Upgrading Climate Models for Diverging Architectures

Naoya Maruyama, RIKEN AICS

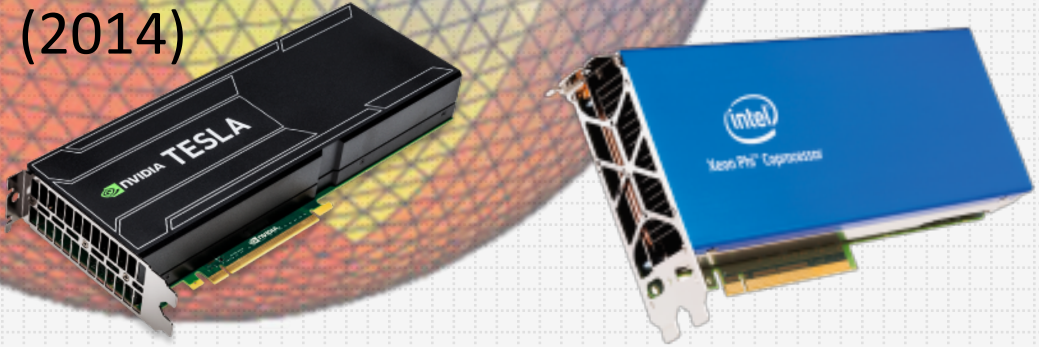
6th AICS International Symposium

February 23, 2016

NICAM: Global Climate Simulations with Non-hydrostatic ICosahedral Atmospheric Model

First global dx=3.5km run in 2004 using the Earth Simulator. Tomita et al. (2005), Miura et al. (2007, Science)

First global dx=0.87km run in 2012 using the K computer. Miyamoto et al. (2014)



AIMES: Advanced Computation and I/O Methods for Earth-System Simulations

- Funded by JST, DFG, and ANR for 3 years (SPPEXA2, 2016 – 2018)
- PI: Thomas Ludwidg (DKRZ), Naoya Maruyama (RIKEN), Takayuki Aoki (Tokyo Tech), Thomas Dubos (Ecole Polytechnique)

JST

DFG

ANR
AGENCE
NATIONALE
DE LA
RECHERCHE

AIMES

AICS

TOKYO TECH

DKRZ

l'X
ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Collaborators

Deutscher Wetterdienst
Wetter und Klima aus einer Hand
DWD

HDF

CSCS
Centro Svizzero di Calcolo
Swiss National Supercomp

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

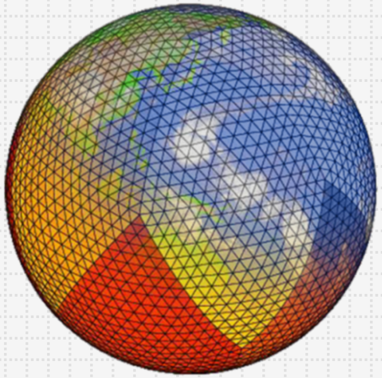
NVIDIA

intel

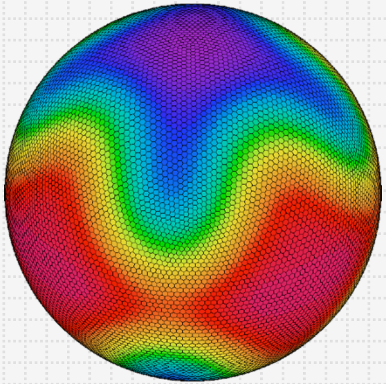
CRAY

BULL

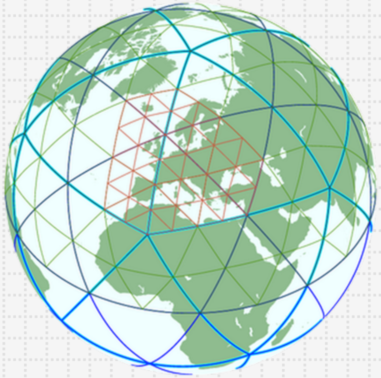
Project Overview



NICAM



DYNAMICO



ICON

Enhance programmability and performance portability

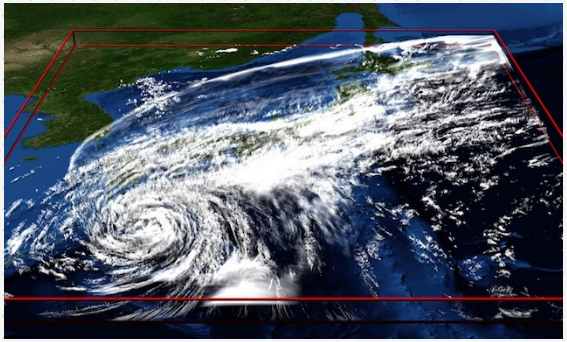
Overcome storage limitations

A common benchmark set for icosahedral models (miniapps)

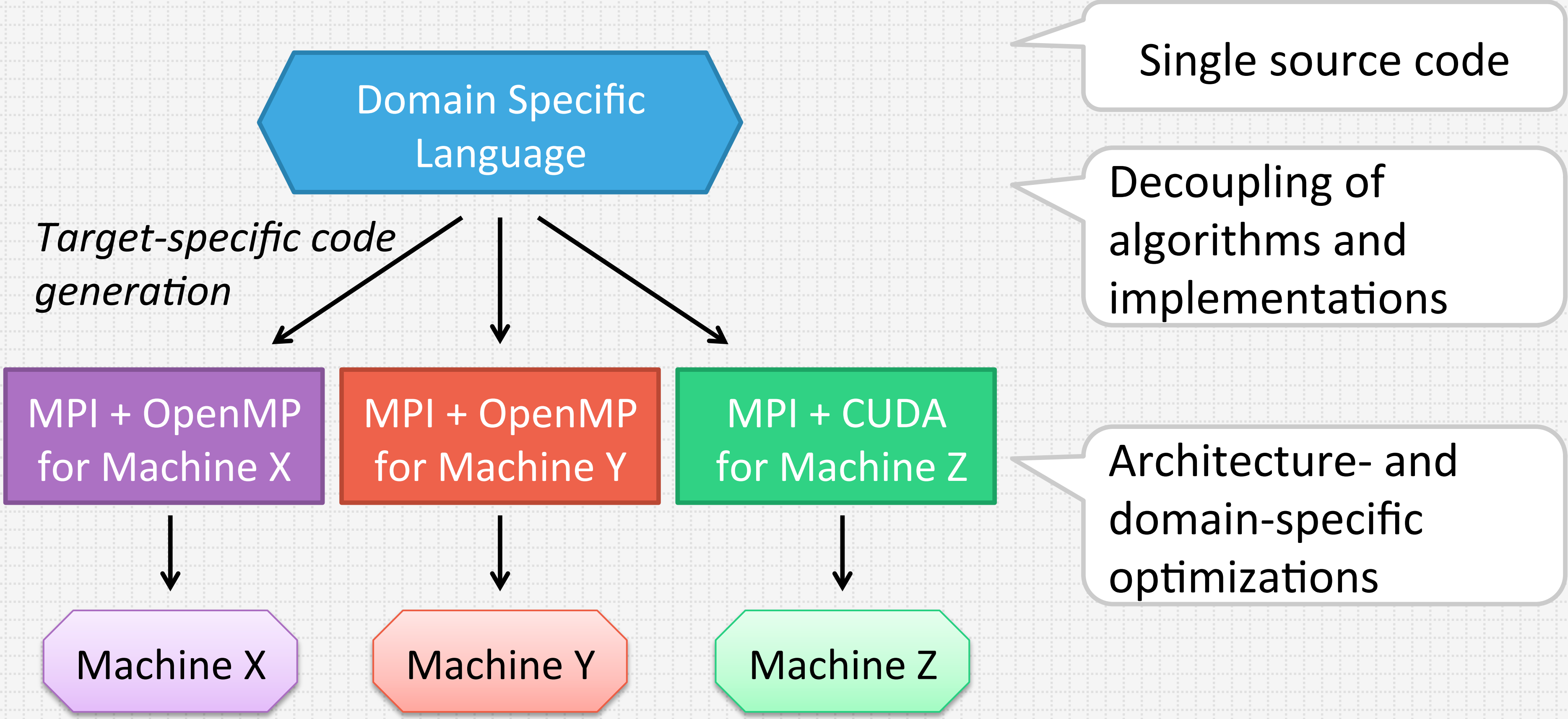
- WP1: Higher-level code design
- WP2: Massive I/O
- WP3: Evaluation
- WP4: Management

Programming Atmospheric Models

- Very large code base written in Fortran (mostly F90, some F03) with MPI and OpenMP
- Key requirement: *Performance Portability*
  - Single unified source code for maintainability
  - Extensive optimizations to alleviate memory-bandwidth bottlenecks
- Early attempts for accelerators
  - Proof-of-concept study at Tokyo Tech (ASUCA on TSUBAME)



High-Level Approach to Performance Portable Climate Models



Domain Specific Language

Target-specific code generation

MPI + OpenMP for Machine X

MPI + OpenMP for Machine Y

MPI + CUDA for Machine Z

Machine X

Machine Y

Machine Z

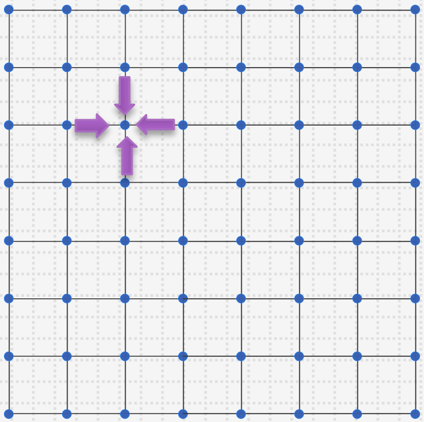
Single source code

Decoupling of algorithms and implementations

Architecture- and domain-specific optimizations

Physis Stencil Framework

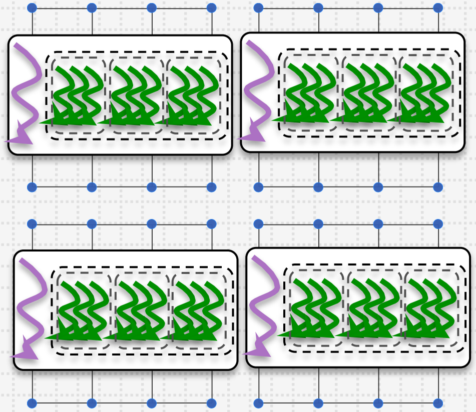
[Maruyama11], http://github.com/naoyam/physis



Stencil DSL

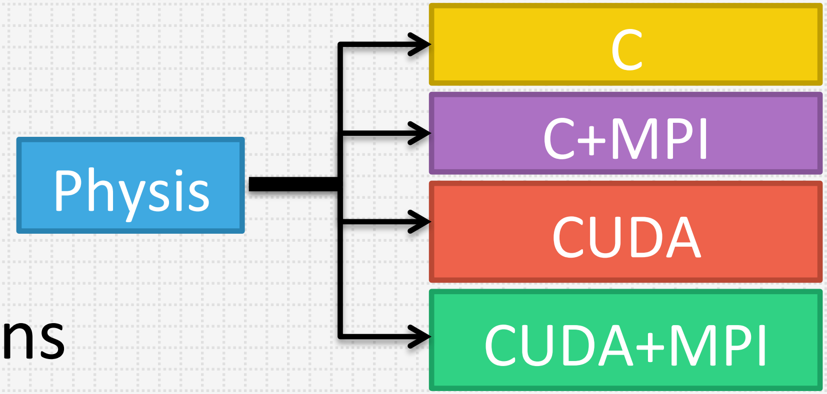
- Declarative
- Portable
- Global-view
- C-based

```
void diffusion(int x, int y, int z,
     PSGrid3DFloat g1, PSGrid3DFloat g2) {
float v = PSGridGet(g1,x,y,z)
 +PSGridGet(g1,x-1,y,z)+PSGridGet(g1,x+1,y,z)
 +PSGridGet(g1,x,y-1,z)+PSGridGet(g1,x,y+1,z)
 +PSGridGet(g1,x,y,z-1)+PSGridGet(g1,x,y,z+1);
PSGridEmit(g2,v/7.0);
}
```



DSL Compiler

- Target-specific code generation and optimizations
- Automatic parallelization



Physis

C

C+MPI

CUDA

CUDA+MPI

Physis DSL Overview

- Focus on stencils with regular multi-dimensional grids
- C with a small number of custom constructs for stencil computations
- Consisting of constructs for:
  - Grid data structures
  - Stencil definitions
  - Control logic

```
void kernel(const int x, const int y, const int z,
            PSGrid3DPoint g1, PSGrid3DDouble g2) {
  double v = PSGridGet(g1,x,y,z).vx
    +PSGridGet(g1,x-1,y,z).vx + PSGridGet(g1,x+1,y,z).vx
    +PSGridGet(g1,x,y-1,z).vy + PSGridGet(g1,x,y+1,z).vy
    +PSGridGet(g1,x,y,z-1).vz + PSGridGet(g1,x,y,z+1).vz;
  PSGridEmit(g2,v/7.0);
}
```

User-Defined Types

Example: 3D grids with multiple fields

```
struct point {
  double vx, vy, vz
};

DeclareGrid3D(struct point, Point);

void foo() {
  PSGrid3DPoint g=PSGrid3dPointNew(N,N,N);
  ...
  PSGridFree(g);
}
```

Data Layout Optimization

- User code always AoS
- Converted to suitable layouts depending on target architectures

User code (AoS)

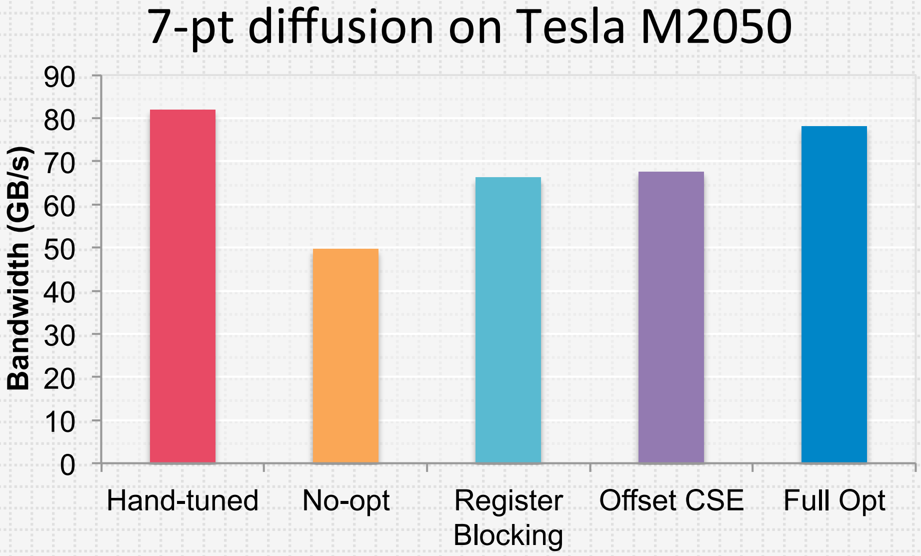
```
struct Point {
    float u;
    float v;
    float w;
    float x[19];
};
DeclareGrid3D(Point, struct Point);

PSGrid3DPoint g;
...
PSGridGet(g.u, i, j, k);
PSGridGet(g.x[l], i, j+1, k);
```

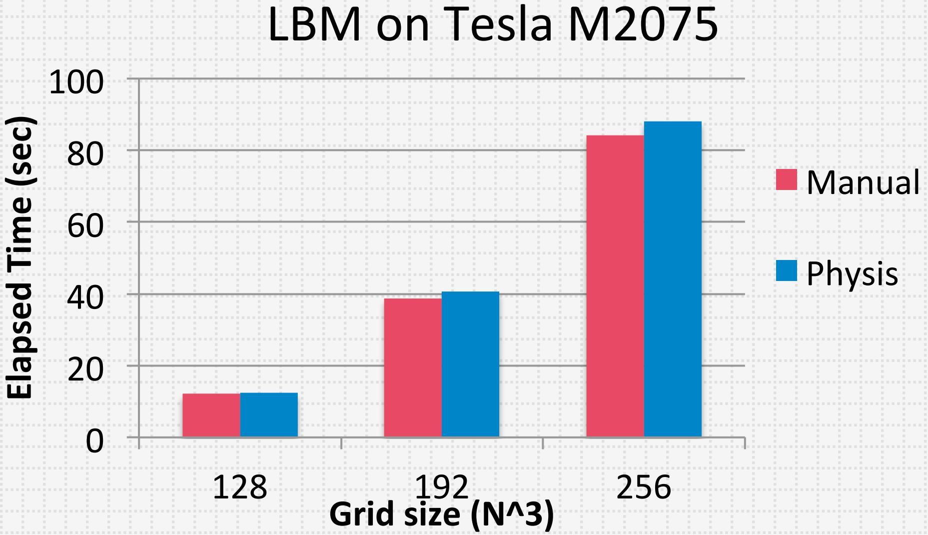
Generated code for GPU (SoA)

```
struct Point {
    float *u;
    float *v;
    float *w;
    float *x;
};
...
// PSGridGet(g.u, i, j, k);
g->u[i+j*nx+k*nx*ny]
// PSGridGet(g.x[2], i, j+1, k);
g->x[i+j*nx+k*nx*ny+l*nx*ny*nz];
```

Performance Results on GPU

- Basic local optimizations at DSL translation time
- Automatic GPU-specific optimizations such as data layout conversions
- Close to hand-optimized performance on NVIDIA GPUs

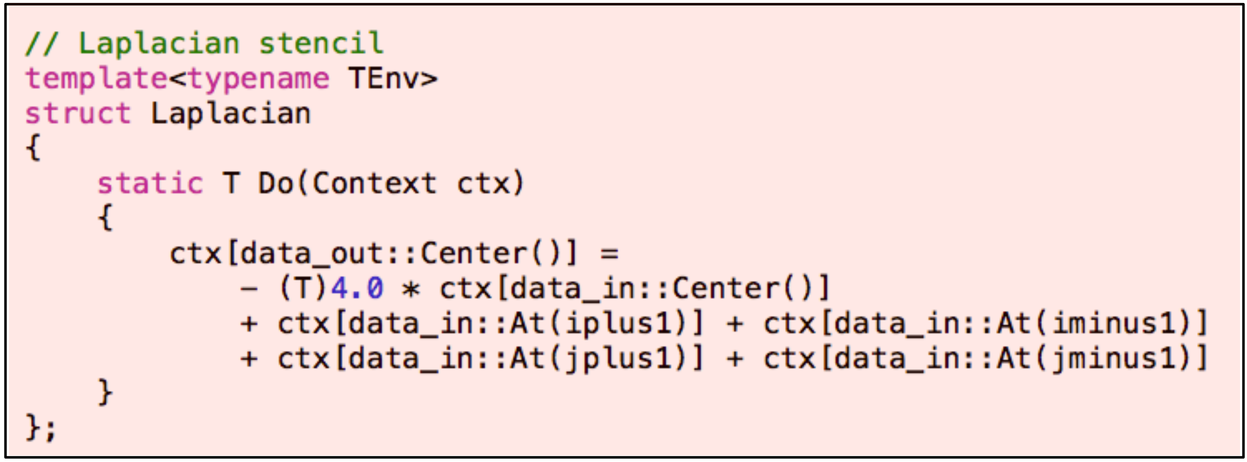




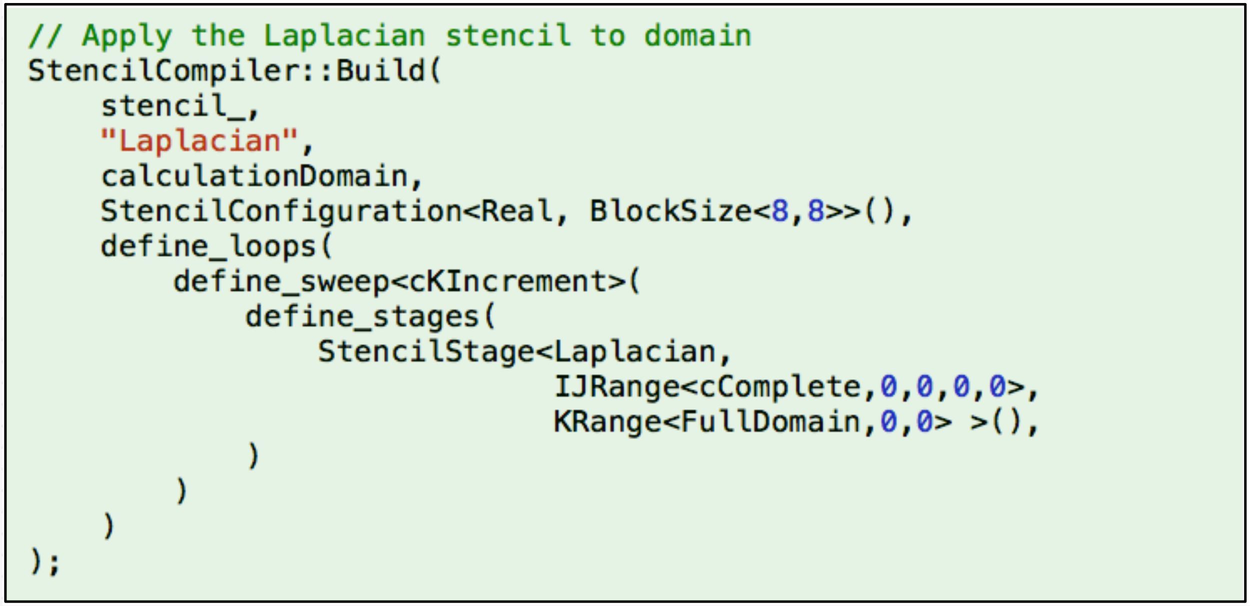
STELLA Stencil DSL

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

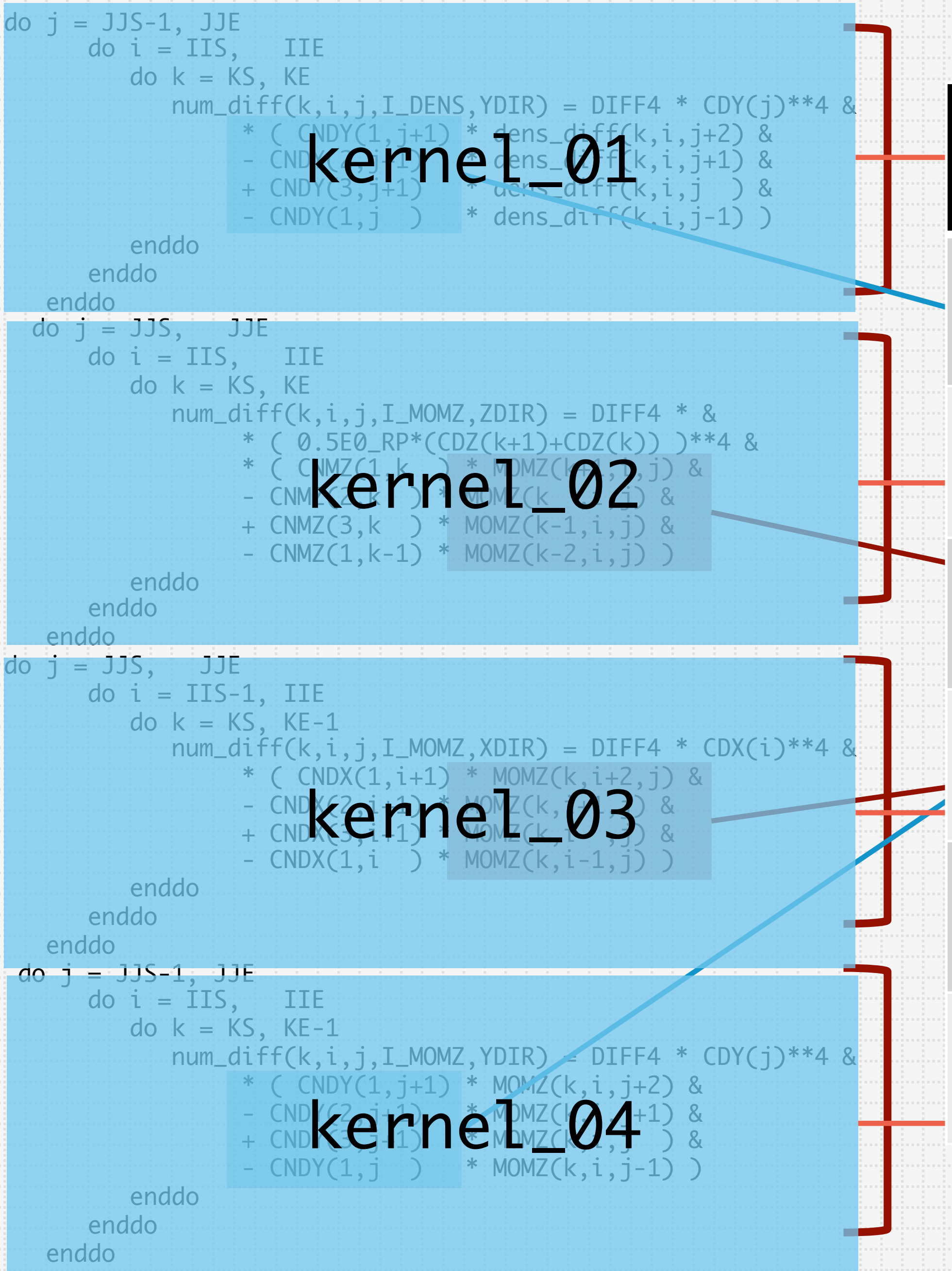
- DSL for the dynamical core of the COSMO model (SwissMeteo)
- Based on C++ meta-programming
- Performance portability over CPUs and GPUs
- Gysi et al. (2015)



```
// Laplacian stencil
template<typename TEnv>
struct Laplacian
{
    static T Do(Context ctx)
    {
        ctx[data_out::Center()] =
            - (T)4.0 * ctx[data_in::Center()]
            + ctx[data_in::At(iplus1)] + ctx[data_in::At(iminus1)]
            + ctx[data_in::At(jplus1)] + ctx[data_in::At(jminus1)]
    }
};
```



```
// Apply the Laplacian stencil to domain
StencilCompiler::Build(
    stencil_,
    "Laplacian",
    calculationDomain,
    StencilConfiguration<Real, BlockSize<8,8>>(),
    define_loops(
        define_sweep<cKIncrement>(
            define_stages(
                StencilStage<Laplacian,
                             IJRange<cComplete,0,0,0,0>,
                             KRange<FullDomain,0,0> >(),
            )
        )
    )
);
```



```
do j = JJS-1, JJE
     do i = IIS,   IIE
        do k = KS, KE
           num_diff(k,i,j,I_DENS,YDIR) = DIFF4 * CDY(j)**4 &
                * ( CNDY(1,j+1) * dens_diff(k,i,j+2) &
                - CNDY(2,j+1) * dens_diff(k,i,j+1) &
                + CNDY(3,j+1) * dens_diff(k,i,j  ) &
                - CNDY(1,j  ) * dens_diff(k,i,j-1) )
        enddo
     enddo
  enddo
```

```
  do j = JJS,   JJE
     do i = IIS,   IIE
        do k = KS, KE
           num_diff(k,i,j,I_MOMZ,ZDIR) = DIFF4 * &
                * ( 0.5E0_RP*(CDZ(k+1)+CDZ(k)) )**4 &
                * ( CNMZ(1,k  ) * MOMZ(k+1,i,j) &
                - CNMZ(2,k  ) * MOMZ(k  ,i,j) &
                + CNMZ(3,k  ) * MOMZ(k-1,i,j) &
                - CNMZ(1,k-1) * MOMZ(k-2,i,j) )
        enddo
     enddo
  enddo
```

```
do j = JJS,   JJE
     do i = IIS-1, IIE
        do k = KS, KE-1
           num_diff(k,i,j,I_MOMZ,XDIR) = DIFF4 * CDX(i)**4 &
                * ( CNDX(1,i+1) * MOMZ(k,i+2,j) &
                - CNDX(2,i+1) * MOMZ(k,i+1,j) &
                + CNDX(3,i+1) * MOMZ(k,i  ,j) &
                - CNDX(1,i  ) * MOMZ(k,i-1,j) )
        enddo
     enddo
  enddo
```

```
 do j = JJS-1, JJE
     do i = IIS,   IIE
        do k = KS, KE-1
           num_diff(k,i,j,I_MOMZ,YDIR) = DIFF4 * CDY(j)**4 &
                * ( CNDY(1,j+1) * MOMZ(k,i,j+2) &
                - CNDY(2,j+1) * MOMZ(k,i,j+1) &
                + CNDY(3,j+1) * MOMZ(k,i,j  ) &
                - CNDY(1,j  ) * MOMZ(k,i,j-1) )
        enddo
     enddo
  enddo
```

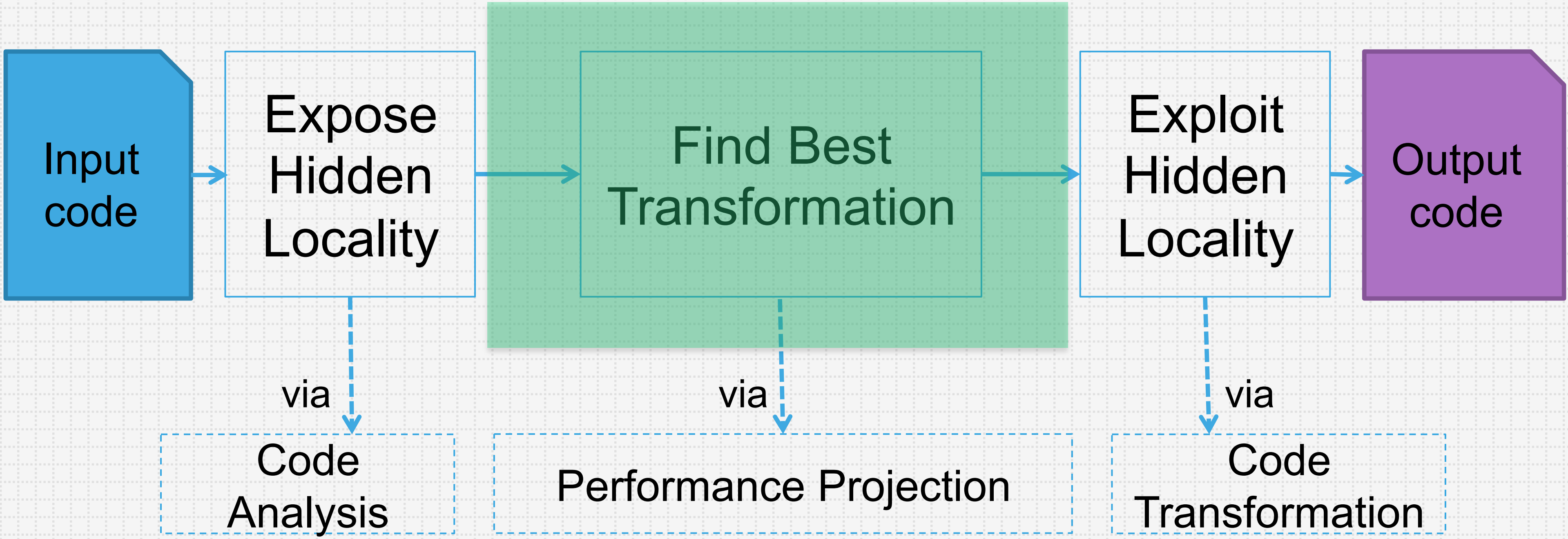
App.	No. Kernels	No. Arrays	Redundancy
SCALE	142	64	41%
WRF	122	46	24%
ASUCA	115	58	17%
MITgcm	94	31	22%
HOMME	43	27	21%
COSMO	35	24	38%

Up to 100s

Tens

1.2x~1.67x

Automated Locality Optimization



Input code

Expose Hidden Locality

Find Best Transformation

Exploit Hidden Locality

Output code

via

Code Analysis

via

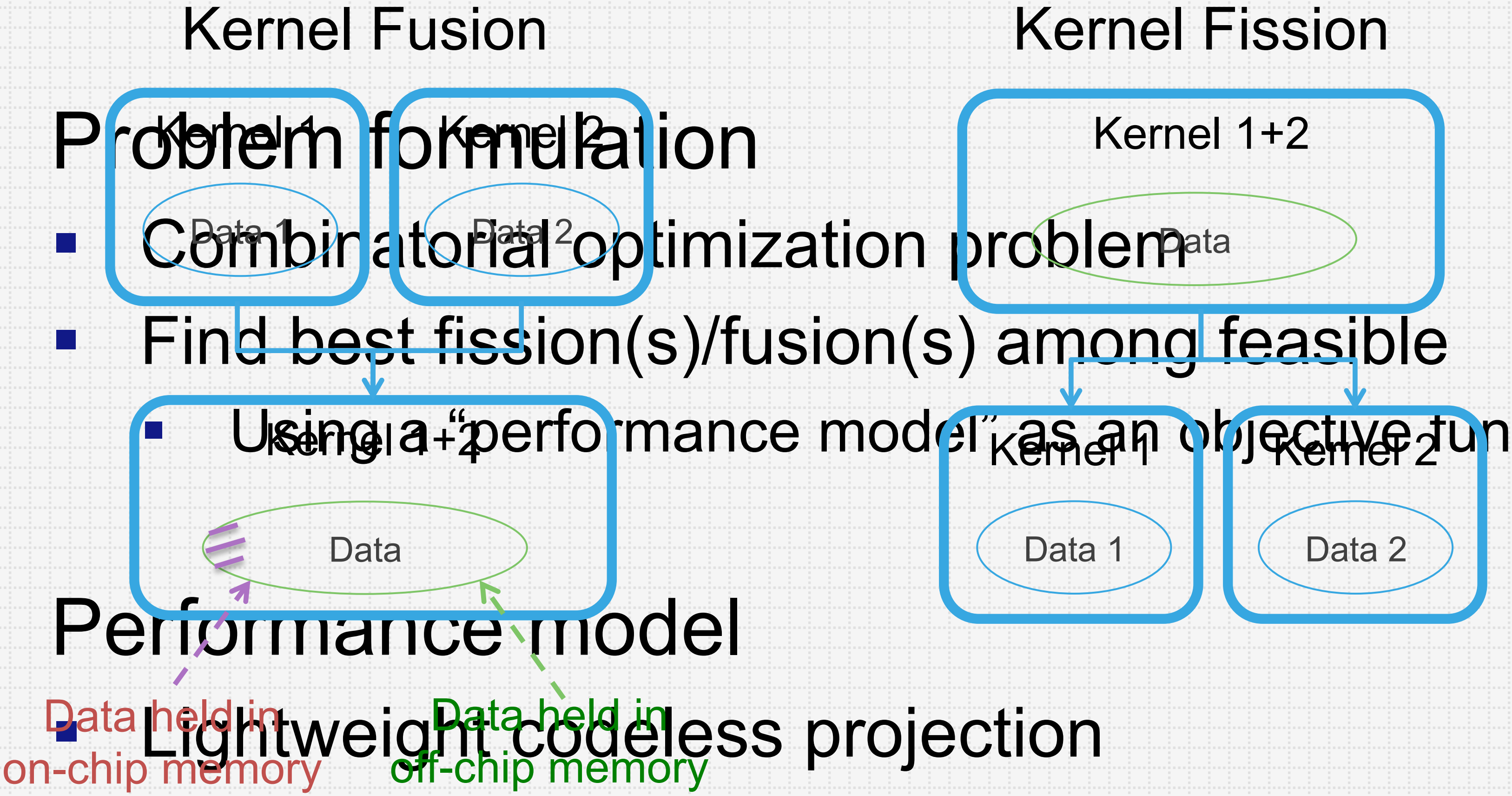
Performance Projection

via

Code Transformation

Transformation

- Types of transformation



Kernel Fusion

Kernel 1
Data 1
Kernel 2
Data 2
Kernel 1+2
Data
Data held in on-chip memory
Data held in off-chip memory

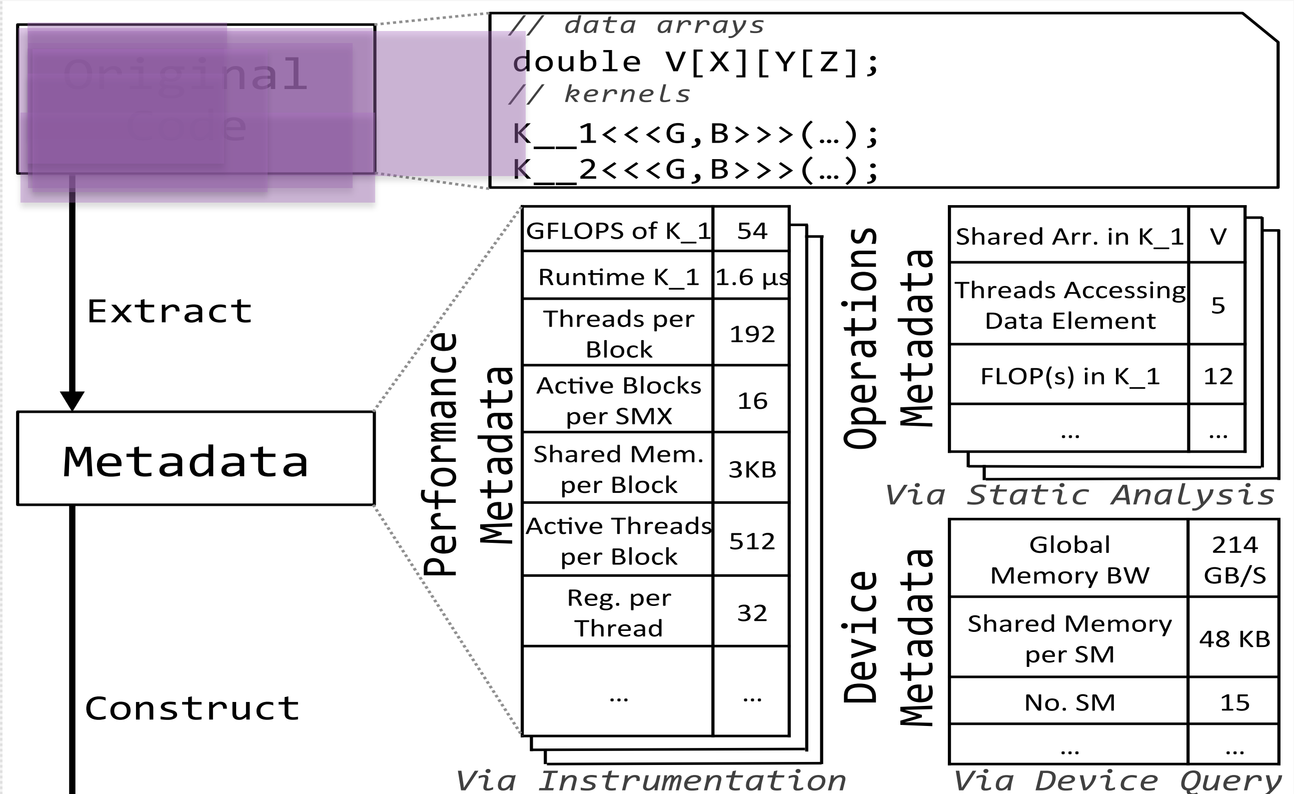
Kernel Fission

Kernel 1+2
Data
Kernel 1
Data 1
Kernel 2
Data 2

- Problem formulation
  - Combinatorial optimization problem
  - Find best fission(s)/fusion(s) among feasible
    - Using a "performance model" as an objective function
- Performance model
  - Lightweight codeless projection

M. Wahib and N. Maruyama, Scalable Kernel Fusion for Memory-Bound GPU Applications, SC'14

End-to-End Transformation



```
// data arrays
double V[X][Y[Z];
// kernels
K__1<<<G,B>>>(…);
K__2<<<G,B>>>(…);
```

Performance Metadata

GFLOPS of K_1	54
Runtime K_1	1.6 μs
Threads per Block	192
Active Blocks per SMX	16
Shared Mem. per Block	3KB
Active Threads per Block	512
Reg. per Thread	32
…	…

Via Instrumentation

Operations Metadata

Shared Arr. in K_1	V
Threads Accessing Data Element	5
FLOP(s) in K_1	12
…	…

Via Static Analysis

Device Metadata

Global Memory BW	214 GB/S
Shared Memory per SM	48 KB
No. SM	15
…	…

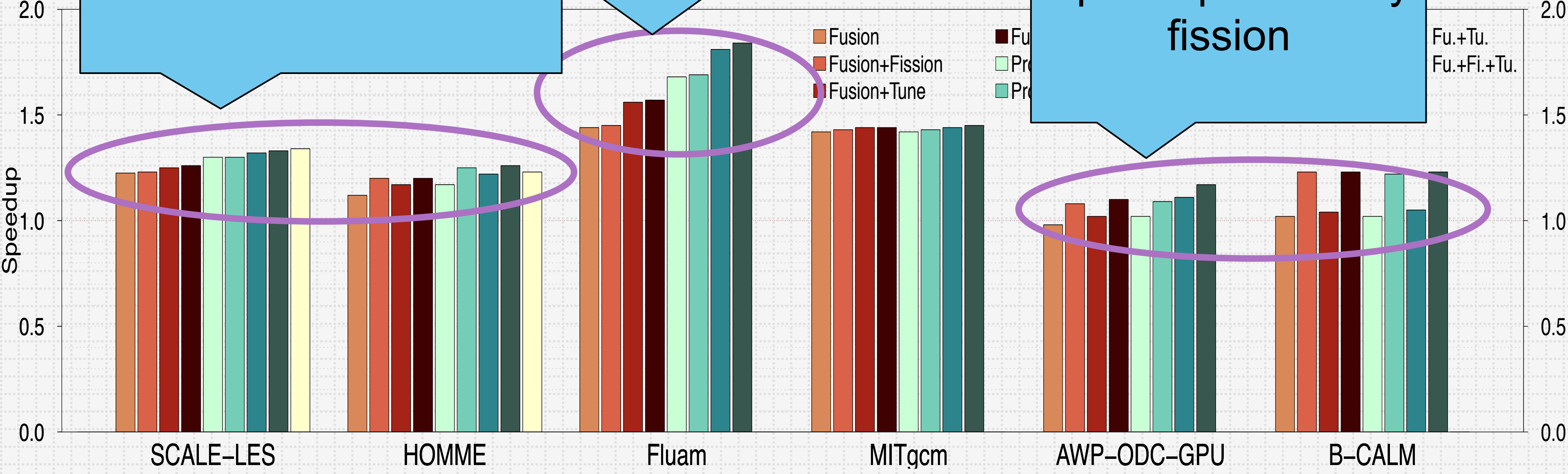
Via Device Query

- [illegible]
- [illegible]
- Output as JSON [illegible] constraints
- Output as .DOT

Case Studies with Production Apps

App.	Description
SCALE [Weather]	Next generation mesoscale weather model [**Four years in development**]
HOMME [Climate]	Dynamical core of Community Atmospheric Model (CAM)
Fluam [Hydrodynamics]	A fluctuating particle hydrodynamics application based on an hybrid Eulerian-Lagrangian approach
MITgcm [Oceanic]	An oceanic general circulation model relying on a finite volume numerical method [**18 years in development**]
AWP-ODC-GPU [Seismic]	An earthquake wave propagation simulator [**ACM Gordon Bell finalist**]
B-CALM [FDTD]	A 3D-FDTD simulator which models the permittivity of dispersive material

Results



Problem:
Generation of code

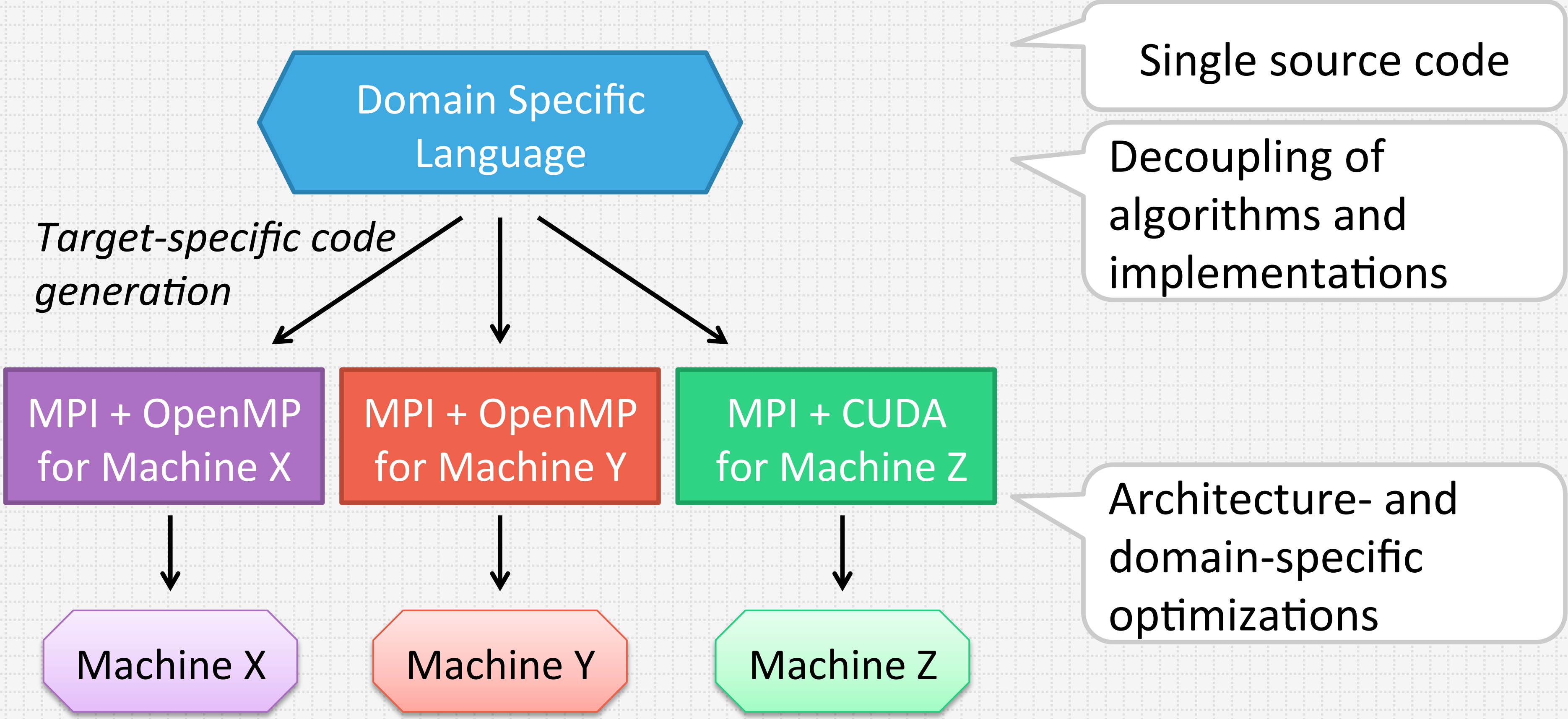
Problem:
Filtering latency-bound kernels

Note:
Speedup driven by fission

Nvidia K40 speedup compared to baseline CUDA version with no kernel fission/fusion

M. Wahib and N. Maruyama, Automated GPU Kernel Transformations in Large-Scale Production Stencil Applications, HPDC'15

High-Level Approach to Performance Portable Climate Models



Domain Specific Language

Target-specific code generation

MPI + OpenMP for Machine X

MPI + OpenMP for Machine Y

MPI + CUDA for Machine Z

Machine X

Machine Y

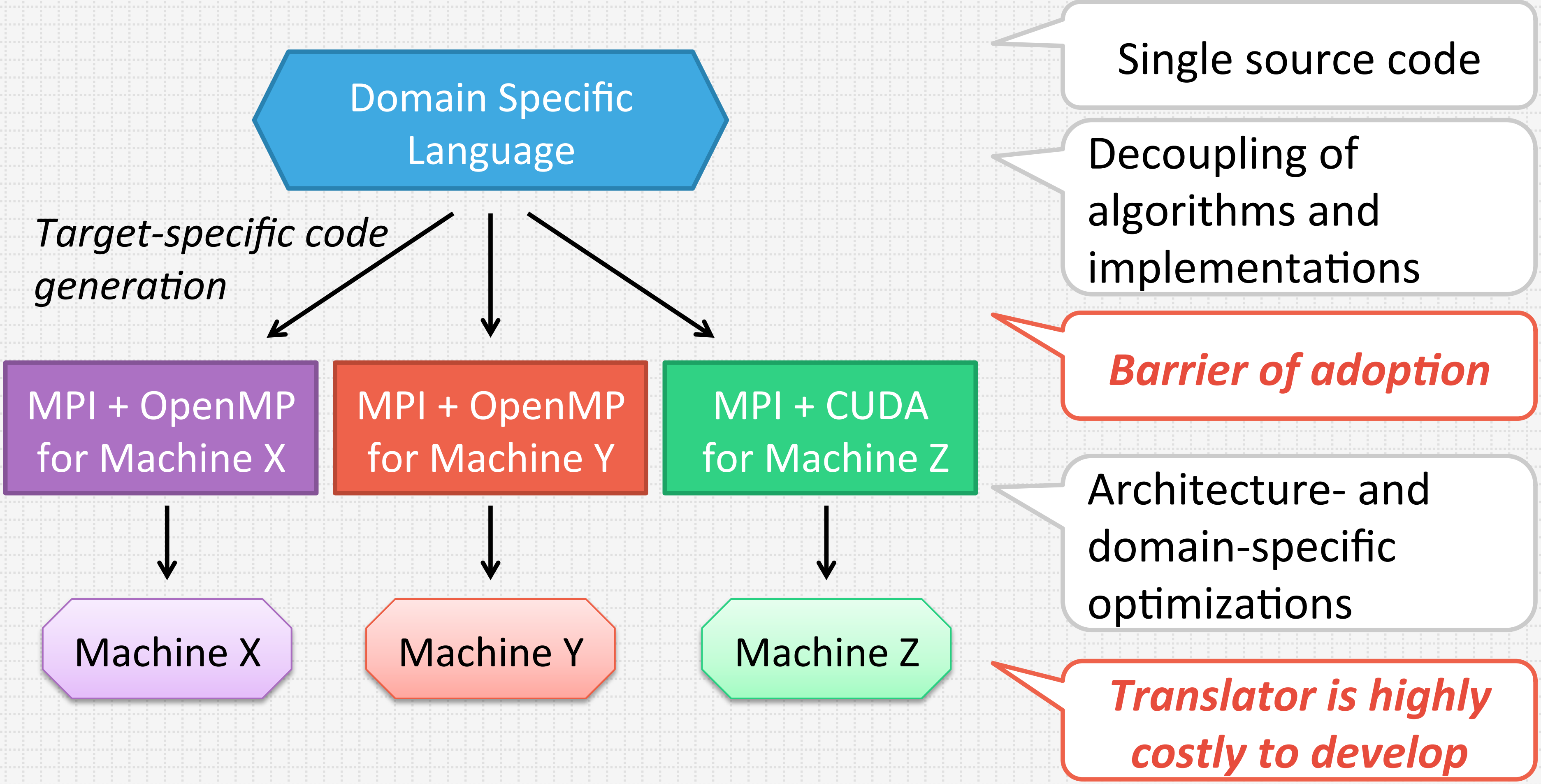
Machine Z

Single source code

Decoupling of algorithms and implementations

Architecture- and domain-specific optimizations

High-Level Approach to Performance Portable Climate Models *in Practice*



Domain Specific Language

Target-specific code generation

MPI + OpenMP for Machine X

MPI + OpenMP for Machine Y

MPI + CUDA for Machine Z

Machine X

Machine Y

Machine Z

Single source code

Decoupling of algorithms and implementations

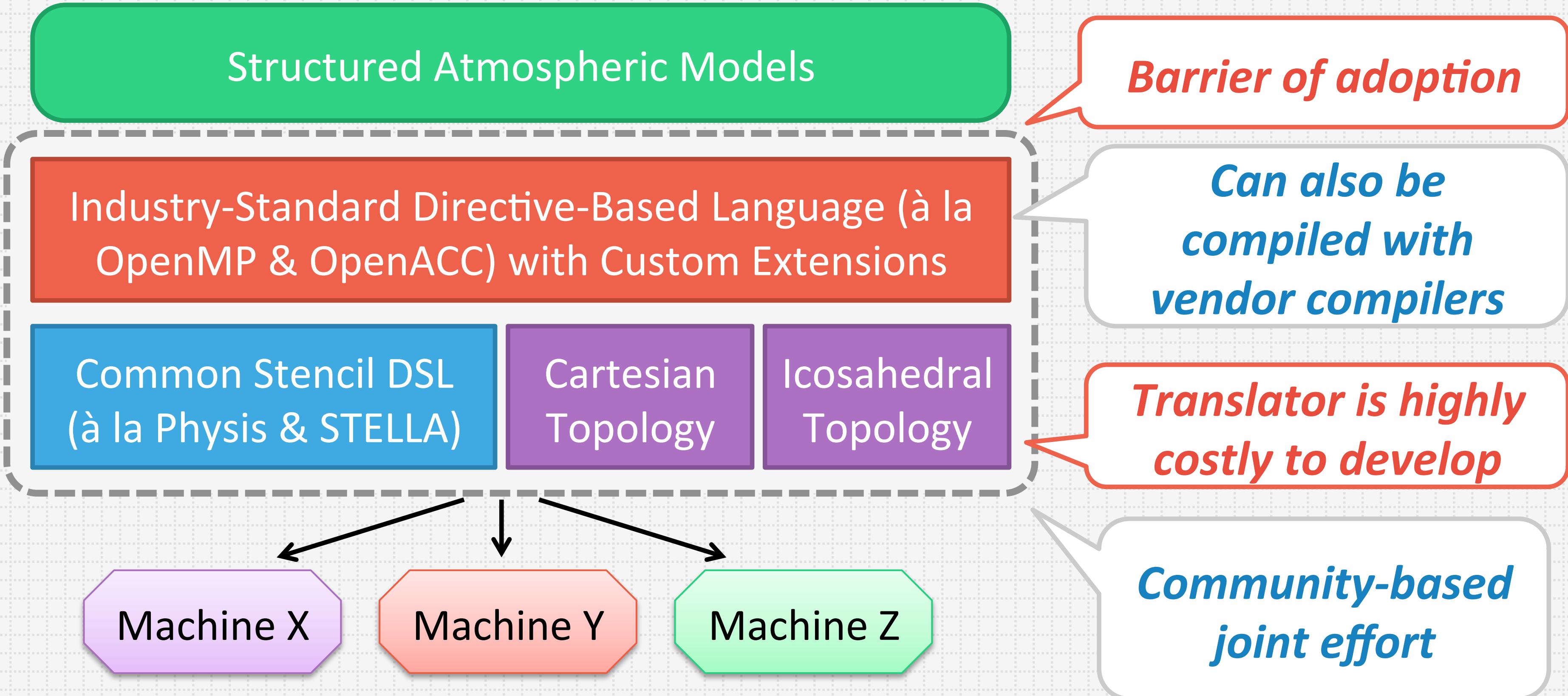
Barrier of adoption

Architecture- and domain-specific optimizations

Translator is highly costly to develop

Towards Sustainable Programming Environment for Atmospheric Models

- Design a programming interface that is compatible with accepted standards (e.g., OpenMP)
- Build a community for joint development



Barrier of adoption

Can also be compiled with vendor compilers

Translator is highly costly to develop

Community-based joint effort

Note: Under discussion with CSCS. Not necessarily reflect the final design.

Summary

- AIMES joint project
  - Extending icosahedral climate models with advanced programming and I/O methods
- Plan
  - Joint effort to build sustainable, performance-portable programming environment for extreme-scale atmospheric models
- Expected outcome
  - Future-proof atmospheric models
  - Foundation for encompassing further aggressive optimizations such as kernel fusion

AIMES

AICS

TOKYO TECH

DKRZ

l'X ÉCOLE POLYTECHNIQUE UNIVERSITÉ PARIS-SACLAY

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

Deutscher Wetterdienst Wetter und Klima aus einer Hand DWD

NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

NVIDIA

intel

BULL

CRAY

Acknowledgments

- Hisashi Yashiro, Mohamed Wahib, Ryuji Yoshida, Hirofumi Tomita (AICS)
- Takayuki Aoki (Tokyo Tech)
- Lucas Benedicic, William Sawyer (CSCS)
- Julian Kunkel (DKRZ), Thomas Dubos (Ecole Polytechnique)