Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

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

Introduction to dawn - Motivation, History, Future

Dawn - Motivation

Model software development starts at numerical discretization of continuous quantities:

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

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
```

Motivation

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

```
#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?

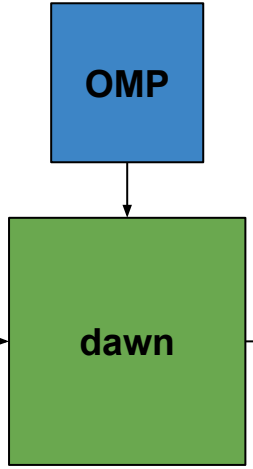
```
#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( 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

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

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




```
!$OMP PARALLEL
!$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
```

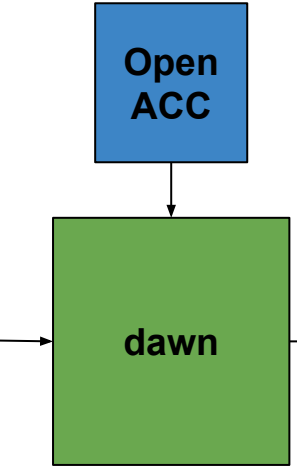
Motivation

No FORTRAN Backend Exists, only for illustration purposes

Idea of dawn / DSLs in general

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

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



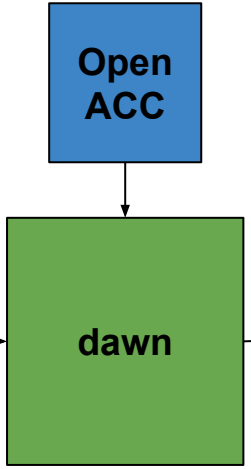

```
!$ACC PARALLEL &
!$ACC PRESENT(ptr_patch, iidx, iblk, pci_c, grad_...)
!$ACC LOOP GANG
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 DO jk = slev, elev
   DO je = i_startidx, i_endidx
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
       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

Motivation

Idea of dawn / DSLs in general

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

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




```
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)
 }
}
```

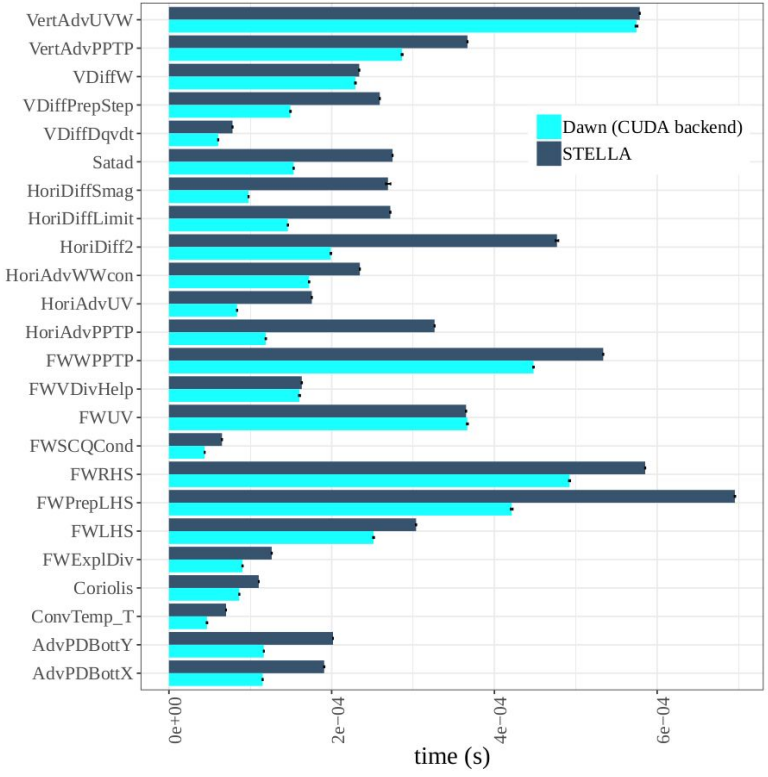
Dawn - History

- 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 - History

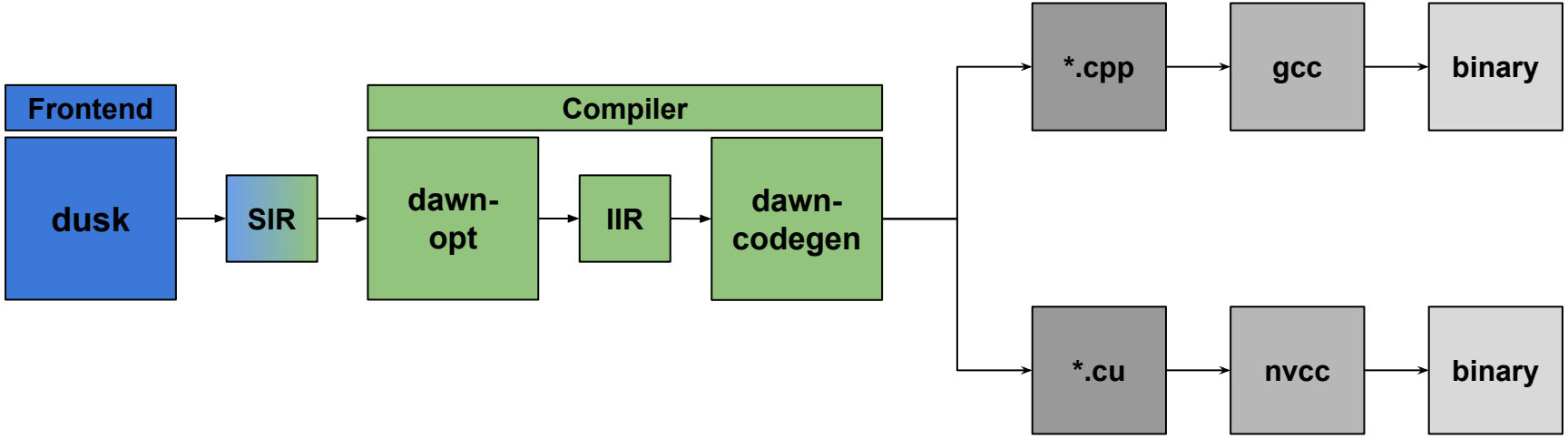
- Dawn was used to successfully translate the complete COSMO dycore
  - advection schemes
  - diffusion
  - tridiagonal solver
  - ...
  -
- Outperforms previous efforts of translating the COSMO dycore using DSLs, at a fraction of the lines of code

Dawn - History



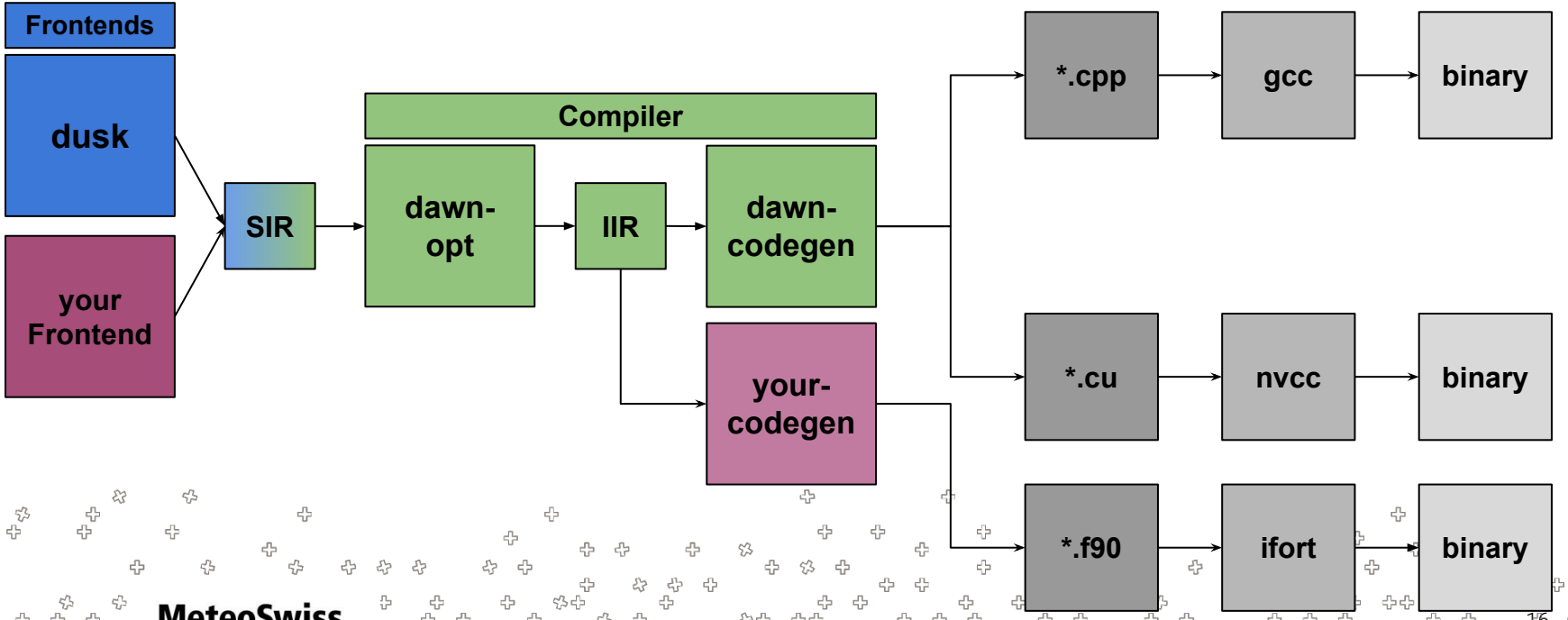
Dawn Design Overview

Dawn - A closer look



Dawn Design Overview

Dawn - A closer look



SIR Snippet


```
"stencils": [
 {
  "ast": {
   "root": {
    "blockStmt": {
     "statements": [
      {
       "verticalRegionDeclStmt": {
        "verticalRegion": {
         "ast": {
          "root": {
           "blockStmt": {
            "statements": [
             {
              "exprStmt": {
               "expr": {
                "assignmentExpr": {
                 "left": {
                  "fieldAccessExpr": {
                   "name": "rot vec",
                   "unstructuredOffset": {}
                  }
                 },
                 "op": "=",
                 "right": {
                  "reductionOverNeighborExpr": {
                   "op": "+",
                   "rhs": {
                    "binaryOperator": {
                     "left": {
                      "fieldAccessExpr": {
                       "name": "vec",
                       "unstructuredOffset": {
                        "hasOffset": true
...
```


- **A**bstract **S**yntax **T**ree representation close to DSL
- Interface for frontends
- Format well suited for translation
- Not meant to be human-readable

IIR Snippet


```
"stencils": [
  {
   "multiStages": [
    {
     "stages": [
      {
       "doMethods": [
        {
         "ast": {
          "block stmt": {
           "statements": [
            {
             "expr stmt": {
              "expr": {
               "assignment_expr": {
                "left": {
                 "field access expr": {
                  "name": "rot vec",
                  "vertical shift": 0,
                  "vertical indirection": "",
                  "zero offset": {},
                  "argument_map": [
                   -1,
                   -1,
                   -1
                  ],
                  "argument_offset": [
                   0,
                   0,
                   0
                  ],
                  "negate_offset": false,
 ...
```


- Also tree-like representation, with ASTs nested inside
- Contains many more details, collected during analyses
- Format well suited for transformations (e.g. through visitors)
- Compiler passes transform valid IIR into valid IIR

IIR Snippet

Hierarchical compartmentalization of ASTs to achieve valid parallelization

"Decorated" AST

Dawn - State of Development / Future

Current State

- Can translate most computations in the ICON dycore
  - roughly 90% feature complete
- Quite a few isolated computations from the ICON dycore translated and end-to-end tested

Currently Under Development

- Global values
- Horizontal subdomain selection

Dawn - State of Development / Future

Future:

- Close gaps to ICON
  - Mostly horizontal indirections for Semi-Lagrangian in the horizontal
- Start to translate a whole ICON module
  - e.g. diffusion

Q&A

Questions?