JEAN-MICHEL GORIUS, TOBIAS WICKY, TOBIAS GROSSER, AND TOBIAS GYSI

A Compiler Intermediate Representation for Stencils

"Climate change is now affecting every country on every continent. It is disrupting national economies and affecting lives, costing people, communities and countries dearly today and even more tomorrow. Weather patterns are changing, sea levels are rising, weather events are becoming more extreme and greenhouse gas emissions are now at their highest levels in history." - **United Nations, Sustainable Development Goals**

Open Climate Compiler Initiative

VULCAN

MeteoSwiss

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

SPCL

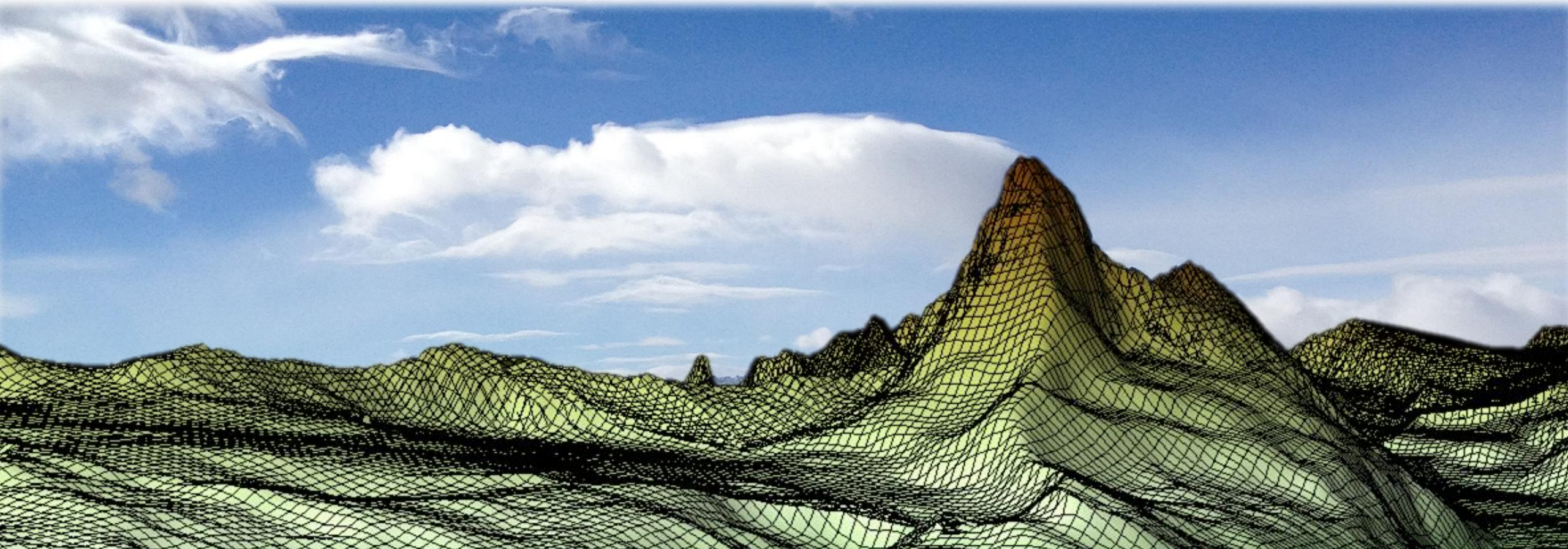
ETH zürich

COSMO Atmospheric Model

- Regional atmospheric model used by 7 national weather services
- Implements many different stencil programs

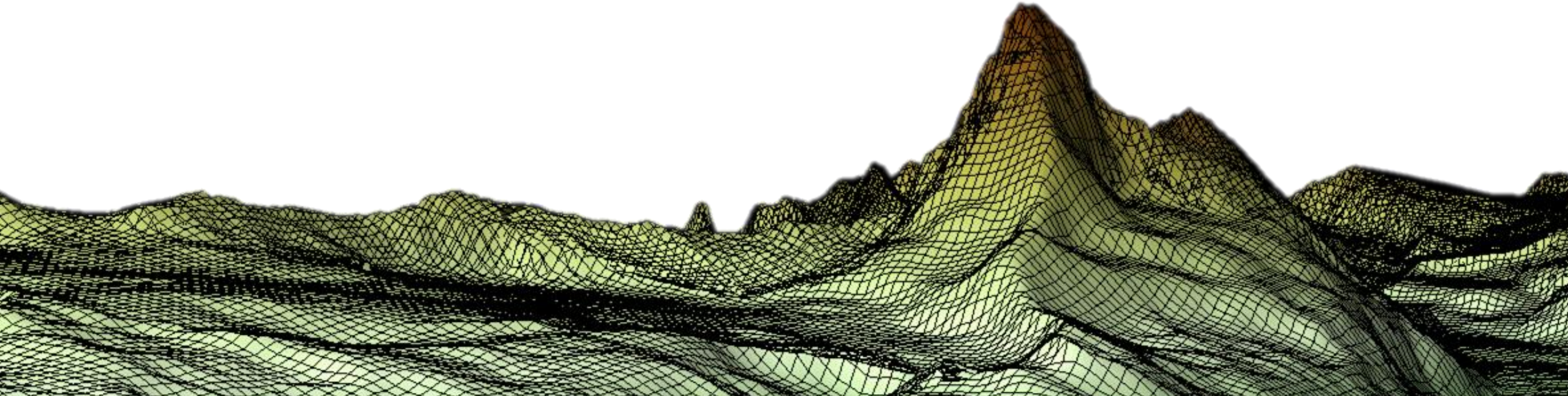
Resolution (35m)

What resolution is needed to predict if there is snow out of the banner cloud at Matterhorn?



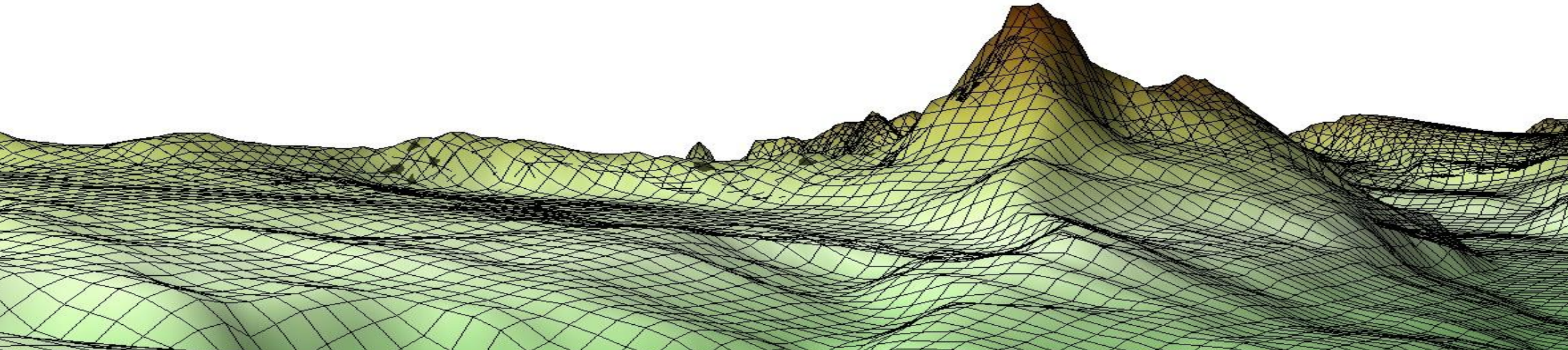
Resolution (35m)

What resolution is needed to predict if there is snow out of the banner cloud at Matterhorn?



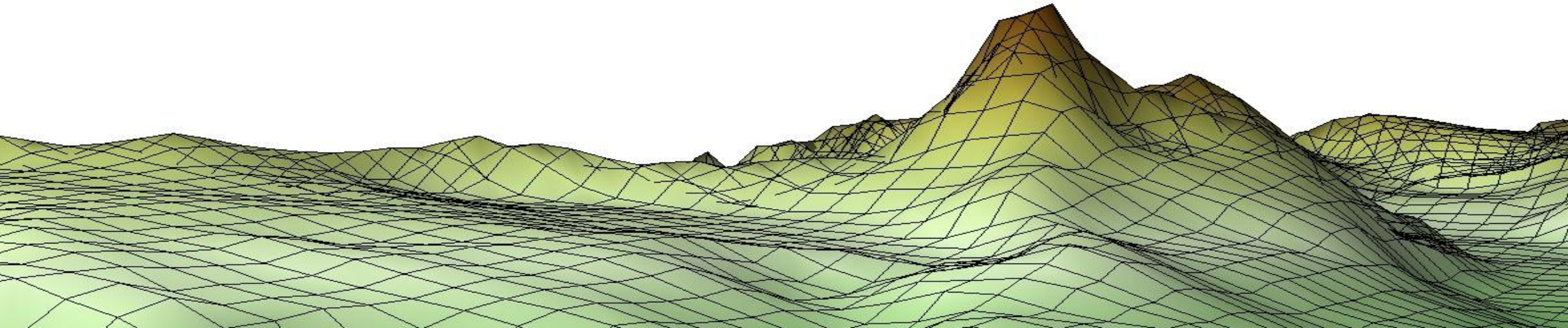
Resolution (70m)

What resolution is needed to predict if there is snow out of the banner cloud at Matterhorn?



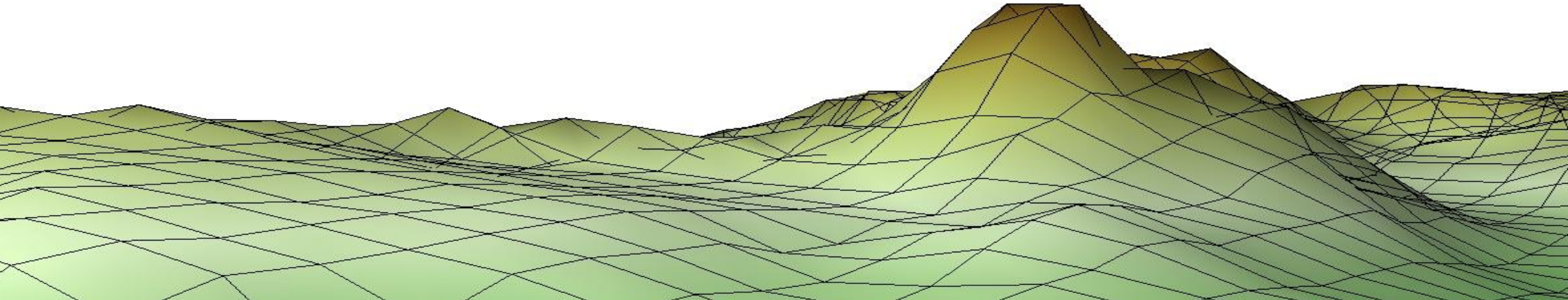
Resolution (140m)

What resolution is needed to predict if there is snow out of the banner cloud at Matterhorn?



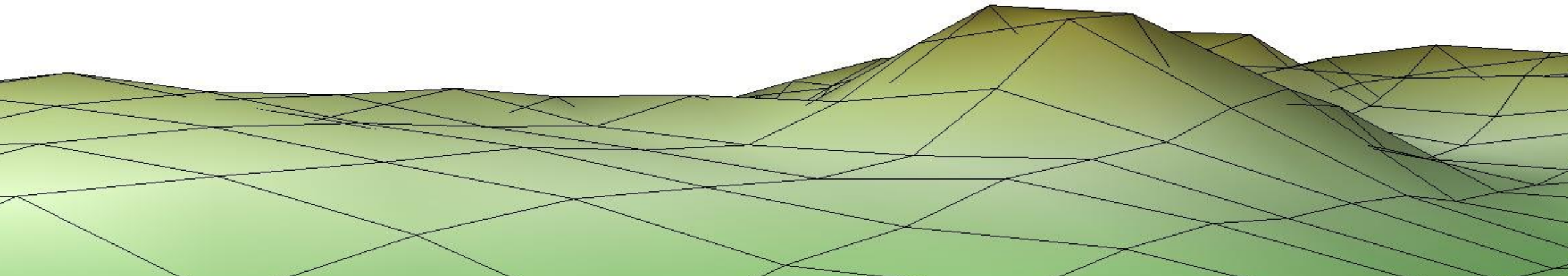
Resolution (280m)

What resolution is needed to predict if there is snow out of the banner cloud at Matterhorn?



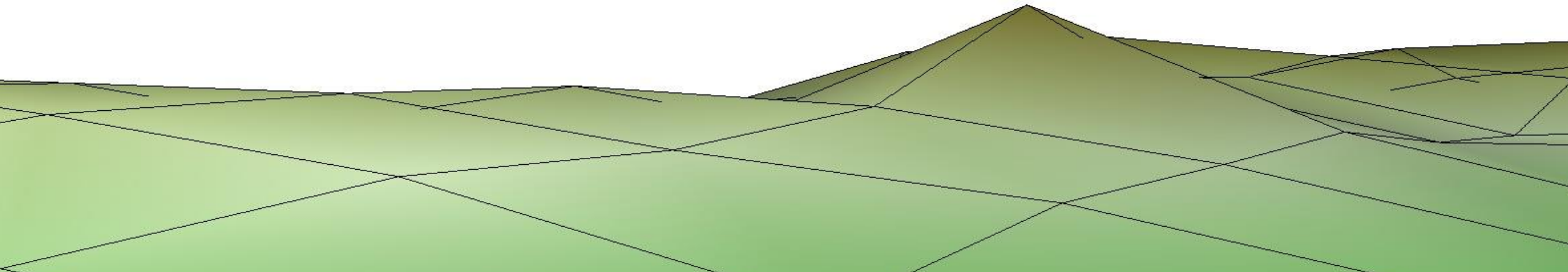
Resolution (560m)

What resolution is needed to predict if there is snow out of the banner cloud at Matterhorn?



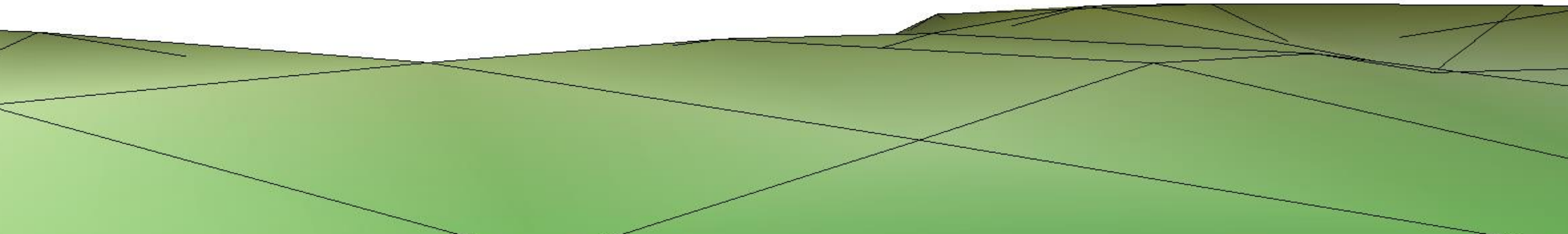
Resolution (1.1km – Weather Forecast Today)

What resolution is needed to predict if there is snow out of the banner cloud at Matterhorn?



Resolution (2.2km – Weather Forecast 2015)

What resolution is needed to predict if there is snow out of the banner cloud at Matterhorn?



Achieving High-Performance, Portability, and Productivity

COSMO

Stella/GridTools

1st GPU model running in production

Dawn (GTClang)

1998 — 2010 — **2016** — 2019

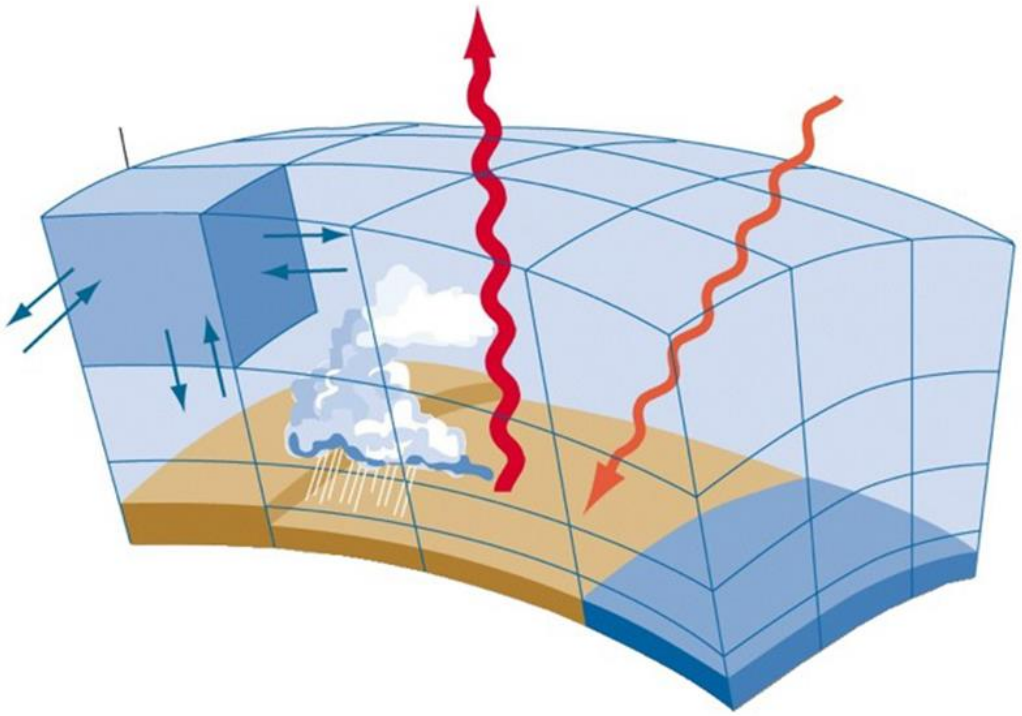
- Fortran code
- optimized for vector machines

- DSL embedded in C++
- GPU and CPU support
- performance & portability

- domain-specific compiler
- front end language agnostic
- powerful analysis and optimization passes
- productivity

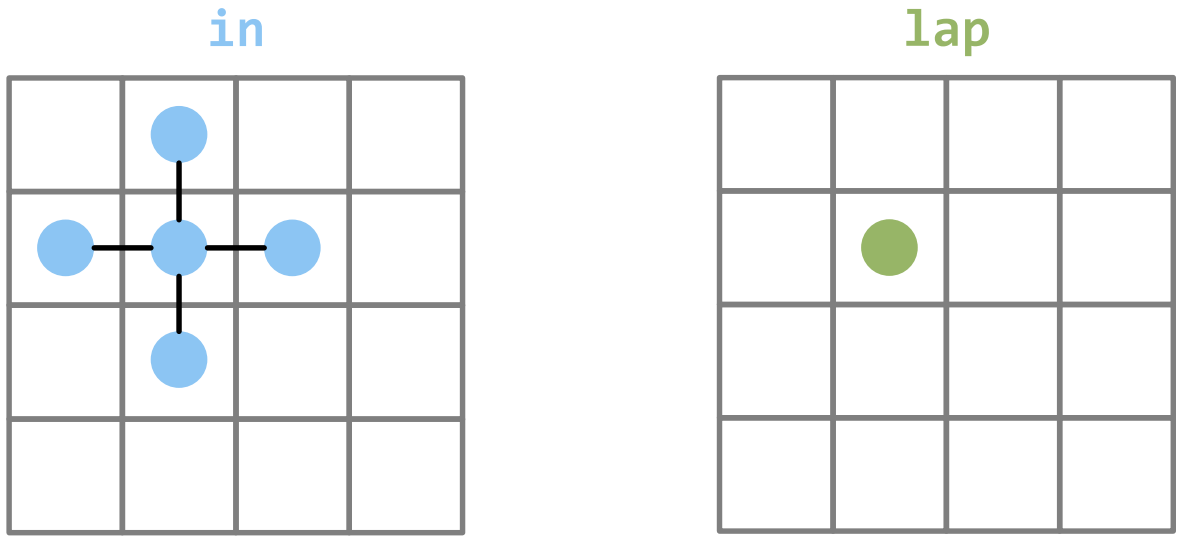
Domain-Science vs Computer-Science

- solve PDE
- finite differences
- structured grid

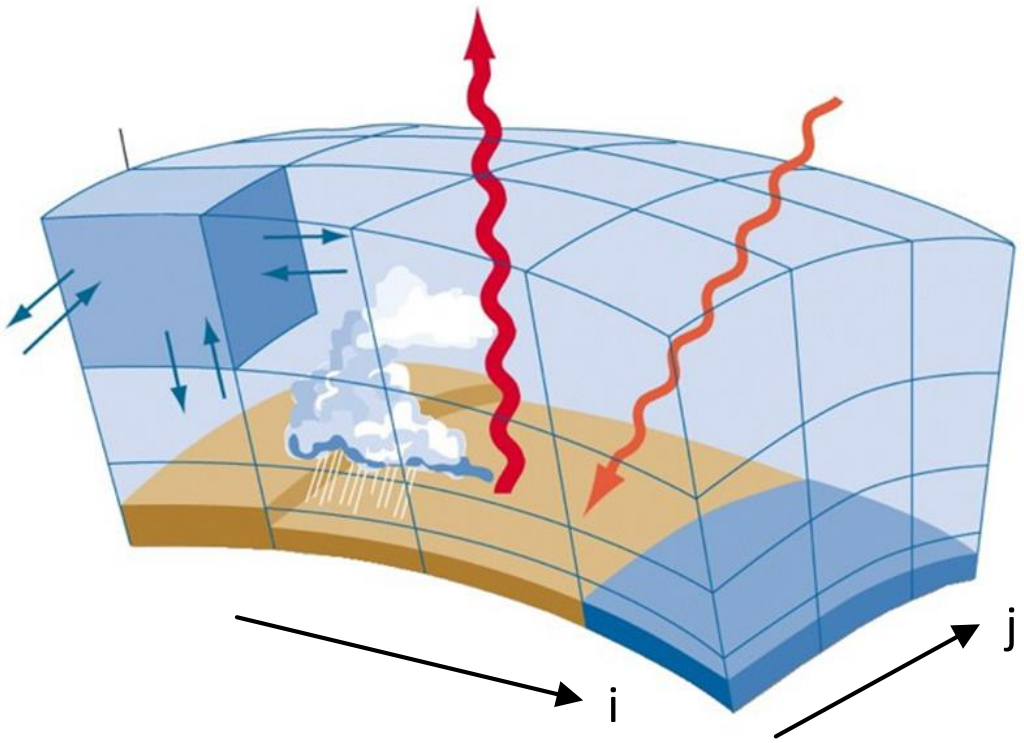


- element-wise computation
- fixed neighborhood

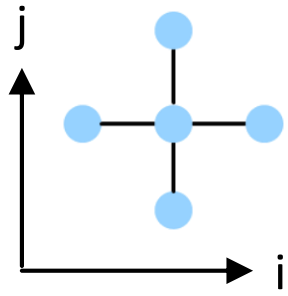
```
lap(i,j) = -4.0 * in(i,j) +
  in(i-1,j) + in(i+1,j) +
  in(i,j-1) + in(i,j+1)
```



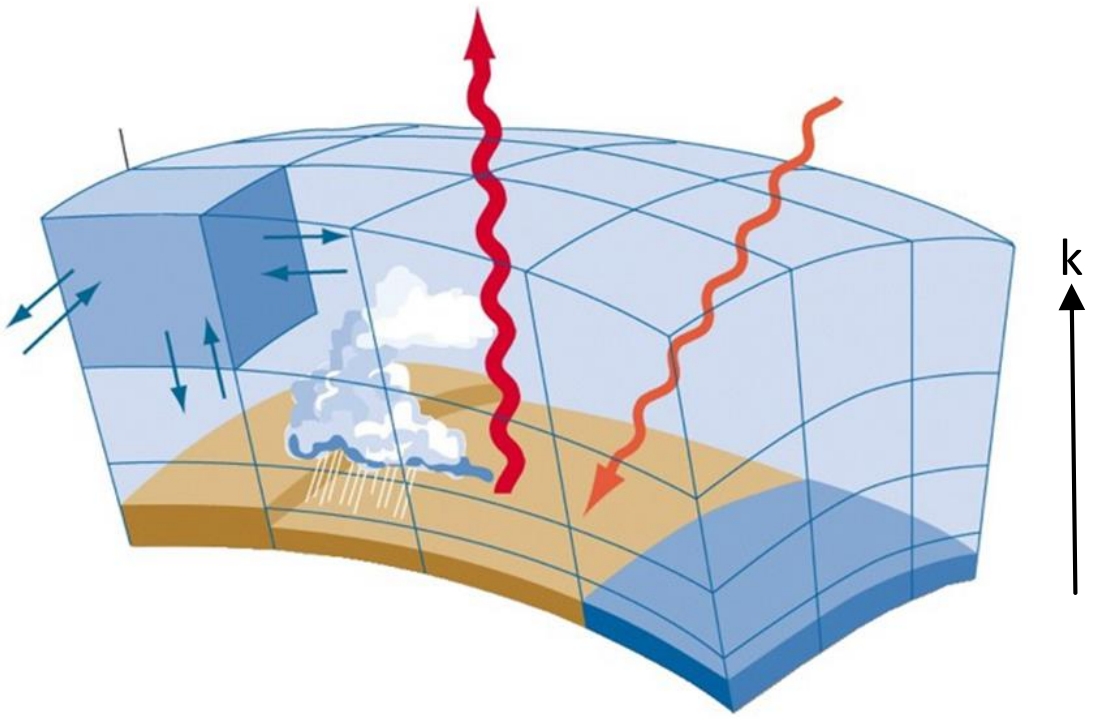
Algorithmic Motifs – Finite Differences



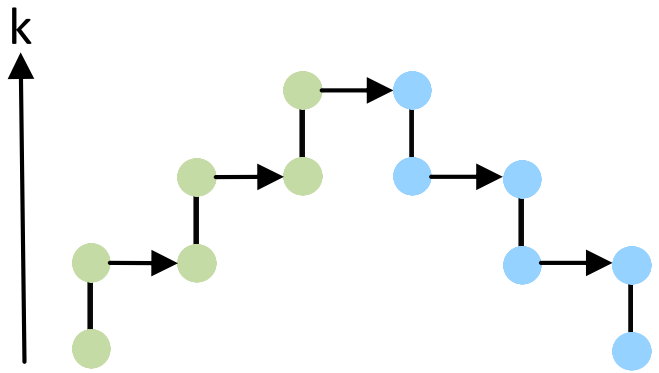
- stencils (no loop carried dependencies)
- mostly horizontal dependencies



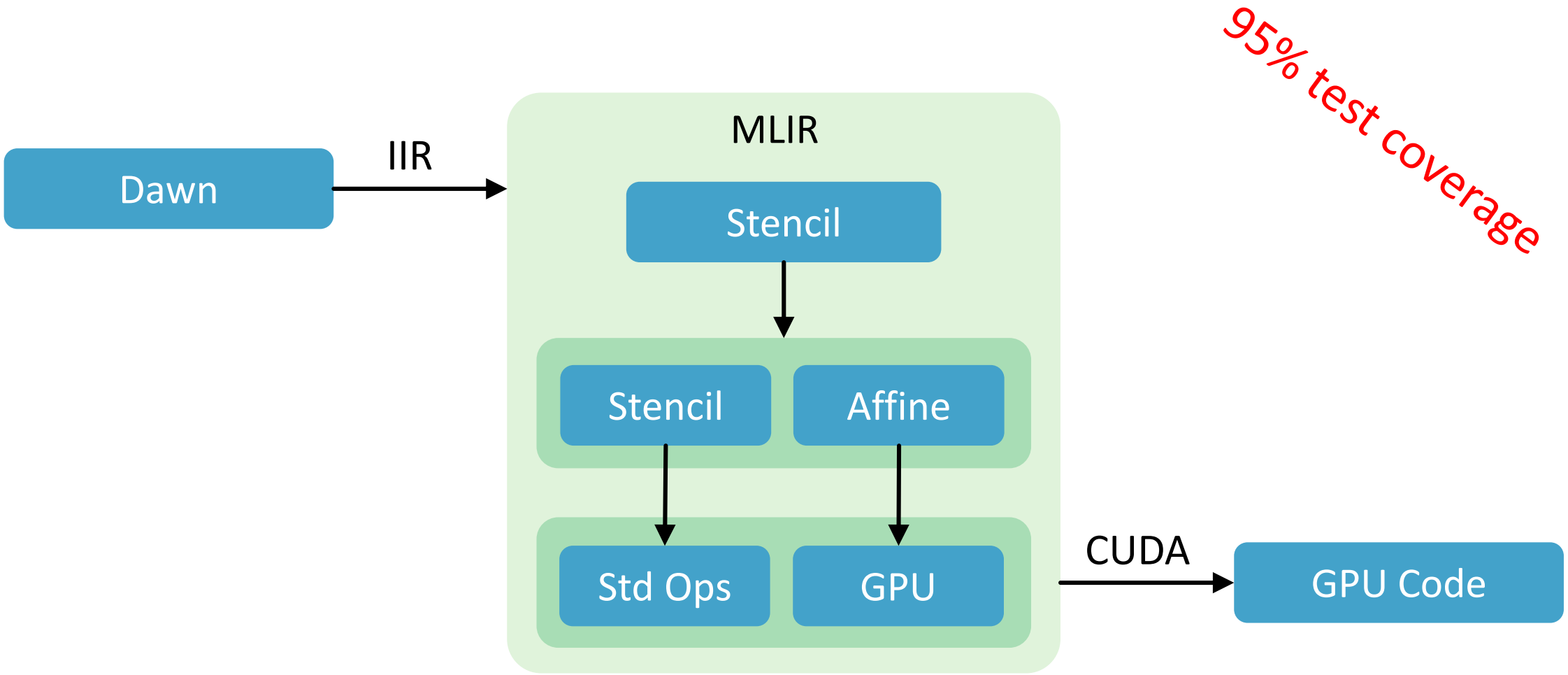
Algorithmic Motifs – Tridiagonal Systems



- vertical dependencies
- loop carried dependencies



Our Current Toolchain



Low-level Dialect (IIR)

```
stencil.iir {
  stencil.stencil(%arg0: !stencil<"field:f64">, %arg1: !stencil<"field:f64">) {
    stencil.multi_stage "Parallel" {
      stencil.stage {
        stencil.do_method [0, 0, 60, 0] {
          %0 = stencil.field_access %arg1 [0, 0, 0] : !stencil<"ptr:f64">
          %1 = stencil.field_access %arg0 [0, 0, 0] : !stencil<"ptr:f64">
          %2 = stencil.get_value %0 : f64
          %3 = stencil.get_value %1 : f64
          %4 = addf %2, %3 : f64
          %cst = constant 4.000000e+00 : f64
          %5 = mulf %4, %cst
          stencil.write %0, %5 : f64
        }
      }
    }
  }
}
```

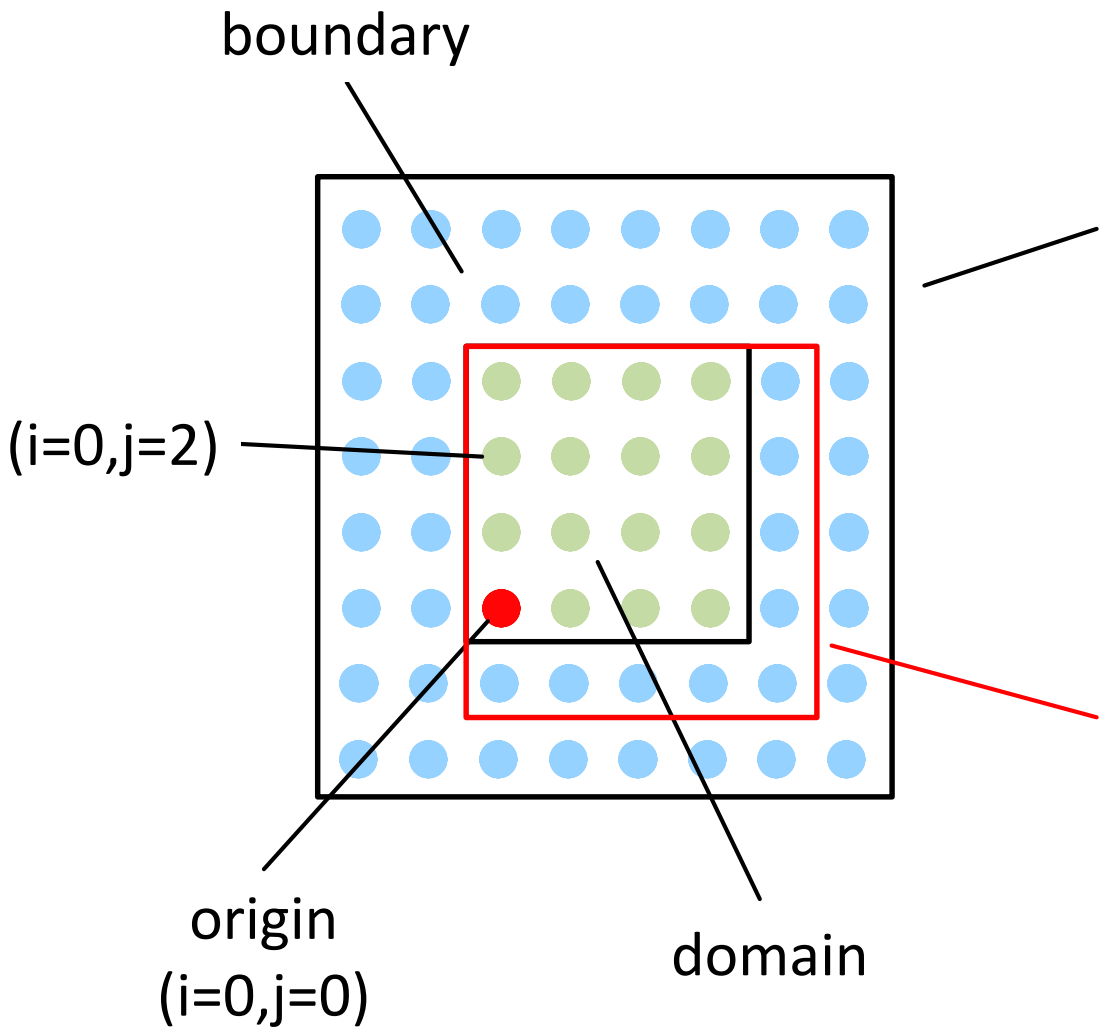
Lowering to “Affine and Standard”

```
func @stencil(%arg0: memref<4096xf64>, %arg1: memref<4096xf64>) {
  affine.for %arg2 = 0 to 10 {
    affine.for %arg3 = 0 to 10 {
      affine.for %arg4 = 0 to 61 {
        %c16 = constant 16 : index
        %c3 = constant 3 : index
        %4 = muli %c16, %c16 : index
        %5 = muli %4, %arg4 : index
        %6 = addi %arg2, %c3 : index
        %7 = muli %c16, %6 : index
        %8 = addi %5, %7 : index
        %9 = addi %arg3, %c3 : index
        %10 = addi %8, %9 : index
        %11 = load %arg1[%10] : memref<4096xf64>
        %12 = load %arg0[%10] : memref<4096xf64>
        %13 = addf %11, %12 : f64
        %cst = constant 4.000000e+00 : f64
        %14 = mulf %13, %cst : f64
        store %14, %arg1[%10] : memref<4096xf64>
  }}}
  return
}
```

Lowering to “CUDA C”

```
__host__ void _stencil_mlir(double* arg0, double* arg1) {
  double* arg1_device;
  cudaMalloc((void **)&arg1_device, 4096*sizeof(double));
  cudaMemcpy(arg1_device, arg1, 4096*sizeof(double), cudaMemcpyHostToDevice);
  double* arg0_device;
  cudaMalloc((void **)&arg0_device, 4096*sizeof(double));
  cudaMemcpy(arg0_device, arg0, 4096*sizeof(double), cudaMemcpyHostToDevice);
  stencil_kernel_stencil_kernel_mlir<<<dim3(10,1,1),dim3(10,1,1)>>>(arg1_device, arg0_device);
  cudaMemcpy(arg1, arg1_device, 4096*sizeof(double), cudaMemcpyDeviceToHost);
  cudaMemcpy(arg0, arg0_device, 4096*sizeof(double), cudaMemcpyDeviceToHost);
}
__global__ void stencil_kernel_stencil_kernel_mlir(double* arg0, double* arg1) {
  int j = blockIdx.x; int i = threadIdx.x;
  for (int k = 0; k < 61; k += 1) {
    int i21 = 16 * 16 * k + 16 * (j + 3) + (i + 3);
    double i24 = arg0[i21];
    double i25 = arg1[i21];
    double i26 = i24 + i25;
    double i28 = i26 * 4.000000;
    arg0[i21] = i28;
  }
}
```

Stencil Dialect Proposal – Stencil Storage



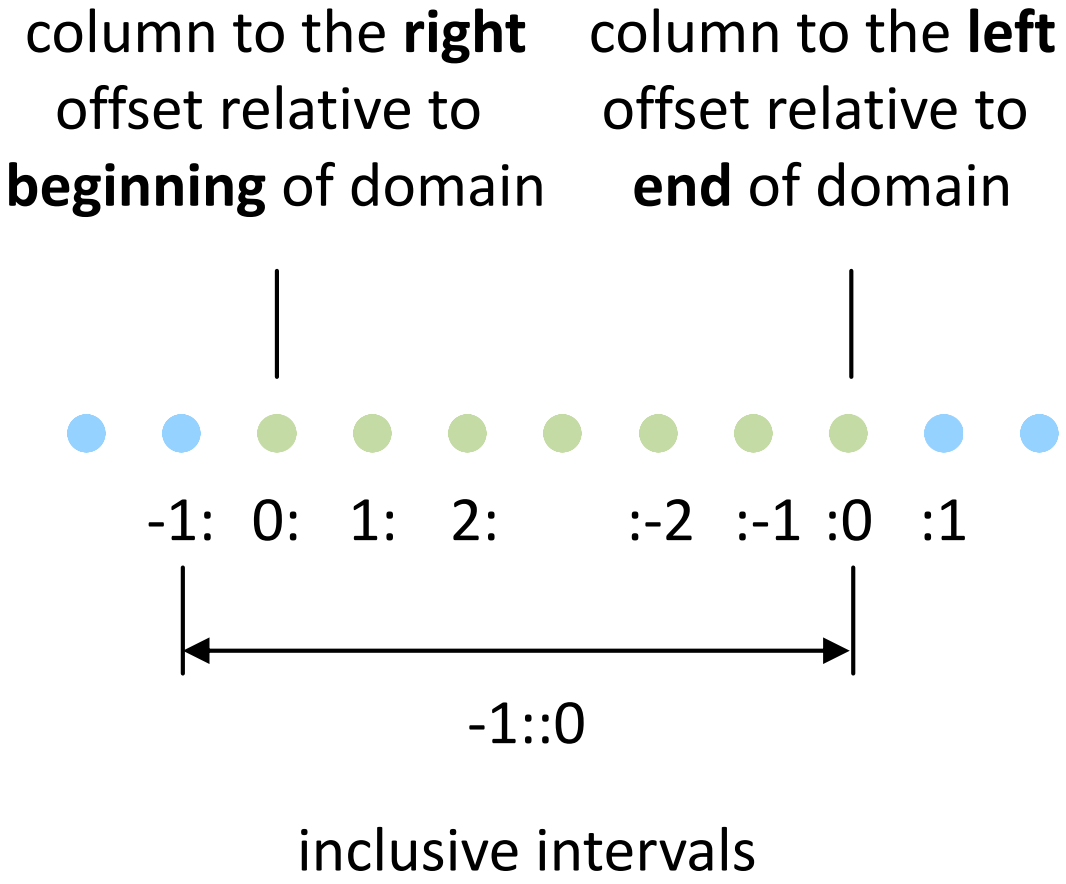
entire field/array

```
!stencil.field<?x?xf64 [-2::2,-2::2]>
```

rectangular view

```
!stencil.view<?x?xf64>
```

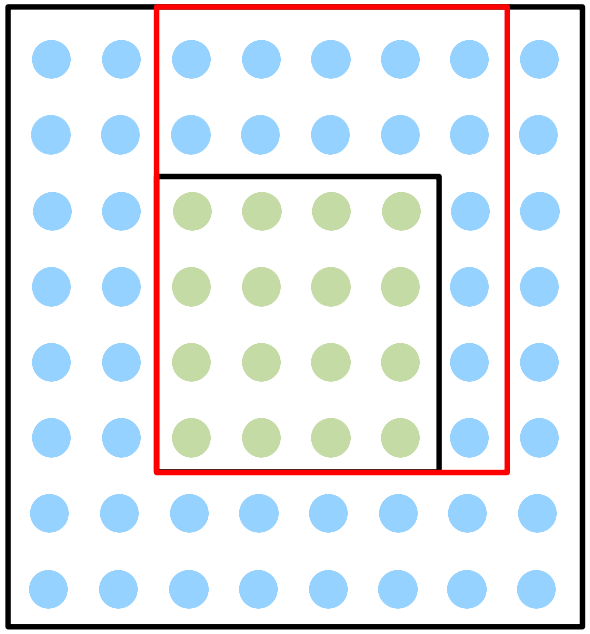
Stencil Dialect Proposal – Range Notation



inclusive intervals

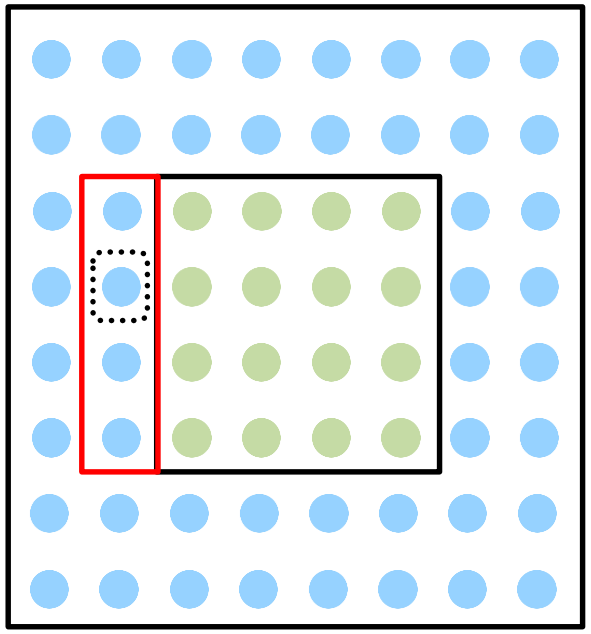
Stencil Dialect Proposal – Defining Ranges

`[0::1,0::2]`



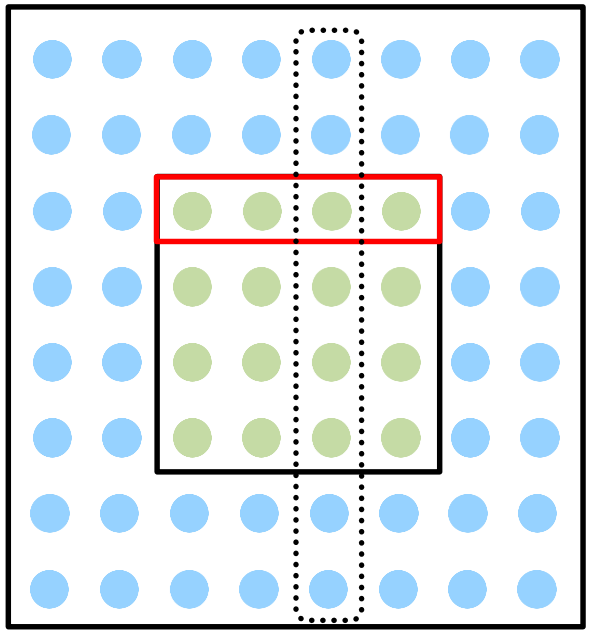
`!stencil.view<?x?xf64>`

`[-1:-1:,0::0]`



`!stencil.view<?x?xf64>`

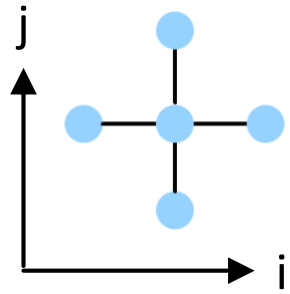
`[0::0,:0]`



`!stencil.view<?x0xf64>`

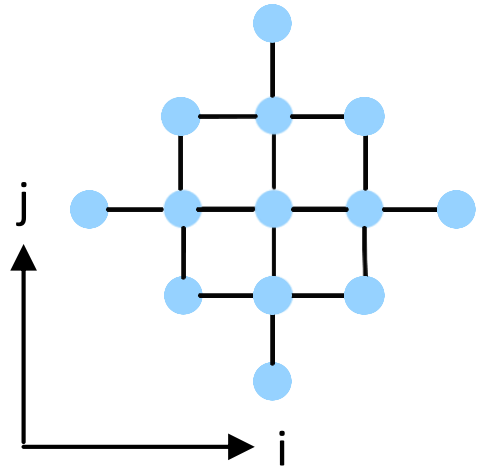
Stencil Dialect Proposal – Stencil Function

```
func @lap(%in : !stencil.view<?x?x?xf64>) -> f64
  attributes { stencil.function } {
    %0 = stencil.access %in[-1,0,0] : f64
    %1 = stencil.access %in[1,0,0] : f64
    %2 = stencil.access %in[0,1,0] : f64
    %3 = stencil.access %in[0,-1,0] : f64
    %4 = stencil.access %in[0,0,0] : f64
    %5 = addf %0, %1 : f64
    %6 = addf %2, %3 : f64
    %7 = addf %5, %6 : f64
    %8 = constant -4.0 : f64
    %9 = mulf %4, %8 : f64
    %10 = addf %9, %7 : f64
    return %10 : f64
}
```



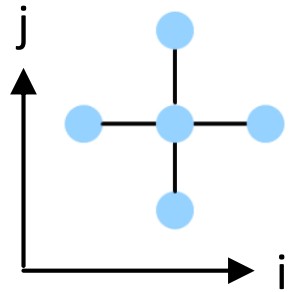
Stencil Dialect Proposal – Laplace of Laplace

```
func @laplap(%in : !stencil.view<?x?x?xf64>) -> f64
  attributes { stencil.function } {
    %0 = stencil.call @lap(%in)[-1,0,0] : f64
    %1 = stencil.call @lap(%in)[1,0,0] : f64
    %2 = stencil.call @lap(%in)[0,1,0] : f64
    %3 = stencil.call @lap(%in)[0,-1,0] : f64
    %4 = stencil.call @lap(%in)[0,0,0] : f64
    %5 = addf %0, %1 : f64
    %6 = addf %2, %3 : f64
    %7 = addf %5, %6 : f64
    %8 = constant -4.0 : f64
    %9 = mulf %4, %8 : f64
    %10 = addf %9, %7 : f64
    return %10 : f64
}
```



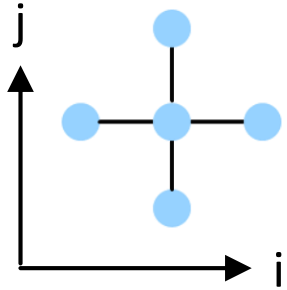
Stencil Dialect Proposal – Stencil Program

```
func @program(%in : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>,
              %out : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>) {
  %0 = stencil.load %in : !stencil.view<?x?x?xf64>
  %1 = stencil.apply @lap(%0) : !stencil.view<?x?x?xf64>
  %2 = stencil.apply @lap(%1) : !stencil.view<?x?x?xf64>
  stencil.store %out[0::0,0::0,0::0], %2
}
```



Stencil Dialect Proposal – Subdomains

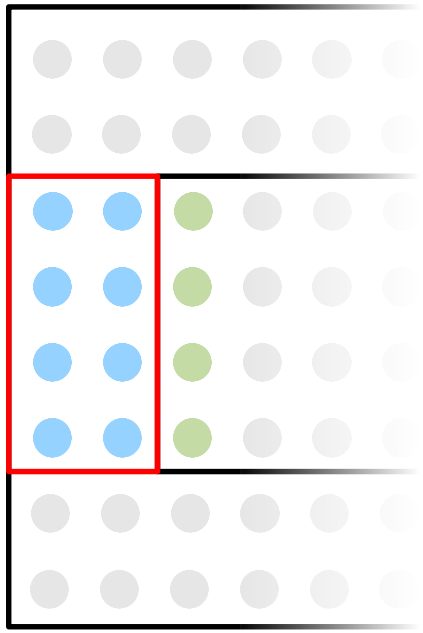
```
func @program(%in : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>,
              %out : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>) {
  %0 = stencil.load %in : !stencil.view<?x?x?xf64>
  %1 = stencil.apply @lap(%0) : !stencil.view<?x?x?xf64>
  %2 = stencil.apply @top(%0) : !stencil.view<?x?x?xf64>
  %3 = stencil.combine %1[0::0,0::0,0::-1], %2 : !stencil.view<?x?x?xf64>
  stencil.store %out[0::0,0::0,0::0], %3
}
```



shape of second subdomain maybe inferred
(or provided explicitly if required)

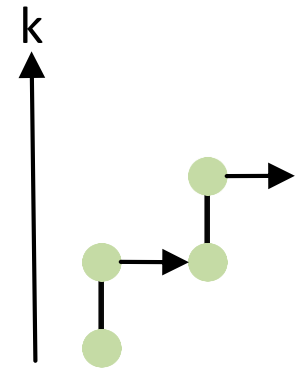
Stencil Dialect Proposal – Boundary Conditions

```
func @program(%in : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>,
              %out : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>) {
  %0 = stencil.load %in : !stencil.view<?x?x?xf64>
  %1 = stencil.apply @lap(%0) : !stencil.view<?x?x?xf64>
  %2 = stencil.slice %1[0:,0::0,0::0] : !stencil.view<0x?x?xf64>
  %3 = stencil.apply @zerograd(%2) : !stencil.view<?x?x?xf64>
  %4 = stencil.combine %1[0::0,0::0,0::0], %3 : !stencil.view<?x?x?xf64>
  stencil.store %out[-2::0,0::0,0::0], %4
}
```



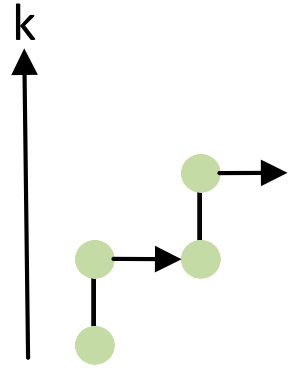
Stencil Dialect Proposal – Loop Carried Dependencies

```
func @program(%in : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>) {
  stencil.for [%k = 1 to 60] {
    %1 = stencil.load %in : !stencil.view<?x?x?xf64>
    %2 = stencil.apply @stencil(%1) : !stencil.view<?x?x?xf64>
    stencil.store %in[0::0,0::0,%k:%k:], %2
  }
}
```



Stencil Dialect Proposal – Loop Carried Dependencies

```
func @program(%in : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>) {
  %0 = stencil.load %in : !stencil.view<?x?x?xf64>
  %acc = stencil.slice %0[0::0,0::0,0:0:] : !stencil.view<?x?x?xf64>
  %res = stencil.for [%k = 1 to 60] (%acc) {
    %1 = stencil.combine %0[0::0,0::0,%k:%k:],
                         %acc[0::0,0::0,%k-1:%k-1:] : !stencil.view<?x?x?xf64>
    %2 = stencil.apply @stencil(%1) : !stencil.view<?x?x?xf64>
    stencil.yield %2[0::0,0::0,%k:%k:], %acc
  }
  stencil.store %in[0::0,0::0,0::0], %res
}
```



Conclusion

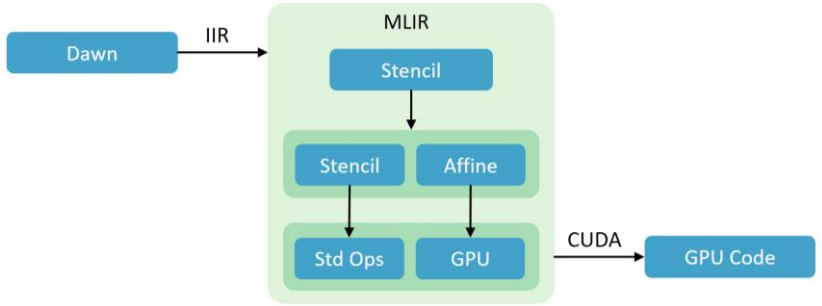
accelerating climate is important

COSMO Atmospheric Model

- Regional atmospheric model used by 7 national weather services
- Implements many different stencil programs

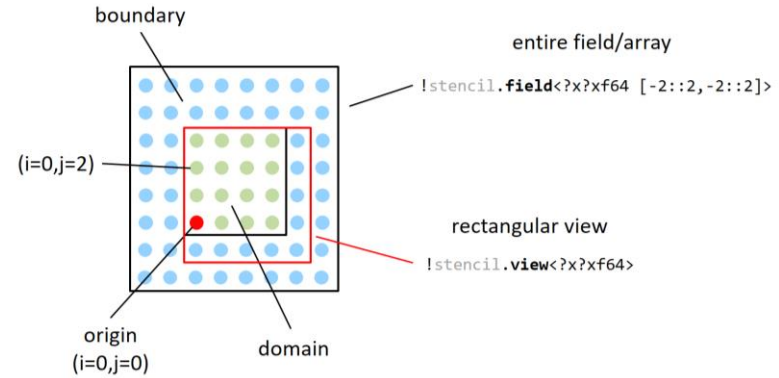
experimental toolchain

Our Current Toolchain



describe memory accesses

Stencil Dialect Proposal – Stencil Storage



explicit data-flow

Stencil Dialect Proposal – Boundary Conditions

```
func @program(%in : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>,
              %out : !stencil.field<?x?x?xf64 [-3::3,-3::3,0::0]>) {
  %0 = stencil.load %in : !stencil.view<?x?x?xf64>
  %1 = stencil.apply @lap(%0) : !stencil.view<?x?x?xf64>
  %2 = stencil.slice %1[0:,0::0,0::0] : !stencil.view<0x?x?xf64>
  %3 = stencil.apply @zerograd(%2) : !stencil.view<?x?x?xf64>
  %4 = stencil.combine %1[0::0,0::0,0::0], %3 : !stencil.view<?x?x?xf64>
  stencil.store %out[-2::0,0::0,0::0], %4
}
```