My_container_tst

2024年2月27日

1. **设计一个容器，能够读取模拟前自行输入的各类参数，并以多叉树的方式进行存储和组织，基本需求分析如下。**

1.1 STL容器一般包含哪些基本方法：

vector容器的API主要包含：赋值、拷贝、插入（头插、尾插、中间插）、删除（头删、尾删、中间删）、函数重载（删除、插入函数）、容量、清空、判断是否为空、重指定大小、返回头元素、返回尾元素、返回中间某元素、查找某元素的位置、查找某位置的元素、修改某位置的元素、替换某元素为、返回某元素的数量、输出元素等。

deque（双端队列）容器的API主要包括（除上述以外）：头插、头删、无容量capacity、

list（列表【双向循环链表】）容器的API主要包括（除上述意外）：元素互换等

1.2 本容器需要包含的方法：

（1） 基本方法：添加节点，节点赋值，获取指定索引的节点值，获取指定节点的子节点，删除节点，修改节点，更新树。

（2）文件读取方法：从指定的YAML文件中加载指定的参数树，解析YAML文件并构建参数树，递归解析YAML节点，将YAML节点值转化为参数树的节点。

2. **该容器是用来存储有限元输入的参数的，其内部存储数据结构是多叉树类型，可以使用标准容器。**
   那么现在主要任务是明确需要输入哪些参数，各个参数都是什么数据类型。

2.1 是否需要根据具体问题确定不同的参数，即物理场数据

（1）热传导问题

（2）电磁学问题

（3）结构力学问题

（4）流体力学问题

2.2 是否需要根据不同子域和影像区划分确定不同的参数

2.3 几何信息、拓扑信息是否需要设定为参数

2.4 数值计算方法为FEM、FVM

2.5 在《项目策划书》中，Domain为时域，而Region为空间域，Parameter类（控制参数模块）定义在Domain中，

图 1：非结构框架基础层次结构

（1）Domain需要通过指针管理下面所有的Region，通过Region的名字进行索引；

（2）FieldField集合指针，如cell、face、node等，上面可能会存储一些速度场数据；

（3）MatrixMatrix集合指针，定义在cell上的U方程的系数矩阵；

（4）Patch集合指针，patchType是Patch的分类，如cell、face、node等，patchName一般定义成邻近进程号；

（5）Equation集合指针：equName代表方程名字，如速度U方程，压力P方程等；

（6）存储Region之间的通信对；

（7）Domain内需要封装一系列方法，包括创建通信对createCommPair()，创建方程，通过键值对获取相应数据，启动求解等。

2.6 在《项目策划书》中，Region是在一个空间域内具有相同计算流程的网格集合，Region中定义了额外的类，其中：

（1）所有Region内定义有内部网格单元类Mesh和物理边界网格单元类Boundary，网格单元Mesh类存储网格拓扑以及格点坐标等网格信息，物理边界网格存储在其派生类Boundary中；

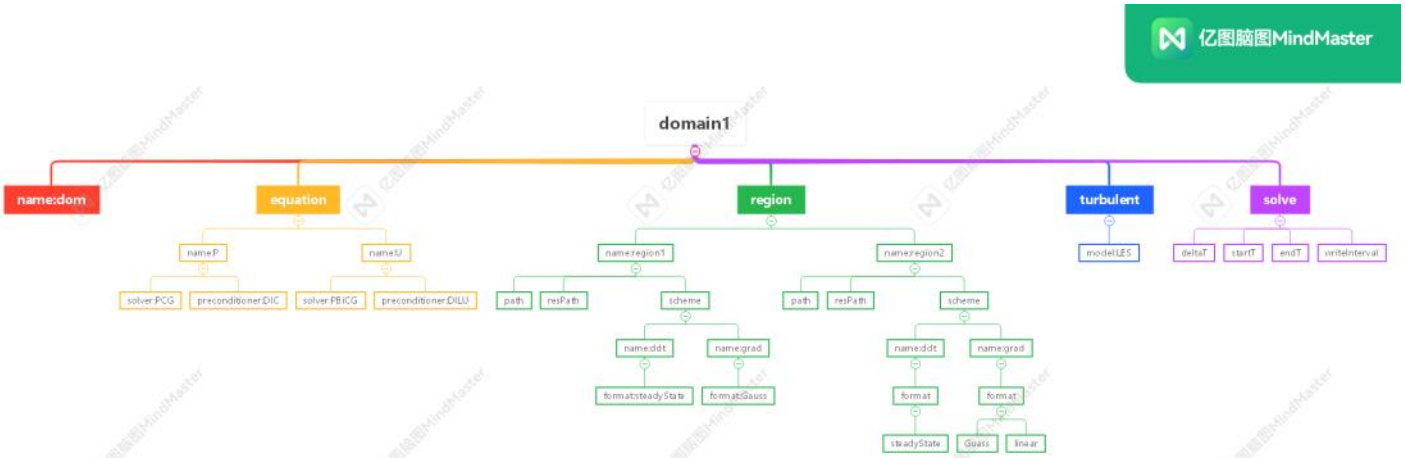
（2）由于Region平铺在不同进程上，进行Region内网格分块与负载均衡后，不同进程间存在进程边界，进程边界网格在程序初始化时将从内部网格单元类Mesh中剥离出来存储在通信拓扑类Patch中；

（3）Region内部流场信息由流场类Field进行管理；

（4）对于Region内网格的几何信息，例如网格面面积、法向量，网格单元体积、体心坐标等几何信息存储在网格信息类MeshInfo中；

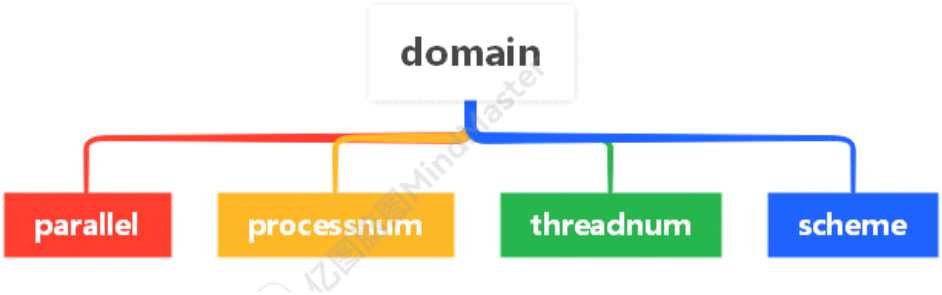
（5）系数矩阵信息存储在Matrix类中，该类包含行列坐标，流场信息等；

2.7 在《项目策划书》中，Parameter类是用来读取配置参数的，本框架采用YAML语言，根据YAML控制参数文件格式，可得以下树形数据结构：

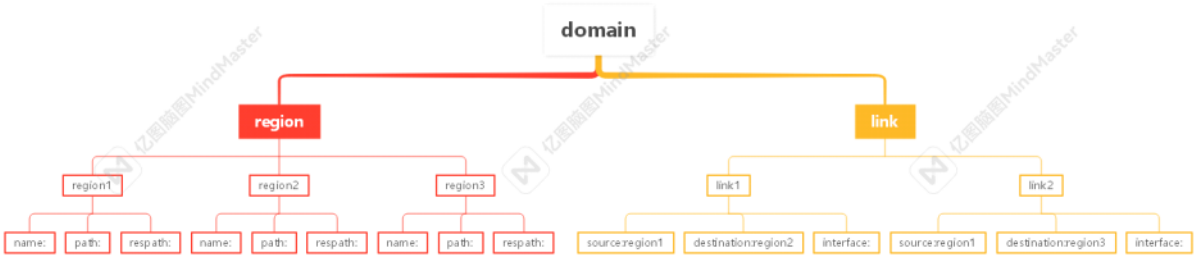


2.8 在已有的HSF代码中，和Region相关的参数（只包含.yaml文件）还包括以下几项：

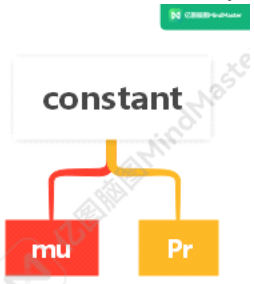
2.8.1 运行时的系统参数（config_system.yaml）



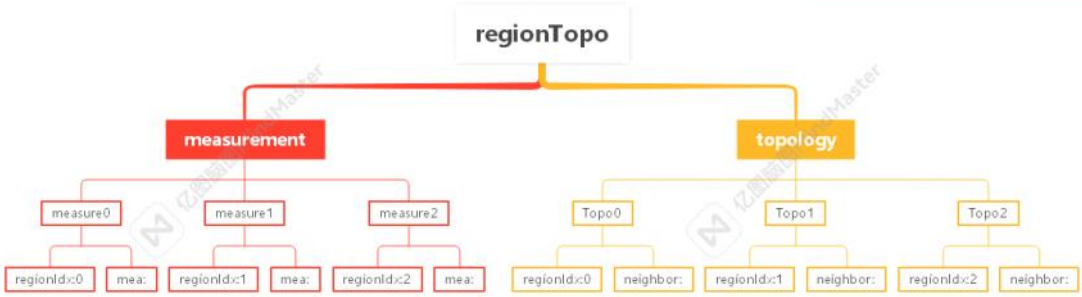
2.8.2 网格片拓扑相关信息（config_topo.yaml）



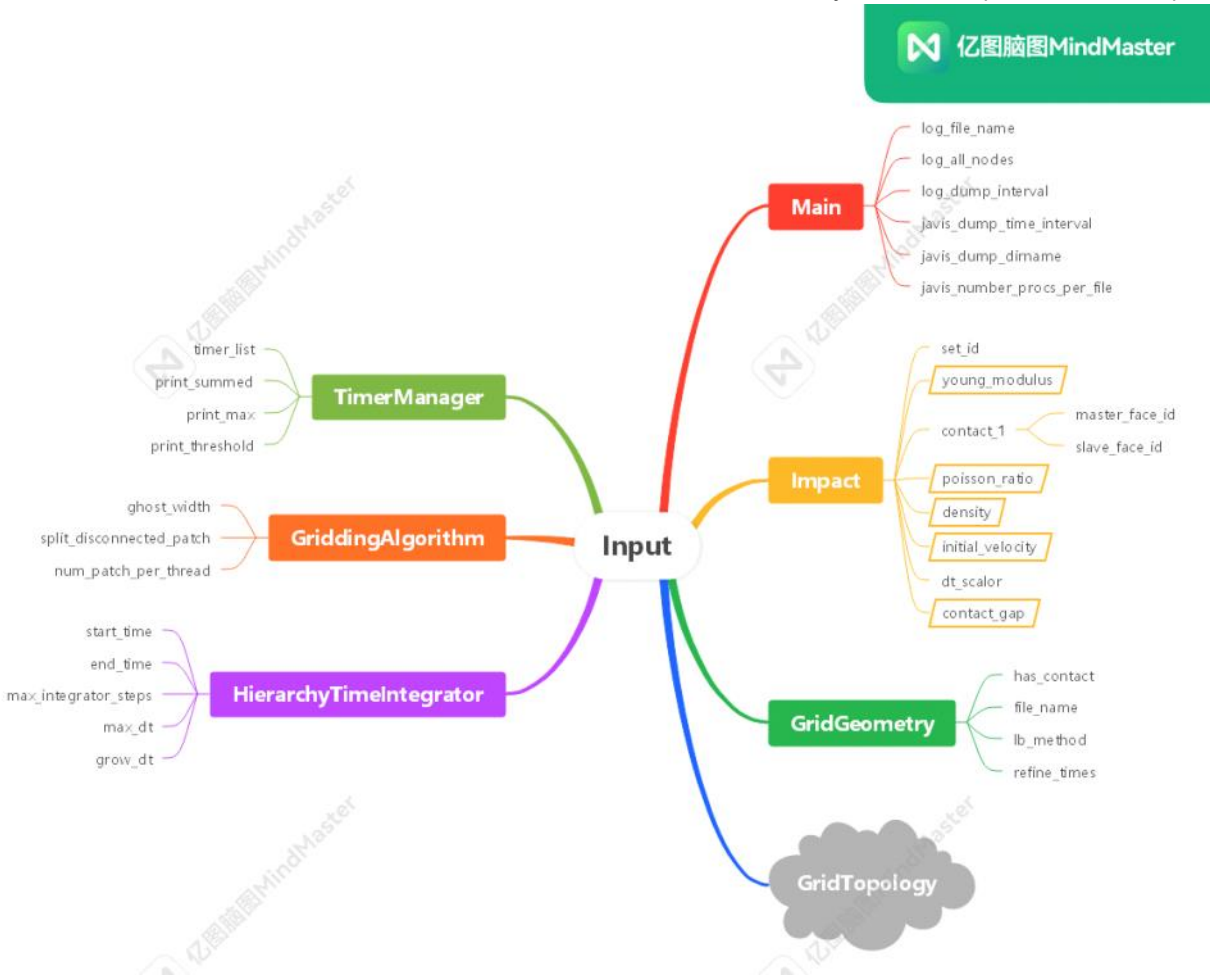
2.8.3 常数信息（constant.yaml）



2.8.4 regionTopo.yaml



2.9 在实际模拟的示例中，用户往往需要输入额外的参数来解决具体执行的案例，此部分参考jaumin的examples中接触例子的Input。



3容器实际结构和所包含的功能

3.1 容器（container）的数据结构

整体思路：设计一个以多叉树（N-ray tree）为通用数据结构的容器，能够读取六个不同的yaml文件，而每个yaml文件包含不同的内容。

（1）由于上述六个yaml文件的数据组织架构类似，均是明显的多叉树形式，其共同特点如下：

①树中需要存储多种类型的数据类型，包括整型、浮点数、字符型和布尔值；

②参数分类比较多，如果用树型数据结构来描述就是，节点的度比较大，但是树的高度/深度不大；

③每个节点下的子树高度不统一。

根据上述描述，我们需要自定义一个多叉树结构，用来灵活存储多种数据类型。

（2）为了解决以上问题，目前提出以下方案：

①使用`std::variant`模板类用于存储不同类型的数据。

```
std::variant<int, double, bool, std::string> value;
```

②自定义树的节点，每个节点可以包含一个`std::variant`类型的数据，和一个子节点列表。其中该节点的子节点列表计划使用`std::map`容器来存储子节点。

③采用已给的PARAFILE示例中的st_tree库，这是一个第三方的开源库，同时兼容STL接口。需要使用特定的C++标准（如C++ 11、14、17）进行构建。

设计容器中包含6种不同的子类，分别对应上述6个yaml文件，6种子容器分别命名状况如下：

3.2 解析YAML文件

整体思路：使用一个YAML解析库，计划使用yaml-cpp/yaml.h开源库，需要通过cmake进行构建和编译。

读取yaml文件格式内容：yamlfile_name["name"].as<string>()

yaml-cpp/yaml.h

3.3 容器类设计

设计一个容器类，需要包含六个'YamlNode'实例，分别对应六个不同的YAML文件。提供方法加载、查询和操作文件。

```
class YamlContainer {
private:
    std::map<std::string, std::shared_ptr<YamlNode>> files;
public:
    void loadFile(const std::string& name, const std::string& filename) {
        auto rootNode = std::make_shared<YamlNode>();
        parseYamlFile(filename, rootNode);
        files[name] = rootNode;
    }
    // 提供查询和操作YamlNode的方法
};
```

3.4 容器的使用

（1）使用容器进行加载文件，判断文件类型、读取文件内容结构以创建对应的数据结构类型。

（2）读取文件内容，对树中的各个节点赋值。

（3）使用容器进行查询或修改操作。

3.5 容器读取工程数据库中的内容

在数值模拟中，除了mesh文件中的数据有专门的读取接口以外，我们往往需要自定义大量的物性参数，这个定义的往往是在实例过程中的输入文件中。

我们可能会输入的工程数据库主要包括以下几类：

（1）材料数据库：主要包含各种工程材料的物理化学性质，如材料的密度分布、力学性质、热传导性质、电性质和材料模型。

（2）流体数据库：如流体的物理性质（粘度、密度等）、湍流模型。

（3）边界条件和初始条件数据库：存储以给定的边界条件和初始条件。

（4）实验数据优化参数数据库：存储实测模型的数据集，用于修正或者对比验证模拟结果。

目前由于暂未确定此类工程文件以何种方式存储（以什么数据格式存储，存储在何种文件格式中），故暂时不考虑如何读取和组织此类数据。