HPC Frameworks for Stencil Applications

Framework-based Stencil Applications

Weather prediction code ASUCA

Numerical weather prediction is one of the major applications in high-performance computing and is accelerated on GPU supercomputers. Obtaining high-performance using thousands of GPUs often needs skillful programming. The Japan Meteorological Agency is developing a high-resolution meso-scale weather prediction code ASUCA. We have implemented it on a multi-GPU platform by using Fortran and C++ frameworks.

Governing equations

- Compressible nonhydrostatic equations
- Flux form, Time-splitting scheme, HEVI scheme
- Generalized coordinate
  Horizontal: Arakawa-C grid, Vertical: Lorenz grid

- Equation of motion

$$\frac{\partial}{\partial t}\left(\frac{\rho u^i}{J}\right)+\frac{\partial}{\partial \hat{x}^j}\left(\frac{\rho u^i \hat{u}^j}{J}\right)+\frac{\partial}{\partial \hat{x}^n}\left(\frac{1}{J}\frac{\partial \hat{x}^n}{\partial x^i}p\right)-\frac{\rho g^i}{J}=\frac{F^i}{J}$$

- Eq. of potential temperature

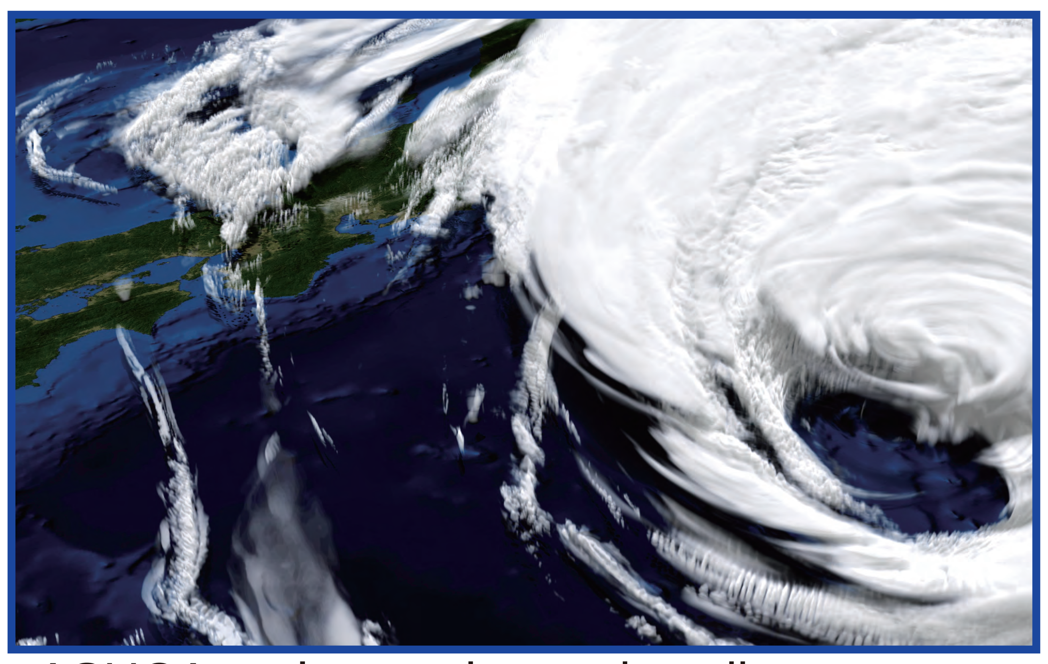
$$\frac{\partial}{\partial t}\left(\frac{\rho\theta_m}{J}\right)+\frac{\partial}{\partial \hat{x}^i}\left(\frac{\rho\theta_m \hat{u}^i}{J}\right)=\frac{F_{\rho\theta_m}}{J}$$

- Continuity equation

$$\frac{\partial}{\partial t}\left(\frac{\rho}{J}\right)+\frac{\partial}{\partial \hat{x}^i}\left(\frac{\rho \hat{u}^i}{J}\right)=\frac{F_\rho}{J}$$

- Equation of state

$$\rho=\frac{p_0}{R\theta_m}\left(\frac{p}{p_0}\right)^{C_v/C_p}$$



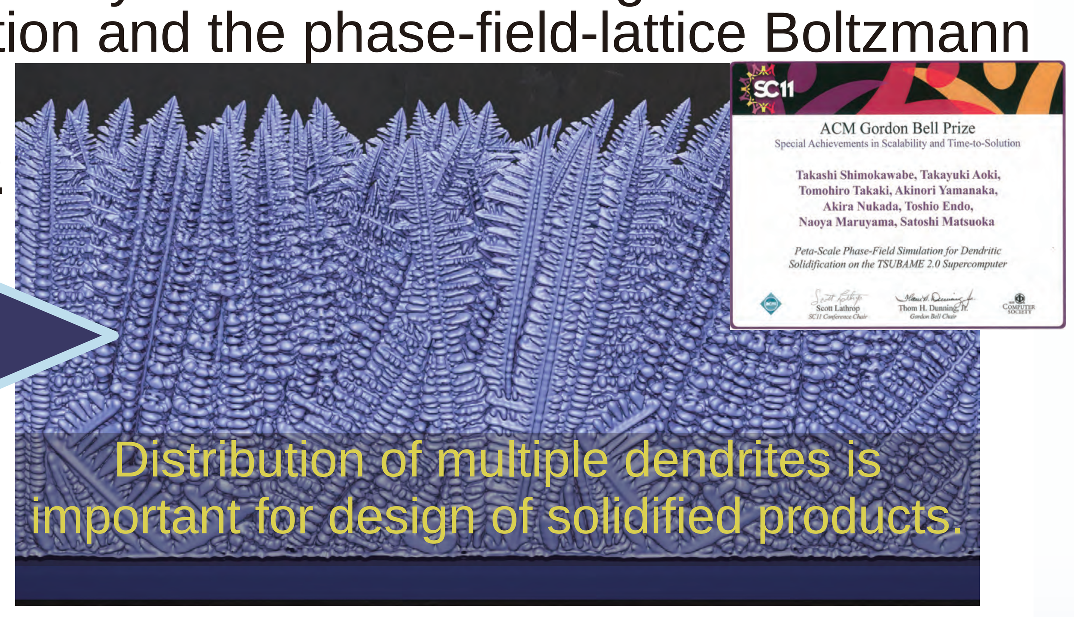
ASUCA real operation to describe a typhoon with 5,376 × 4,800 × 57 mesh using 672 GPUs of the TSUBAME 2.5.

Phase-field simulations (ongoing)

Collaboration with Prof. Takaki (Kyoto institute of technology)

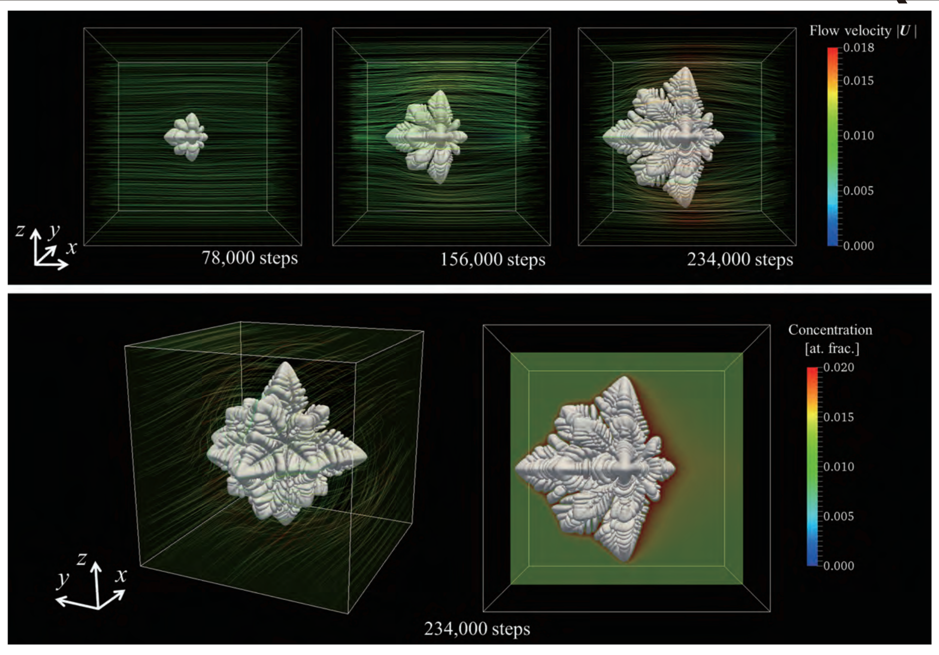
The mechanical properties of metal materials largely depend on their intrinsic internal microstructures. The phase-field simulation is the most powerful method known to simulate the micro-scale dendritic growth during solidification in a binary alloy. We are introducing our frameworks into 2011 Gordon Bell application and the phase-field-lattice Boltzmann model.

2011 ACM Gordon Bell Prize



Dendritic growth in the binary alloy solidification with 4096 × 1024 × 4096 (768 GPUs of TSUBAME2.0)

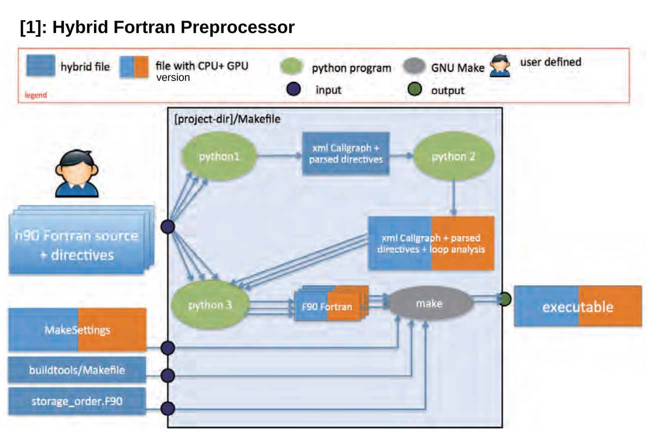
Phase-field-lattice Boltzmann (PFLB) model

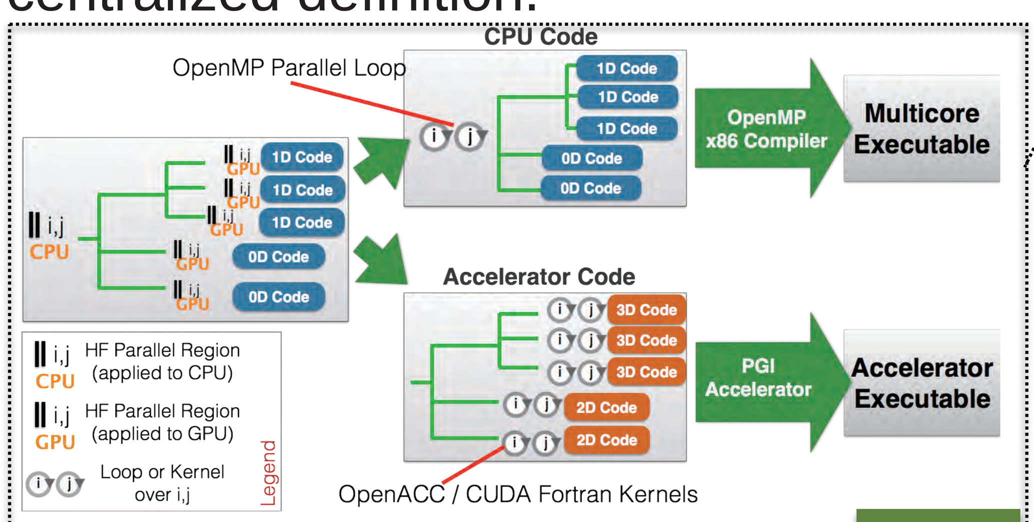


The melt convection is one of the dominant factors determining the dendrite morphology. Recently, we have developed a phase-field-lattice Boltzmann (PFLB) model which can express the dendrite motion and growth in the melt convection. On the other hand, the PFLB simulation needs large computational cost.

Fortran Approach

In order to support GPU accelerated Fortran user code for the ASUCA weather prediction model, a second approach has been developed: Hybrid Fortran, a Fortran parser, preprocessor and directive language that supports multiple parallelizations (fine grained / coarse grained) and automatically adjusts the storage order according to a centralized definition.



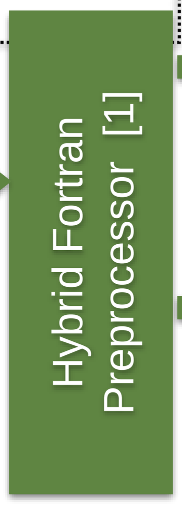



```fortran
subroutine calculate_all_columns(a, sums)
    real, intent(in), dimension(NZ, NX, NY) :: a
    real, intent(out), dimension(NX, NY) :: sums
    integer(4) :: i, j

    @domainDependant{attribute(autoDom), template(KIJ)}
    a, sums
    @end domainDependant

    @parallelRegion{appliesTo(CPU), domName(i,j), domSize(NX,NY)}
    sum_column(a, sums)
    @end parallelRegion
end subroutine

subroutine sum_column(a, sum)
    real, intent(in), dimension(NZ) :: a
    real, intent(out) :: sum
    integer(4) :: k

    @domainDependant{attribute(autoDom), domName(i,j), domSize(NX,NY)}
    a, sum
    @end domainDependant

    @parallelRegion{appliesTo(GPU), domName(i,j), domSize(NX,NY)}
    sum = 0.0
    do k=1, NZ
        sum = sum + a(k)
    end do
    @end parallelRegion
end subroutine
```

CPU parallelization (coarse grained)

Privatization of scalars / 1D arrays in parallel domain

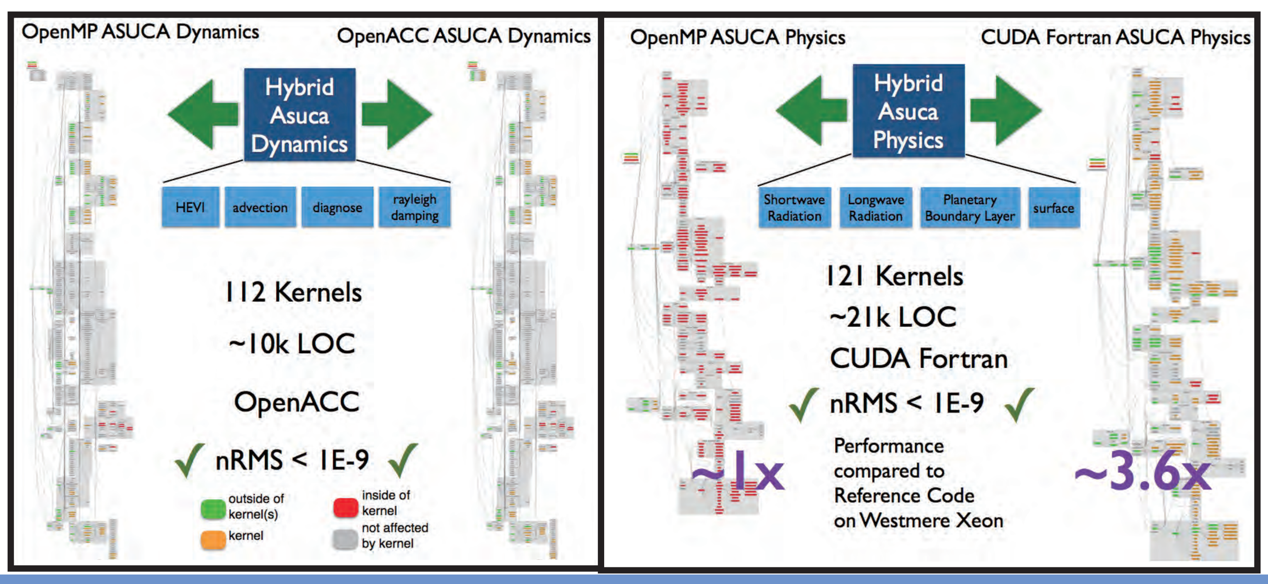
GPU parallelization (fine grained)




```fortran
subroutine calculate_all_columns(a, sums)
    real, intent(in), dimension(NZ, NX, NY) :: a
    real, intent(out), dimension(NX, NY) :: sums
    integer(4) :: i, j

    !$omp parallel do
    do j=1, NY
        do i=1, NX
            sum_column(a(:,i,j), sums(i,j))
        end do
    end do
end subroutine

subroutine sum_column(a, sum)
    real, intent(in), dimension(NZ) :: a
    real, intent(out) :: sum
    integer(4) :: k

    sum = 0.0
    do k=1, NZ
        sum = sum + a(k)
    end do
end subroutine
```

storage order according to centralized definition

```fortran
subroutine calculate_all_columns(a, sums)
    real, intent(in), dimension(NX, NY, NZ) :: a
    real, intent(out), dimension(NX, NY) :: sums

    sum_column(a, sums)
end subroutine

subroutine sum_column(a, sum)
    real, intent(in), dimension(NX, NY, NZ) :: a
    real, intent(out), dimension(NX, NY) :: sums
    integer(4) :: k, i, j

    !$acc kernels
    do j=1, NY
        do i=1, NX
            sum(i,j) = 0.0
            do k=1, NZ
                sum(i,j) = sum(i,j) + a(i,j,k)
            end do
        end do
    end do
    !$acc end kernels
end subroutine
```



Both the ASUCA dynamical core and physical processes have been ported using this approach. The physics run with no slowdown on CPU compared to the reference codebase, while achieving 3.6 speedup on GPU.

C++ Approach

- The proposed framework is designed for stencil applications with explicit time integration running on regular structured grids.
- The framework supports execution on NVIDIA's GPUs and CPU.
- The framework is written in the C++ language and CUDA and can be used in the user code developed in the C++ language.
  --> Improving portability of both framework and user code
- To perform stencil computations on grids, the programmer only defines C++ functions that update a grid point, which is applied to entire grids by the framework.

Stencil Computation with the Framework

- User-written function (C++ functor) that updates a grid point
- ArrayIndex3D represents the coordinate of the point where this function is applied.

Diffusion computation

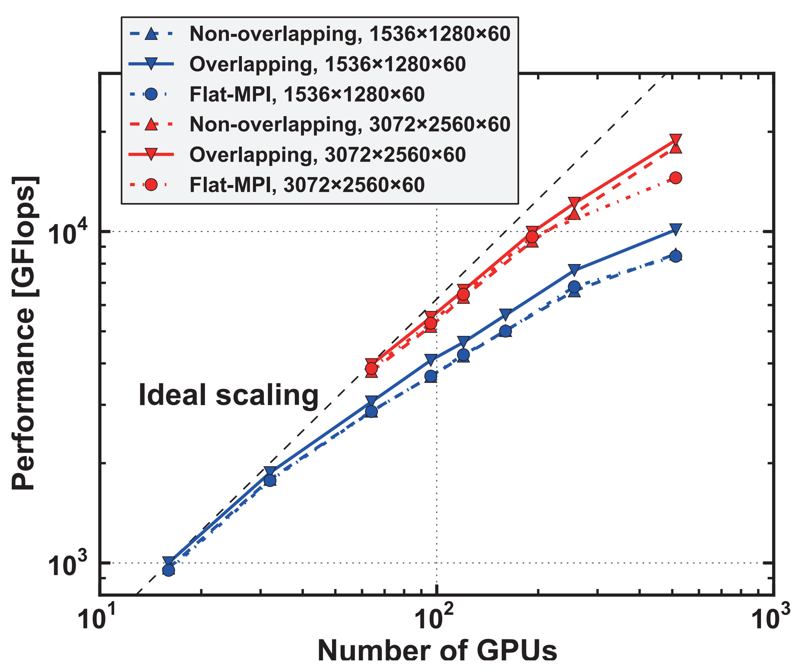
```cpp
struct Diffusion3d {
  __host__ __device__
  void operator()(const ArrayIndex3D &idx,
    float ce, float cw, float cn, float cs, float ct, float cb, float cc,
    const float *f, float *fn) {
    fn[idx.ix()] = cc*f[idx.ix()] + ce*f[idx.ix<1,0,0>()] + cw*f[idx.ix<-1,0,0>()]
                          + cn*f[idx.ix<0,1,0>()] + cs*f[idx.ix<0,-1,0>()]
                          + ct*f[idx.ix<0,0,1>()] + cb*f[idx.ix<0,0,-1>()];
}};
```

- The functor is executed over all grid points by Loop3D provided by the framework.

```cpp
Loop3D loop3d(nx+2*mgnx, mgnx, mgnx, ny+2*mgny, mgny, mgny, nz+2*mgnz, mgnz, mgnz);
loop3d.run(Diffusion3d(), ce, cw, cn, cs, ct, cb, cc, f, fn);
```

User-written function   Parameters are provided to the user-written function.

Strong Scaling results of Framework-based ASUCA



TSUBAME 2.5 supercomputer at the Tokyo Institute of Technology

- Total 4224 NVIDIA Tesla K20X GPUs
- Each node of TSUBAME 2.5
  - 3 Tesla K20X GPUs attached to the PCI Express bus 2.0 × 16 (8 GB/s)
  - 2 sockets of the Intel CPU Xeon X5670(Westmere-EP) 2.93 GHz 6-core
  - 2 QDR InfiniBand

Reference: T. Shimokawabe, T. Aoki and N. Onodera **"High-productivity Framework on GPU-rich Supercomputers for Operational Weather Prediction Code ASUCA,"** *in Proceedings of the 2014 ACM/IEEE conference on Supercomputing (SC'14)* , New Orleans, LA, USA, Nov 2014.