Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Home Affairs FDHA
Federal Office of Meteorology and Climatology MeteoSwiss

Preparing dawn for Weather and Climate Models on Triangular Grids

ENES, 26.05.2020

Matthias Röthlin

Dawn?

Dawn - Compiler toolchain to enable generation of high-level DSLs for geophysical fluid dynamics models

Agenda

- Motivation & Dawn Overview
- Summary of Developments so far
- Dawn for Triangular Meshes
- Outlook

Motivation

Model software development starts at numerical discretization of continuous quantities:

$$\underline{\nabla}_{\underline{n}}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

Motivation

- (very) straight forward implementation
- "actual science" + mesh

```
DO jk = slev, elev
 DO je = i_startidx, i_endidx
   grad_norm_psi_e(je,jk) =
       (psi_c(iidx(je,2),jk)-psi_c(iidx(je,1),jk))/lhat(je)
 ENDDO
END DO
```

Motivation

- turns out mesh is too large for one machine, add blocks

```
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, jb, i_startblk, i_endblk, &
                    i_startidx, i_endidx, rl_start, rl_end)
 DO jk = slev, elev
   DO je = i_startidx, i_endidx
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
```

Motivation

- code doesn't perform, add directives to exploit shared memory machines

```
#ifdef _OMP
!$OMP PARALLEL
!$OMP DO PRIVATE(jb, i_startidx, i_endidx, je, jk)
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, jb, i_startblk, i_endblk, &
                    i_startidx, i_endidx, rl_start, rl_end)
 DO jk = slev, elev
   DO je = i_startidx, i_endidx
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP END DO NOWAIT
!$OMP END PARALLEL
#endif
```

Motivation

- code needs to target another architecture...
- ... with different optimal memory layout

```
#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```


Grants No 675191 & 823988

Motivation

$$\underline{\nabla}_{\underline{n}}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

Motivation

What if

- Requirements change, e.g. it turns out that this gradient should have been approximated using a higher order stencil?
- A third (fourth...) architecture needs to be supported?
- The mesh library needs to be replaced?

```
#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

Motivation

Idea of dawn / DSLs in general

$$\underline{\nabla_n}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
grad_norm_psi_e =
      reduce( psi_c,
              CELL > EDGE,
              [1/lhat, -1/lhat] )
```

OMP → **dawn**

```
!$OMP PARALLEL
!$OMP DO PRIVATE(jb, i_startidx, i_endidx, je, jk)
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
!$OMP END DO NOWAIT
!$OMP END PARALLEL
```

Motivation

Idea of dawn / DSLs in general

$$\nabla_n \psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
grad_norm_psi_e =
      reduce( psi_c,
              CELL > EDGE,
              [1/lhat, -1/lhat] )
```

Open ACC

dawn

```
!$ACC PARALLEL &
!$ACC PRESENT(ptr_patch, iidx, iblk, pci_c,
grad_...)
!$ACC LOOP GANG
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 DO jk = slev, elev
   DO je = i_startidx, i_endidx
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
!$ACC END PARALLEL
!$ACC END DATA
```


Grants No 675191 & 823988
Matthias Röthlin

Motivation

No FORTRAN Backend Exists, only for illustration purposes

Idea of dawn / DSLs in general

$$\nabla_n \psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
grad_norm_psi_e =
      reduce( psi_c,
              CELL > EDGE,
              [1/lhat, -1/lhat] )
```

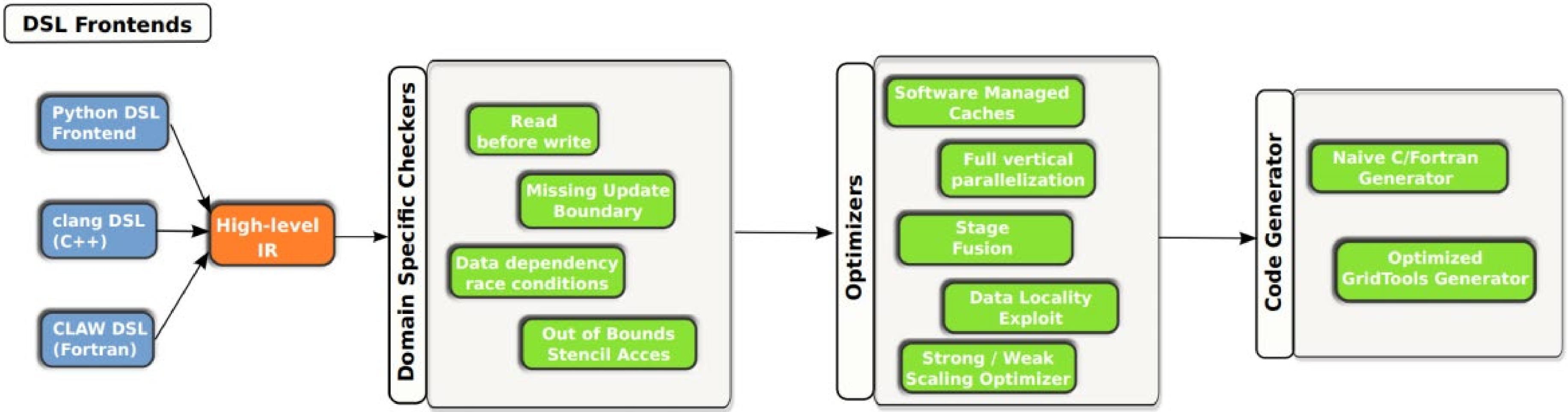
Open ACC

dawn

```
for(int k = 0 + 0; k < m_k_size; ++k) {
 for(auto const& loc : getEdges(LibTag{}, m_mesh)) {
   for(auto inner_loc :
     grad_norm_psi_e(loc, k + 0) = reduce(
         LibTag{}, m_mesh, loc, (::dawn::float_type)0.0,
         std::vector<dawn::LocationType>
         {dawn::Edges, dawn::Cells},
         [&](auto& lhs, auto red_loc1, auto const& weight)
         {
           lhs += weight * psi_c(red_loc1, k + 0);
           return lhs;
         },
         std::vector<::dawn::float_type>({-1.0 ,1.0});
   }
   grad_norm_psi_e(loc, k + 0) /= lhat_e(loc, k + 0)
 }
}
```

Dawn Overview

Dawn - A closer look

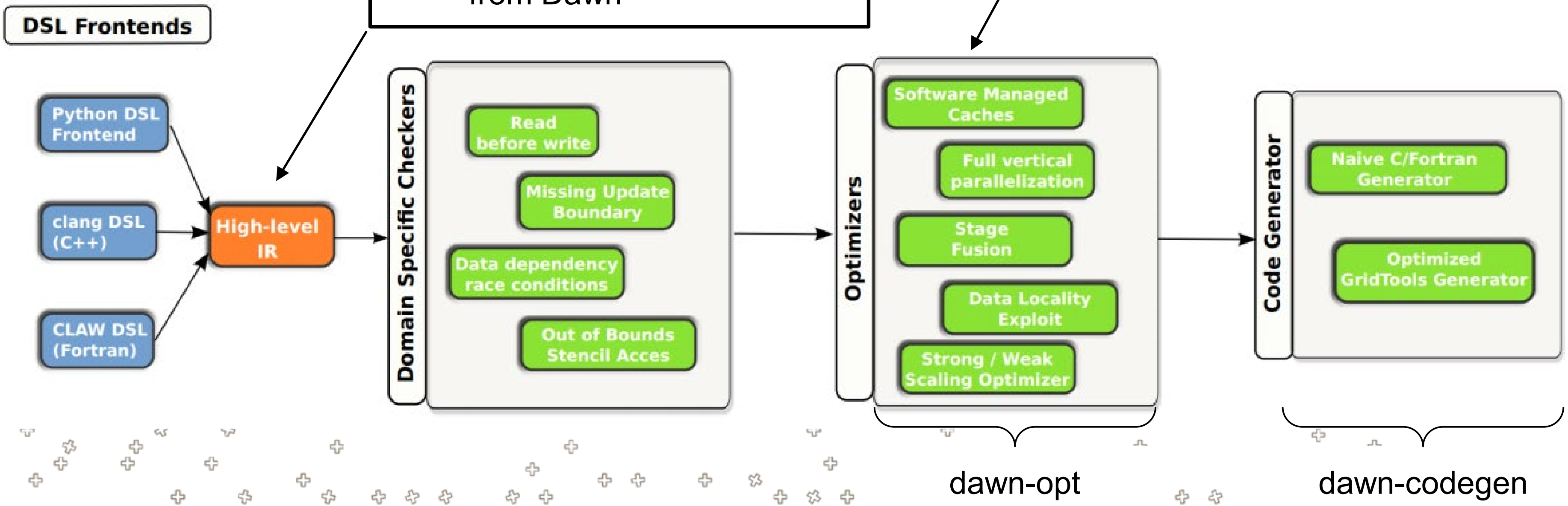


Dawn Overview

Dawn - A closer look

- High Level Intermediate Representation
- Decouples Front Ends from Dawn

- Various optimization passes
- Insertion of pragmas only very small subset of possibilities



Dawn - Developments Summary

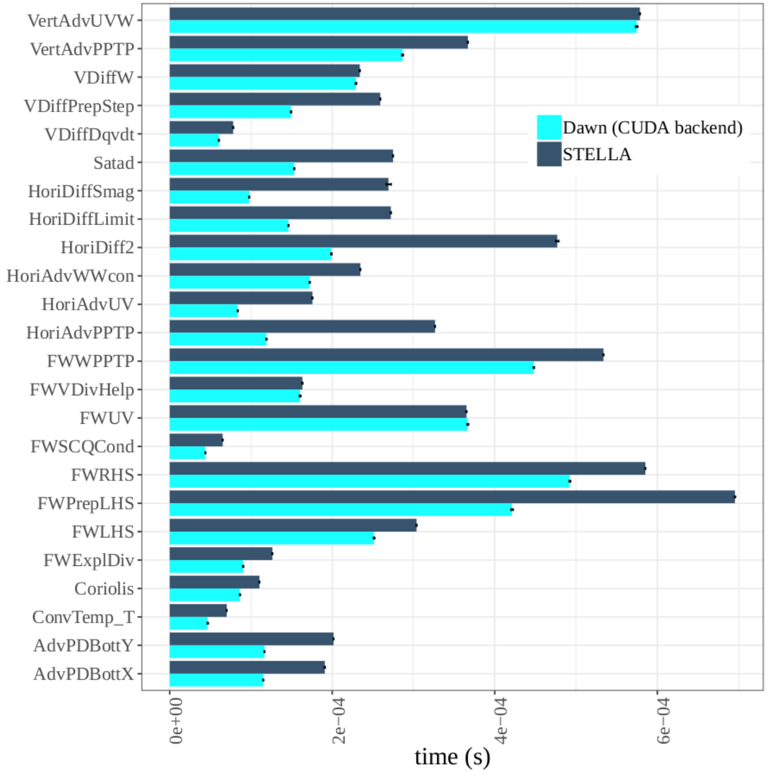
- Dawn currently ships with a frontend called "gtclang"
  - embedded into C++
  - complete w.r.t COSMO dycore in particular / Finite Differences in general
- Wide array of optimization strategies
- Various backends
  - C++ naive
  - gridtools MC / GPU
  - cuda

Dawn - Developments Summary

- Dawn was used to successfully translate the complete COSMO dycore
  - advection schemes
  - diffusion
  - tridiagonal solver
  - ...

- Outperforms previous efforts of translating the COSMO dycore using DSLs, at a fraction of the lines of code

Dawn - Developments Summary



Dawn for Triangular Meshes

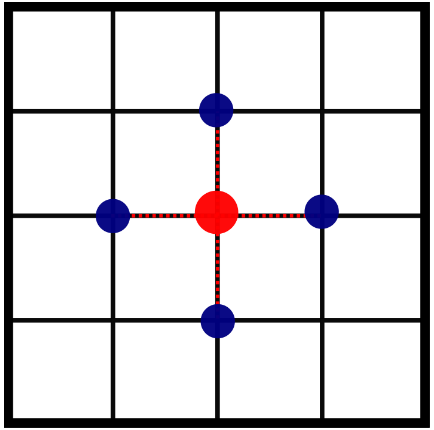
- Dawn is to be used to translate the ICON dycore
- Switch from Finite Differences on a Cartesian mesh to more general computations on a icosahedral triangle mesh
- New concepts are required to map these kind of computations, starting from the frontend all throughout the tool chain until code generation

Dawn for Triangular Meshes

Basic Idea: Reductions instead of "North-East-South-West" accesses

```
field lapl, u
lap(i,j) = -4*u(i,j)
          + u(i-1,j) + u(i+1,j)
          + u(i,j-1) + u(i,j+1)
```

cartesian access



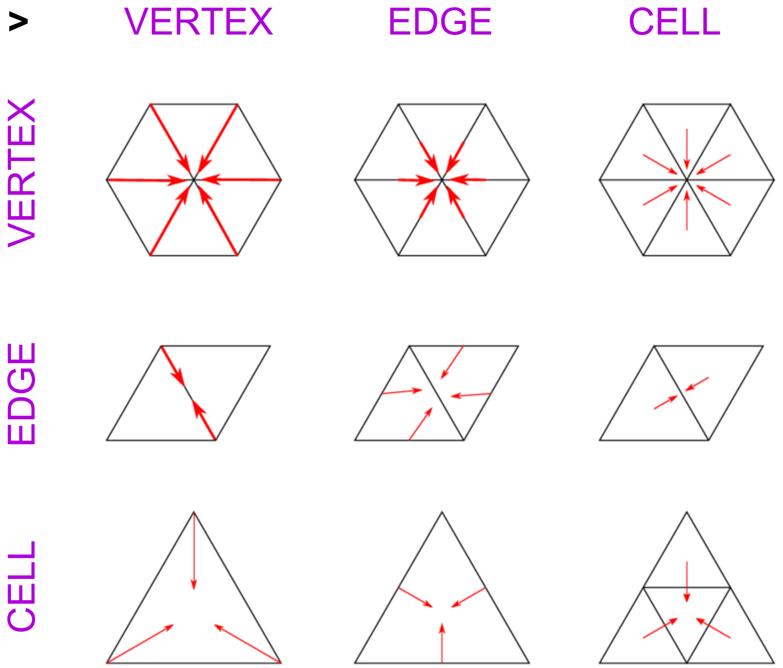
```
VertxField lapl, u
lapl = reduce( VERTEX>VERTEX
               u )
lapl = lapl - 4*u
```

unstr. access

Dawn for Triangular Meshes

Basic Idea: Reductions

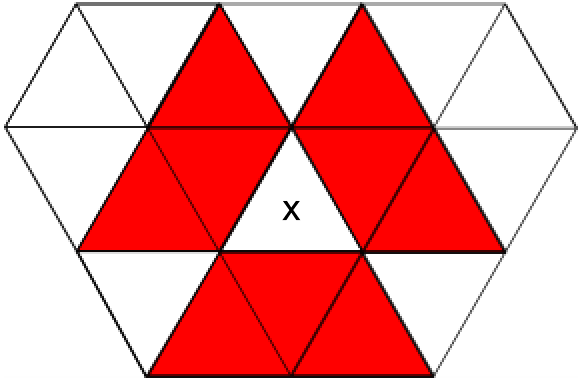
```
cellField div_c;
edgeField flux, n;
div_c = reduce(
          CELL>EDGE,
          flux*n)
```

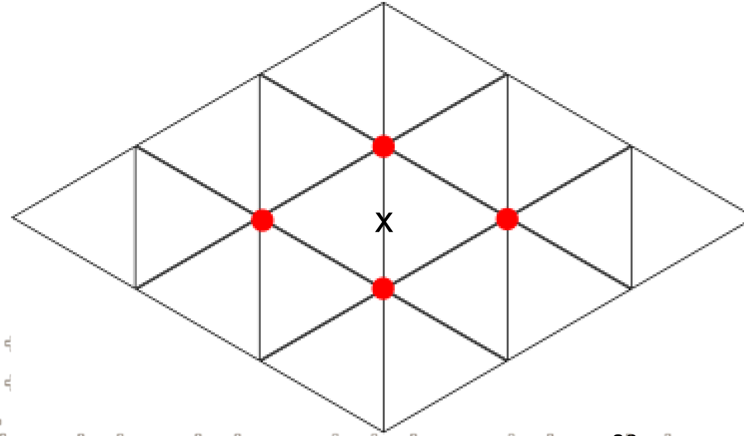


Dawn for Triangular Meshes

Typical computations in ICON are more general

- **larger neighborhoods** / "wider" stencils
- weighted reductions (e.g. gradients)
- "sparse dimensions"
  - arrays of higher rank which store values for each neighbor in the reduction



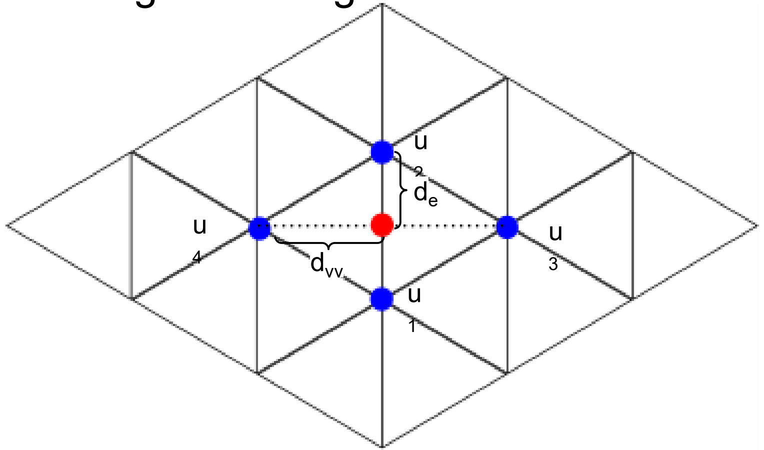


MeteoSwiss esiwace CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE
Grants No 675191 & 823988

Dawn for Triangular Meshes

Worked Example - Laplacian Finite Difference Stencil on a hexagonal triangle mesh

$$\nabla^2(u) \approx \frac{u_1 + u_2 - 2 \cdot u}{d_e^2} + \frac{u_3 + u_4 - 2 \cdot u}{d_{vv}^2}$$

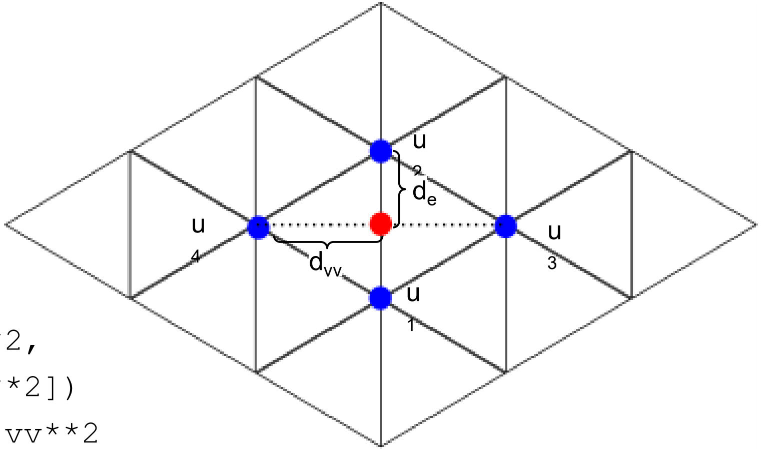


Dawn for Triangular Meshes

Worked Example - Formula to DSL

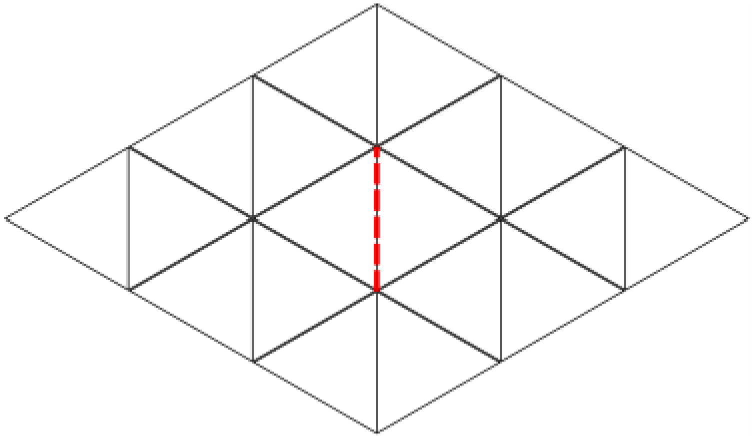
$$\nabla^2(u) \approx \frac{u_1 + u_2 - 2 \cdot u}{d_e^2} + \frac{u_3 + u_4 - 2 \cdot u}{d_{vv}^2}$$

```
vertexField u;
edgeField lapl;
lapl = reduce(EDGE>CELL>VERTEX,
              u,
              [1/d_e**2 , 1/d_e**2,
               1/d_vv**2, 1/d_vv**2])
lapl = lapl - 2*u/d_e**2 - 2*u/d_vv**2
```



Dawn for Triangular Meshes

Worked Example - Building a the reductions "Neighbor Chain"

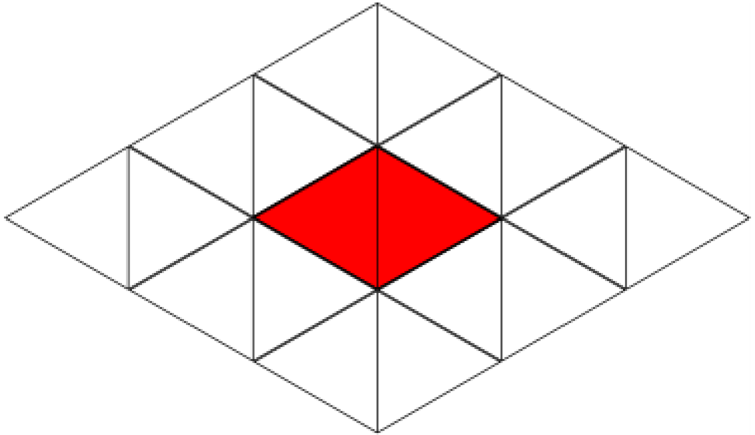


EDGE>CELL>VERTEX

Grants No 675191 & 823988

Dawn for Triangular Meshes

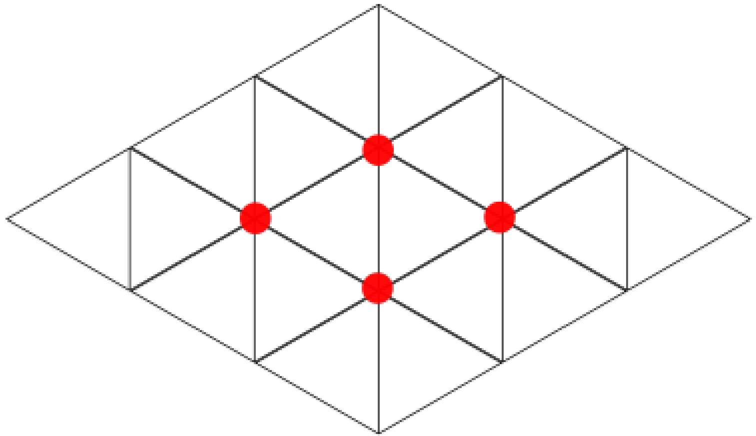
Worked Example - Building a the reductions "Neighbor Chain"



EDGE>**CELL**>VERTE
X

Dawn for Triangular Meshes

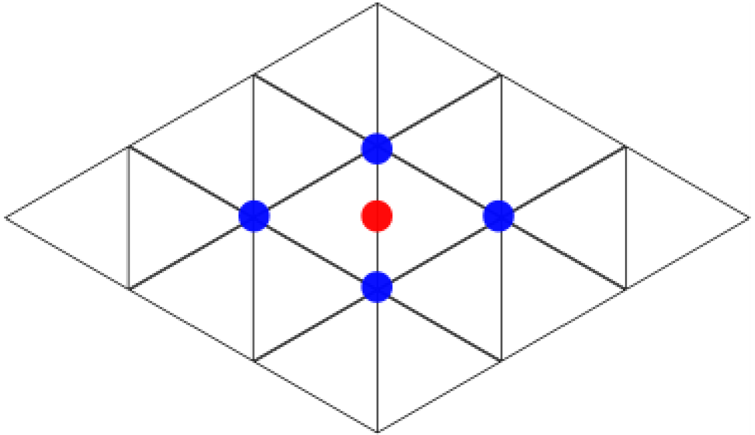
Worked Example - Building a the reductions "Neighbor Chain"



EDGE>CELL>**VERTE**
X

Dawn for Triangular Meshes

Worked Example - Building a the reductions "Neighbor Chain"



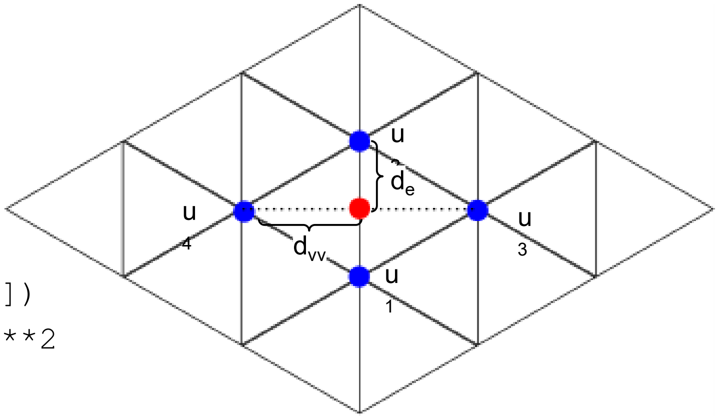
EDGE>CELL>VERTE
X

Dawn for Triangular Meshes

Worked Example - Formula to DSL

$$\nabla^2(u) \approx \frac{u_1 + u_2 - 2 \cdot u}{d_e^2} + \frac{u_3 + u_4 - 2 \cdot u}{d_{vv}^2}$$

```
vertexField u;
edgeField lapl;
lapl = reduce(EDGE>CELL>VERTEX,
              u,
              [1/d_e**2 , 1/d_e**2,
               1/d_vv**2, 1/d_vv**2])
lapl = lapl - 2*u/d_e**2 - 2*u/d_vv**2
```

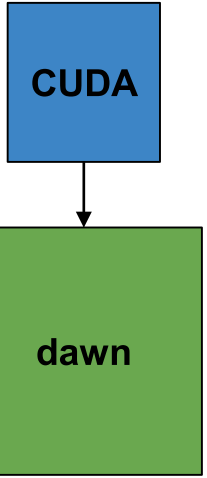


Worked Example - DSL to Codegen

```
vertexField u;
edgeField lapl;
lapl = reduce(EDGE>CELL>VERTEX,
              u,
              [1/d_e**2 , 1/d_e**2,
               1/d_vv**2, 1/d_vv**2])
lapl = lapl - 2*u/d_e**2 - 2*u/d_vv**2
```

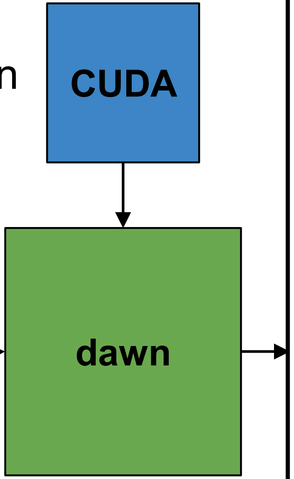
Worked Example - DSL to Codegen

```
vertexField u;
edgeField lapl;
lapl = reduce(EDGE>CELL>VERTEX,
              u,
              [1/d_e**2 , 1/d_e**2,
               1/d_vv**2, 1/d_vv**2])
lapl = lapl - 2*u/d_e**2 - 2*u/d_vv**2
```



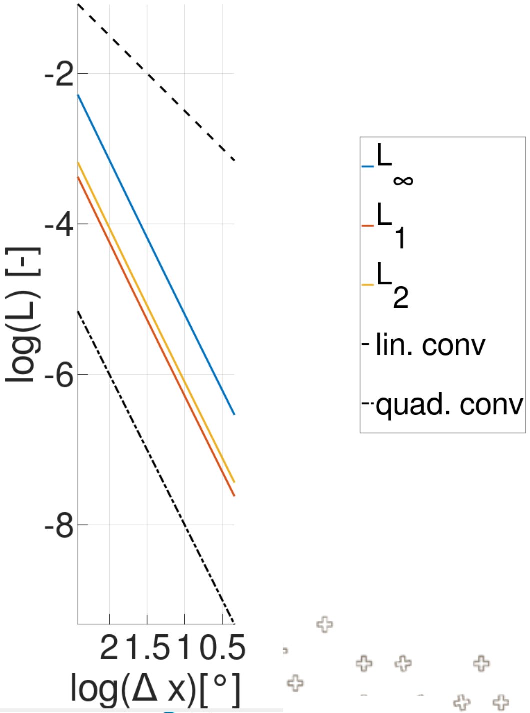
Worked Example - DSL to Codegen

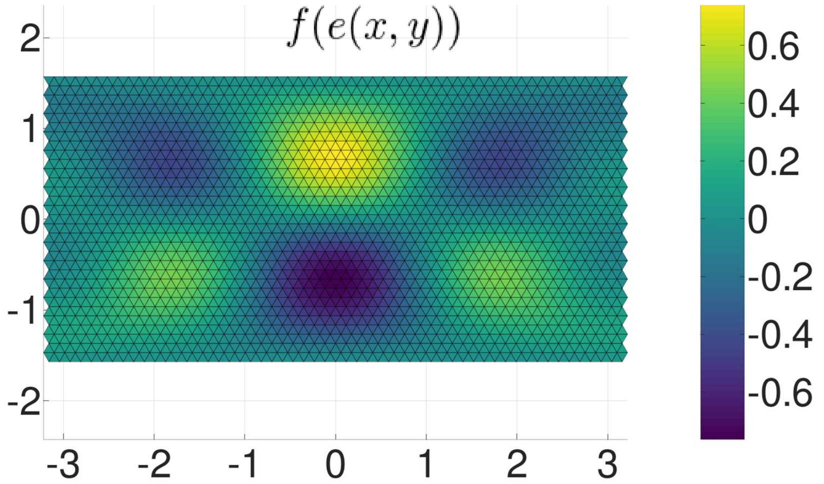
```
vertexField u;
edgeField lapl;
lapl = reduce(EDGE>CELL>VERTEX,
              u,
              [1/d_e**2 , 1/d_e**2,
               1/d_vv**2, 1/d_vv**2])
lapl = lapl - 2*u/d_e**2 - 2*u/d_vv**2
```

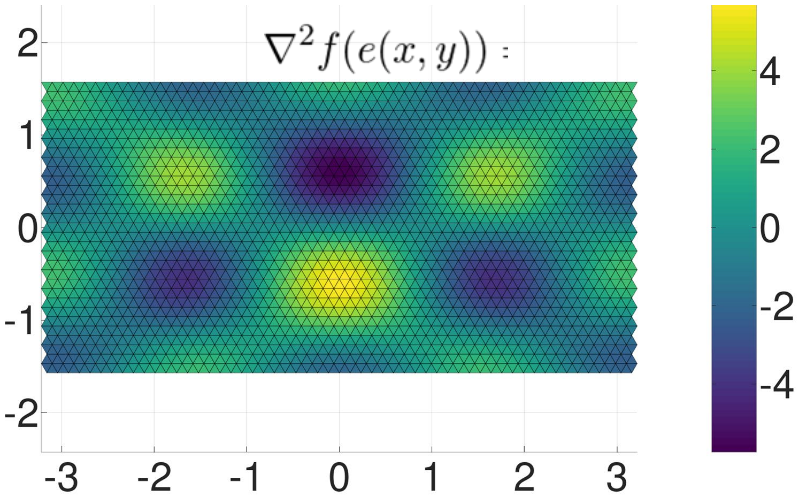



```
template <int E_C_V_SIZE>
__global__ void ICON_laplacian__ms439_s466_kernel(
   int NumEdges, int NumVertices, int kSize, const int*
ecvTable,
   const ::dawn::float_type* __restrict__ u,
    ::dawn::float_type* __restrict__ lapl) {
 unsigned int pidx = blockIdx.x * blockDim.x + threadIdx.x;
 unsigned int kidx = blockIdx.y * blockDim.y + threadIdx.y;
 int klo = kidx * LEVELS_PER_THREAD;
 int khi = (kidx + 1) * LEVELS_PER_THREAD;
 if(pidx >= NumEdges) {return;}
 for(int kIter = klo; kIter < khi; kIter++) {
   int offsetIdx = kIter * NumEdges + pidx;
   ::dawn::float_type weights[4] = {
       1. / d_e[offsetIdx] **2,  1. / d_e[offsetIdx] **2,
       1. / d_vv[offsetIdx] **2, 1. / d_vv[offsetIdx] **2};
   for(int nbhIter = 0; nbhIter < E_C_V_SIZE; nbhIter++) {
     int nbhIdx = kIter * NumVertices +
                       ecvTable[pidx * E_C_V_SIZE + nbhIter];
     if(nbhIdx == DEVICE_MISSING_VALUE) {continue;}
     lapl[offsetIdx] += u[nbhIdx] * weights[nbhIter];
   }
   lapl[offsetIdx] =
       lapl[offsetIdx] - 2*u[offsetIdx]/d_e[offsetIdx]
                       - 2*u[offsetIdx] / d_vv[offsetIdx]
 }
}
```

$$\begin{cases} u(x,y) := \frac{1}{4}\sqrt{\frac{105}{2\pi}} \cos 2x \cos^2 y \sin y \ , \\ v(x,y) := -\frac{1}{2}\sqrt{\frac{15}{2\pi}} \cos x \cos y \sin y \ . \end{cases}$$







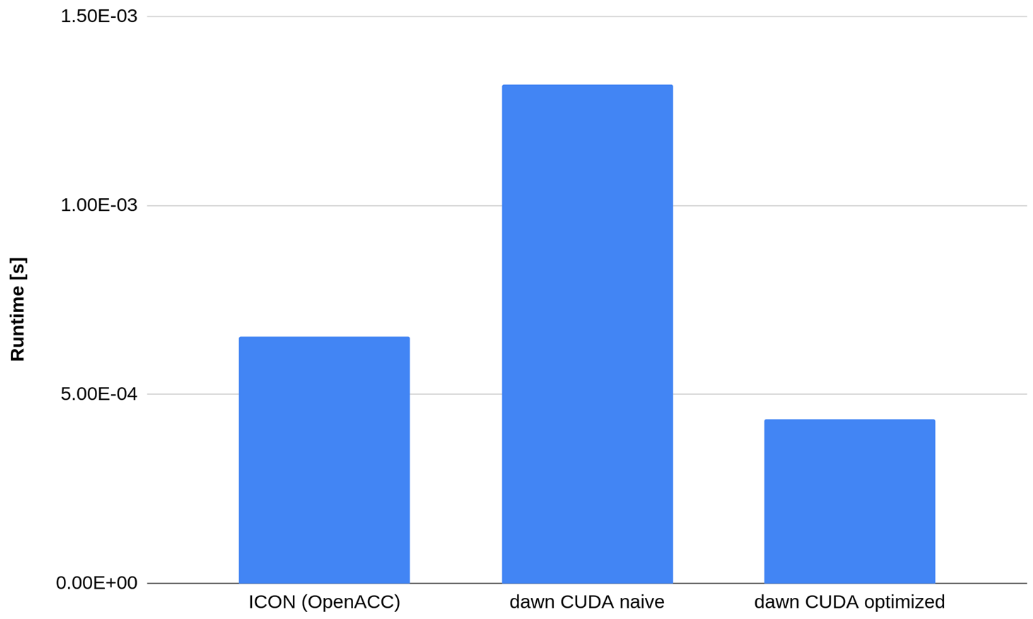
MeteoSwiss

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Grants No 675191 & 823988

Dawn for Triangular Meshes

Worked Example - Timings: Tesla P100, Single GPU, 1.3M Edges, Laplacian + Smagorinsky Diffusion



Dawn - Outlook

- Short to Midterm: Translate functional subunits of the ICON dycore using the DSL
- Long Term: Translate entire ICON dycore using the DSL
  - Efficiently target pre-exascale machines
  - Keep code readable
  - Ensure code can be further developed scientifically

- dawn is open source / open development: github

BACKUP

Example of a CUDA Optimization Strategy: Fuse Reductions

```
# dvt_tang for smagorinsky
dvt_tang = reduce(
    u_vert * dual_normal_x + v_vert * dual_normal_y,
    Edge > Cell > Vertex,
    [-1.0, 1.0, 0.0, 0.0],
)

# dvt_norm for smagorinsky
dvt_norm = reduce(
    u_vert * dual_normal_x + v_vert * dual_normal_y,
    Edge > Cell > Vertex,
    [0.0, 0.0, -1.0, 1.0],
)
```

MeteoSwiss
esiwace CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE
Grants No 675191 & 823988

```
# dvt_tang for smagorinsky
float lhs_495 = 0.0;
float weights_495[4] = {-1.0, 1.0, 0.0, 0.0};
for(int nbhIter = 0; nbhIter < E_C_V_SIZE; nbhIter++) {
  int nbhIdx = ecvTable[pidx * E_C_V_SIZE + nbhIter];
  if(nbhIdx == DEVICE_MISSING_VALUE) {continue;}
  int denseIdx = kIter * NumVertices + nbhIdx;
  int sparseIdx = kIter * NumEdges * E_C_V_SIZE + ...
  lhs_495 += weights_495[nbhIter] *
             (u_vert[denseIdx] * dual_normal_x[sparseIdx] +
              v_vert[denseIdx] * dual_normal_y[sparseIdx]);
}
dvt_tang[denseIdx] = lhs_495;
# dvt_norm for smagorinsky
float lhs_517 = 0.0;
float weights_517[4] = {0.0, 0.0, -1.0, 1.0};
for(int nbhIter = 0; nbhIter < E_C_V_SIZE; nbhIter++) {
  int nbhIdx = ecvTable[pidx * E_C_V_SIZE + nbhIter];
  if(nbhIdx == DEVICE_MISSING_VALUE) {continue;}
  int denseIdx = kIter * NumVertices + nbhIdx;
  int sparseIdx = kIter * NumEdges * E_C_V_SIZE + ...
  lhs_517 += weights_495[nbhIter] *
             (u_vert[denseIdx] * dual_normal_x[sparseIdx] +
              v_vert[denseIdx] * dual_normal_y[sparseIdx]);
}
dvt_norm[denseIdx] = lhs_517;
```

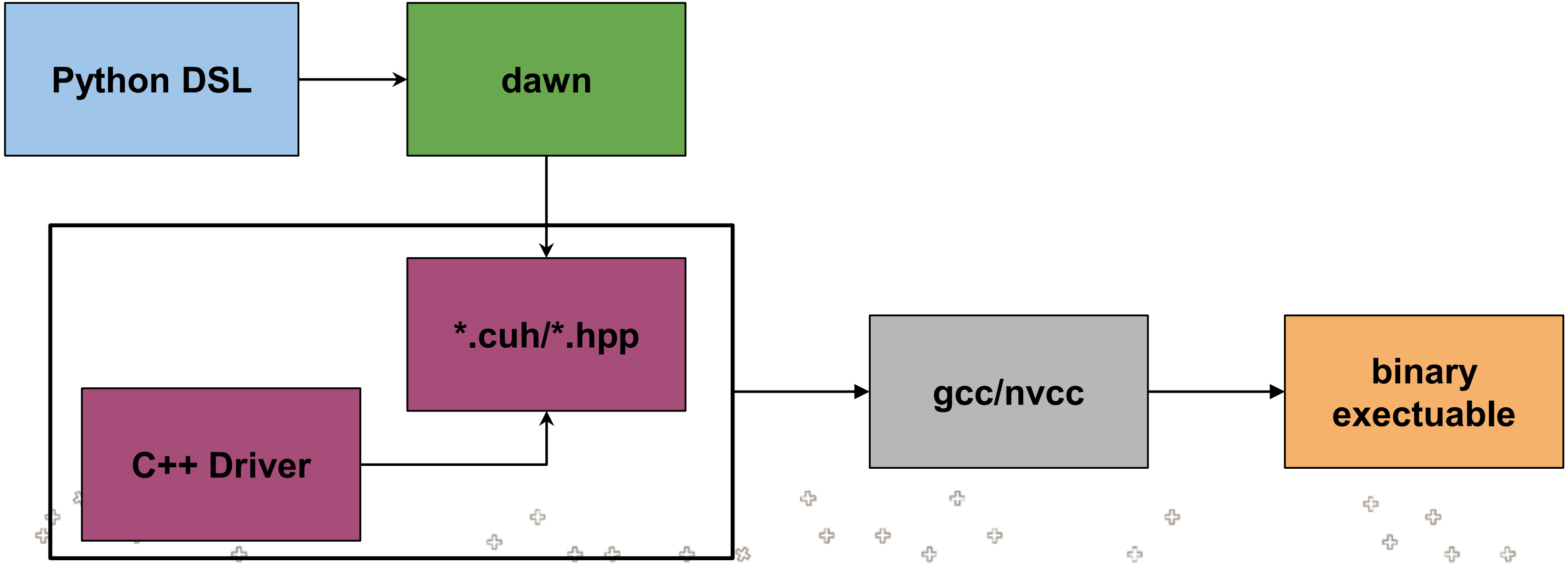
```
# dvt_tang for smagorinsky
dvt_tang = reduce(
    u_vert * dual_normal_x + v_vert * dual_normal_y,
    Edge > Cell > Vertex,
    [-1.0, 1.0, 0.0, 0.0],
)
```

```
# dvt_norm for smagorinsky
dvt_norm = reduce(
    u_vert * dual_normal_x + v_vert * dual_normal_y,
    Edge > Cell > Vertex,
    [0.0, 0.0, -1.0, 1.0],
)
```

```
# fused reductions
float lhs_495 = 0.0;
float lhs_517 = 0.0;
float weights_495[4] = {-1.0, 1.0, 0.0, 0.0};
float weights_517[4] = {0.0, 0.0, -1.0, 1.0};
for(int nbhIter = 0; nbhIter < E_C_V_SIZE; nbhIter++) {
  int nbhIdx = ecvTable[pidx * E_C_V_SIZE + nbhIter];
  if(nbhIdx == DEVICE_MISSING_VALUE) {continue;}
  int denseIdx = kIter * NumVertices + nbhIdx;
  int sparseIdx = kIter * NumEdges * E_C_V_SIZE + ...
  float rhs = (u_vert[denseIdx] * dual_normal_x[sparseIdx] +
               v_vert[denseIdx] * dual_normal_y[sparseIdx]);
  lhs_495 += weights_495[nbhIter] * rhs;
  lhs_517 += weights_517[nbhIter] * rhs;
}
dvt_tang[denseIdx] = lhs_495;
dvt_norm[denseIdx] = lhs_517;
```

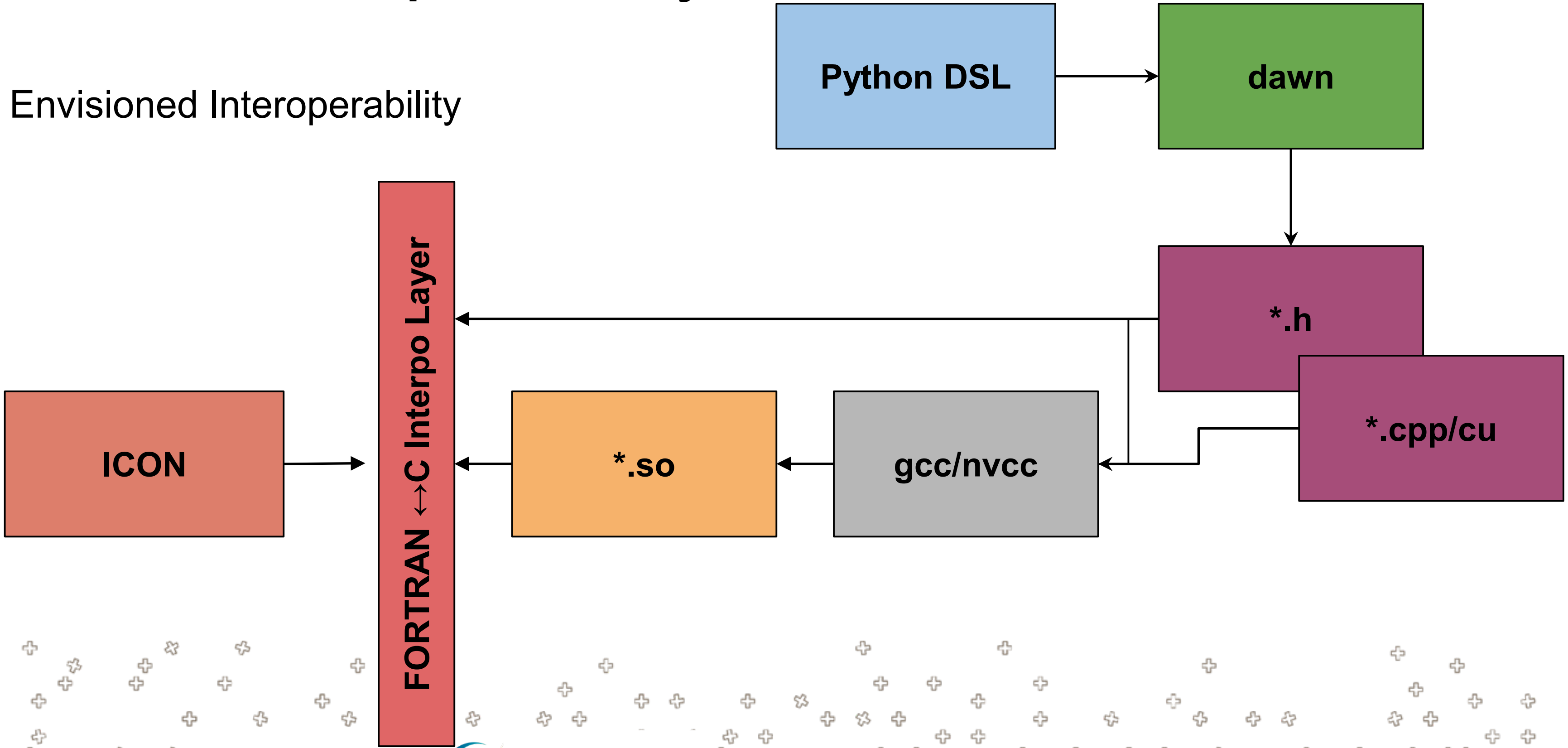
Interoperability

Current work flow:



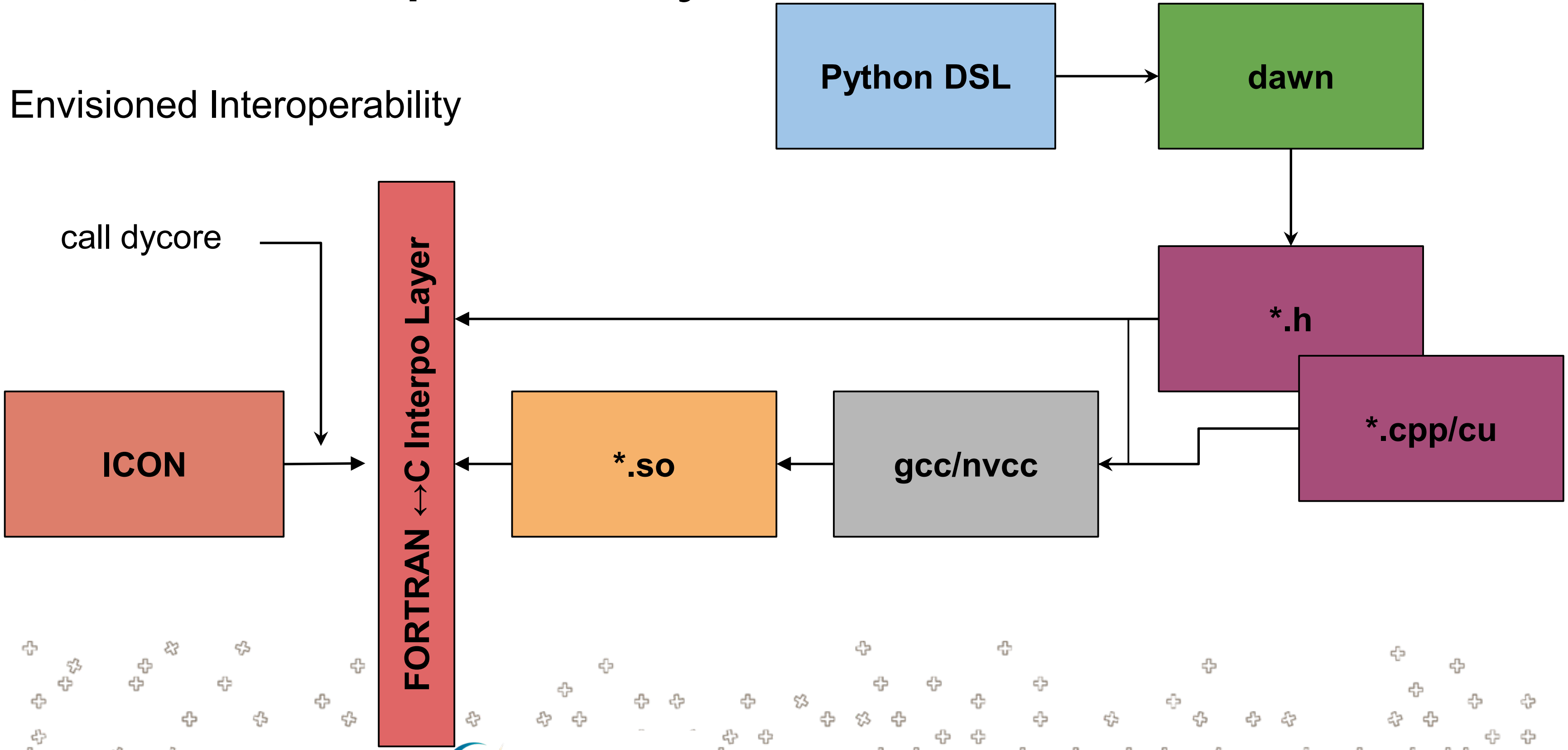
Interoperability

Envisioned Interoperability



Interoperability

Envisioned Interoperability



Interoperability

```
SUBROUTINE vn_adv_horizontal(
           p_vn, p_vort, p_delp_c,
           pt_patch, pt_int,
           p_ddt_vn, p_kin, p_vt,
           p_delp_v,
           opt_rlstart, opt_rlend  )
! <SNIP>
!===========================================
! Calculate the gradient of kinetic energy,
! and accumulate velocity tendency
!===========================================

 CALL grad_fd_norm( p_kin, pt_patch,
              z_tmp_e, opt_rlstart=4 )
```

```
SUBROUTINE grad_fd_norm(
       psi_c, ptr_patch, grad_norm_psi_e,
       opt_slev, opt_elev, opt_rlstart,
       opt_rlend )
TYPE(t_patch), TARGET, INTENT(in) :: ptr_patch
! <SNIP>
!$ACC DATA ....          &
!$ACC
!$OMP PARALLEL
!$OMP DO PRIVATE(jb,i_startidx,i_endidx,je,jk)
 DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, jb, i_startblk, i_endblk, &
                    i_startidx, i_endidx, rl_start, rl_end)
   !$ACC PARALLEL IF( i_am_accel_node .AND. acc_on )
   !$ACC LOOP GANG
   DO je = i_startidx, i_endidx
     !$ACC LOOP VECTOR
     DO jk = slev, elev
      grad_norm_psi_e(je,jk,jb) =  &
         &  ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) - &
         &    psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )  &
         &  * ptr_patch%edges%inv_dual_edge_length(je,jb)
     ENDDO
   END DO
   !$ACC END PARALLEL
 END DO
! <SNIP>
```

MeteoSwiss

esiwace CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Grants No 675191 & 823988

Interoperability

```
SUBROUTINE vn_adv_horizontal(
           p_vn, p_vort, p_delp_c,
           pt_patch, pt_int,
           p_ddt_vn, p_kin, p_vt,
           p_delp_v,
           opt_rlstart, opt_rlend  )
! <SNIP>
!===========================================
! Calculate the gradient of kinetic energy,
! and accumulate velocity tendency
!===========================================

 CALL grad_fd_norm( p_kin, pt_patch,
              z_tmp_e, opt_rlstart=4 )
```

Stencil is rewritten using our DSL

```
grad_norm_psi_e =
      reduce( psi_c,
              C > E,
              [1/lhat, -1/lhat] )
```

Interoperability

```
SUBROUTINE vn_adv_horizontal(
           p_vn, p_vort, p_delp_c,
           pt_patch, pt_int,
           p_ddt_vn, p_kin, p_vt,
           p_delp_v,
           opt_rlstart, opt_rlend  )
! <SNIP>
!===========================================
! Calculate the gradient of kinetic energy,
! and accumulate velocity tendency
!===========================================

 CALL grad_fd_norm( p_kin, pt_patch,
              z_tmp_e, opt_rlstart=4 )
```

Code is generated using dawn

Header

```
void grad_fd_norm_dawn(
  dawn::cell_field_t<double>& psi_c,
  dawn::edge_field_t<double>& grad_norm_psi_e,
  dawn::edge_field_t<double>& inf_edge_length);
```

Implementation

```
void grad_fd_norm_dawn(
  dawn::cell_field_t<double>& psi_c,
  dawn::edge_field_t<double>& grad_norm_psi_e,
  dawn::edge_field_t<double>& inf_edge_length)
 for(int k = 0; k < k_size; ++k) {
   for(auto const& loc : getEdges(m_mesh)) {
     for(auto inner_loc :
       grad_norm_psi_e(loc, k) = reduce(...
```

Interoperability

```
SUBROUTINE vn_adv_horizontal(
           p_vn, p_vort, p_delp_c,
           pt_patch, pt_int,
           p_ddt_vn, p_kin, p_vt,
           p_delp_v,
           opt_rlstart, opt_rlend  )
! <SNIP>
!===========================================
! Calculate the gradient of kinetic energy,
! and accumulate velocity tendency
!===========================================

 CALL grad_fd_norm( p_kin, pt_patch,
               z_tmp_e, opt_rlstart=4 )
```

Compile into library

Header

```
void grad_fd_norm_dawn(
  dawn::cell_field_t<double>& psi_c,
  dawn::edge_field_t<double>& grad_norm_psi_e,
  dawn::edge_field_t<double>& inf_edge_length);
```

*.so

MeteoSwiss
esiwace CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE
Grants No 675191 & 823988

Interoperability

```
SUBROUTINE vn_adv_horizontal(
           p_vn, p_vort, p_delp_c,
           pt_patch, pt_int,
           p_ddt_vn, p_kin, p_vt,
           p_delp_v,
           opt_rlstart, opt_rlend  )
! <SNIP>
!==========================================
! Calculate the gradient of kinetic energy,
! and accumulate velocity tendency
!==========================================

 CALL grad_fd_norm_dsl( p_kin, pt_patch,
              z_tmp_e, opt_rlstart=4 )
```

Call FORTAN Interop layer

```
MODULE grad_fd_norm_dsl( p_kin,...
! Pack / transform fields for C/C++
!
CALL grad_fd_norm_dawn

!
! Unpack Fields
```

```
void
grad_fd_norm_daw
n(...
```

*.so

MeteoSwiss
Operation Center 1
CH-8058 Zurich-Airport
T +41 58 460 91 11
www.meteoswiss.ch

MeteoSvizzera
Via ai Monti 146
CH-6605 Locarno-Monti
T +41 58 460 92 22
www.meteosvizzera.ch

MétéoSuisse
7bis, av. de la Paix
CH-1211 Genève 2
T +41 58 460 98 88
www.meteosuisse.ch

MétéoSuisse
Chemin de l'Aérologie
CH-1530 Payerne
T +41 58 460 94 44
www.meteosuisse.ch