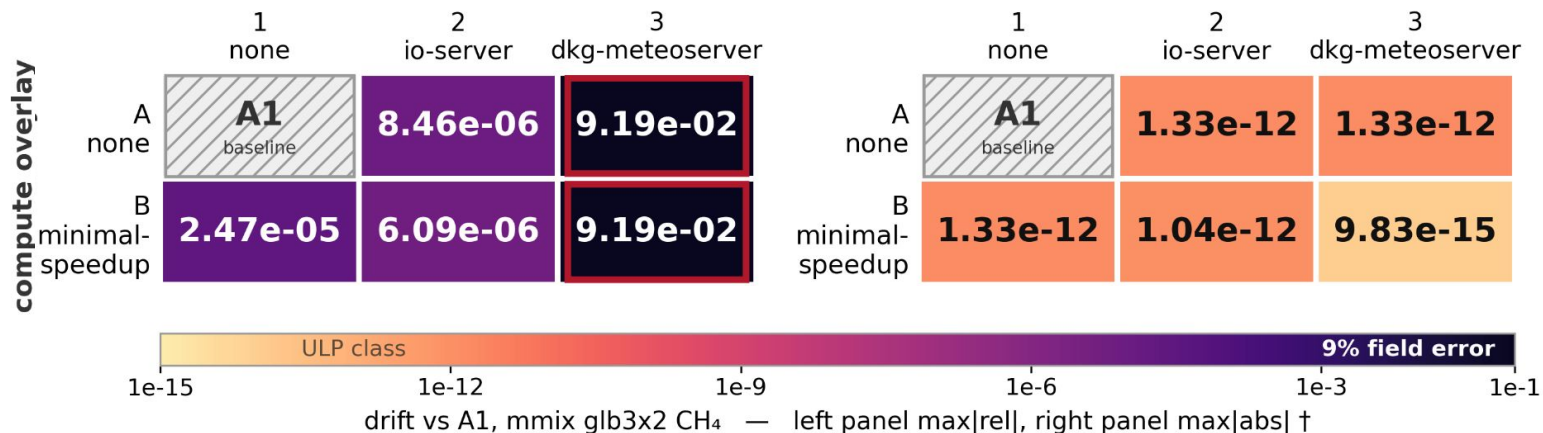
before fix

max|rel| vs A1 · 2026-05-26

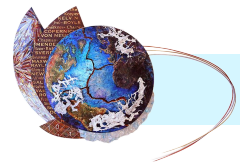
after fix

max|abs| vs A1 † · post-merge

meteo-IO architecture



A 9% field error, invisible to wall clock, exit codes, and observation output — caught only by a field-level diff. Root cause: two lines (czeta uninitialized on IO ranks).



Framing–Roofline

Why TM5 is memory-bandwidth bound, not compute bound

