Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

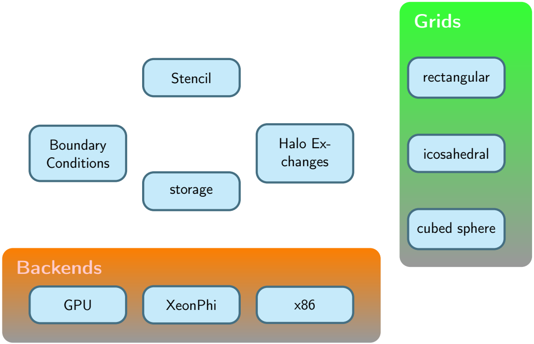
Gridtools Project

Carlos Osuna, Meteoswiss (ETH)
carlos.osuna@meteoswiss.ch

COSMO Developer Workshop, January 2016

What is Gridtools

- Set of grid tools, including DSL for stencil codes, for solving PDEs on (mostly) structured (but not only rectangular) grids

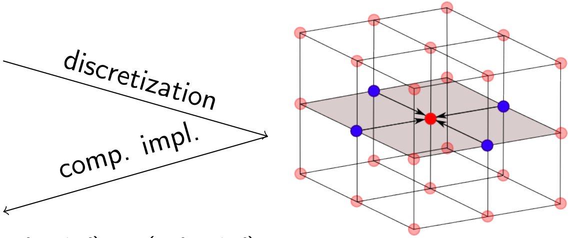


- Provides a separation of concerns: Separates model and algorithm from hardware specific implementation and optimization
- Supports multiple backends

Separation Of Concerns

User description of mathematical model

$$\frac{\partial U}{\partial t} = -\alpha \nabla^2 (\nabla^2 U)$$



$$\begin{aligned}
lap(i,j) &= 4u(t,i,j) - u(t,i+1,j) - u(t,i-1,j) - \\
&\quad u(t,i,j+1) - u(t,i,j-1) \\
u(t+1,i,j) &= 4lap(t,i,j) - lap(t,i+1,j) - lap(t,i-1,j) - \\
&\quad lap(t,i,j+1) - lap(t,i,j-1)
\end{aligned}$$

Separation Of Concerns

Generated Kernel for GPU

```
const int i = threadIdx.x;
const int j = threadIdx.y;
int i_h = 0;
int j_h = 0;

if(j < 2)
{
i_h = i;
j_h = (j==0 ? -1 : blockDim.y);
}
else if(j < 4 && i <= blockDim.y)
{
i_h = (j==2 ? -1 : blockDim.x);
j_h = i;
}

for(int k=0; k < kdim; ++k)
{
lap(i,j) = - 4.0 * phi(i,j,k)
+ phi(i+1,j,k) + phi(i-1,j,k)
+ phi(i,j+1,k) + phi(i,j-1,k);
```

```
if(i_h != 0 || j_h != 0)
lap(i_h, j_h) =
- 4.0 * phi(i_h,j_h,k)
+ phi(i_h+1,j_h,k) + phi(i_h-1,j_h,k)
+ phi(i_h,j_h+1,k) + phi(i_h,j_h-1,k);
__syncthreads();
flx(i,j,k) = lap(i+1,j,k) - lap(i,j,k);
fly(i,j,k) = lap(i,j+1,k) - lap(i,j,k);
if(i_h < 0)
flx(i_h,j_h,k) = lap(i_h+1,j_h,k) -
lap(i_h,j_h,k);
if(j_h < 0)
fly(i,j_h,k) = lap(i,j_h+1,k) -
lap(i,j_h,k);
__syncthreads();
result(i,j) = phi(i,j,k) - alpha(i,j,k)*(
flx(i,j,k) - flx(i-1,j,k) +
fly(i,j,k) - fly(i,j-1,k));
}
```

People

C++

Paolo Crosetto (C2SM)

Mauro Bianco (CSCS)

Carlos Osuna (C2SM)

Python

Lucas Benedicic (CSCS)

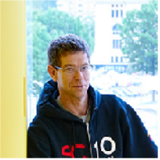
Lead

Oliver Fuhrer (Meteoswiss)

Thomas Schulthess (CSCS)

Consulting

Will Sawyer (CSCS)

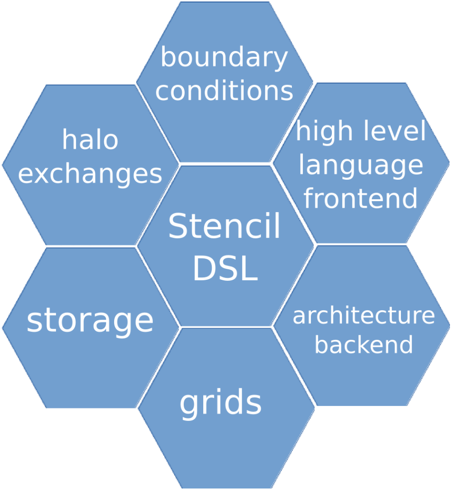


Why GridTools beyond STELLA?

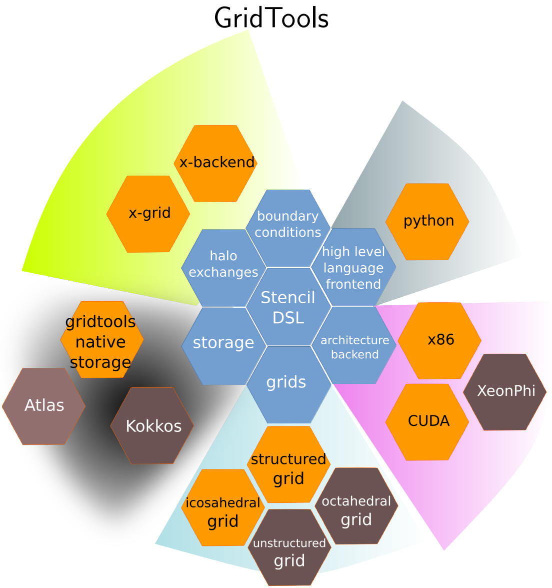
- Generalizes STELLA concepts (beyond COSMO and rectangular grids)
    1. Generic Boundary Conditions
    2. Handle staggered grids
    3. Multidimensional storage (time, vector fields, ...)
    4. C++11
- Improve and simplify syntax
- Development environment in python
- Set of tools that support multiple grids
- Support for multiple types of stencil codes -> Shared maintenance

Library Design

GridTools



Library Design



Improved Syntax

- Describe stencil operators

```
template<typename TEnv>
struct Laplace
{
  STENCIL_STAGE(TEnv)
  STAGE_PARAMETER(u)
  STAGE_PARAMETER(lap)

  static void Do(Context ctx, FullDomain)
  {
    ctx[lap::Center()] = -4*ctx[Center()] +
    (ctx[u::At(iminus1)] + ctx[u::At(iplus1)]) +
    (ctx[u::At(jminus1)] + ctx[u::At(jplus1)]);
  }
};
```

```
struct Laplace
{
  typedef in_accessor<0, range<-1,1-1,1> > u;
  typedef out_accessor<1> lap;

  template<typename Evaluation>
  static void Do(Evaluation const& eval,
                 full_domain)
  {
    eval(lap()) = eval(-4*u() +
      u(i+1) + u(i-1) +
      u(j+1) + u(j-1));
  }
};
```

Multidimensional storages for time integration

```
template<typename TEnv>
struct Diff
{
  typedef in_accessor<0> u_in;
  typedef in_accessor<1, range<-1,1-1,1> > lap;
  typedef out_accessor<2> u_out;

  template<typename Evaluation>
  static void Do(Evaluation const& eval,
      full_domain)
  {
    eval(u_out()) = -4*eval(u_in()) +
      eval(lap(i+1)) + eval(lap(i-1)) +
      eval(lap(j+1)) + eval(lap(j-1));
  }
};
```

Double buffer fields

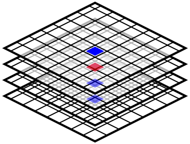
```
template<typename TEnv>
struct Diff
{
  typedef inout_accessor<0> u;
  typedef in_accessor<1, range<-1,1-1,1> > lap;
  typedef dimension<4> time;

  template<typename Evaluation>
  static void Do(Evaluation const& eval,
      full_domain)
  {
    eval(u(time(1))) = -4*eval(u()) +
      eval(lap(i+1)) + eval(lap(i-1)) +
      eval(lap(j+1)) + eval(lap(j-1));
  }
};
```

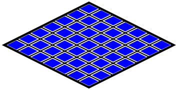
Time in multidimensional fields

Symplified Syntax

- Caches



KCaches



IJCaches

STELLA

```
KCache<v_stage, cFill, KWindow<-2,1>, KRange<FullDomain,0,0> >()
IJCache<lap, cLocal, IJWindow<0,0,0,0>, KRange<FullDomain,0,0> >()
```

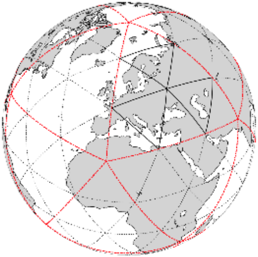
Gridtools

```
cache<K, fill>(v_stage())
cache<IJ, local>(lap())
```

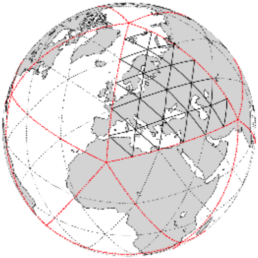
Gridtools on Other Grids

The Horizontal Grid

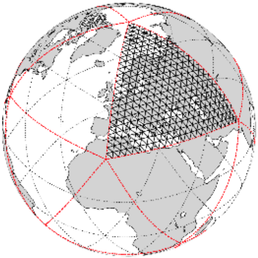
Spherical geodesic grids derived from the icosahedron



R3B00



R3B01

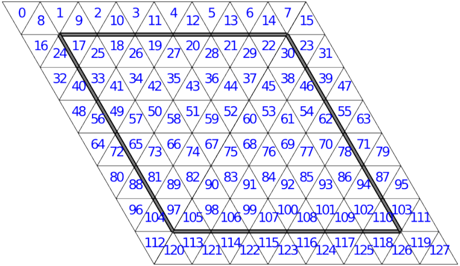


R3B03 optimized

Gridtools on Other Grids

Indexing Structured Grids

Global models typically explicitly use indirect indexing.
GridTools hides data accesses from the user and can use more efficient backends using structured grids and direct indexing



Implications of direct addressing for $\triangle^2$

	Time (s)
Direct addressing	0.0025
Indirect addressing	0.0037

ICON Example

```
!$OMP PARALLEL DO
DO jb = i_startblk, i_endblk
#ifdef __LOOP_EXCHANGE
  DO jc = i_startidx, i_endidx
    DO jc = slev, elev
#else
  DO jk = slev, elev
    DO jc = i_startidx, i_endidx
#endif
        div_vec_c(jc,jk,jb) = vec_e(iid(ic,jb,1),jk,iblk(ic,jb,1)) * ptr_int%geofac_div(jc,1,jb) + &
          vec_e(iid(ic,jb,2),jk,iblk(ic,jb,2)) * ptr_int%geofac_div(jc,2,jb) + &
          vec_e(iid(ic,jb,3),jk,iblk(ic,jb,3)) * ptr_int%geofac_div(jc,3,jb)
    ENDDO
  ENDDO
ENDDO
```

ICON

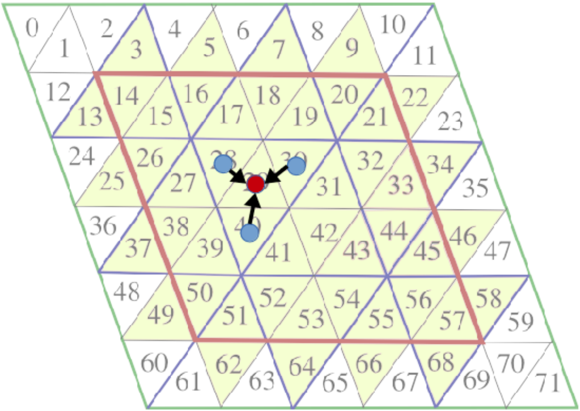
Gridtools

```
struct Div {
  typedef in_accessor<0, grid::edges, radius<1> > vec;
  typedef out_accessor<1, grid::cells> div_vec;
  typedef boost::mpl::vector<vec, div_vec> arg_list;

  template<typename Evaluation>
  static void Do(Evaluation const& eval, full_domain)
  {
     eval(div_vec())= eval(on_edges<vec>());
  }
};
```

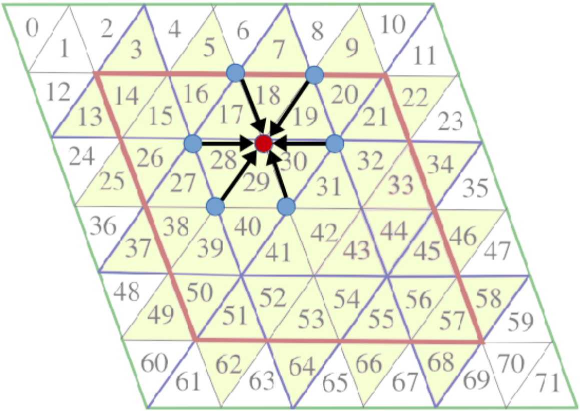
Grid Operations

on_cells<u>()



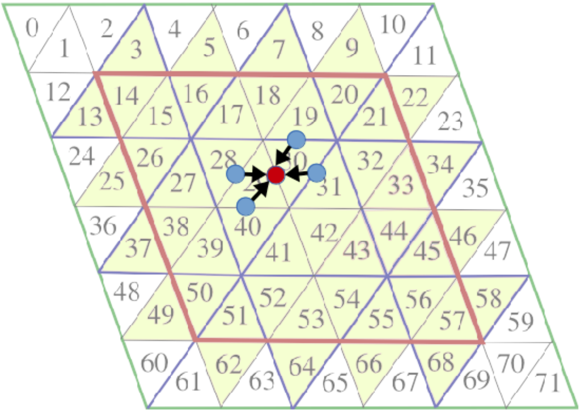
Grid Operations

on_vertexes<u>()



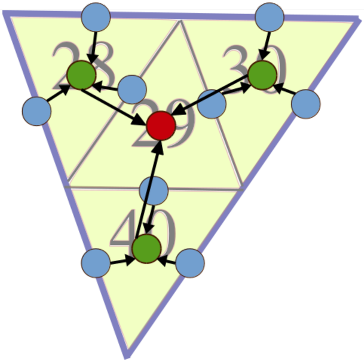
Grid Operations

on_edges<u>()



Grid Operations

on_cells<on_edges<u> >()



Development Environment in Python

GridTools

- generalizes STELLA
- improves/simplifies the DSL syntax

But

- does not reduce amount of code in the dynamical core
- not interactive/prototyping language (C++)

Development Environment in Python

- Python as a prototyping language for stencils
- Just in time compilation in C++
- IPython: interatice visualization & analysis of operators
- Final C++ gridtools can be generated from python stencils

```
In [2]: from tests.test_stencils import Laplace
from gridtools import MultiStageStencil

class LaplaceStencil(MultiStageStencil):
""" A Laplace operator.– """
def kernel(self, out_data, in_data):
# iterate over the fields's interior points
for p in self.get_interior_points(out_data,
halo=(−1,1,−1,1), k_direction="forward"):
out_data[p] = −4*in_data[p] +
in_data[p +(−1,0,0)] + in_data[p +(1,0,0)] +
in_data[p +(0,−1,0)] + in_data[p +(0,1,0)]
```

Conclusions

- Gridtools is the successor of STELLA. Designed as a set of modular tools for PDEs
- For rectangular grids, it generalizes STELLA concepts and simplifies the syntax
- It inherits the performance of STELLA for GPU & CPU backends
- Support for multiple grids, sharing the same DSL syntax.
- Provides a development environment in python helpful to prototype dynamical cores.