IMPLEMENTING A PORTABLE GPU BACKEND FOR X3D2 USING OPENMP OFFLOADING

Paul Bartholomew

epcc

Xcompact3d

- High-order CFD code targeting DNS/LES of turbulent flows

$$\frac{du}{dt} = -\nabla p + \frac{1}{Re}\nabla^2 u - \nabla \cdot uu$$

- Compact finite difference schemes

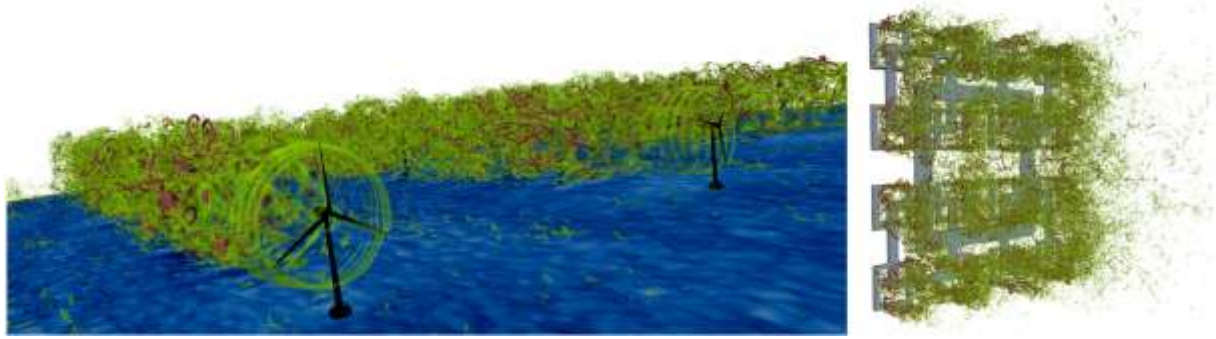
$$\alpha f'_{i-1} + f'_i + \alpha f'_{i+1} = R(f)|_i$$

- Spectral Poisson solver (FFT)

$$\nabla^2 p = \nabla \cdot u^\star \rightarrow p$$
$$u = u^\star - \Delta t \nabla p; \nabla \cdot u = 0$$

- Pencil-based decomposition

$$N^3 \rightarrow N^2\ parallelism$$



Wind farm (left) and fractal grid (right) simulated using Xcompact3d. Bartholomew *et al.* SoftwareX (2020).

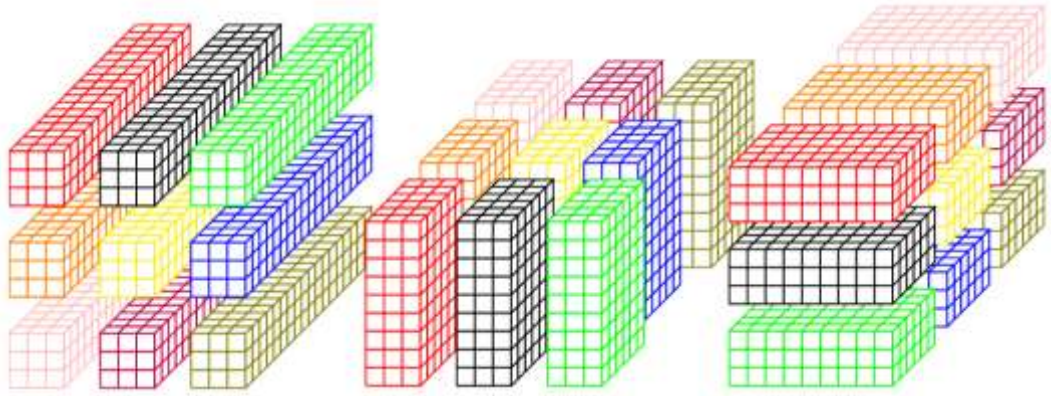
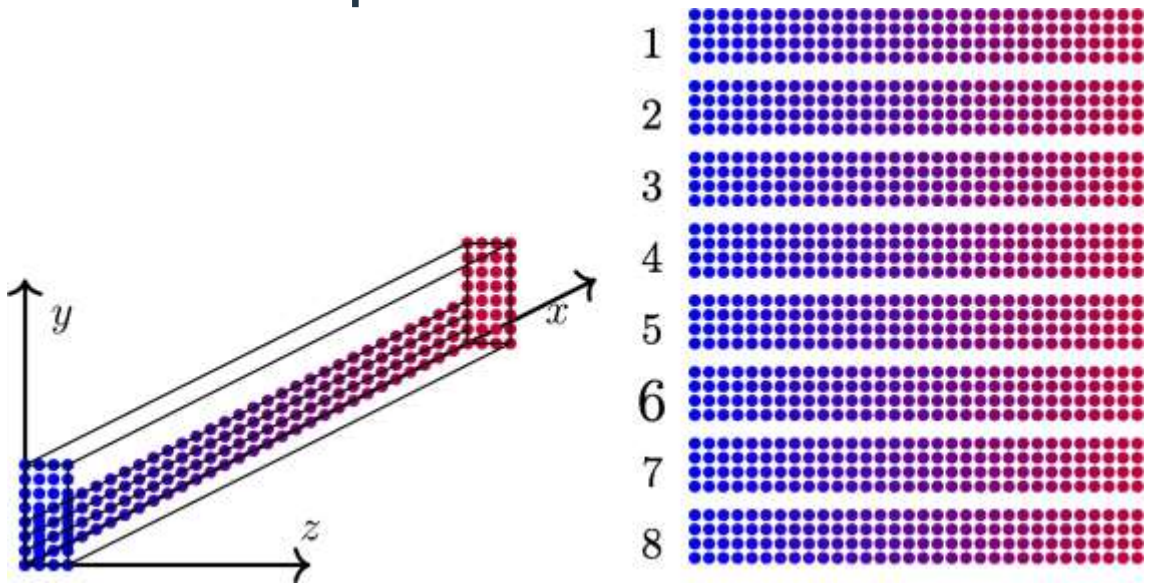


Illustration of pencil-based decomposition. Bartholomew & Laizet (2019).

epcc

x3d2 redesign

- Xcompact3d designed for small-node, distributed parallelism
  - Exploits embarrassing parallelism of independent tridiagonal systems/FFTs
- Poorly suited to modern fat nodes with wide vectors/GPUs
- Data layout rewritten to break data dependencies

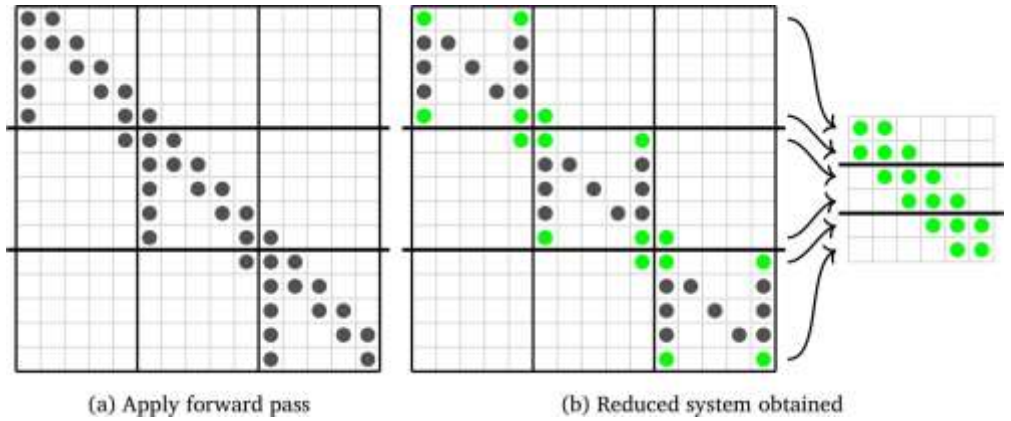


Vectorised data layout. Akkurt *et al.* Computer Physics Communications (2025).

epcc

x3d2 redesign

- Xcompact3d designed for small-node, distributed parallelism
  - Exploits embarrassing parallelism of independent tridiagonal systems
- Poorly suited to modern fat nodes with wide vectors/GPUs
- Data layout rewritten to break data dependencies
- Distributed tridiagonal solver



DistD2 tridiagonal solver. Akkurt *et al.* Computer Physics Communications (2025).

x3d2 backends

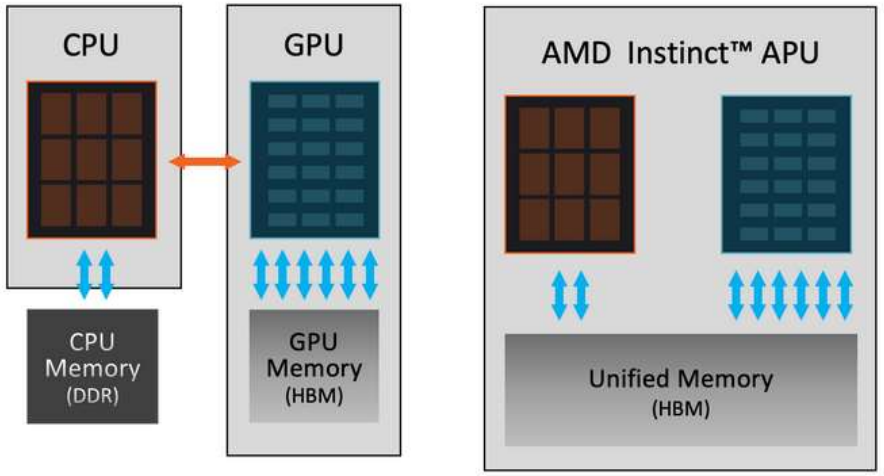
- Multicore CPUs with MPI+OMP
  - Explicit use of SIMD on inner loops
- NVIDIA GPUs with CUDA Fortran
- AMD GPUs?
  - HIP modules? See, *e.g.* NEKO[1]
  - OMP offloading?
    - Compare performance *vs* CUDA Fortran on NVIDIA
    - Target kernels for optimisation with HIP

[1] Jansson *et al.* Computers & Fluids (2024).

epcc

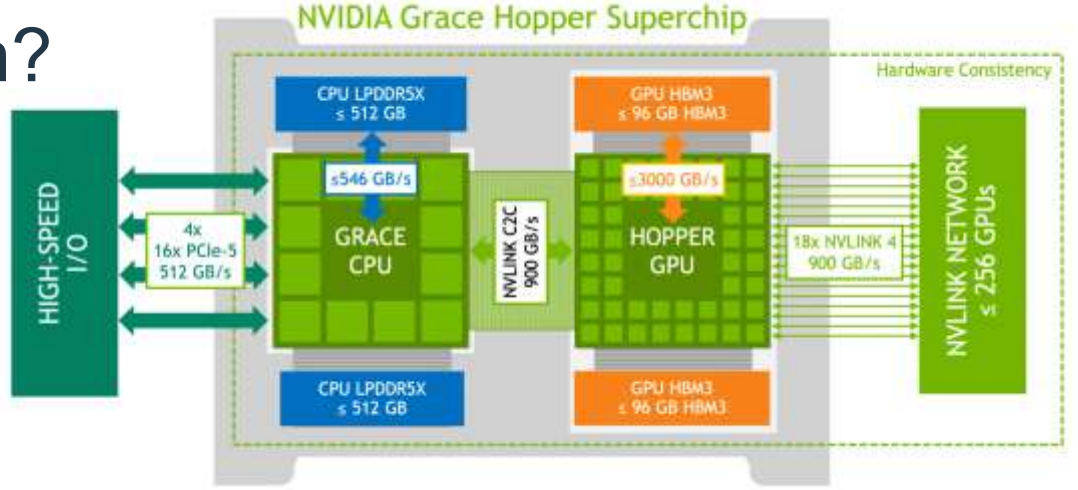
Investigating USM

- Modern GPU superchips (NVIDIA GH200, AMD MI300A) feature Unified Shared Memory (USM)
- Accelerates host-device data transfer
- Do we need to worry about data motion?



(a) Discrete CPU and GPU. (b) APU.

Tandon & Chang. https://rocm.blogs.amd.com/software-tools-optimization/mi300a-programming/README.html



Evans et al. https://developer.nvidia.com/blog/nvidia-grace-hopper-superchip-architecture-in-depth/

epcc

Investigating USM

- Simple benchmark to reflect x3d2's CUDA Fortran backend
- Timed execution
- OpenMP reports data transfers

```
! Initialise
a(:) = 1.0

! Set
!$omp target teams distribute parallel do
do i = 1, n
   b(i) = a(i)
end do
!$omp end target teams distribute parallel do

! Kernel
!$omp target teams distribute parallel do
do i = 1, n
   b(i) = 2 * b(i)
end do
!$omp end target teams distribute parallel do

! Get
!$omp target teams distribute parallel do
do i = 1, n
   a(i) = b(i)
end do
!$omp end target teams distribute parallel do

if (any(a /= 2.0)) then
   error stop
else
   print *, "PASS"
end if
```

```
$ NVCOMPILER_ACC_NOTIFY=3 OMP_TARGET_OFFLOAD=MANDATORY time ./main1
  upload CUDA data  file=.../main.f90 function=main line=16 device=0 threadid=1 variable=b(:) bytes=4000000000
  upload CUDA data  file=.../main.f90 function=main line=16 device=0 threadid=1 variable=a(:) bytes=4000000000
  launch CUDA kernel file=.../main.f90 function=main line=16 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvke
  download CUDA data  file=.../main.f90 function=main line=20 device=0 threadid=1 variable=a(:) bytes=4000000000
  download CUDA data  file=.../main.f90 function=main line=20 device=0 threadid=1 variable=b(:) bytes=4000000000
  upload CUDA data  file=.../main.f90 function=main line=23 device=0 threadid=1 variable=b(:) bytes=4000000000
  launch CUDA kernel file=.../main.f90 function=main line=23 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvke
  download CUDA data  file=.../main.f90 function=main line=27 device=0 threadid=1 variable=b(:) bytes=4000000000
  upload CUDA data  file=.../main.f90 function=main line=30 device=0 threadid=1 variable=a(:) bytes=4000000000
  upload CUDA data  file=.../main.f90 function=main line=30 device=0 threadid=1 variable=b(:) bytes=4000000000
  launch CUDA kernel file=.../main.f90 function=main line=30 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvke
  download CUDA data  file=.../main.f90 function=main line=34 device=0 threadid=1 variable=b(:) bytes=4000000000
  download CUDA data  file=.../main.f90 function=main line=34 device=0 threadid=1 variable=a(:) bytes=4000000000
   PASS
  93.92user 4.44system 1:39.31elapsed 99%CPU (0avgtext+0avgdata 7925760maxresident)k
  0inputs+0outputs (0major+162431minor)pagefaults 0swaps
```

Benchmark code (left), timings + data transfers (right).

Investigating USM

- Simple benchmark to reflect x3d2's CUDA Fortran backend
- Applied data motion controls → 2x speedup

```
! Initialise
a(:) = 1.0

! Set
!$omp target teams distribute parallel do map(to:a) map(from:b)
do i = 1, n
   b(i) = a(i)
end do
!$omp end target teams distribute parallel do

! Kernel
!$omp target teams distribute parallel do map(tofrom:b)
do i = 1, n
   b(i) = 2 * b(i)
end do
!$omp end target teams distribute parallel do

! Get
!$omp target teams distribute parallel do map(to:b) map(from:a)
do i = 1, n
   a(i) = b(i)
end do
!$omp end target teams distribute parallel do

if (any(a /= 2.0)) then
   error stop
else
   print *, "PASS"
end if
```

```
$ NVCOMPILER_ACC_NOTIFY=3 OMP_TARGET_OFFLOAD=MANDATORY time ./main2
  upload CUDA data  file=.../main2.f90 function=main line=16 device=0 threadid=1 variable=a$sd1(:) bytes=128
  upload CUDA data  file=.../main2.f90 function=main line=16 device=0 threadid=1 variable=b$sd2(:) bytes=128
  upload CUDA data  file=.../main2.f90 function=main line=16 device=0 threadid=1 variable=descriptor bytes=128
  upload CUDA data  file=.../main2.f90 function=main line=16 device=0 threadid=1 variable=a(:) bytes=4000000000
  upload CUDA data  file=.../main2.f90 function=main line=16 device=0 threadid=1 variable=descriptor bytes=128
  launch CUDA kernel file=.../main2.f90 function=main line=16 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvk
  download CUDA data  file=.../main2.f90 function=main line=20 device=0 threadid=1 variable=b(:) bytes=4000000000
  upload CUDA data  file=.../main2.f90 function=main line=23 device=0 threadid=1 variable=b$sd2(:) bytes=128
  upload CUDA data  file=.../main2.f90 function=main line=23 device=0 threadid=1 variable=descriptor bytes=128
  upload CUDA data  file=.../main2.f90 function=main line=23 device=0 threadid=1 variable=b(:) bytes=4000000000
  launch CUDA kernel file=.../main2.f90 function=main line=23 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvk
  download CUDA data  file=.../main2.f90 function=main line=27 device=0 threadid=1 variable=b(:) bytes=4000000000
  upload CUDA data  file=.../main2.f90 function=main line=30 device=0 threadid=1 variable=a$sd1(:) bytes=128
  upload CUDA data  file=.../main2.f90 function=main line=30 device=0 threadid=1 variable=b$sd2(:) bytes=128
  upload CUDA data  file=.../main2.f90 function=main line=30 device=0 threadid=1 variable=descriptor bytes=128
  upload CUDA data  file=.../main2.f90 function=main line=30 device=0 threadid=1 variable=b(:) bytes=4000000000
  upload CUDA data  file=.../main2.f90 function=main line=30 device=0 threadid=1 variable=descriptor bytes=128
  launch CUDA kernel file=.../main2.f90 function=main line=30 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvk
  download CUDA data  file=.../main2.f90 function=main line=34 device=0 threadid=1 variable=a(:) bytes=4000000000
   PASS
  44.21user 4.76system 0:49.96elapsed 98%CPU (0avgtext+0avgdata 7925760maxresident)k
  0inputs+0outputs (0major+200171minor)pagefaults 0swaps
```

Benchmark code with data motion clauses (left), timings + data transfers (right).

Investigating USM

- Simple benchmark to reflect x3d2's CUDA Fortran backend
- Made work array device-resident → ~100x speedup

```
! Initialise
a(:) = 1.0
!$omp target enter data map(alloc:b)

! Set
!$omp target teams distribute parallel do map(to:a)
do i = 1, n
   b(i) = a(i)
end do
!$omp end target teams distribute parallel do

! Kernel
!$omp target teams distribute parallel do
do i = 1, n
   b(i) = 2 * b(i)
end do
!$omp end target teams distribute parallel do

! Get
!$omp target teams distribute parallel do map(from:a)
do i = 1, n
   a(i) = b(i)
end do
!$omp end target teams distribute parallel do

if (any(a /= 2.0)) then
   error stop
else
   print *, "PASS"
end if

!$omp target exit data map(delete:b)
```

```
$ NVCOMPILER_ACC_NOTIFY=3 OMP_TARGET_OFFLOAD=MANDATORY time ./main3
  upload CUDA data  file=.../main3.f90 function=main line=17 device=0 threadid=1 variable=descriptor bytes=128
  upload CUDA data  file=.../main3.f90 function=main line=17 device=0 threadid=1 variable=a$sd1(:) bytes=128
  upload CUDA data  file=.../main3.f90 function=main line=17 device=0 threadid=1 variable=descriptor bytes=128
  upload CUDA data  file=.../main3.f90 function=main line=17 device=0 threadid=1 variable=a(:) bytes=4000000000
  launch CUDA kernel file=.../main3.f90 function=main line=17 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvk
  upload CUDA data  file=.../main3.f90 function=main line=24 device=0 threadid=1 variable=descriptor bytes=128
  launch CUDA kernel file=.../main3.f90 function=main line=24 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvk
  upload CUDA data  file=.../main3.f90 function=main line=31 device=0 threadid=1 variable=descriptor bytes=128
  upload CUDA data  file=.../main3.f90 function=main line=31 device=0 threadid=1 variable=a$sd1(:) bytes=128
  upload CUDA data  file=.../main3.f90 function=main line=31 device=0 threadid=1 variable=descriptor bytes=128
  launch CUDA kernel file=.../main3.f90 function=main line=31 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvk
  download CUDA data  file=.../main3.f90 function=main line=35 device=0 threadid=1 variable=a(:) bytes=4000000000
   PASS
  0.52user 3.86system 0:05.37elapsed 81%CPU (0avgtext+0avgdata 4027392maxresident)k
  0inputs+0outputs (0major+62607minor)pagefaults 0swaps
```

Benchmark code with device-resident work array (left), timings + data transfers (right).

Investigating USM

- Simple benchmark to reflect x3d2's CUDA Fortran backend
- Enabled USM → reference code approaches speed of optimised version

```
! Initialise
a(:) = 1.0

! Set
!$omp target teams distribute parallel do
do i = 1, n
   b(i) = a(i)
end do
!$omp end target teams distribute parallel do

! Kernel
!$omp target teams distribute parallel do
do i = 1, n
   b(i) = 2 * b(i)
end do
!$omp end target teams distribute parallel do

! Get
!$omp target teams distribute parallel do
do i = 1, n
   a(i) = b(i)
end do
!$omp end target teams distribute parallel do

if (any(a /= 2.0)) then
   error stop
else
   print *, "PASS"
end if
```

```
$ NVCOMPILER_ACC_NOTIFY=3 OMP_TARGET_OFFLOAD=MANDATORY time ./main1
  upload CUDA data  file=.../main.f90 function=main line=16 device=0 threadid=1 variable=b(:) bytes=4000000000
  upload CUDA data  file=.../main.f90 function=main line=16 device=0 threadid=1 variable=a(:) bytes=4000000000
  launch CUDA kernel file=.../main.f90 function=main line=16 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvke
  download CUDA data  file=.../main.f90 function=main line=20 device=0 threadid=1 variable=a(:) bytes=4000000000
  download CUDA data  file=.../main.f90 function=main line=20 device=0 threadid=1 variable=b(:) bytes=4000000000
  upload CUDA data  file=.../main.f90 function=main line=23 device=0 threadid=1 variable=b(:) bytes=4000000000
  launch CUDA kernel file=.../main.f90 function=main line=23 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvke
  download CUDA data  file=.../main.f90 function=main line=27 device=0 threadid=1 variable=b(:) bytes=4000000000
  upload CUDA data  file=.../main.f90 function=main line=30 device=0 threadid=1 variable=a(:) bytes=4000000000
  upload CUDA data  file=.../main.f90 function=main line=30 device=0 threadid=1 variable=b(:) bytes=4000000000
  launch CUDA kernel file=.../main.f90 function=main line=30 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvke
  download CUDA data  file=.../main.f90 function=main line=34 device=0 threadid=1 variable=b(:) bytes=4000000000
  download CUDA data  file=.../main.f90 function=main line=34 device=0 threadid=1 variable=a(:) bytes=4000000000
   PASS
  93.92user 4.44system 1:39.31elapsed 99%CPU (0avgtext+0avgdata 7925760maxresident)k
  0inputs+0outputs (0major+162431minor)pagefaults 0swaps
```

```
$ NVCOMPILER_ACC_NOTIFY=3 OMP_TARGET_OFFLOAD=MANDATORY time ./main1.usm
  launch CUDA kernel file=.../main1.f90 function=main line=16 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvk
  launch CUDA kernel file=.../main1.f90 function=main line=23 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvk
  launch CUDA kernel file=.../main1.f90 function=main line=30 device=0 host-threadid=0 num_teams=0 thread_limit=0 kernelname=nvk
   PASS
  1.65user 3.34system 0:06.01elapsed 83%CPU (0avgtext+0avgdata 4027392maxresident)k
  0inputs+0outputs (1major+14943minor)pagefaults 0swaps
```

Benchmark code (left), timings + data transfers for reference and USM builds (top/bottom).

x3d2 OpenMP TARGET status

- Device-resident field type

```
! Constructs a device-resident field
type(omptgt_field_t) function omptgt_field_init(ngrid, next, id) result(f)
  integer, intent(in) :: ngrid
  class(field_t), pointer, intent(in) :: next
  integer, intent(in) :: id

  f%refcount = 0
  f%next => next
  f%id = id

  f%dev_id = omp_get_default_device()
  f%dev_ptr = omp_target_alloc(ngrid * c_sizeof(0.0_dp), f%dev_id)
  call c_f_pointer(f%dev_ptr, f%p_data_tgt, shape=[ngrid])

end function omptgt_field_init
```

epcc

x3d2 OpenMP TARGET status

- Device-resident field type
- BLAS1-like operations defined on fields, *e.g.*

$$\frac{d\phi}{dt} = f(\phi^n) \Rightarrow \phi^{n+1} = \phi^n + f(\phi^n)\Delta t$$

```
subroutine vecadd_offload_(dims, a, x, b, y)
  integer
  real(dp subroutine vecadd_offload(self, a, x, b, y)
  real(dp
  real(dp   class(omptgt_backend_t) :: self
  real(dp   real(dp), intent(in) :: a
            type(omptgt_field_t), intent(in) :: x
  integer   real(dp), intent(in) :: b
            type(omptgt_field_t), intent(inout) :: y
  !$omp t                                                          ddr(x, y)
  do k =    integer, dimension(3) :: dims
    do j
      do    dims = self%allocator%get_padded_dims(x%dir)
        y
      end   call vecadd_offload_(dims, a, x%data_tgt, b, y%data_tgt)
    end d
  end do  end subroutine
  !$omp end target teams distribute parallel do
end subroutine
```

High-level interface

Low-level implementation

epcc

x3d2 OpenMP TARGET status

- Device-resident field type
- BLAS1-like operations defined on fields, *e.g.*

$$\frac{d\phi}{dt} = f(\phi^n) \Rightarrow \phi^{n+1} = \phi^n + f(\phi^n)\Delta t$$

- Tridiagonal solver work in progress, *e.g.*

$$f(\phi) = \frac{\partial \phi}{\partial x} \Rightarrow \phi' = A^{-1}R(\phi)$$

```
!$omp target teams distribute parallel do private(j) collapse(2) has_device_addr(du, u, thom_s)
do k = 1, n_groups
  do i = 1, SZ
    do j = 5, n_rhs - 4
      du(i, j, k) = c_m4*u(i, j - 4, k) + c_m3*u(i, j - 3, k) &
                  + c_m2*u(i, j - 2, k) + c_m1*u(i, j - 1, k) &
                  + c_j*u(i, j, k) &
                  + c_p1*u(i, j + 1, k) + c_p2*u(i, j + 2, k) &
                  + c_p3*u(i, j + 3, k) + c_p4*u(i, j + 4, k) &
                  - du(i, j - 1, k)*thom_s(j)
    end do
  end do
end do
!$omp end target teams distribute parallel do
```

Forward pass

```
!$omp target teams distribute parallel do collapse(2) private(j) has_device_addr(du, thom_f, thom_w, strch)
do k = 1, n_groups
  do i = 1, SZ
    do j = n_tds - 1, 1, -1
      ! du(j) = (du(j) - f*du(j+1)/strch(j))*w*strch(j)
      du(i, j, k) = (du(i, j, k)*strch(j) - thom_f(j)*du(i, j + 1, k))*thom_w(j)
    end do
  end do
end do
!$omp end target teams distribute parallel do
```

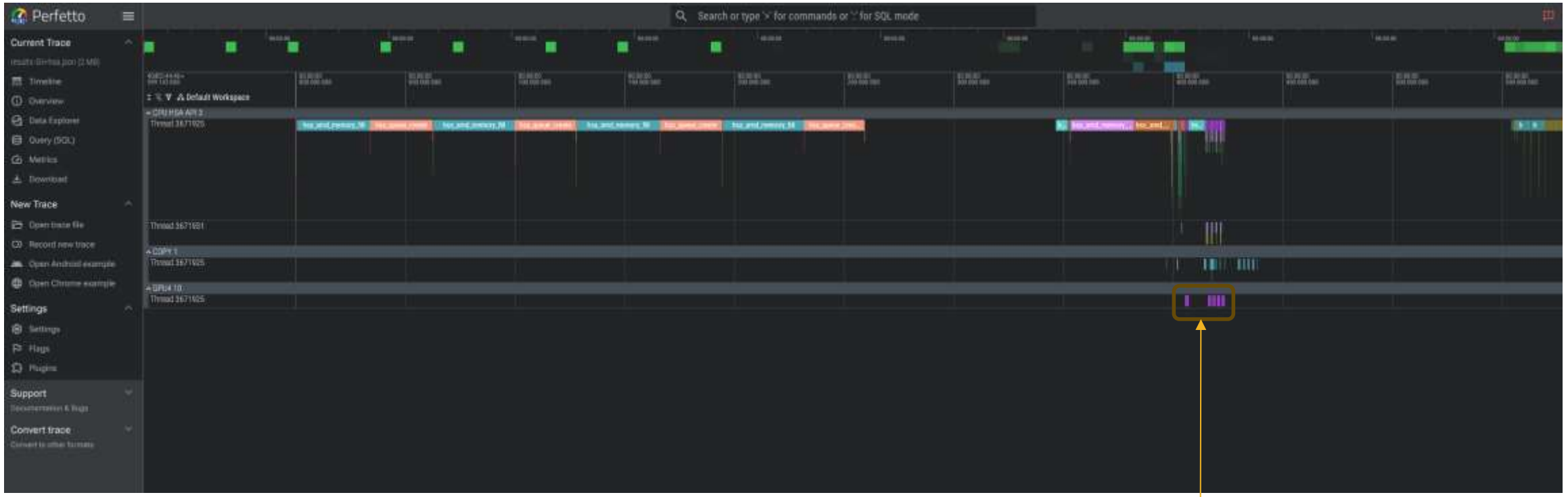
Backward pass

epcc

Testing the tridiagonal solver

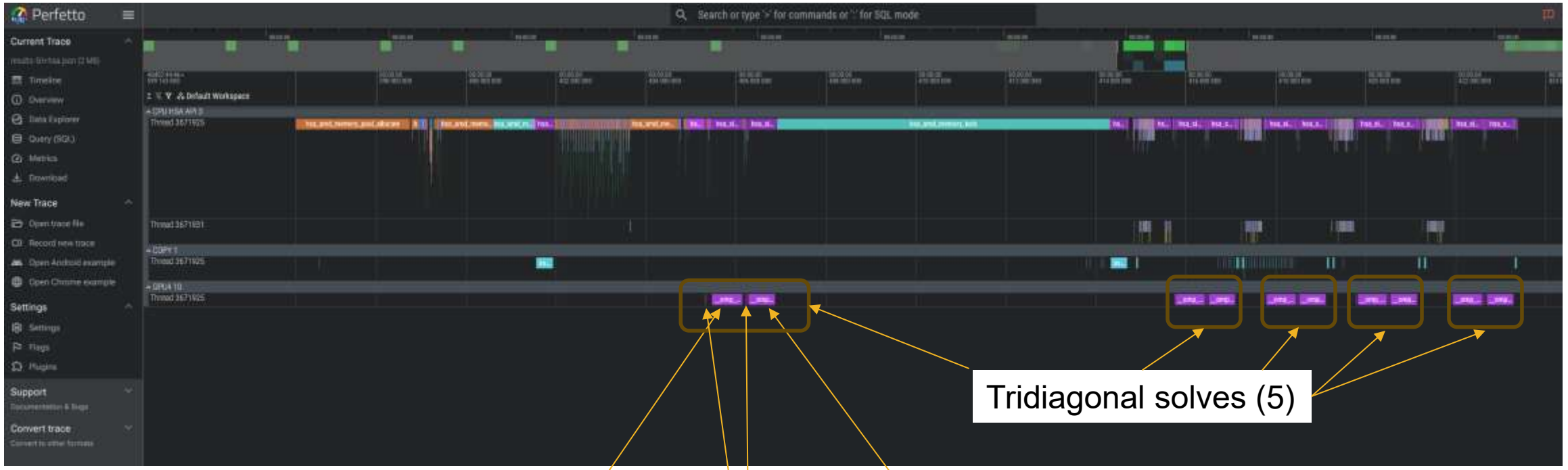
```
 Total time 2.3382840156555176
 Check performance:
Achieved BW:   40.094 GiB/s
Device BW:   9918.213 GiB/s
Effective BW util: % 0.40
 Check error:
 min: -1.5946508265249585E-07 max:  1.5942257444723218E-07
 +++: -.9999988211479345 .9999988211514741
 error norm 7.084328587195277E-09
 Check second derivatives... PASSED
 ALL TESTS PASSED SUCCESSFULLY.
```

Testing the tridiagonal solver



epcc

Testing the tridiagonal solver



epcc

x3d2 next steps

- Integrate offload code into main code
- Poisson solver
  - FFT: rocFFT (cuFFT already implemented)
  - GEMM: P.Costa *et al.* (arXiv:2603.09528)
    - Mesh flexibility
    - Favourable arithmetic intensity for GPUs?
    - Exploits vendor libraries (hipBLAS/cuBLAS)

- Acknowledgements
  - Thanks to ARCHER2 eCSE-GPU funding and DIRAC for AMD GPU access