介绍一个非结构网格CFD的特定域语言OP2库

李健

中国地质大学（武汉）海洋学院

2022. 09. 15

主要内容

- 特定域语言
- 非结构网格的一些概念
- OP2库介绍
- OP2 API介绍
- OP2的研究计划

- 我国超算“花样”太多、种类太多，超算软件的移植、优化成本太高。
- （解决超算软件移植优化问题）要想办法，比如建立一个编译优化平台。
- HPC、AI算力建在西部是合适的，数据中心搬到西部则不可行。
- 预计未来三五年后，计算机会把HPC、AI、大数据计算融合在一起。
- “算力网络”就是把很多机器连在一起做事，但这件事做起来很费劲。

郑纬民

（中国工程院院士、清华大学计算机系教授）

一、特定域语言

●软件开发者希望获得新功能，但又担心开发成本，如MPI+X（CUDA, OpenCL, SYCL, AVX/NEON/SVE）；

●不同的（新的）硬件架构，需要新的代码优化（后期维护成本高）

解决？

从并行化实施的角度，高级别抽象，将数值算法和并行化方法分离；

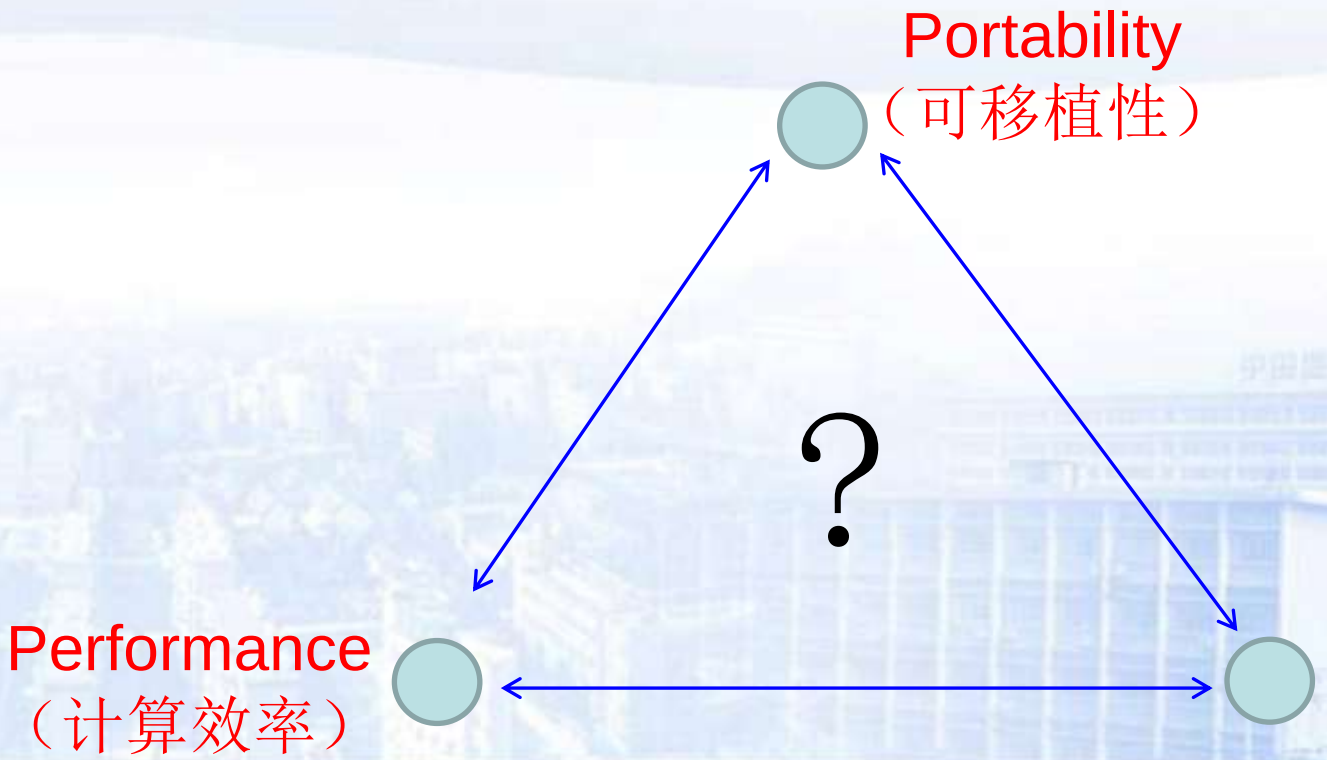
改变后端(backend)即实现最优化效率；

有利于代码长期维护。

软件开发的终极目标：抽象！

一、特定域语言(Domain-Specific Language)

定义： A language or active library that exploits domain knowledge for **productivity** and **efficiency**.



- 应用数学
- IT工程师
- 专业领域专家
- 全栈开发

一、特定域语言

CFD相关的DSL研究：

Liszt（Stanford，停止开发）

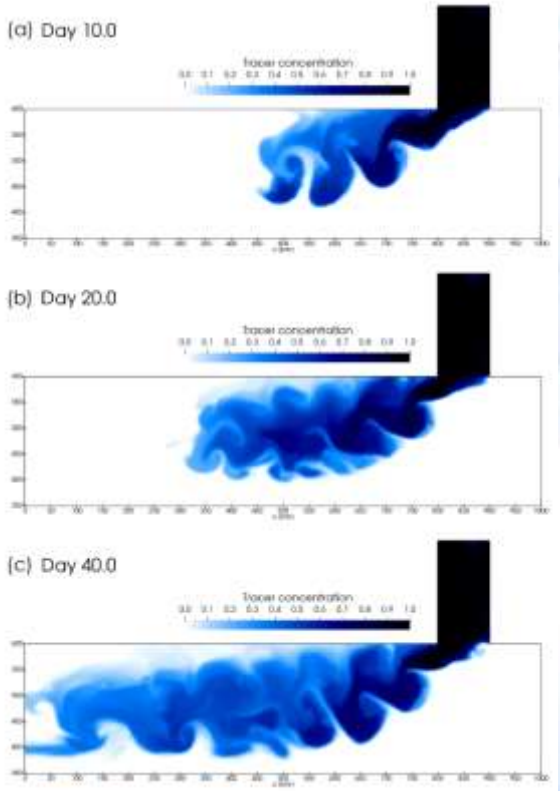
OP2（非结构网格）/OPS（多块结构网格）

FEniCS/Firedrake（pyOP2）（Python语言的DSL，DG法）

PyFR（使用Mako模板的DSL，FR法）

Devito（有限差分，结构网格）

GridTools，Psyclone（气象模型）



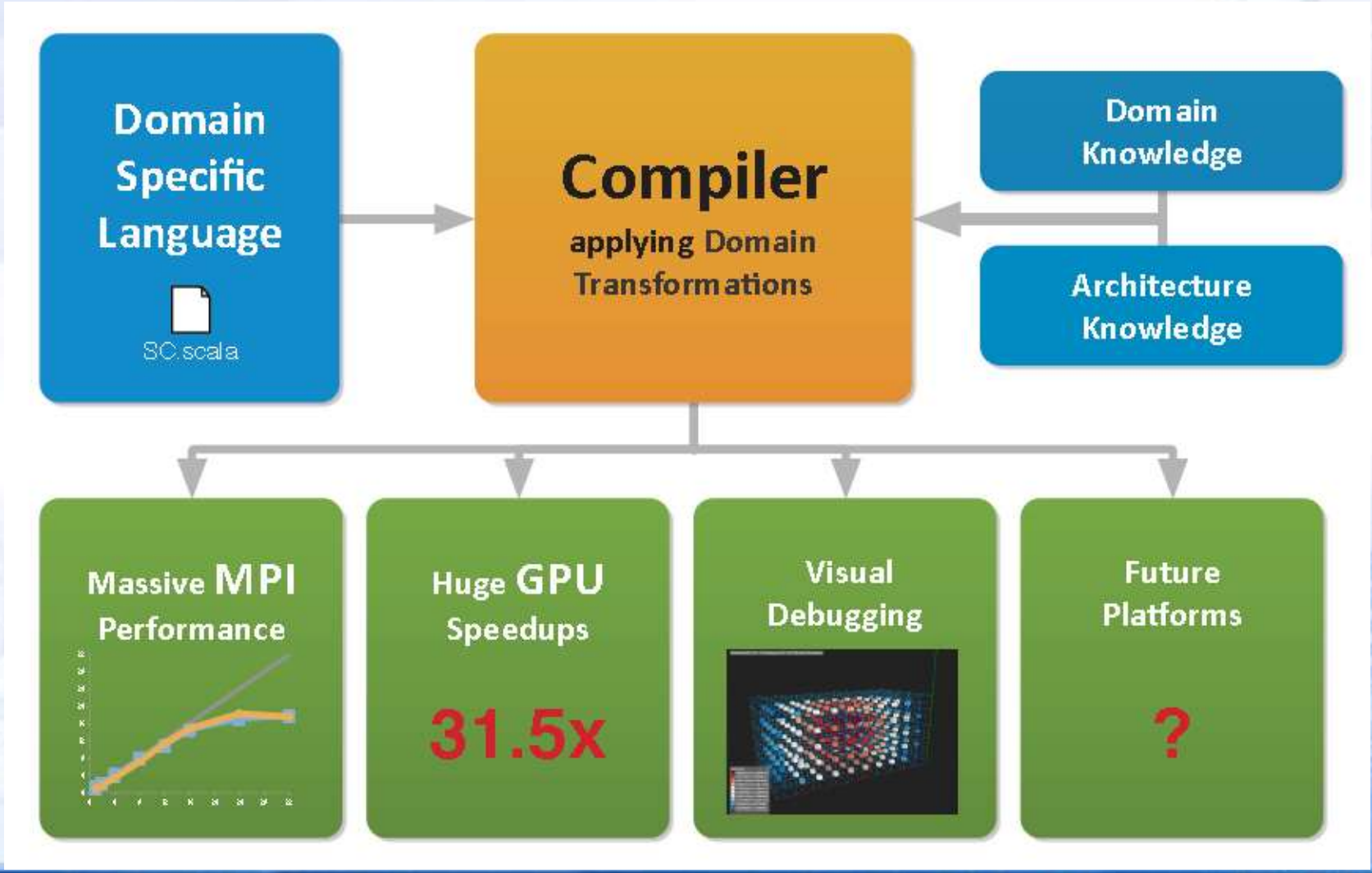
Firedrake项目可认为是fluidity项目的衍生品。

Meander（弯道环流），Python代码一共185行；

河床演变的Python代码一共763行。

Tuomas Kärnä et al. Thetis coastal ocean model: discontinuous Galerkin discretization for the three-dimensional hydrostatic equations. Geosci. Model Dev., 11, 4359-4382, 2018

一、特定域语言


Domain Specific Language
SC.scala
Compiler
applying Domain Transformations
Domain Knowledge
Architecture Knowledge
Massive MPI Performance
Huge GPU Speedups
31.5x
Visual Debugging
Future Platforms
?

一、特定域语言

4个优势：

- 实用性(Productivity)：将计算科学与计算机科学分离
- 移植性(Portability)：在不同硬件平台上都能运行
- 性能(Performance)：针对不同平台的性能高度优化
- 创新(Innovation)：在硬件架构和编程模型引入革新的技术

一、特定域语言

我们可以使用程序来分析：

- 实施区域分解算法
- 将源码快速转换为针对特定硬件架构的源码
- 针对cache和硬件架构，优化数据布局
- 根据特定架构的机器，选择存在冲突的并行化模式

一、特定域语言

对DSL的批评：

- 使用特殊的语法（我还得学习一种编程语言？）
- DSL语言设计的很差（缺乏普适性？）
- 很难与其他库或语言结合（DSL与非DSL部分的交互？）
- 没有开发环境（怎么调试？性能分析工具？）
- 构建完整的编译器成本很高

一、特定域语言

几种抽象化并行模型：

- 科学计算库，如：Trilinos 、PETSc
- C++模板(STL)，如：Kokkos，RAJA
- 特定域语言，如:FEniCS,firedrake,PyOP2， Exastencils,PSyclone (UK)
- 胶水语言(Python-C/C++交互）：PyFR，ExaHyPE-Engine

软件开发的终极目标：抽象(abstraction)

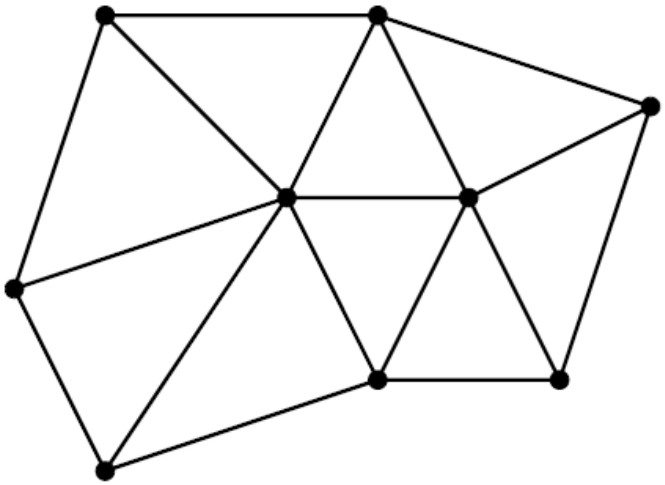
Istvan Z. Reguly, Gihan R. Mudalige.2020. Productivity, Performance, and Portability for Computational Fluid Dynamics Applications. Computers & Fluids

二、非结构网格的一些概念

非结构化网格广泛地应用于计算科学和工程领域的程序开发，包括CFD领域有。

非结构化网格使用连接信息定义网格拓扑关系。OP2库将非结构网格数据分为4类：

- Set (集: 单元，节点，边);
- Data on sets (流速, 水深, …);
- Connectivity (or mapping) between the sets;
- Operations over sets (读写, 增量循环,…)



这样所有的非结构网格(mesh)或图(graph)可使用OP2抽象库来定义。

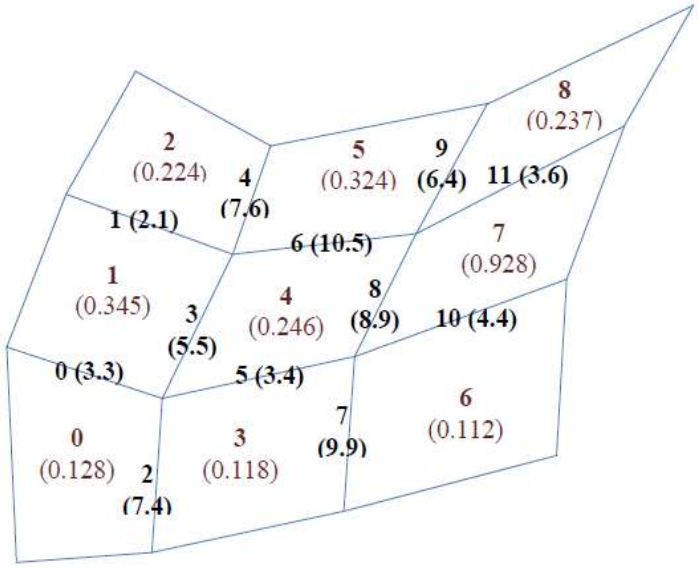
二、非结构网格的一些概念

一个set包括：nodes, edges, triangular/quadrilateral faces (cells, elements). 与这些set联系的是data（如：节点坐标和边上的权重）和set间的映射来定义一个set的单元如何与周围的单元连接的。

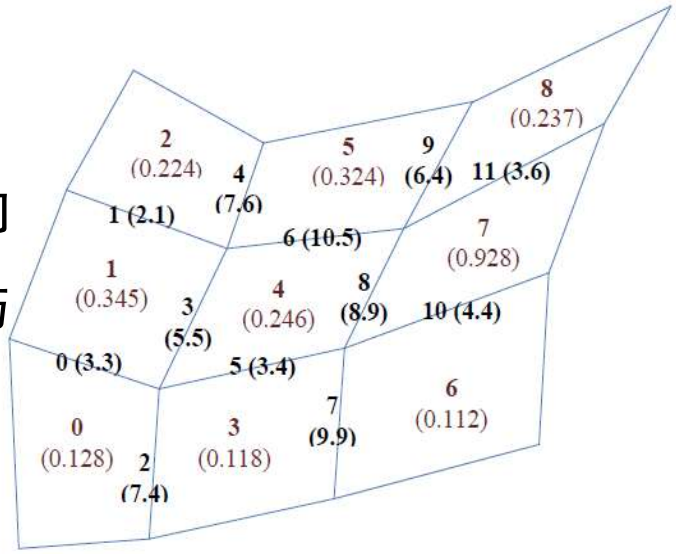
OP2库支持C/C++和FORTRAN编程的应用程序。如图1定义了3个数据集(set)：内部edges, cells (四边形)和边界edges.

对应地，OP2 API定义如下：

```
int nedges = 12; int ncells = 9; int nbedges = 12;
op_set edges = op_decl_set(nedges, "edges");
op_set cells = op_decl_set(ncells, "cells");
op_set bedges = op_decl_set(nbedges, "bedges");
```



二、非结构网格的一些概念



连接关系通过数据集之间的映射表来声明。仅考虑内部区域的边时，edge_to_cell整数型数组给出单元与内部边的连接关系信息：

```
int edge_to_cell[24] = {0,1, 1,2, 0,3, 1,4, 2,5, 3,4,
                        4,5, 3,6, 4,7, 5,8, 6,7, 7,8 };
op_map pecell = op_decl_map(edges, cells, 2,
                edge_to_cell, "edge_to_cell_map");
```

属于数据集edges的每个单元映射到在数据集cells中的2个单元上。op_map声明定义该映射，其中pecell维度是2，因此它的标记0和标记1映射到单元0和单元1；标记2和标记3映射到单元1和单元2，等等……

二、非结构网格的一些概念

当声明一个映射时，首先传递源(source)数据集(如edges)，然后是目标数据集（如cells）。然后，传递每个数据集的维度（如：2；pecell映射每条边到2个单元）。一旦数据集和连接关系定义好了，可将数据(data)与这些数据集联系起来。下面是单元和边上分别存储的数据（双精度）。注意到每个单元数据集上声明了一个双精度的数据，每个单元也可定义矢量的一串数据（如：每个单元上存储3个数据：XYZ坐标值）：

```
double cell_data[9] = {0.128, 0.345, 0.224, 0.118, 0.246,
                       0.324, 0.112, 0.928, 0.237};
double edge_data[12] = {3.3, 2.1, 7.4, 5.5, 7.6, 3.4,
                        10.5, 9.9, 8.9, 6.4, 4.4, 3.6};

op_dat dcells   = op_decl_dat(cells, 1, "double",
                              cell_data, "data_on_cells");
op_dat dedges   = op_decl_set(edges, 1, "double",
                              edge_data, "data_on_edges");
```

二、非结构网格的一些概念

非结构网格模型计算都可描述为数据集(sets)的操作，包括：

1. 执行数据集的循环计算
2. 通过映射关系访问数据
3. 实施一些计算，然后（可能通过映射关系）写回到数组去。

   如果通过映射，循环计算涉及间接访问，OP2将其标记为一次indirect loop；如果不是，就是direct loop。

OP2 API提供并行循环计算声明语法，这允许用户在这些循环计算中声明对数据集的并行方式。

二、非结构网格的一些概念

考虑下列的串行方式的循环计算，对图1网格中的每个内部边(edge)进行操作。

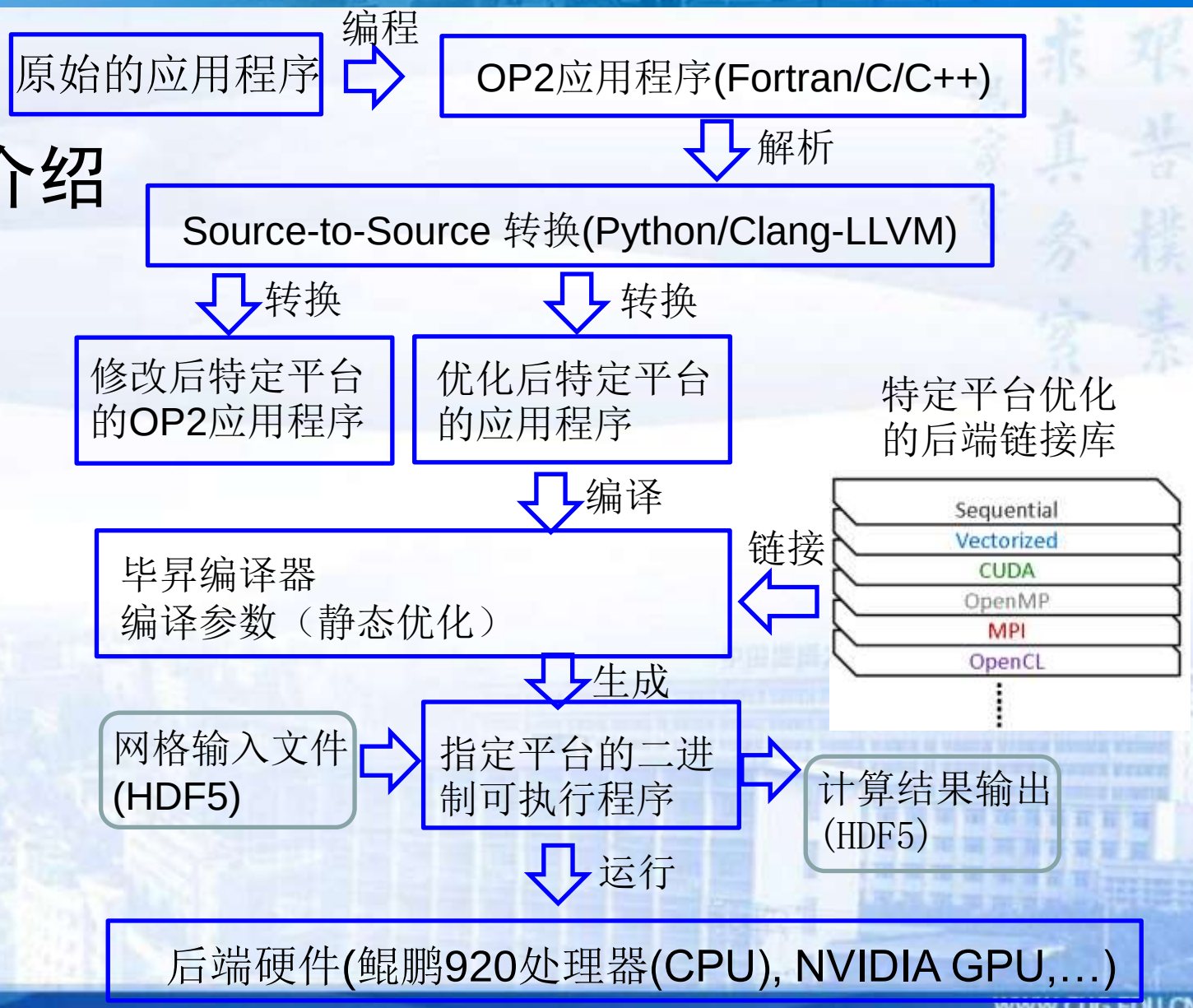
每个单元使用**连接到该单元的边及对应的相邻单元上**的数值更新单元存储数值：

```
void res_seq_loop(int nedges, int *edge_to_cell,
                  double *edge_data, double *cell_data)
{
  for (int i = 0; i<nedges; i++){
      cell_data[edge_to_cell[2*i]]   += edge_data[i];
      cell_data[edge_to_cell[2*i+1]] += edge_data[i];
  }
}
```

结合基本的核函数，开发者可使用OP2 API声明该循环计算如下：

```
void res(double* edge, double* cell0, double* cell1){
  *cell0 += *edge;
  *cell1 += *edge;
}
op_par_loop(res,"residual_calculation", edges,
   op_arg(dedges, -1, OP_ID,  1, "double", OP_READ),
   op_arg(dcells,  0, pecell, 1, "double", OP_INC),
   op_arg(dcells,  1, pecell, 1, "double", OP_INC));
```

三、OP2库介绍


原始的应用程序
编程
OP2应用程序(Fortran/C/C++)
解析
Source-to-Source 转换(Python/Clang-LLVM)
转换
转换
修改后特定平台的OP2应用程序
优化后特定平台的应用程序
特定平台优化的后端链接库
编译
链接
毕昇编译器
编译参数（静态优化）
Sequential
Vectorized
CUDA
OpenMP
MPI
OpenCL
生成
网格输入文件(HDF5)
指定平台的二进制可执行程序
计算结果输出(HDF5)
运行
后端硬件(鲲鹏920处理器(CPU), NVIDIA GPU,…)

三、OP2库介绍

（1）荷载均衡的区域分解(ParMETIS, PTSCOTCH)

（2）数据竞争：层级着色方法(较原子操作，效率提高)

（3）间接内存访问：非结构网格重编码，改善数据空间局部性

（4）SoA与AoS的自动转换

（5）多GPU并行的GPUDirect技术

（6）HDF5的I/O

（7）支持多种编译器

- OpenMP (OpenMP4.x-simd)
- CUDA
- MPI
- MPI + OpenMP (OpenMP4.x-simd)
- MPI + CUDA (GPUDirect)

三、OP2库介绍

●显格式（模拟无粘性流动：间断解问题）；

●隐格式（OP2很少用，因为隐格式的计算瓶颈是线性方程组求解，已有各类并行化的求解器）；

●OP2已经有PyOP2，是firedrake的一部分。

OP2只能使用静态非结构网格，不支持使用动态网格(AMR)；

OP2不支持CPU与GPU混合执行核函数。

四、OP2 API介绍

对单元做循环

Read NonZero（cells）

```
// 保存旧时刻的数值解
op_par_loop(save_soln, "save_soln", cells,
            op_arg_dat(p_q, -1, OP_ID, 4, "double", OP_READ),
            op_arg_dat(p_qold, -1, OP_ID, 4, "double", OP_WRITE));

// save.h 核函数
inline void save_soln(const double *q, double *qold) {
  for (int n = 0; n < 4; n++)
      qold[n] = q[n];
}
```

四、OP2 API介绍

save_soln核函数的OpenMP版本：

```
void op_par_loop_save_soln(char const *name, op_set set,
      op_arg arg0,
      op_arg arg1){
.....
 // execute plan
   #pragma omp parallel for
   for ( int thr=0; thr<nthreads; thr++ ){
     int start  = (set->size* thr)/nthreads;
     int finish = (set->size*(thr+1))/nthreads;
     for ( int n=start; n<finish; n++ ){
       save_soln(
         &((double*)arg0.data)[4*n],
         &((double*)arg1.data)[4*n]);
      }
    }
.....
```

四、OP2 API介绍

MPI + OpenMP版本核函数：

```
void op_par_loop_save_soln(char const *name, op_set set,
  op_arg arg0,
  op_arg arg1){
……
op_mpi_halo_exchanges(set, nargs, args);

// execute plan
#pragma omp parallel for
……

// combine reduction data
  op_mpi_set_dirtybit(nargs, args);

……
```

四、OP2 API介绍

CUDA版本的save_soln核函数：

```
__device__ void save_soln_gpu( const double *q, double *qold) {
  for (int n = 0; n < 4; n++)
    qold[n] = q[n];
}

// 调用CUDA kernel function
__global__ void op_cuda_save_soln(

//process set elements
  for ( int n=threadIdx.x+blockIdx.x*blockDim.x; n<set_size;
              n+=blockDim.x*gridDim.x ){

    //user-supplied kernel call
    save_soln_gpu(arg0+n*4,
            arg1+n*4);
```

四、OP2 API介绍　以airfoil mini-app为例介绍OP2 API

开发者只要将串行的legacy code写为OP2 API形式的串行代码，剩下的工作几乎都交给python转换脚本完成即可（也需要手动检查与调整）。

```
#include "op_seq.h“

// 并行执行循环计算的核函数
#include "adt_calc.h"
#include "bres_calc.h"
#include "res_calc.h"
#include "save_soln.h"
#include "update.h”

// 主程序
int main(int argc, char **argv) {
  // OP initialisation
  op_init(argc, argv, 2);
```

```
op_printf("initialising flow field \n");

char file[] = “new_grid.h5”;   // 计算网格文件

// declare sets, pointers, datasets and global constants
// 直接从HDF5文件读取相关参数
  op_set nodes = op_decl_set_hdf5(file, "nodes");
  op_set edges = op_decl_set_hdf5(file, "edges");
  op_set bedges = op_decl_set_hdf5(file, "bedges");
  op_set cells = op_decl_set_hdf5(file, "cells");

  op_map pedge = op_decl_map_hdf5(edges, nodes, 2, file, "pedge");
  op_map pecell = op_decl_map_hdf5(edges, cells, 2, file, "pecell");
  op_map pbedge = op_decl_map_hdf5(bedges, nodes, 2, file, "pbedge");
  op_map pbecell = op_decl_map_hdf5(bedges, cells, 1, file, "pbecell");
  op_map pcell = op_decl_map_hdf5(cells, nodes, 4, file, "pcell");
```

```
// CFD 模型的一些常数参数
op_get_const_hdf5("gam", 1, "double", (char *)&gam, "new_grid.h5");
op_get_const_hdf5("gm1", 1, "double", (char *)&gm1, "new_grid.h5");
……
op_decl_const(1, "double", &gam);
op_decl_const(1, "double", &gm1);
……

op_diagnostic_output();  // 输出关于sets, mapping和datasets的诊断信息

op_dump_to_hdf5(“new_grid_out.h5”); //再输出到h5文件，可以用h5diff工具，看一下读取HDF5文件对不对

// 非结构网格的区域分解，可以使用PTSCOTCH和ParMETIS，有geom, kway和geomkway三种算法
op_partition("PTSCOTCH", "KWAY", edges, pecell, p_x);

if (renumber) op_renumber(pecell);  // 区域分解后的网格编号调整，增加数据局部性
```

```
int g_ncell = op_get_size(cells);

op_timers(&cpu_t1, &wall_t1);

// 时间层推进计算
 niter = 1000;
 for (int iter = 1; iter <= niter; iter++) {

 //  save old flow solution
    op_par_loop(save_soln, "save_soln", cells,
                op_arg_dat(p_q, -1, OP_ID, 4, "double", OP_READ),
                op_arg_dat(p_qold, -1, OP_ID, 4, "double", OP_WRITE));

。 。 。 。 。 。


 op_exit();
```

四、OP2 API介绍

从以上DSL编程我们可以看出其难点在于：

1）开发者要了解非结构网格的概念和存储；

2）开发者要了解数值算法；

3）开发者要了解并行机制（MPI+X）。

关键就是：开发者要能写出正确的用 OP2 API编写的串行的CFD代码。

五、OP2的研究计划

- 在线可视化
- 使用外部求解器（隐格式CFD应用）
- Just-in-runtime编译
- 混合CPU与GPU执行
- FORTRAN语言的转换
- CPU基于任务的异步并行优化（HPX）
- GPU集群的异步并行优化（HPXCL）

五、OP2的研究计划

以OP2为测试平台的异步并行模式优化研究

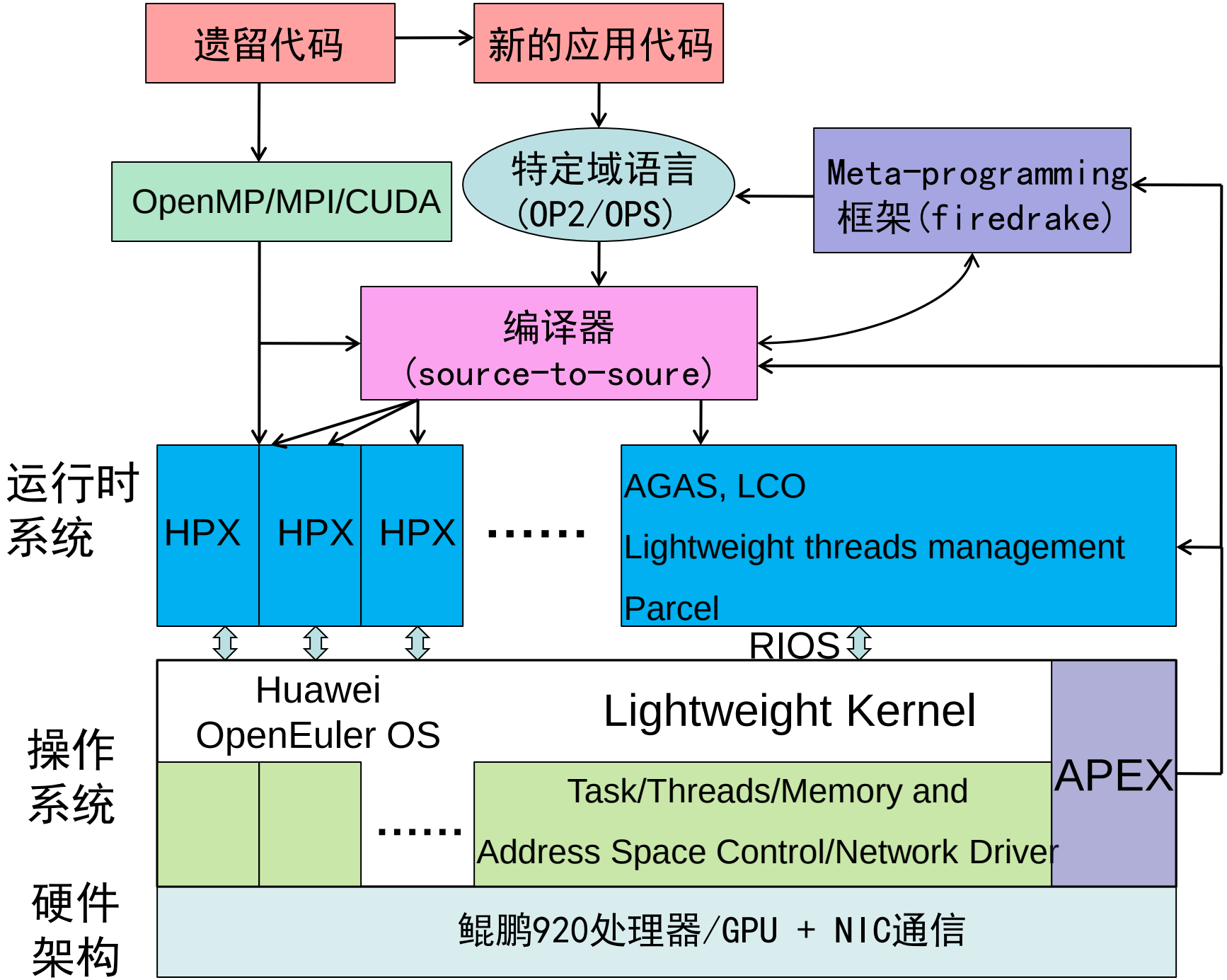
HPX其实就是将传统的异步计算思想，以新的C++编程方法和异步思想，抽象统一，可以以MPI，CUDA，。。。为后端，最低程度的侵入原始代码。 HPX概念举例：

Locality：执行的同步域，类似于SMP机器上的一个NUMA域或者集群中的一个计算节点；

Component：是一个C++类（包含server/client），可以远程访问。

Active Global Address Space (AGAS)：全局地址空间。

Parcel：HPX中通信数据的component


遗留代码
新的应用代码
OpenMP/MPI/CUDA
特定域语言
(OP2/OPS)
Meta-programming
框架(firedrake)
编译器
(source-to-soure)
运行时
系统
HPX
HPX
HPX
.....
AGAS, LCO
Lightweight threads management
Parcel
RIOS
操作
系统
Huawei
OpenEuler OS
Lightweight Kernel
APEX
.....
Task/Threads/Memory and
Address Space Control/Network Driver
硬件
架构
鲲鹏920处理器/GPU + NIC通信