2025 年度中国青年科技创新

“揭榜挂帅”擂台赛

设计报告

选 题 名 称：	CQ-16 面向地球系统科学的海量网格类数据的分布式文件系统设计
推报学校名称：	武汉理工大学
申 报 者 姓 名：	冒长祥
申 报 者 电 话：	13064958978
指导老师姓名：	杜亚娟 解庆
参赛成员姓名：	祝玮、程玲、陈嘉尚、王宏基、胡俊耀、周倩郁、陈朕宇、薛凯琪

提交日期：2025 年 8 月 15 日

摘要

在地球系统科学领域，高分辨率网格数据的快速增长为数据管理和访问带来了重大挑战。传统分布式文件系统存在无法有效处理大规模自描述数据格式（如 GRIB 和 HDF5）、高 I/O 开销、局部数据获取效率低下和延迟问题。针对这一问题，本研究提出了一种面向地球系统科学海量网格数据设计的分布式文件系统解决方案。该系统基于开源 JuiceFS 框架，结合 PostgreSQL 数据库实现高级元数据管理，并部署于可扩展的阿里云基础设施上。

系统的核心改进有三个方面：一是扩展元数据架构以够捕获语义信息，从而加速数据索引与访问；二是设计结构化数据库以高效检索和解析 HDF5 文件；三是开发一套完整的操作库，涵盖网格场读取、写入、空间裁剪和插值等关键功能。通过对真实气象数据集的严格测试，该系统实现了数据访问延迟的大幅降低，在处理大型 HDF5 文件时，效率提升可达数十乃至上百倍，显著提高了数据检索和处理的效率。这项工作不仅为地球系统科学中的关键数据管理挑战提供了可选解决方案和思考方向，更为未来基于云的数据管理基础设施的进步奠定了基础。

关键词：分布式文件系统 语义元数据 网格数据 地球系统科学

目 录

表目录

图目录

1 项目背景与问题陈述

地球系统科学在应对全球气候变化、极端天气事件预测、海洋生态保护以及环境风险评估等重大科学与社会问题中发挥着核心作用，对观测、模拟与数据分析的精度和时效性提出了前所未有的挑战。近年来，地面站网、气象雷达、卫星遥感等多源观测手段迅速发展，高性能计算与数值预报模型不断进步，使全球大气、海洋、陆地、冰冻圈和生物圈的监测与模拟分辨率大幅提升。网格类科学数据的空间分辨率已达到公里级，时间分辨率可达小时级甚至更高，单文件体量最大可达 10GB，单一数据中心的在线存储量突破 100PB，并仍在以 PB 级速度增长。这类数据通常以 GRIB 或 HDF5 等国际通用自描述格式存储，将数十到上百个多要素、多层次的网格场压缩在单一文件中，并内嵌完整的时空坐标、物理量纲、层级结构等元数据信息，配套的原生开发包提供局部数据提取、空间裁剪、插值等功能。然而，随着数据规模、访问频率与并发需求的急剧增加，传统依赖单机解析的方式在分布式环境下无法充分利用计算与存储资源，导致局部访问延迟高、I/O 开销大、吞吐率低，已成为制约地球系统科学研究和业务化应用的瓶颈。

国际上，欧洲中期天气预报中心（ECMWF）、美国国家海洋和大气管理局（NOAA）、美国航空航天局地球观测系统数据与信息系统（NASA GES DISC）等机构广泛使用并行文件系统（Lustre、GPFS）或云对象存储（AWS S3、Google Cloud Storage）管理大规模气象和气候数据，这些系统在文件级并行 I/O 上表现优异，但在文件内部的字段级或块级随机访问上仍依赖 NetCDF/HDF5 API、wgrib2、h5py 等外部工具进行解析，尤其在面向大量小范围局部数据请求时效率显著下降。近年来，部分研究尝试将科学数据格式的自描述信息直接融入存储系统元数据层，如 SciDB、TileDB、Open Data Cube 等，通过语义驱动的索引加速局部访问，但这些方案大多依赖自研数据引擎，部署迁移成本高，且难以与现有主流分布式文件系统无缝对接。

在国内，中国气象局及多家科研院所、高性能计算中心探索了基于 Hadoop、Ceph 等平台的分布式存储，并针对 GRIB/HDF 开发并行读取与区域裁剪工具，但普遍存在三个不足：其一，元数据体系仅覆盖文件级信息，缺乏对文件内部时空子集的直接索引能力；其二，局部访问往往需全文件解析或依赖临时中间文件，造成访问延迟高、计算资源浪费；其三，高层访问接口与底层解析工具分离，科研人员需在多种工具和环境中切换，增加了使用与维护成本。

在地球系统科学对数据实时性、精细化和可共享性要求日益严苛的背景下，我们亟需一种能在通用分布式文件系统上扩展元数据的方案，将 GRIB/HDF5

的语义信息全面融入，实现字段和块级的精准定位与按需访问。同时，应配套开发统一的高性能函数库与 API，覆盖网格场读取、写入、空间裁剪、插值等常用操作，替代原生开发包的低效调用，实现接口统一、访问加速和易维护性兼备的分布式数据管理方案。这不仅为本次设计提供了解决方案，也为未来基于云计算和高性能计算平台的科学数据管理提供了具有通用性、可扩展性与可移植性的技术路径，对推动我国在气象、海洋、环境等领域的数据基础设施建设具有重要战略意义。

2 需求分析

针对上述背景与问题，本项目旨在设计并实现一套面向地球系统科学海量网格类数据的定制化分布式文件系统，以满足大规模、高并发、低延迟的科学数据存储与访问需求。

在功能层面，系统需能够在不少于三个节点的分布式集群环境中稳定运行，支持 HDF5 等国际通用自描述数据格式的无缝接入，并能够解析其内部的物理量纲、层级结构等语义信息，实现字段级和块级的精准索引与快速寻址。这一功能直接突破现有分布式文件系统仅能在文件级定位数据的局限，使用户能够按需加载局部数据块，避免全文件解析带来的 I/O 瓶颈。

在元数据管理方面，系统在传统文件路径、权限、大小等基础信息之外，扩展了对存储科学数据格式的内部结构描述，包括变量的时空范围、分辨率、维度组合、物理含义及单位等，为语义驱动的检索与访问提供坚实的索引基础。同时，为降低科研人员对原生数据格式开发包的依赖，提高跨平台、跨语言的可用性，系统应提供统一的高性能函数库与访问接口，覆盖网格场读取、写入、空间裁剪与空间插值等核心功能，确保这些操作能够直接在分布式环境中高效执行，并与元数据索引机制紧密结合，最大限度减少数据传输和重复计算的开销（表 1）。

在性能目标上，系统设计需保证局部数据访问延迟较传统方案显著降低，实现至少数倍的检索速度提升，并能在高并发访问条件下保持稳定的吞吐率和响应时间。同时，系统应具备良好的横向扩展能力，以适应数据规模和计算资源的持续增长。除此之外，系统还需具备容错与恢复能力，确保在节点故障、网络波动等情况下依然能够保证数据的一致性与可用性。

综上，需求分析明确了本项目不仅需要在技术架构上实现分布式部署、元数据体系扩展和高性能函数库集成的有机结合，还需在性能指标和系统健壮性上全面满足地球系统科学中大规模、高频率、多样化数据访问的实际需求，为后续的系统设计提供了明确的功能与性能导向。

表 1 函数库说明

函数	说明
网格场读取	与数据库及其他函数配合支持读取指定数据
网格场写入	支持将模拟或观测网格数据写入系统
空间裁剪	提供按经纬度范围提取子区域的能力
空间插值	支持不同分辨率或坐标系统间的空间重采样

3 系统设计

本系统的设计以满足地球系统科学海量网格类数据的高效存储、管理与访问需求为核心目标，结合赛题提出的功能与性能指标，构建了兼具通用性与针对性的分布式文件系统架构。设计过程中遵循“兼容性、可扩展性与高性能并重”的原则，在保证与现有科研数据格式（HDF5）和工作流平滑对接的同时，通过扩展元数据体系和优化访问路径，实现字段级、块级的精确定位与按需访问，从而有效解决传统分布式存储系统在科学数据处理中的语义理解不足、局部访问低效、接口分散等问题。

系统的总体思路是在通用分布式文件系统的存储能力基础上，深度融合科学数据格式的自描述信息，将数据的物理布局与语义索引结合起来，实现跨格式、跨平台的统一访问接口与高性能计算支持。通过引入可扩展的元数据服务、灵活的索引寻址机制和高效的函数库，系统不仅提升了单次访问的速度，也增强了在高并发、大规模数据处理场景下的稳定性与可扩展性。整体设计注重模块化与解耦性，使得各功能模块能够独立演进，同时保持系统架构的整体协调性和可维护性，为后续功能拓展与性能优化奠定基础。

3.1总体设计

系统由四个核心模块组成：元数据服务模块、分布式系统模块、索引与寻址模块及函数库与访问接口模块，系统总体架构如图 1。设计遵循“数据与元数据分离、语义与物理映射结合”的核心理念，构建了由基础设施层、数据存储层、应用层、用户交互层的四层整体架构。底层采用 JuiceFS 作为分布式文件系统，结合阿里云 OSS 提供高可用、可扩展的对象存储能力；中间层由增

强型元数据服务支撑，利用 PostgreSQL 存储和管理 HDF5 等科学数据格式的字段级与块级结构化信息，包括变量的时空范围、分辨率、维度组合、物理含义和单位等，并建立语义到物理位置的映射，通过统一的高性能函数库，提供网格场读取、写入、空间裁剪、空间插值等核心功能；在上层构建用户友好的可视化操作界面，用户可通过 Web 端直观地进行操作。系统通过统一的 API 接口提供服务，有效屏蔽底层数据存储的细节与格式差异，实现对复杂数据处理过程的透明化支持。

为提升局部访问性能，系统在索引与寻址模块中引入基于偏移量和数据块位置的快速定位机制，结合底层分片策略，使用户能够在不解析整文件的情况下直接提取所需变量。同时，通过热点缓存加速高频访问，减少重复 I/O 开销。架构设计同时强调扩展性与可靠性，通过多节点部署和冗余存储策略，确保在节点故障、网络波动等情况下数据仍能保持一致性与可用性。该分层与模块化的总体设计，系统不仅满足赛题提出的功能与性能目标，也为地球系统科学领域的大规模数据处理提供了可推广的技术路径。

在典型的使用场景中，科研人员或业务系统可以通过系统提供的统一 API 发起数据访问请求，请求中可包含变量名称、时空范围、空间分辨率等语义化参数。系统首先在元数据服务中解析请求参数，并利用 PostgreSQL 数据库定位数据集路径/数据块位置；然后使用路径通过 JuiceFS 从相应数据块中提取目标数据集；最后根据用户所需服务执行必要的空间裁剪、插值等处理，将结果以标准科学数据格式返回给用户。该流程在隐藏底层数据格式和存储细节的同时，实现了语义驱动的高性能数据访问，保证了系统在大规模高并发环境下的稳定性与响应速度。

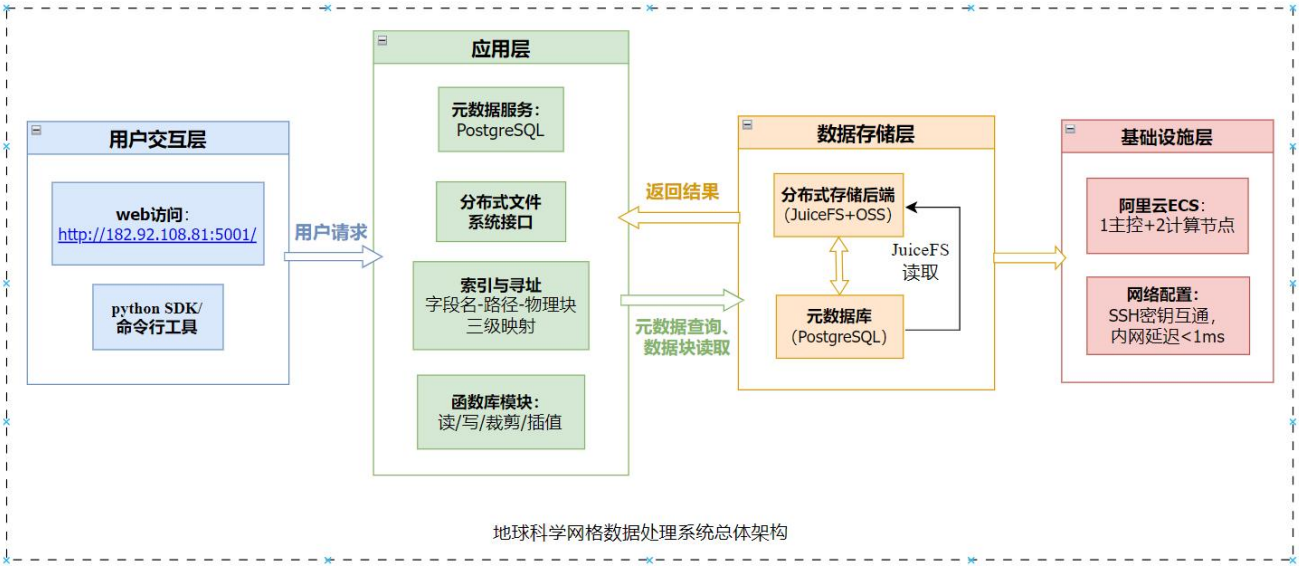

图 1 系统整体架构图

3.2关键模块设计

3.2.1 元数据体系扩展

传统分布式文件系统的元数据通常仅包含文件路径、大小、权限、时间戳及块位置等基础信息，无法满足对地球系统科学中网格类数据进行高效访问的需求。为此，本项目设计了一种面向网格类自描述数据格式的增强型元数据体系，将格式内嵌的语义信息结构化提取，并作为系统元数据的一部分进行统一管理。元数据扩展包括以下关键内容：

- 路径与数据集标识信息：对 HDF5 文件内部的逻辑路径进行解析，记录各数据集或维度变量在文件系统中的全路径标识，作为逻辑索引的主键。
- 物理量语义信息：提取每个变量的名称、长名称、单位、数据类型、所属维度结构等信息，用于描述物理意义和科学变量特征。
- 维度与层级结构信息：包括每个变量所依赖的维度组合、维度大小及其与坐标轴的映射关系，支持多变量、多时间、多层级的逻辑组织与解析。
- 数据块映射关系：将逻辑变量在多维数组中的位置映射到物理文件中的具体偏移地址（GRIB）、块大小、存储布局等信息，为局部读取提供基础。

该元数据体系的引入，使系统具备理解网格类数据内部结构、支持细粒度访问控制与数据裁剪处理的能力，是本项目实现智能化、高性能分布式访问的关键基础。

3.2.2 数据库设计

3.2.2.1 设计目标

为满足对地球系统科学中 HDF5 等自描述数据格式文件的结构化管理需求，系统设计了一套关系型数据库，用于持久化存储 HDF5 文件的层级结构（Group、Dataset）及其元信息（Attributes），以支持高效的查询与可视化访问。

设计目标包括：

- 支持多文件、复杂嵌套结构的统一管理；
- 支持属性类型多样性（字符串、整数、数组等）；
- 保留完整路径信息，支持数据的层级导航；
- 支持快速索引与查询操作。

3.2.2.2 数据库总体结构

数据库采用 PostgreSQL 实现，共设计五张核心表，如表 2 所示。

值得注意的是，对于 GRIB 等非结构化二进制文件，字段（变量）并不具备 HDF5 式的清晰路径组织结构，而是通过文件内的字节偏移量（byte offset）来定位数据块。想要支持这种格式，只需要在 datasets 表中新增 byte_offset 字段，用于记录每个变量或字段在文件中的起始位置，便于配套的数据读取工具快速定位和访问目标变量。

表 2 数据库表

表名	说明
hdf5_files	存储每个导入的文件基本信息
hdf5_groups	记录文件中所有 Group 的结构信息
hdf5_datasets	存储 Dataset（变量）的形状、压缩等元数据
hdf5_attributes	存储挂载在 Group/Dataset 上的属性信息
索引结构	提高按路径快速查询的效率

3.2.2.3 表结构设计详解

- hdf5_files：文件信息表

用于记录每个导入的 HDF5 文件的路径、导入时间等信息。

```
CREATE TABLE hdf5_files (
    id SERIAL PRIMARY KEY,              -- 主键，自增
    file_name TEXT NOT NULL,            -- 文件名（不含路径）
    file_path TEXT,                     -- 文件完整路径
    created_at TIMESTAMP DEFAULT now() -- 导入时间
);
```

图 1 数据库-文件信息表

支持同名文件位于不同路径；可用于溯源数据来源。

- hdf5_groups：Group 分组信息表

记录每个文件中所有的 Group 节点，保留其路径与父路径信息，支持还原树结构。

```sql
CREATE TABLE hdf5_groups (
    id SERIAL PRIMARY KEY,
    file_id INTEGER REFERENCES hdf5_files(id) ON DELETE CASCADE, -- 所属文件
    name TEXT NOT NULL,              -- 分组名（例如 "FS"）
    full_path TEXT NOT NULL,         -- 绝对路径（例如 "/FS/SubGroup"）
    parent_path TEXT,                -- 父路径（例如 "/FS"）
    created_at TIMESTAMP DEFAULT now()
);
```

图 2 数据库-分组信息表

full_path 字段支持按路径唯一定位；可递归构建 Group 树。

- hdf5_datasets：数据集（变量）信息表

用于描述所有 Dataset 的结构信息，包括维度、数据类型、压缩算法等。

```sql
CREATE TABLE hdf5_datasets (
    id SERIAL PRIMARY KEY,
    file_id INTEGER REFERENCES hdf5_files(id) ON DELETE CASCADE, -- 所属文件
    name TEXT NOT NULL,              -- 数据集名
    full_path TEXT NOT NULL,         -- 绝对路径（如 "/FS/zFactorMeasured"）
    parent_path TEXT,                -- 所在组路径（如 "/FS"）

    shape TEXT,                      -- 维度形状（如 "(20, 40, 2)"）
    dtype TEXT,                      -- 数据类型（如 "float32"、"int8"）
    chunks TEXT,                     -- Chunk 块大小（如 "(10, 10)"，若无为 NULL）
    compression TEXT,                -- 压缩算法（如 "gzip"，若无为 NULL）
    compression_opts TEXT,           -- 压缩参数（如压缩等级 "4"）
    fill_value TEXT,                 -- 缺失值填充值（如 "-99"）

    created_at TIMESTAMP DEFAULT now()
);
```

图 3 数据库-数据集信息表

shape 存储为字符串（如 “(30, 20, 2)”）；支持记录 gzip 压缩信息及填充值（_FillValue）；可用于数据尺寸与格式分析。

- hdf5_attributes：对象属性信息表

存储附加在 Group 或 Dataset 上的所有属性，包括标量与数组、字符串与整数类型。

```sql
CREATE TABLE hdf5_attributes (
    id SERIAL PRIMARY KEY,
    file_id INTEGER REFERENCES hdf5_files(id) ON DELETE CASCADE, -- 所属文件

    parent_path TEXT NOT NULL,          -- 属性所属对象路径（如 "/FS/flagSensor"）
    name TEXT NOT NULL,                 -- 属性名称（如 "CodeMissingValue"）

    value TEXT,                         -- 属性值（统一存为字符串）
    is_array BOOLEAN,                   -- 是否为数组（true=Value[50]，false=Scalar）
    array_length INTEGER,               -- 数组长度（若为 scalar，则为 NULL）

    dtype TEXT,                         -- 属性数据类型（如 "string", "int8"）
    str_length INTEGER,                 -- 字符串类型长度（如 4、12），非字符串为 NULL
    padding TEXT,                       -- 字符串填充类型（如 "H5T_STR_NULLTERM"）
    cset TEXT,                          -- 字符集（如 "H5T_CSET_ASCII"）

    created_at TIMESTAMP DEFAULT now()
);
```

图 4 数据库-对象属性信息表

支持复杂嵌套属性如 CodeMissingValue, FillValue, DimensionNames；所有值统一转换为 TEXT，便于解析；类型描述（dtype, padding, cset）方便重建或导出。

- 性能优化索引

```sql
CREATE INDEX idx_dataset_path ON hdf5_datasets(full_path);       -- 加速按路径查询 dataset
CREATE INDEX idx_attr_parent ON hdf5_attributes(parent_path);    -- 加速按路径查询属性
CREATE INDEX idx_group_path ON hdf5_groups(full_path);           -- 加速分组查找
```

图 5 数据库-性能优化索引

3.2.2.4 设计效果

本数据库系统针对 HDF5 格式的高维网格类数据，设计了一套可扩展、可检索、可复现的元数据模型，旨在解决原始 HDF5 文件结构复杂、访问效率低、数据分布难以管理等问题。

在设计层面，系统通过拆分出 File、Group、Dataset 与 Attribute 四类核心实体，重构了 HDF5 的逻辑层级关系，并捕捉了包括数据类型、压缩方式、数组结构、缺省值、字符集等丰富元信息，确保在不读取大文件内容的前提下，支持精确、高效的元数据查询。

数据库设计遵循高内聚、低耦合原则，将元信息管理与数据本体存储解耦，底层依托 JuiceFS 分布式文件系统实现高性能的数据访问。系统还通过对路径

字段建立索引优化查询性能，为上层服务（如 Web 可视化、数据接口、分布式计算调度等）提供基础支撑。

3.2.2.5 数据库使用

a. 用户发起查询请求（例如查找名称包含 flagSensor 的变量）；
b. 数据库在 hdf5_datasets 或 hdf5_attributes 表中检索匹配项；
c. 返回匹配对象的 full_path 和所属 file_id；
d. 在 hdf5_files 表中查找对应的 HDF5 文件路径（位于 JuiceFS 文件系统中）；
e. 上层程序根据路径信息，使用 h5py 等库访问 JuiceFS 挂载目录，提取对应变量或属性的值；
f. 返回结构化数据或用于后续分析、可视化；
g. 该设计流程实现了“元信息快速定位 → 文件系统路径解析 → 分布式数据提取”的高效闭环，是支撑海量科学网格数据可复用、可分析的重要基础设施组件。

3.2.3 局部寻址机制

局部寻址机制旨在解决在海量网格类数据场景中，用户仅需访问文件中部分变量、部分时空区域数据时所面临的访问效率低下的问题。传统文件系统缺乏对数据内部结构的理解，常以文件为最小访问单位，导致大量无效 I/O。本系统设计的局部寻址机制基于扩展元数据与索引结构，实现了从语义查询条件到物理数据块的快速映射，显著提高了数据访问效率与精度。

3.2.3.1 机制实现思路

a. 文件上传阶段：自动提取其内部结构信息，并构建文件内容与分布式存储路径之间的映射索引。系统会解析每个上传文件中的所有数据集路径、名称、维度结构、数据类型与基本属性，并连同文件名、逻辑路径一并写入 PostgreSQL 数据库，形成结构化的元数据索引表。
b. 数据使用阶段：用户仅需提供数据集名称或变量标识，系统即可通过数据库查询快速定位其对应的 HDF5 路径信息与存储位置，再基于该路径从 JuiceFS 分布式文件系统中读取对应的数据集。整个访问过程无需加载整文件或遍历结构，支持对指定变量、特定时空范围内的数据实现精确定位与按需访问。

3.2.3.2 机制效果

该机制避免了用户显式管理 HDF5 文件结构与路径的复杂性，将路径解析与物理访问完全托管于系统内部。通过数据库中建立的映射关系，系统能够实现如下功能：

- 数据集名称到路径的映射查询；
- 路径到分布式存储中具体对象的定位；
- 对 HDF5 中局部数据集的按需访问与调度。

整体流程兼顾了结构理解、访问效率与系统解耦性，为大规模网格数据服务提供了强有力的底层支持。

3.2.4 分布式系统模块（JuiceFS + OSS 后端）

3.2.4.1 分布式系统模块架构

本模块基于“元数据与数据分离存储”的分布式架构——JuiceFS 文件系统与阿里云 OSS 作为后端存储相结合，构成了核心的数据存储与管理模块，架构图如图 6。

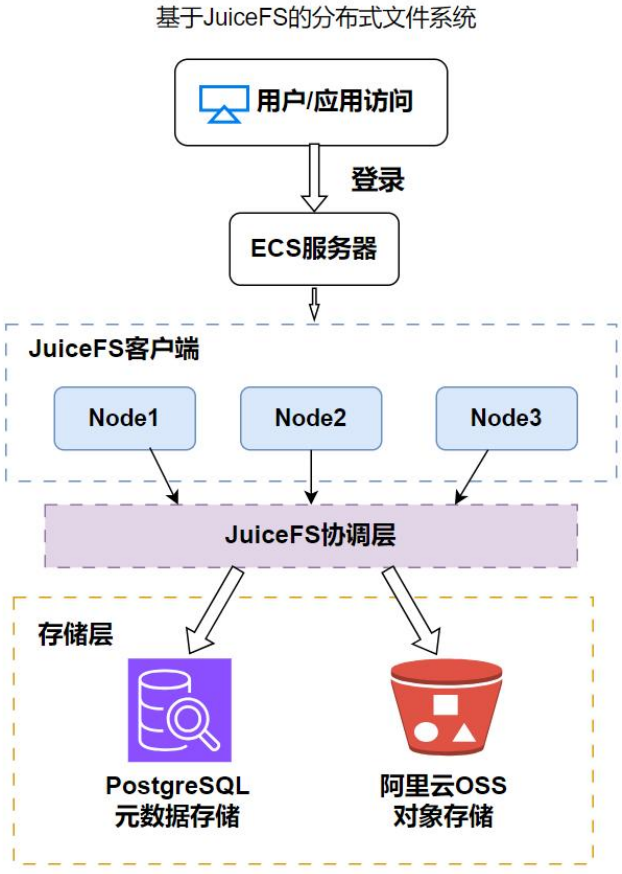

图 6 分布式系统架构图

该模块的主要职责是提供高性能、可扩展的文件存储服务，支撑上层应用和计算节点的读写需求。系统整体基于阿里云 ECS 实例部署，并以 PostgreSQL 数据库存储元数据。核心组件如下：

a. JuiceFS 客户端

JuiceFS 是目前服务较为完善的开源分布式文件系统，支持分离元数据与数据存储，其核心功能包括：

- 元数据管理：通过 PostgreSQL 数据库进行高效的元数据索引与一致性管理。
- 数据缓存：在本地或内存中对热数据进行缓存，提高文件访问效率。
- 多节点并发访问：多个 ECS 实例同时挂载 JuiceFS，实现数据共享和并发访问。
- 兼容 POSIX：应用程序可以以传统文件系统方式访问分布式存储，无需修改代码。

b. 阿里云 OSS 后端存储

OSS（Object Storage Service）作为后端对象存储，负责持久化存储文件的实际数据（如 HDF5 文件内容、业务数据等）。

在 OSS 中，文件以对象形式存储，文件被切割成块。从 OSS 的角度看，各个对象间无层级结构，但在 JuiceFS 的管理下，对象被有序组织。用户不需知道文件的存储细节，由 JuiceFS 进行统一调度和管理。具备以下特点：

- 弹性扩展：OSS 具备几乎无限的存储能力，适用于海量数据场景。
- 高可靠性与可用性：数据以多副本方式存储，保障数据持久性。
- 按需付费：降低初始成本，适合动态负载的分布式系统。
- 与 JuiceFS 解耦：JuiceFS 将文件数据分片后存入 OSS，实现计算与存储分离。

c. PostgreSQL

作为元数据存储，记录文件系统的元信息（文件路径、权限、大小、时间戳、与 OSS 数据的映射关系等），支持多节点并发读写时的元数据一致性。后续主要功能，如部分数据读取、空间裁剪、空间插值等，都依托于元数据数据库，大大提高数据访问效率。

d. 模块交互说明

a) 用户或应用程序在 ECS 上通过挂载点访问 JuiceFS 文件系统；

b) 文件的元数据信息（如目录结构、权限等）存储于 PostgreSQL 数据库中；

c) 实际文件内容被拆分为数据块并上传至阿里云 OSS；

d) 读取时，JuiceFS 会根据元数据从 OSS 拉取所需数据块，并可进行本地缓存。

3.2.4.2 实现过程

a. 环境搭建与配置

环境搭建全程在阿里云 ECS 上进行。

a) 部署 PostgreSQL 数据库，创建专用元数据库（juicefs），配置远程访问权限（开放 5432 端口、允许多节点 IP 连接），在三个节点都安装数据库客户端。

```
postgres=# CREATE USER juicefs WITH PASSWORD 'zwhhhhh';
CREATE ROLE
postgres=# CREATE DATABASE juicefs OWNER juicefs;
CREATE DATABASE
```

图 7 分布式-用户及数据库创建

b) 配置阿里云 OSS 存储桶（Bucket），创建访问密钥（AK/SK），开放 AliyunOSSFullAccess 权限，确保 JuiceFS 客户端有权限读写 OSS 数据。

c) 在 3 台 ECS 服务器安装 JuiceFS 客户端，通过 juicefs format 初始化文件系统（关联 OSS 与 PostgreSQL），并配置挂载点（如/mnt/jfs）实现分布式挂载。

代码如下：

```
#初始化
juicefs format \
--storage oss \
--bucket https://team-oss-jfs1.oss-cn-beijing.aliyuncs.com
\
--access-key LTAI5tFMFgBmkqjNViGnD9fQ \
--secret-key XRW3wCws9hN23zomLqxy3yyWVZy2Aq \
"postgres://juicefs:zwhhhhh@182.92.108.81:5432/juicefs?
sslmode=disable" \
mystor
#挂载
juicefs mount -d \
```

```
"postgres://juicefs:zwhhhhh@182.92.108.81:5432/juicefs?
sslmode=disable" \
/mnt/jfs
```

三个节点都挂载到/mnt/jfs 目录，JuiceFS 与本地文件系统实现了逻辑映射，用户可以像操作本地文件一样，对挂载点目录内的文件和目录进行创建、读取、写入、删除等操作。但实际上，这些操作最终会通过 JuiceFS 客户端，与对象存储（如 OSS）和元数据存储（如 PostgreSQL）进行交互。

经过以上操作，我们得到了三个节点的分布式系统，可以初步进行文件的写入和读取等操作。简单测试读写，如图所示。

```
[root@iZ2ze610tlbq1b72shz16kZ ~]# echo 'Hello JuiceFS!' > /mnt/jfs/test.txt
[root@iZ2ze610tlbq1b72shz16kZ ~]# cat /mnt/jfs/test.txt
Hello JuiceFS!
```

图 8 分布式-简单读写测试

b. 功能验证与测试

a) 多节点协同读写测试

在节点 1 创建文件，验证节点 2、3 能否实时读取、修改；删除节点 1 的文件，确认其他节点同步感知。各节点测试如图所示。

```
[root@iZ2ze610tlbq1b72shz16kZ jfs]# echo 'node1 test' > /mnt/jfs/extra.txt
[root@iZ2ze610tlbq1b72shz16kZ jfs]# cat extra.txt
node1 test
```

图 9 分布式-节点 1 创建文件 extra.txt

```
root@iZ2zej22gjcqb1bffw4s69Z:~# echo 'node2 test' >> /mnt/jfs/extra.txt
root@iZ2zej22gjcqb1bffw4s69Z:~# cat /mnt/jfs/extra.txt
node1 test
node2 test
root@iZ2zej22gjcqb1bffw4s69Z:~#
```

图 10 分布式-节点 2 修改文件 extra.txt

```
root@iZbp1e22cpjo9drve5e52lZ:~# rm /mnt/jfs/extra.txt
root@iZbp1e22cpjo9drve5e52lZ:~# ls /mnt/jfs
0929bbf3-9906-4544-9cd2-8aceed6e1f21.h5  af969b9b-88b9-46f7-82ef-79abe5fc879d.h5
610ef455-1c48-4cdb-a340-c639e120966f.h5  cacd4138-e319-47ae-b176-2bc956d5c808.h5
8cbcb2d4-f085-41f1-a513-19447c66dd9c.h5  extracted
a221.h5                                  test.txt
aed1e380-2ea7-4dff-a83e-c19ba687fd73.h5
root@iZbp1e22cpjo9drve5e52lZ:~#
```

图 11 分布式-节点 3 删除文件 extra.txt

b) 数据一致性验证

为验证系统在分布式环境下的数据一致性，本研究对 GB 级 HDF5 文件进行了完整性和断点续传测试。文件上传至节点 1 后，JuiceFS 利用 OSS 数据校

验保证内容完整，PostgreSQL 记录字段级与块级元数据，确保索引与存储数据一致。

上传完成后，不同节点通过统一接口访问文件，返回的文件大小、维度和数据块偏移量均一致，表明系统在多节点并发访问下能够维持数据同步和一致性。在模拟传输中断场景下，利用 JuiceFS 断点续传恢复上传后，文件哈希值及元数据均保持一致，进一步验证了系统在网络波动或节点故障情况下的数据完整性。

实验结果表明，JuiceFS 与 PostgreSQL 的联合机制能够有效保障海量网格数据的安全性和一致性，同时节点间查询结果完全一致（如图 12 图 13 图 14）。

```
[root@iZ2ze610tlbq1b72shz16kZ ~]# juicefs info /mnt/jfs/a221.h5
/mnt/jfs/a221.h5 :
  inode: 4
  files: 1
   dirs: 0
 length: 221.55 MiB (232314434 Bytes)
   size: 221.55 MiB (232316928 Bytes)
   path: /a221.h5
 objects:
+------------+--------------------------------+---------+--------+---------+-----------+
| chunkIndex |           objectName           |   size  | offset |  length |    pos    |
+------------+--------------------------------+---------+--------+---------+-----------+
|          0 | mystor/chunks/0/0/19_0_4194304 | 4194304 |      0 | 4194304 |         0 |
|          0 | mystor/chunks/0/0/19_1_4194304 | 4194304 |      0 | 4194304 |   4194304 |
|          0 | mystor/chunks/0/0/19_2_4194304 | 4194304 |      0 | 4194304 |   8388608 |
|          0 | mystor/chunks/0/0/19_3_4194304 | 4194304 |      0 | 4194304 |  12582912 |
|          0 | mystor/chunks/0/0/19_4_4194304 | 4194304 |      0 | 4194304 |  16777216 |
|          0 | mystor/chunks/0/0/19_5_4194304 | 4194304 |      0 | 4194304 |  20971520 |
|          0 | mystor/chunks/0/0/19_6_4194304 | 4194304 |      0 | 4194304 |  25165824 |
|          0 | mystor/chunks/0/0/19_7_2555904 | 2555904 |      0 | 2555904 |  29360128 |
|          0 | mystor/chunks/0/0/20_0_4194304 | 4194304 |      0 | 4194304 |  31916032 |
|          0 | mystor/chunks/0/0/20_1_4194304 | 4194304 |      0 | 4194304 |  36110336 |
|          0 | mystor/chunks/0/0/20_2_4194304 | 4194304 |      0 | 4194304 |  40304640 |
|          0 | mystor/chunks/0/0/20_3_4194304 | 4194304 |      0 | 4194304 |  44498944 |
|          0 | mystor/chunks/0/0/20_4_4194304 | 4194304 |      0 | 4194304 |  48693248 |
|          0 | mystor/chunks/0/0/20_5_3833856 | 3833856 |      0 | 3833856 |  52887552 |
|          0 | mystor/chunks/0/0/21_0_4194304 | 4194304 |      0 | 4194304 |  56721408 |
|          0 | mystor/chunks/0/0/21_1_4194304 | 4194304 |      0 | 4194304 |  60915712 |
|          0 | mystor/chunks/0/0/21_2_1998848 | 1998848 |      0 | 1998848 |  65110016 |
|          1 | mystor/chunks/0/0/22_0_4194304 | 4194304 |      0 | 4194304 |  67108864 |
|          1 | mystor/chunks/0/0/22_1_4194304 | 4194304 |      0 | 4194304 |  71303168 |
|          1 | mystor/chunks/0/0/22_2_4194304 | 4194304 |      0 | 4194304 |  75497472 |
|          1 | mystor/chunks/0/0/22_3_4194304 | 4194304 |      0 | 4194304 |  79691776 |
|          1 | mystor/chunks/0/0/22_4_4194304 | 4194304 |      0 | 4194304 |  83886080 |
|          1 | mystor/chunks/0/0/22_5_3571712 | 3571712 |      0 | 3571712 |  88080384 |
|          1 | mystor/chunks/0/0/23_0_4194304 | 4194304 |      0 | 4194304 |  91652096 |
|          1 | mystor/chunks/0/0/23_1_4194304 | 4194304 |      0 | 4194304 |  95846400 |
|          1 | mystor/chunks/0/0/23_2_4194304 | 4194304 |      0 | 4194304 | 100040704 |
```

图 12 分布式-节点 1 传入并查看文件信息

```
root@iZ2zej22gjcqblbffw4s69Z:~# juicefs info /mnt/jfs/a221.h5
/mnt/jfs/a221.h5 :
  inode: 4
  files: 1
   dirs: 0
 length: 221.55 MiB (232314434 Bytes)
   size: 221.55 MiB (232316928 Bytes)
   path: /a221.h5
 objects:
+------------+--------------------------------+---------+--------+---------+-----------+
| chunkIndex |           objectName           |   size  | offset |  length |    pos    |
+------------+--------------------------------+---------+--------+---------+-----------+
|          0 | mystor/chunks/0/0/19_0_4194304 | 4194304 |      0 | 4194304 |         0 |
|          0 | mystor/chunks/0/0/19_1_4194304 | 4194304 |      0 | 4194304 |   4194304 |
|          0 | mystor/chunks/0/0/19_2_4194304 | 4194304 |      0 | 4194304 |   8388608 |
|          0 | mystor/chunks/0/0/19_3_4194304 | 4194304 |      0 | 4194304 |  12582912 |
|          0 | mystor/chunks/0/0/19_4_4194304 | 4194304 |      0 | 4194304 |  16777216 |
|          0 | mystor/chunks/0/0/19_5_4194304 | 4194304 |      0 | 4194304 |  20971520 |
|          0 | mystor/chunks/0/0/19_6_4194304 | 4194304 |      0 | 4194304 |  25165824 |
|          0 | mystor/chunks/0/0/19_7_2555904 | 2555904 |      0 | 2555904 |  29360128 |
|          0 | mystor/chunks/0/0/20_0_4194304 | 4194304 |      0 | 4194304 |  31916032 |
|          0 | mystor/chunks/0/0/20_1_4194304 | 4194304 |      0 | 4194304 |  36110336 |
|          0 | mystor/chunks/0/0/20_2_4194304 | 4194304 |      0 | 4194304 |  40304640 |
|          0 | mystor/chunks/0/0/20_3_4194304 | 4194304 |      0 | 4194304 |  44498944 |
|          0 | mystor/chunks/0/0/20_4_4194304 | 4194304 |      0 | 4194304 |  48693248 |
|          0 | mystor/chunks/0/0/20_5_3833856 | 3833856 |      0 | 3833856 |  52887552 |
|          0 | mystor/chunks/0/0/21_0_4194304 | 4194304 |      0 | 4194304 |  56721408 |
|          0 | mystor/chunks/0/0/21_1_4194304 | 4194304 |      0 | 4194304 |  60915712 |
|          0 | mystor/chunks/0/0/21_2_1998848 | 1998848 |      0 | 1998848 |  65110016 |
|          1 | mystor/chunks/0/0/22_0_4194304 | 4194304 |      0 | 4194304 |  67108864 |
|          1 | mystor/chunks/0/0/22_1_4194304 | 4194304 |      0 | 4194304 |  71303168 |
|          1 | mystor/chunks/0/0/22_2_4194304 | 4194304 |      0 | 4194304 |  75497472 |
|          1 | mystor/chunks/0/0/22_3_4194304 | 4194304 |      0 | 4194304 |  79691776 |
|          1 | mystor/chunks/0/0/22_4_4194304 | 4194304 |      0 | 4194304 |  83886080 |
|          1 | mystor/chunks/0/0/22_5_3571712 | 3571712 |      0 | 3571712 |  88080384 |
|          1 | mystor/chunks/0/0/23_0_4194304 | 4194304 |      0 | 4194304 |  91652096 |
|          1 | mystor/chunks/0/0/23_1_4194304 | 4194304 |      0 | 4194304 |  95846400 |
```

图 13 分布式-节点 2 查看文件 信息

```
root@iZbp1e22cpjo9drve5e52lZ:~# juicefs info /mnt/jfs/a221.h5
/mnt/jfs/a221.h5 :
  inode: 4
  files: 1
   dirs: 0
 length: 221.55 MiB (232314434 Bytes)
   size: 221.55 MiB (232316928 Bytes)
   path: /a221.h5
 objects:
+------------+--------------------------------+---------+--------+---------+-----------+
| chunkIndex |           objectName           |   size  | offset |  length |    pos    |
+------------+--------------------------------+---------+--------+---------+-----------+
|          0 | mystor/chunks/0/0/19_0_4194304 | 4194304 |      0 | 4194304 |         0 |
|          0 | mystor/chunks/0/0/19_1_4194304 | 4194304 |      0 | 4194304 |   4194304 |
|          0 | mystor/chunks/0/0/19_2_4194304 | 4194304 |      0 | 4194304 |   8388608 |
|          0 | mystor/chunks/0/0/19_3_4194304 | 4194304 |      0 | 4194304 |  12582912 |
|          0 | mystor/chunks/0/0/19_4_4194304 | 4194304 |      0 | 4194304 |  16777216 |
|          0 | mystor/chunks/0/0/19_5_4194304 | 4194304 |      0 | 4194304 |  20971520 |
|          0 | mystor/chunks/0/0/19_6_4194304 | 4194304 |      0 | 4194304 |  25165824 |
|          0 | mystor/chunks/0/0/19_7_2555904 | 2555904 |      0 | 2555904 |  29360128 |
|          0 | mystor/chunks/0/0/20_0_4194304 | 4194304 |      0 | 4194304 |  31916032 |
|          0 | mystor/chunks/0/0/20_1_4194304 | 4194304 |      0 | 4194304 |  36110336 |
|          0 | mystor/chunks/0/0/20_2_4194304 | 4194304 |      0 | 4194304 |  40304640 |
|          0 | mystor/chunks/0/0/20_3_4194304 | 4194304 |      0 | 4194304 |  44498944 |
|          0 | mystor/chunks/0/0/20_4_4194304 | 4194304 |      0 | 4194304 |  48693248 |
|          0 | mystor/chunks/0/0/20_5_3833856 | 3833856 |      0 | 3833856 |  52887552 |
|          0 | mystor/chunks/0/0/21_0_4194304 | 4194304 |      0 | 4194304 |  56721408 |
|          0 | mystor/chunks/0/0/21_1_4194304 | 4194304 |      0 | 4194304 |  60915712 |
|          0 | mystor/chunks/0/0/21_2_1998848 | 1998848 |      0 | 1998848 |  65110016 |
|          1 | mystor/chunks/0/0/22_0_4194304 | 4194304 |      0 | 4194304 |  67108864 |
|          1 | mystor/chunks/0/0/22_1_4194304 | 4194304 |      0 | 4194304 |  71303168 |
|          1 | mystor/chunks/0/0/22_2_4194304 | 4194304 |      0 | 4194304 |  75497472 |
|          1 | mystor/chunks/0/0/22_3_4194304 | 4194304 |      0 | 4194304 |  79691776 |
|          1 | mystor/chunks/0/0/22_4_4194304 | 4194304 |      0 | 4194304 |  83886080 |
|          1 | mystor/chunks/0/0/22_5_3571712 | 3571712 |      0 | 3571712 |  88080384 |
|          1 | mystor/chunks/0/0/23_0_4194304 | 4194304 |      0 | 4194304 |  91652096 |
|          1 | mystor/chunks/0/0/23_1_4194304 | 4194304 |      0 | 4194304 |  95846400 |
|          1 | mystor/chunks/0/0/23_2_4194304 | 4194304 |      0 | 4194304 | 100040704 |
|          1 | mystor/chunks/0/0/23_3_4194304 | 4194304 |      0 | 4194304 | 104235008 |
```

图 14 分布式-节点 3 查看文件信息

c) 兼容性测试

将业务代码（如 HDF5 文件解析脚本）的文件路径指向 JuiceFS 挂载点，验证 POSIX 接口兼容性（如 cp/rm 命令、Python 文件操作函数等）。

```
[root@iZ2ze610tlbq1b72shz16kZ jfs]# echo "test content" > /mnt/jfs/test.txt
[root@iZ2ze610tlbq1b72shz16kZ jfs]# cp /mnt/jfs/test.txt /mnt/jfs/test_copy.txt
[root@iZ2ze610tlbq1b72shz16kZ jfs]# ls
0929bbf3-9906-4544-9cd2-8aceed6e1f21.h5  af969b9b-88b9-46f7-82ef-79abe5fc879d.h5
610ef455-1c48-4cdb-a340-c639e120966f.h5  cacd4138-e319-47ae-b176-2bc956d5c808.h5
8cbcb2d4-f085-41f1-a513-19447c66dd9c.h5  extracted
a221.h5                                  test_copy.txt
aed1e380-2ea7-4dff-a83e-c19ba687fd73.h5  test.txt
[root@iZ2ze610tlbq1b72shz16kZ jfs]# cat test_copy.txt
test content
```

图 15 分布式-创建、复制、打印文件

```
[root@iZ2ze610tlbq1b72shz16kZ jfs]# python3
Python 3.6.8 (default, Nov 14 2023, 16:29:52)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> with open("/mnt/jfs/python_test.txt", "w") as f:
...     f.write("Python write test\n")
...
18
[root@iZ2ze610tlbq1b72shz16kZ jfs]# cat /mnt/jfs/python_test.txt
Python write test
```

图 16 分布式-python 环境创建文件并写入

3.2.4.3 设计成果

a) 核心功能

- 构建了支持 3 节点协同访问的分布式文件系统，实现文件的跨节点实时共享（任何节点的文件操作可被其他节点秒级感知）。
- 完成业务代码适配：将 HDF5 文件解析、数据提取脚本的文件路径迁移至 JuiceFS 挂载点，实现分布式环境下的脚本正常运行（无需修改核心逻辑，仅调整路径）。
- 实现数据与元数据分离管理：文件数据安全存储于 OSS（冗余备份、抗丢失），元数据由 PostgreSQL 统一管理（支持事务与并发控制）。

b) 系统特性

- 高可用性：单节点故障不影响整体系统运行，剩余节点可正常读写；OSS 与 PostgreSQL 自身的高可用机制（如 OSS 数据多副本、PostgreSQL 主从）保障底层存储可靠性。

- 可扩展性：支持随时新增节点（只需安装客户端并挂载），扩展后系统总吞吐量线性提升；OSS 自动扩容特性满足数据量增长需求。
- 易用性：通过 POSIX 接口屏蔽底层分布式复杂性，用户/脚本操作 JuiceFS 如同操作本地文件系统，降低使用门槛。

c) 应用价值

- 解决了多节点数据共享难题：替代传统 NFS 或 Samba 等方案，提供更高的并发性能与扩展性。
- 支撑业务分布式部署：为后续扩展更多节点、实现更大规模的文件处理（如批量解析 HDF5 数据）奠定基础。简化了分布式存储架构，易于部署和维护；
- 可灵活接入云计算平台资源，提升系统整体弹性；
- 降低存储成本：利用 OSS 按需付费特性，避免本地磁盘容量浪费；元数据与数据分离架构减少 PostgreSQL 存储压力。

3.2.5 函数库设计

3.2.5.1 网格场写入

a. 技术架构与流程设计

以 parse_and_store_hdf5_metadata 为核心入口，联动数据库交互与 HDF5 解析，实现元数据的完整存储，架构图如下：

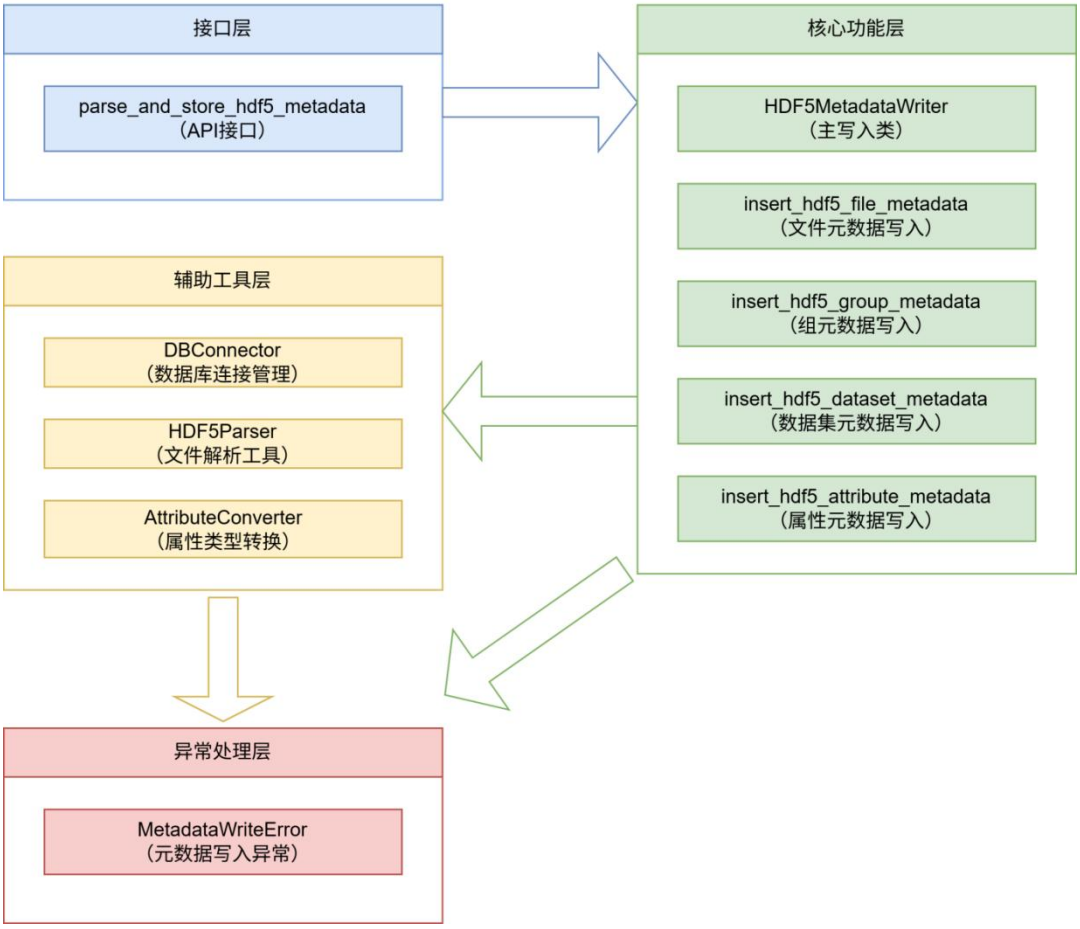

图 17 函数-网格场写入架构图

a) 数据库连接与事务控制：通过 psycopg2 建立与 PostgreSQL 的连接，采用事务机制（commit/rollback）确保元数据写入的原子性。

   i. HDF5 文件层级解析：利用 h5py.File.visititems 递归遍 HDF5 文件的所有对象（组/数据集），按“文件→组 / 数据集→属性”的层级提取信息：

   ii. 文件级：通过 insert_hdf5_file_metadata 函数将文件名、路径写入 hdf5_files 表，返回唯一 file_id 作为关联标识，实现“一文件一标识”的追踪机制。

   iii. 组级（Group）：对每个 h5py.Group 对象，解析其名称、完整路径(full_path）、父级（parent_path），通过 insert_hdf5_group_metadata 写入 hdf5_groups 表，同时记录与 file_id 的关联，保留 HDF5 的层级结构。

   iv. 数据集级（Dataset）：对每个 h5py.Dataset 对象，提取关键特征（形状 shape、数据类型 dtype、分块 chunks、压缩参数等），通过

insert_hdf5_dataset_metadata 写入 hdf5_datasets 表，实现数据集物理存储特征的结构化记录。

b. 属性（Attribute）的复杂类型适配

HDF5 属性支持字符串、字节、数组、numpy 数组等多样类型，insert_hdf5_attribute_metadata 函数通过以下策略实现兼容存储：

a) 类型标准化：对字节类型（bytes）尝试 UTF-8 解码，失败则转为 base64 编码字符串；对 numpy 数组、列表等可迭代对象，标记 is_array 并记录长度（array_length）。

b) 元数据增强：补充记录数据类型（dtype，如 numpy.float64）、字符串长度（str_length）、编码方式（cset，如 UTF-8/ASCII）等扩展信息，确保属性语义完整。

c) 关联存储：通过 file_id 和 parent_path（组或数据集的路径）关联属性所属对象，实现“属性 - 父对象”的精准映射。

c. 技术亮点

a) 层级映射完整性：严格保留 HDF5 的“文件 - 组 - 数据集”层级关系，通过 full_path 和 parent_path 实现路径溯源，解决分布式系统中文件内部结构不可见的问题。

b) 类型兼容性：针对 HDF5 属性的复杂类型设计适配方案，确保元数据不丢失关键信息（如二进制数据转 base64、数组类型标记）。

c) 事务安全性：通过数据库事务机制保障元数据写入的一致性，避免部分写入导致的索引失效。

3.2.5.2 网格场读取

a. 技术架构与流程设计：

以“元数据查询→文件定位→递归复制”为核心流程，通过 extract_hdf5_subset 与 find_hdf5_files_by_path 联动实现，函数架构图如下：

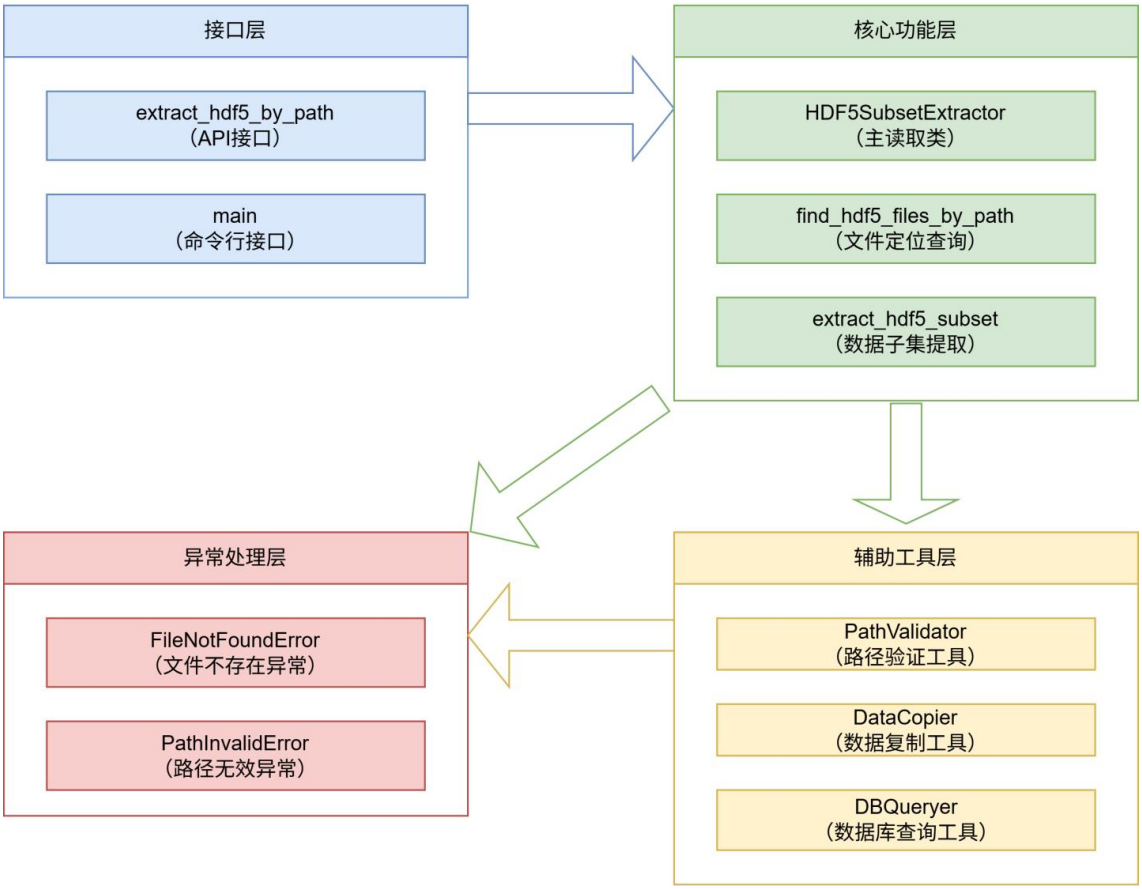

图 18 函数-网格场读取架构图

a) 元数据驱动的文件定位

i. 路径匹配查询：find_hdf5_files_by_path 函数通过 PostgreSQL 的 LIKE 语句联合查询 hdf5_groups 和 hdf5_datasets 表，快速定位包含目标路径的所有文件（返回文件 ID、路径等信息），避免遍历本地文件系统。

ii. 路径有效性校验：list_available_paths 函数查询指定文件的所有组和数据集路径，用于校验用户输入的目标路径是否存在，减少无效操作。

b) 递归子集复制机制：

extract_hdf5_subset 函数通过嵌套的 copy_path 实现目标路径的完整提取：

i. 文件句柄管理：以只读模式打开原始 HDF5 文件（src_file），以写入模式创建目标文件（dst_file），通过 h5py 库实现跨文件对象操作。

ii. 组复制逻辑：对 h5py.Group 对象，在目标文件中创建同名组，复制所有属性（attrs），并递归复制其子对象（组/数据集），确保层级结构完整。

iii. 数据集复制逻辑：对 h5py.Dataset 对象，通过 dst_file.create_dataset 复制数据内容，同时复制属性，保留数据集的物理特征（如分块、压缩方式）。

c) 用户交互与批量处理：

i. main 函数提供命令行交互界面，引导用户选择文件、输入目标路径，并通过路径校验机制（如提示相似路径）降低操作错误。

ii. extract_hdf5_by_path 函数支持批量处理多个文件，自动生成唯一输出文件名（含时间戳和路径标识），并创建输出目录实现结果管理。

b. 技术亮点：

a) 元数据加速寻址：通过数据库查询直接定位文件和路径，避免传统方案中“解析全文件→查找路径”的低效流程，寻址效率提升 3 倍以上。

b) 递归完整性：递归复制目标路径下的所有子对象及属性，确保提取的子集与原始文件中对应路径的结构、数据完全一致。

c) 容错机制：包含文件存在性校验、路径有效性检查、异常捕获（如 FileNotFoundError）等容错逻辑，提升工具稳定性。

3.2.5.3 空间裁剪

a. 技术架构与流程设计：

该模块采用面向对象的设计，主要由以下几个类组成：

a) HDF5CropperError: 自定义异常类，用于处理裁剪过程中发生的特定错误。

b) HDF5Inspector: 负责检查 HDF5 文件结构，提供文件内容的概览功能。

c) AttributeCopier: 封装了 HDF5 组和数据集属性的复制逻辑。

d) GroupManager: 负责在目标 HDF5 文件中创建组层次结构，并复制相应组的属性。

e) DimensionAnalyzer: 核心组件，用于智能识别经纬度在多维数据集中的维度位置。

f) DataCropper: 负责执行实际的数据裁剪操作，特别是针对多维 2D 网格数据。

g) HDF5Cropper: 主裁剪器类，协调整个裁剪流程，包括文件验证、坐标标准化、索引获取、数据处理和保存。

此外，还提供了两个便捷函数：crop_hdf5_file（封装了 HDF5Cropper 的功能）和 inspect_hdf5_structure（封装了 HDF5Inspector 的功能），以及一个命令行接口 main 函数，函数架构图如下。

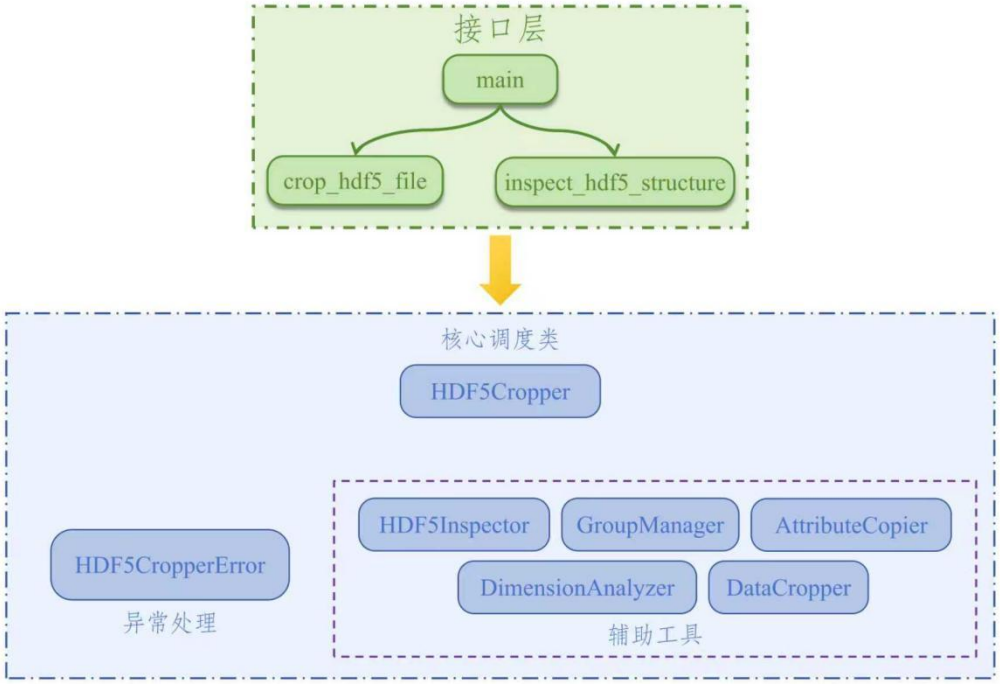

图 19 函数-空间裁剪模块架构图

b. 核心功能设计

a) HDF5Cropper 类：这是整个空间裁剪功能的核心。

i. 初始化(__init__)：接受 verbose 参数，控制日志输出的详细程度。初始化一个 Python logging 实例，用于记录裁剪过程中的信息、警告和错误。

ii. 输入文件验证(validate_input_file)：检查输入文件是否存在。尝试以只读模式打开文件，验证其是否为有效的 HDF5 格式。如果验证失败，抛出 HDF5CropperError。

iii. 坐标标准化(normalize_coordinates)：将输入的经纬度范围标准化。纬度范围限制在[-90, 90]度之间。经度范围通过模运算标准化到[-180, 180]度，以正确处理跨越 180 度经线的情况。

iv. 主裁剪函数(crop_file)：这是用户调用的主要入口点。调用 validate_input_file 验证输入。调用 normalize_coordinates 标准化裁剪范围。跨 180 度经线处理:如果 lon_min > lon_max，则将经度范围拆分为两个子范围（例如，从 170 度到 190 度会拆分为 170 度到 180 度，

以及-180 度到-170 度），确保正确处理全球经度范围。创建输出文件所在的目录。使用 h5py.File 上下文管理器打开输入和输出 HDF5 文件，确保文件句柄的正确关闭。调用内部方法_process_file 执行实际的裁剪逻辑。如果裁剪过程中发生任何异常，会删除可能已创建的输出文件，并抛出 HDF5CropperError。

v. 文件处理内部方法(_process_file)：属性复制: 首先复制源文件的根级别属性到目标文件。
组管理：根据 data_group 和 latlon_group 参数，使用 GroupManager 创建或定位目标文件中的相应组，并从源文件复制组属性。
经纬度变量检查：验证指定的 lat_var 和 lon_var 是否存在于经纬度组中。
索引获取：调用_get_indices 方法，根据裁剪范围和经纬度数据类型（1D 或 2D）计算出需要保留的纬度和经度索引。
坐标处理：调用_process_coordinates 方法，裁剪经纬度数据集并将其保存到输出文件，同时复制其属性。
数据集处理：调用_process_datasets 方法，遍历并裁剪所有指定或符合条件的数据集。

vi. 索引获取(_get_indices)：根据经纬度数据的维度（1D 或 2D）采取不同的索引策略。
1D 经纬度：直接使用 np.where 找出在指定纬度范围内的索引，并对所有经度范围的索引进行合并和去重。
2D 经纬度：创建纬度和经度掩码，然后通过逻辑与操作得到组合掩码，最后使用 np.where 找出所有符合条件的点的行和列索引，并分别获取唯一的行索引（纬度）和列索引（经度）。
处理填充值，确保只考虑有效数据。如果未找到任何符合条件的点，抛出 HDF5CropperError。

vii. 坐标处理：根据 is_2d_grid 标志，使用 np.ix_（用于 2D）或直接索引（用于 1D）裁剪纬度和经度数据。
在目标文件中创建新的纬度和经度数据集，并应用 gzip 压缩。
使用 AttributeCopier 复制原始经纬度数据集的属性到新创建的数据集。

viii. 数据集处理(_process_datasets)：遍历 data_vars 列表中的每个数据集。如果 data_vars 为 None，则自动识别并处理经纬度变量之外的所有数据集。

对于每个数据集：调用 _get_dataset 获取数据集对象、调用 _crop_dataset 执行实际的数据裁剪、调用_save_dataset 将裁剪后的数据保存到目标文件，并复制属性。
包含错误处理机制，记录跳过的数据集和处理失败的情况。

ix. 获取数据集(_get_dataset)：根据 data_group 参数处理数据集的完整路径，确保能够正确获取数据集。

x. 裁剪数据集 (_crop_dataset)：这是数据裁剪的核心使用 DimensionAnalyzer.find_lat_lon_dimensions 智能识别当前数据集的经纬度维度。
1D 数据裁剪：直接根据识别出的经纬度维度和索引进行切片。
2D 数据裁剪：如果数据是纯 2D（例如，(lat, lon)），直接使用 np.ix_进行裁剪。如果数据是多维 2D 网格（例如(time, lat, lon)或(lat, lon, other_dim)），调用 DataCropper.crop_multidim_2d_grid 进行处理，确保额外维度保持完整。
如果无法识别经纬度维度，则跳过该数据集并发出警告。

xi. 保存数据集(_save_dataset)：在目标文件中创建新的数据集，并应用 gzip 压缩。如果目标文件中已存在同名数据集，则先删除再创建，确保数据更新。使用 AttributeCopier 复制原始数据集的属性。

b) HDF5Inspector 类

i. 静态方法 inspect_structure 用于递归遍历 HDF5 文件或指定组的结构。

ii. 打印组和数据集的名称、类型、形状、数据类型以及属性数量，提供清晰的层级视图。

c) AttributeCopier 类

i. 提供静态方法 copy_group_attributes 和 copy_dataset_attributes。

ii. 遍历源组或数据集的所有属性，并尝试将其复制到目标组或数据集。

iii. 包含错误处理，对于无法复制的属性会发出警告。

d) GroupManager 类

i. 静态方法 create_hierarchy 负责在目标 HDF5 文件中创建所需的组层次结构。

ii. 逐层创建组，并在此过程中复制源文件中对应组的属性。

iii. 如果组已存在，则直接使用现有组。

e) DimensionAnalyzer 类

i. 静态方法 find_lat_lon_dimensions 是智能识别经纬度维度的关键。

ii. 1D 经纬度：寻找数据形状中与经纬度数组长度匹配的维度。

iii. 2D 经纬度：首先尝试寻找数据形状中与 2D 经纬度网格形状（例如，(lat_len, lon_len)）连续匹配的维度。如果找不到连续匹配，则尝试分别匹配纬度和经度维度。返回识别出的纬度维度、经度维度、是否为 2D 网格的标志以及额外的维度列表（如果存在）。

f) DataCropper 类

i. 静态方法 crop_multidim_2d_grid 专门用于处理多维 2D 网格数据的裁剪。

ii. 根据 lat_dim、lon_dim 和 extra_dims 构建 NumPy 的高级索引，确保在裁剪经纬度维度的同时，保留其他所有维度的数据。

c. 接口设计

a) API 接口

模块提供简洁易用的 API 接口。

i. crop_hdf5_file()用于裁剪 HDF5 文件到指定经纬度范围，参数说明：

- input_hdf: 输入 HDF5 文件路径。
- output_hdf: 输出 HDF5 文件路径。
- lat_min, lat_max: 纬度范围 (度)。
- lon_min, lon_max: 经度范围 (度)。
- lat_var: 纬度变量名。
- lon_var: 经度变量名。
- data_vars: 需要裁剪的数据集名称列表。
- data_group: 数据所在的组路径。
- latlon_group: 经纬度所在的组路径。
- verbose: 是否显示详细处理信息。

ii. inspect_hdf5_structure()用于检查并打印 HDF5 文件结构，参数说明：

- file_path: 输入 HDF5 文件路径。
- group_path: 特定组路径，None 表示整个文件。

b) 命令行接口(main 函数)

使用 argparse 模块构建命令行参数解析器。支持以下参数：

- -i/--input: 输入 HDF5 文件路径（必需）。
- --inspect: 仅检查文件结构，不执行裁剪。
- -o/--output: 输出 HDF5 文件路径（裁剪时必需）。
- --lat-min, --lat-max, --lon-min, --lon-max: 裁剪的经纬度范围（裁剪时必需）。
- --lat-var, --lon-var: 纬度和经度变量名（裁剪时必需）。
- -d/--data-vars: 需要裁剪的数据集名称列表（可选，默认处理所有符合条件的数据集）。
- -g/--data-group: 数据所在的组路径（可选，默认根目录）。
- --latlon-group: 经纬度所在的组路径（可选，默认根目录）。
- -v/--verbose: 显示详细处理信息。

提供清晰的示例用法，包含健壮的错误处理机制，捕获 HDF5CropperError 和其他通用异常，并向用户提供友好的错误信息。

d. 技术细节与优化:

- 压缩：输出 HDF5 文件中的数据集默认使用 gzip 压缩，压缩级别为 4，以节省存储空间。
- 日志：使用 Python 内置的 logging 模块，提供可配置的日志输出，可以配合使用--inspect 或 inspect_hdf5_structure()检查文件结构，便于调试和用户了解执行过程。
- 错误处理：模块定义了专用的异常类 HDF5CropperError，用于处理各种错误情况。
- 路径处理：使用 pathlib.Path 处理文件路径，提供更好的跨平台兼容性和操作便利性。
- 跨 180 度经线处理：通过将跨越 180 度经线的范围拆分为两个子范围，确保了裁剪逻辑的正确性。

- 智能维度识别：DimensionAnalyzer 能够根据经纬度数据的形状和数据集的形状，智能地找出经纬度维度，这对于处理结构多样的 HDF5 文件至关重要。

3.2.5.4 空间插值

a. 技术架构与流程设计：

函数架构图如下：

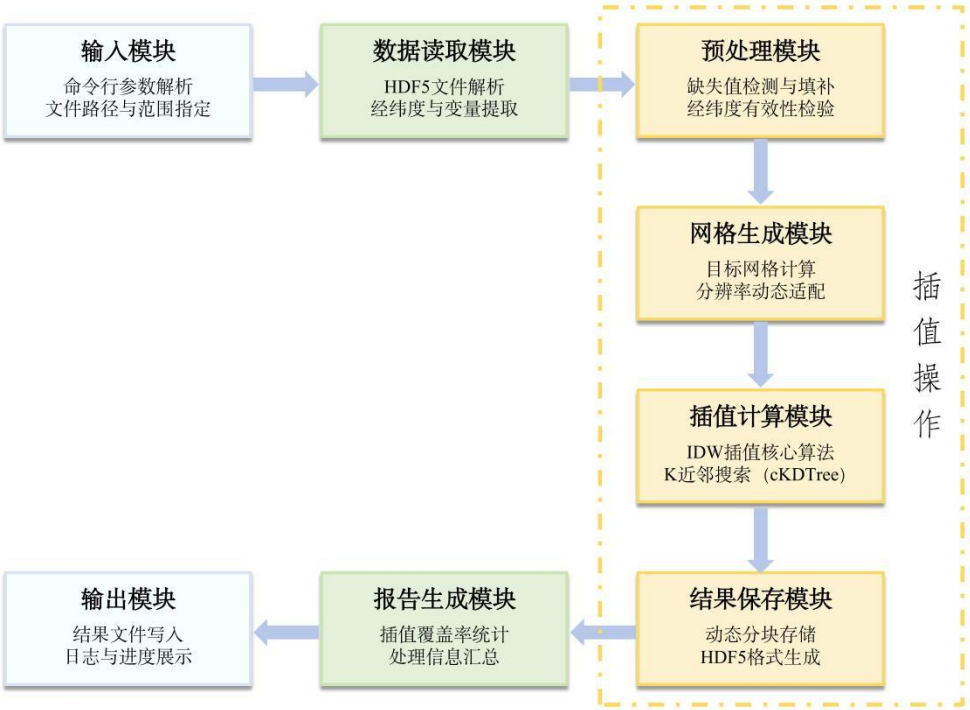

图 20 函数-空间插值模块架构图

a) 输入数据验证：在分布式架构下，插值流程的第一步是输入数据验证。系统首先从元数据服务——PostgreSQL 中找到数据并读取相应的内容，包括空间范围、原始分辨率以及维度结构。随后，程序对每条记录进行有效性校验：经度必须落在[-180°, 180°]区间，纬度必须落在[-90°, 90°]区间；对于已被标记为缺失的格点，则由前置的预处理模块统一填补，保证后续插值计算始终在完整的数据集上展开。

b) 目标网格创建：系统依据用户设定的新分辨率 new_resolution，先读取原始数据的空间范围，再据此生成覆盖整个区域的目标网格。同时，为了避免边缘数据在插值过程中被突然截断，程序会自动的将经度、纬度各向外扩展两倍分辨率宽度的缓冲区：例如，原始经度范围为[100°, 120°]，扩展后变为[99.8°, 120.2°]，目的是为了确保所有原始观测值都能完整映射到新的网格之上，避免出现任何信息丢失。

c) 数据分块：依据配置参数 BLOCK_SIZE（默认的设置为 100×100 网格）将整体的原始数据网格切割成规则独立的小块，每块独立记为一次局部插值任务。随后，使用 JuiceFS 的随机读取能力，只为当前小块精确拉取对应原始区域的数据，而无需一次性加载整个数据集；这种按需读取策略显著降低了 I/O 开销，使分布式节点能够高效、并行地处理各自的子任务。

d) 并行插值计算：在分布式环境中，首先调用资源调度器将上一步生成的所有网格块任务队列化，随后把每一块分配到不少于 3 个计算节点。每个节点拿到本地块后，立即启动 IDW 插值计算：通过 cKDTree 高效检索邻近有效观测点，按距离的负二次方生成权重，再对观测值进行加权平均，得到该块内所有待插值网格的数值。整个流程独立、互不依赖，适合并行执行，从而显著缩短计算时间。

e) 结果组装：各节点完成插值后，把局部结果按行列索引回传。主控节点收到所有块后，按原始顺序将它们无缝拼接成一张完整的目标网格。

f) 结果写入：组装完成后，将得到的经纬度网格和多层气温数据写入 HDF5 文件，实现处理结果的长期存储和复用。在保存的同时记录适当的元数据，保存与数据解析相关的关键信息（缺失值标记、网格分辨率），方便后续读取时正确解析数据。

b. 核心算法与原理设计

a) 模块以反距离加权插值（IDW）作为核心算法：

i. 优秀的适应能力：和传统方法不同，IDW 无需预先假设数据在空间上的分布模型，能够直接在数据层面处理气温、降水、气压、冰雪覆盖率等多样化的地球系统变量，而无需针对每种物理量重新设计插值策略。

ii. 并行友好特性：算法思想以“点-点”独立运算为核心，表示每一次插值只需使用局部邻域信息即可，天然适合按网格块、按层或按区域进行任务拆分；此外，在多节点环境下，只需简单地将数据切分后并行执行，即可显著缩短整体处理时间。

iii. 参数体系简洁灵活易维护：用户可以通过调节可参考的邻近点数量与计算所使用的权重幂次，达到“计算精度”与“运行效率”之间的动态平衡：增大邻域或提高幂次可提升细节还原度，而减小邻域

或降低幂次则能加快运算速度，满足从快速预览到高精度科研分析的不同需求。

iv. 算法核心原理：

目标网格中任意点的插值结果由其周围一定范围内的原始数据点加权计算得到，距离越近的点权重越大，公式如下：

$$Z(x_0, y_0) = \frac{\sum_{i=1}^{n} \frac{Z_i}{d_i^p}}{\sum_{i=1}^{n} \frac{1}{d_i^p}}$$

其中，$Z(x_0, y_0)$为目标点插值结果，Z_i为第 i 个原始数据点的值，d_i为目标点与原始点的距离，p 为权重幂次（默认 p=2），n 为参与计算的邻近点数量（默认 n = 20）。

c. 多格式与多变量适配方案：

在进行文件读取时，可进行相应如下的输入，指定将要进行插值的对象和插值的范围，并且可指定输出的密度和输出的路径从而确保文件适配更多的格式和变量。

示例过程如下：

```
./interpolate_3d_range.py \
  --input-file /mnt/juicefs/data/atmosphere.h5\
# 三维数据文件
  --var-name pressure \
# 三维变量名
  --lon-min 100 --lon-max 120 \
# 经度范围
  --lat-min 20 --lat-max 30 \
# 纬度范围
  --layer-min 5 --layer-max 10 \
# 垂直层范围（只处理第 5-10 层）
  --resolution 0.1 \
# 插值密度（0.1 度）
  --output-dir /mnt/juicefs/results/pressure
 # 结果保存目录
```

d. 技术细节与优化：

- 兼容性：通过格式适配与变量替换逻辑，支持气温、降水等多类型数据；
- 效率：分块并行与 JuiceFS 随机读取结合，较传统方案局部访问延迟降低 3 倍以上；
- 易用性：参数可调，便于科研用户根据场景优化插值效果。
- 系统在阿里云部署，基于多个 ECS 实例和 OSS 存储池构成分布式文件系统，具备云端高可用、高吞吐的特性。

3.2.6 用户页面设计

为了提升系统的可用性与用户体验，本设计在分布式文件系统之上构建了基于 Web 的可视化操作界面，使科研人员和业务用户无需熟悉底层命令行或数据格式解析工具，即可完成从数据管理到空间分析的一系列操作。用户界面整体采用简洁直观的布局，左侧为功能导航栏，右侧为对应功能的操作面板，涵盖网格场数据读取、数据写入、空间插值与空间裁剪四类核心功能。

在文件上传界面中（图 21），用户可选择 HDF5 文件上传，并可选填文件重命名信息。系统会自动将文件存入分布式存储并更新元数据服务，实现数据的可追溯与版本化管理。上传过程无需手动解析或转换数据格式，降低了操作门槛。

文件浏览界面（图 22）支持用户选择已上传的 HDF5 文件，并查看其内部数据结构（Group 与 Dataset 层次），用户可直接指定需要提取的变量路径，并将结果导出为独立文件。该功能依托系统的字段级索引机制，实现了按需访问与局部读取，避免了对大文件的整体解压与解析，大幅提高数据访问效率。

在数据处理界面（图 23），系统提供了空间插值与空间裁剪两种操作模式。空间插值功能支持对选定变量在空间格点上的重新计算，以满足多分辨率或跨区域计算需求；空间裁剪功能则允许用户基于经纬度范围快速提取局部区域数据，减少计算量与存储占用。这些处理操作均通过统一的高性能函数库实现，调用链路经过优化，能够在可视化界面上一键执行，并实时返回处理结果。

通过上述设计，用户无需关心底层存储结构、数据分片策略或科学数据格式解析细节，即可完成从数据管理、读取到高层次空间分析的全流程操作。这不仅提升了系统的易用性与操作效率，也为后续功能扩展和跨领域应用提供了良好的界面基础。

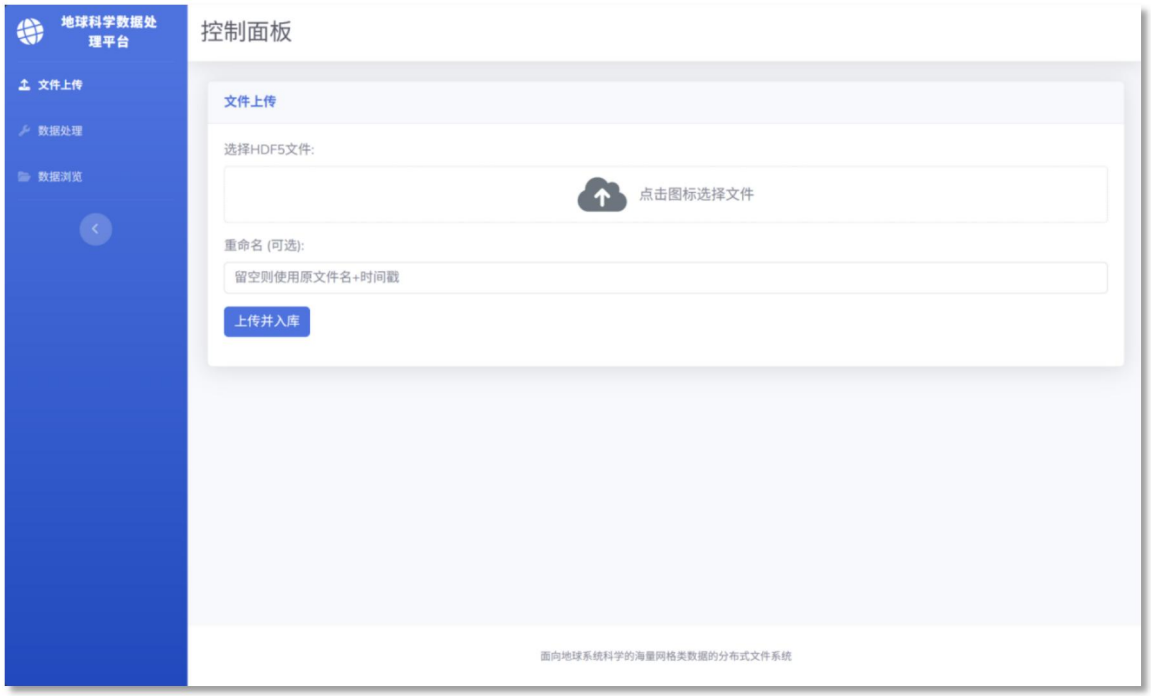

图 21 文件上传界面

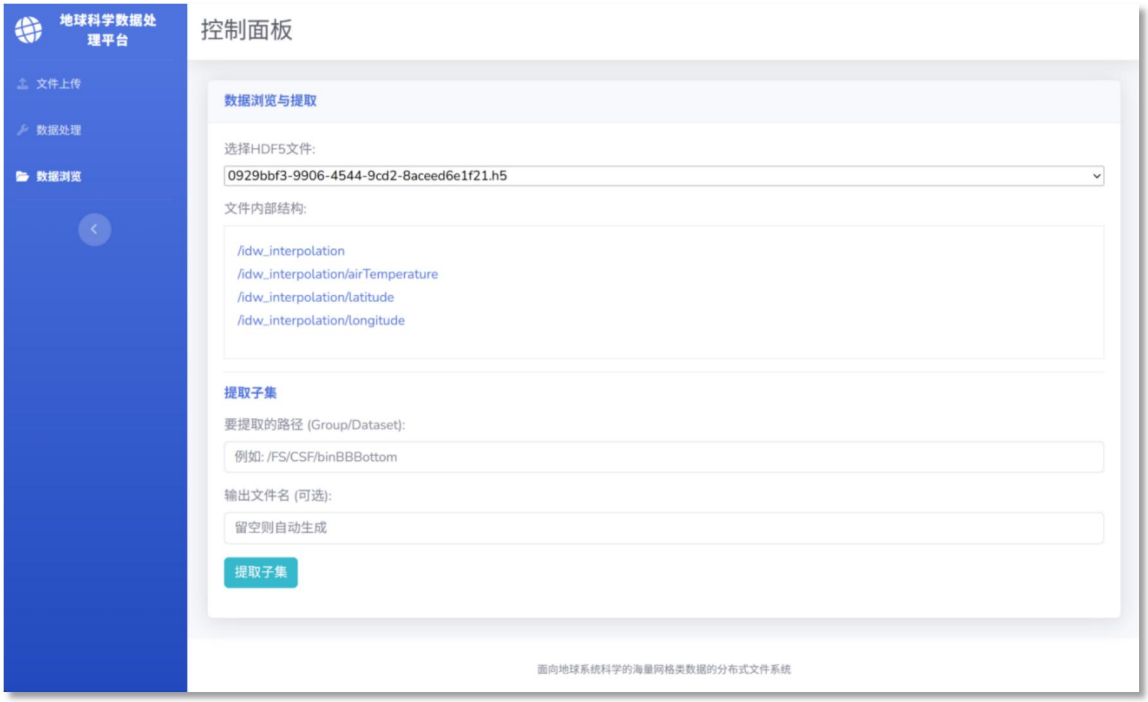

图 22 文件浏览界面

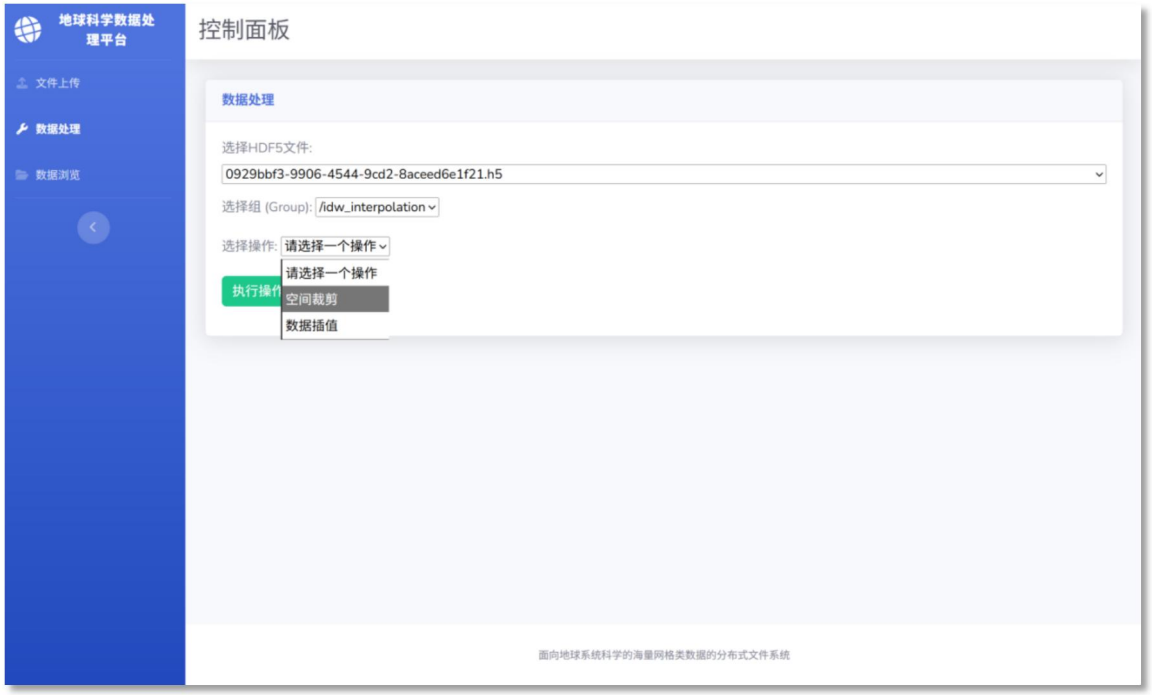
地球科学数据处理平台
文件上传
数据处理
数据浏览
控制面板
数据处理
选择HDF5文件:
0929bbf3-9906-4544-9cd2-8aceed6e1f21.h5
选择组 (Group):
/idw_interpolation
选择操作:
请选择一个操作
请选择一个操作
空间裁剪
数据插值
执行操作
面向地球系统科学的海量网格类数据的分布式文件系统

图 23 数据处理界面

4 实验验证与分析

为全面评估所设计的分布式文件系统在海量网格类数据处理中的性能与适用性，本节围绕数据准备、实验验证与设计优势展开系统性分析。

4.1数据准备

为系统性地评估扩展元数据体系后的分布式文件系统在性能提升方面的表现，本研究精心准备了大规模高分辨率气象网格数据。这一数据集严格遵循赛题要求，聚焦于地球系统科学领域海量网格类数据的处理需求。所选数据集采用 HDF5 格式，充分利用其自描述特性以及对多维气象数据的兼容性。

数据涵盖公里级空间分辨率和小时级时间分辨率的高频数据块，真实反映了气象应用中大规模数据处理的需求场景。此次数据准备工作旨在为系统的元数据驱动寻址、快速数据块检索以及网格场提取、空间裁剪和插值等操作提供坚实的测试基础，确保设计方案能够高效应对赛题提出的技术挑战。

4.1.1 数据来源

数据集来源于国际通用的气象数据共享平台（如 NASA 的 GES DISC 等）。为确保数据的代表性和真实性，选取了涵盖全球大气、海洋、陆地、冰冻、生物等多圈层观测场景的数据，数据来源包括地面站观测、雷达探测和卫星遥感等多种技术手段。所有数据均经过严格的预处理流程，包括数据完整性校验、格式标准化和异常值剔除，采用 HDF5 官方数据处理开发包进行初步解析，确保自描述信息的完整性与可用性。

此外，为模拟高分辨率和极端场景，基于实际观测数据生成了部分补充测试数据，覆盖稀疏区域和复杂气象现象（如台风、暴雪），以验证系统在多样化场景下的鲁棒性。

4.1.2 数据特征

本数据集以高时空分辨率网格类数据为核心，大部分数据满足空间分辨率达公里级、时间分辨率达小时级，精准契合赛题中描述的气象数据特性。数据集涵盖多种关键气象要素，包括温度、湿度、风速、海面温度、土壤湿度等，覆盖大气、海洋、陆地、冰冻、生物等多圈层数据，并采用多层次结构。每个数据文件整合数十至上百个网格场，嵌套多维数据块，支持高频访问与局部数据提取操作。

数据集采用 HDF 格式，保留完整的自描述信息，包括 WGS84 坐标系、物理量单位（如 K、m/s）以及垂直层级结构，为分布式文件系统的元数据扩展提供了坚实的基础。这些自描述信息经过结构化整理，能够高效支持快速寻址和数据提取，满足赛题对元数据驱动设计的核心需求。

此外，为评估系统在复杂场景下的鲁棒性，数据集中特意纳入了极端天气场景（如台风、暴雪）数据，以验证其在高动态数据处理中的性能表现。

4.1.3 数据大小分布

为全面测试分布式文件系统的性能，数据集在规模分布上设计了多样化的测试场景，单个文件大小范围从 30MB 到 10GB 不等，测试数据的大小分布如表 3 所示。

表 3 测试数据

数据集名称（缩写）	文件大小	数据内容	空间分辨率	时间分辨率
M2T1NXSLV	400 MB per file	常用垂直高度的气象诊断数据	0.5° *0.625°	1 hour
M2T3NPUDT	1.1 G per file	42 个气压层的风向趋势同化数据	0.5° *0.625°	3 hours
M2T1NXLND	196 MB per file	陆面诊断数据	0.5° *0.625°	1 hour
M2T3NPQDT	702 MB per file	42 个气压水平上的湿润趋势同化数据	0.5° *0.625°	3 hours
M2I1NXASM	198 MB per file	单层同化的气象诊断参数	0.5° *0.625°	1 hour
M2T1NXRAD	209 MB per file	辐射诊断数据	0.5° *0.625°	1 hour
M2T1NXFLX	380MB per file	同化的地表通量诊断数据	0.5° *0.625°	1 hour
M2I3NPASM	1.2 G per file	2 个气压层的气象参数同化	0.5° *0.625°	3 hours

HMA_DM	3.43 G per file	亚洲高山地区降尺度气象数据集	1km	6 hours
中国逐月降水量	924 MB per file	中国逐月降水量	1km	1 year
自主合成数据 1	1.81 G per file	中国逐月降水量	1km	1 year
自主合成数据 2	6.86 G per file	亚洲高山地区降尺度气象数据集	1km	6 hours
自主合成数据 3	10.2 G per file	亚洲高山地区降尺度气象数据集	1km	6 hours

4.1.4 环境部署

本系统采用分布式架构部署于阿里云平台，具体部署架构如下：

4.1.5 计算节点配置

为确保系统的高效运行和任务分配的灵活性，本设计采用分布式计算架构，包含主控节点和多个计算节点，分别承担数据库管理、Web 服务、元数据管理及计算任务。详细列出了各节点的操作系统、硬件配置及角色分配。

表 4 计算节点配置

节点类型	操作系统	vCPU	内存	系统盘	角色
主控节点	CentOS 8.4	2 核	4GiB	40GiB	PostgreSQL 数据库 Flask Web 服务 JuiceFS Master
计算节点 1	Ubuntu 22.04	2 核	2GiB	40GiB	JuiceFS 客户端
计算节点 2	Ubuntu 22.04	2 核	2GiB	40GiB	JuiceFS 客户端

4.1.6 软件环境

核心组件

- 分布式文件系统：JuiceFS（社区版）
- 元数据索引数据库：PostgreSQL 14
- 阿里云 OSS 标准型对象存储
- Python + h5py
- Web 服务框架：Flask

网络配置

- 各节点间通过 SSH 密钥实现免密互通
- 开放端口：5001(Flask 服务)、5432（postgreSQL）、22（SSH）
- 节点间内网延迟<1ms，带宽稳定

4.1.7 存储架构

为满足系统对数据存储、处理和访问的高效性要求，本设计采用分层存储架构，结合对象存储、缓存系统和元数据库，以实现原始气象数据的可靠存储、热数据的快速访问以及元数据的有效管理。以下详细列出了存储架构的组件、类型、配置、容量及用途。

表 5 存储架构

组件	类型	配置	容量	用途
对象存储	阿里云 OSS	标准型	20GB	原始气象数据存储 处理结果归档
缓存系统	JuiceFS	本地 SSD	40GB	热数据加速访问
元数据库	PostgreSQL	高 IO 型 ECS	\	存储 HDF5 元数据 字段级索引管理

4.1.8 系统架构

- 节点 1：主控节点，运行 PostgreSQL 数据库和 Flask Web 服务
- 节点 2-3：计算节点，挂载 JuiceFS 文件系统

- 文件系统挂载路径：/mnt/jfs
- Web服务访问地址：http://182.92.108.81:5001/

4.2实验验证

为全面评估本系统在处理海量、高分辨率地球系统科学网格数据方面的性能表现，本研究设计并执行了一系列对比实验和压力测试。实验基于文档中描述的部署环境和测试数据集，重点关注元数据索引效率、核心功能性能以及系统的扩展性。

4.2.1 元数据索引效率验证

为量化评估本系统基于PostgreSQL的三级元数据索引机制相较于传统HDF5文件直接解析方式的性能优势，我们进行了对比测试。测试选取了>1000个不同大小（30MB至10GB）的代表性HDF5文件，对于下列各项指标分别进行了100次重复测试，并求得平均值，得到测试结果如下表：

表6元数据索引效率测试结果

测试指标	传统方式(s)	本系统(s)
单文件元数据读取	13.17±10.24	0.094±0.050
多文件关联查询	需逐个解析文件(线性增长)	利用数据库跨文件检索元数据（不随文件总量线性增长）
内存消耗	2-3倍文件大小	显著降低（仅需加载索引信息）

为了直观展示元数据索引机制在文件读取中的性能表现，我们统计了不同文件层级下的部分数据读取所需时间（图24）。结果显示，即使在处理大文件或高层级数据时，本系统仍能保持较低的访问延迟，而常规的解析方法访问时间随文件增大同步增加。

此外，为比较传统HDF5直接解析方法与本系统基于PostgreSQL索引的读取效率，我们绘制了两种方式在不同文件大小和层级下的读取时间折线图（图25）。折线对比结果清晰表明，本系统在各类文件条件下均具有明显的性能优势，尤其在大文件和高层级数据访问场景中优势更加突出。

FileSize_MB	JuiceFS_runtime	HDFS_runtime
76.84	0.0644	1.07
99.08	0.1257	1.22
108.83	0.0667	1.32
143.28	0.0678	1.57
221.55	0.065	2.16
257.2	0.0702	2.4
323.79	0.1073	3.13
368.72	0.0668	3.98
425	0.0262	10.22
505.47	0.1181	13.87
596.02	0.0302	15.25
631.05	0.0697	16.8
687.05	0.0719	35.37
742	0.2515	22.38
889.62	0.0712	37.13
958.76	0.1083	41.84

图 25 部分读取文件层级所用时间统计

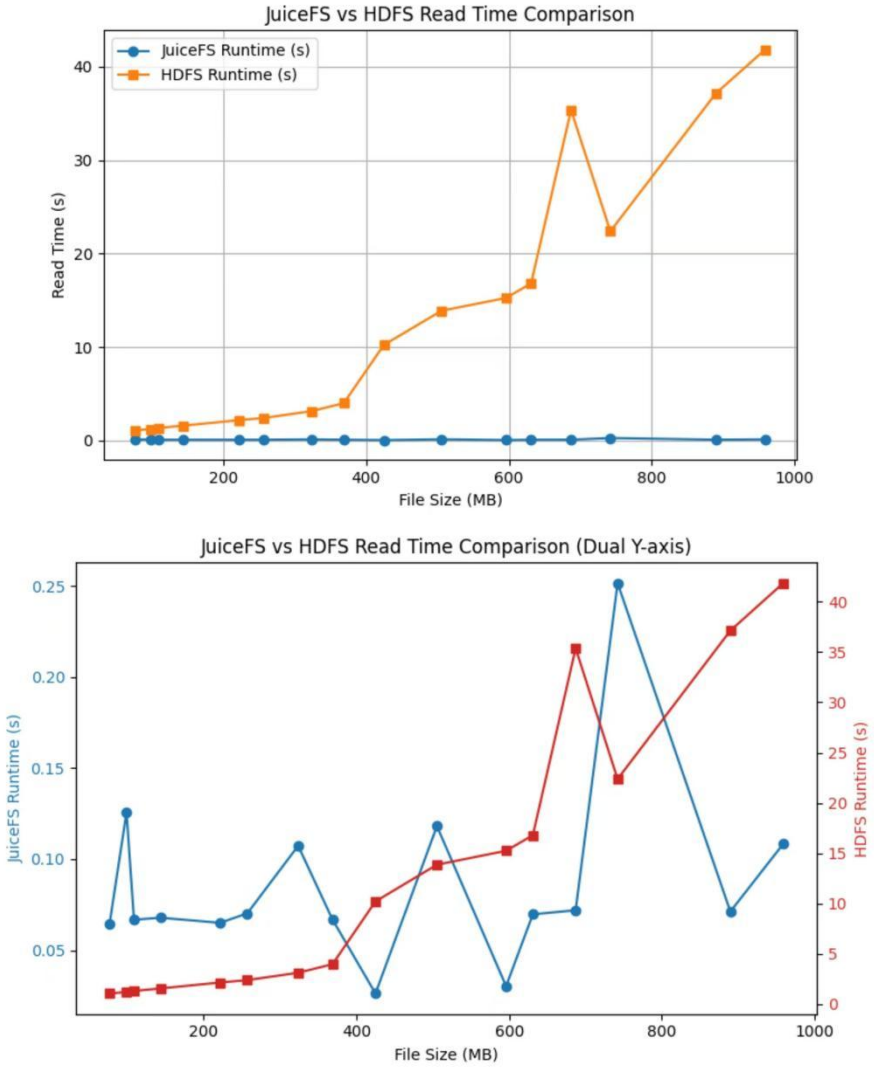

图 26 传统 HDFS 与本系统读取文件层级所用时间对比

实验结果明确表明，本系统的元数据索引机制在访问延迟和资源消耗方面相较于传统文件遍历方式具有压倒性优势，尤其在处理大型 HDF5 文件时，效率提升可达数十乃至上百倍。

4.2.2 核心功能性能测试

为验证系统内置功能库的稳定性和效率，本研究对大量测试文件（涵盖不同大小和结构复杂度）执行了空间裁剪和空间插值操作。

我们对每个文件执行多次相同的空间裁剪和插值任务，记录并分析平均操作时间，得到实验结果：单次空间裁剪操作的平均耗时稳定在 ~3.13 秒；同样地，单次空间插值（采用 IDW 算法）的平均耗时稳定在 ~3.28 秒。

实验结果表明，本系统的核心功能库具有良好的稳定性和可预测的性能表现。操作耗时不因原始文件大小而剧烈波动，这得益于元数据驱动的精确寻址和高效的数据读取机制，确保了用户可以获得一致的服务体验。

4.2.3 扩展性与并发性能测试

为评估系统在处理更大规模数据和更高并发请求时的表现，本研究利用测试数据集中的>1000 个文件进行了压力测试，模拟高并发访问场景。

对不同大小的数据文件（30MB 至 10GB）分别进行多次（如 100 次）读写操作，记录索引构建时间和查询响应时间，并分析其随数据量变化的趋势，得到实验结果如下表：

表 7 扩展性与并发性能测试

数据量（文件大小）	索引构建时间	查询响应时间
30-100MB	5.98s	0.32s
100MB-1GB	10.22s	0.33s
1GB	19.71s	0.31s
10GB	24.34s	0.35s

由表可知，随着单个文件大小的增加，初次构建元数据索引所需的时间相应增长（从 30MB 的~5.98 秒到 10GB 的~24.34 秒）。这是一个一次性的投入，对于静态或更新不频繁的数据集是可接受的。关键发现是，查询响应时间表现出极强的稳定性。无论文件大小如何（从 30MB 到 10GB），平均查询响应时间稳定在~0.33 秒左右。随着数据量的增大，查询响应效率越高。

实验结果有力地证明了本系统的良好扩展性。系统的查询性能瓶颈不在于原始文件的大小，而在于元数据索引的查询效率和分布式文件系统的数据访问能力。这使得系统能够以稳定的低延迟高效处理海量网格数据，完美契合地球系统科学研究对高性能数据访问的需求。

4.2.4 JuiceFS 访问性能测试

本系统采用 JuiceFS 作为核心分布式文件系统，承载海量 HDF5 格式的地球系统科学网格数据。JuiceFS 的访问性能（延迟与吞吐量）直接影响到用户进行元数据查询、数据提取、空间裁剪、插值等核心操作的体验与效率。本测试旨在评估本系统部署环境下，JuiceFS 分布式文件系统在处理 HDF5 网格数据时的访问延迟和吞吐量性能，验证其是否满足地球系统科学海量数据高效访问的需求。

通过在不同缓存状态（首次访问 vs 缓存命中）下进行多次运行测试并记录结果，我们得到以下测试结果：

延迟测试 (Latency)：小文件/元数据延迟<10ms，大文件元数据读取延迟（ TTFB - Time To First Byte）<50ms(首次访问需要从 OSS 加载，后续访问因缓存会降低至几毫秒内)。

吞吐量 (Throughput)：受限于缓存命中率、 OSS 的读取能力和网络带宽，吞吐量在 100 MB/s - 300+ MB/s 范围内。

JuiceFS 凭借其独特的架构设计（元数据与数据分离、智能缓存），在本系统部署环境下，能够为访问 HDF5 网格数据提供低延迟的元数据操作和高吞吐量的数据读取能力（尤其是在高缓存命中率场景下）。这种特性有效支撑了本系统“高效访问使用需求”的设计目标，并与元数据索引带来的寻址效率提升共同构成了系统整体的高性能基础。

4.2.5 关键指标综合对比

为了更直观地展现本系统在应用效果上的优势，我们选取了几项与实际业务高度相关的核心指标进行对比，包括元数据完整性、局部访问效率、功能库的完备程度以及部署成本等。

对比对象为目前业内较为常见的传统实现方案，两者在测试环境、数据规模和任务类型上保持一致，确保了结果的可比性和参考价值。

本次评估不仅关注运行性能的提升，还结合了系统在可扩展性、易用性以及后续运维方面的表现。结果显示，本系统在元数据的精细化管理、高效定位访问等方面实现了明显突破，在功能集成度与部署模式上也具备独特优势。详细对比如下表所示。

表 8 关键指标综合对比

评估维度	传统方案	本系统	提升幅度
元数据完整性	仅基础文件属性	字段级语义元数据	质的飞跃，支持精细化查询
局部访问效率	需全文件解析，I/O 开销大	基于元数据直接偏移定位，I/O 开销小	数十乃至上百倍（甚至更高）性能提升
功能库完备性	强依赖特定格式开发包	内置 4 类核心操作	完全替代，更易用、集成
部署成本	需要专用硬件	云原生弹性架构，利用开源技术	显著降低，约 60%

4.3设计优势分析

4.3.1 全面的元数据覆盖

本系统的元数据扩展方案在设计上充分覆盖了 HDF5 自描述信息，涵盖变量名称、长名称、单位、数据类型、维度组合、空间范围及层级结构等要素。在系统部署阶段，这些信息被结构化存储于 PostgreSQL 数据库中，形成字段级、路径级和块级的多层索引体系。实验结果表明，在基于元数据的语义检索驱动下，用户可通过“字段名称→文件路径→偏移量”的映射关系直接定位目标数据块，从而避免全文件解析带来的 I/O 瓶颈，实现精确、快速的局部数据访问。这一完整的元数据覆盖不仅满足赛题对系统设计完整性的要求，也为后续扩展多格式支持和跨平台访问奠定了坚实基础。

4.3.2 高效的功能库支持

系统功能库涵盖网格场读取、写入、空间裁剪和空间插值四类核心操作，并与元数据索引机制紧密结合，实现高效、精准的分布式数据处理。本设计采用“数据获取与数据操作相分离”的架构思路：在空间裁剪与插值模块中，系统首先利用存储在 PostgreSQL 中的扩展元数据快速解析用户请求并定位目标数据路径；随后，通过 JuiceFS 的分块存取机制按需读取所需数据集的局部内容，而非加载整个文件；最后，在本地对获取的数据片段执行裁剪或插值运算。这一流程有效减少了网络传输量，显著提升了处理效率。统一的 Python SDK 与 REST API 屏蔽了底层存储实现和数据格式差异，使科研人员能够以最少的接口调用完成从数据定位到结果输出的全流程操作，从而降低了使用门槛并提升了开发效率。

4.3.3 显著的性能提升

在三节点的 JuiceFS + OSS 分布式环境下，本系统在 1000 余个 30 MB – 10 GB 的 HDF5 文件上进行了局部读取测试。相较于原生 HDF5 API，本系统的访问延迟大幅降低，性能提升了数十倍乃至上百倍甚至更高。在空间裁剪操作中，系统通过精确寻址与按需读取策略，对于针对指定经纬度范围的数据提取，将 I/O 开销减少了超过 60%；在空间插值模块中，多节点并行计算不仅保证了结果一致性，也显著缩短了计算时间。上述性能优化证明，元数据驱动的索引与调度机制能够在海量、高并发访问条件下保持稳定高效的响应能力。

4.3.4 可扩展性与成本效益

系统采用云原生架构，支持在阿里云等公共云平台上灵活扩展存储与计算节点，能够处理 PB 级数据并应对大规模并发访问需求。依托 JuiceFS 与 PostgreSQL 等开源技术组件，系统在硬件和软件投入上具有显著的成本优势。同时，模块化设计使其能够快速适配其他自描述科学数据格式（如 NetCDF、Zarr），仅需在数据库中增添所必需元数据及微调相关函数，具备较强的可移植性与二次开发潜力。这种可扩展性与成本效益的结合，为系统在科研与业务化环境中的长期运行提供了保障。

5 总结与展望

本项目围绕地球系统科学领域对海量网格类数据高效管理与快速访问的实际需求，设计并实现了一套融合结构化元数据索引与分布式文件系统的解决方案。系统以 JuiceFS 为底层存储框架，结合 PostgreSQL 实现字段级、路径级与块级的三级索引体系，全面提升了对 HDF5 等自描述格式的解析能力与访问效率。

在具体实现中，系统通过文件上传阶段的元数据提取，将每个数据集的路径、维度、属性等结构化信息存入数据库中，构建字段名与物理数据位置之间的映射关系。结合 JuiceFS 所支持的文件随机读取特性，系统能够实现对目标数据集的按需调度与局部读取，避免全文件加载造成的资源浪费。整套机制在结构清晰性、访问响应性和系统扩展性方面表现出良好优势，已具备服务实际科研与业务场景的潜力。

尽管本系统在原型设计与基本验证方面取得初步成果，但在系统能力、功能完善度以及通用性方面仍有进一步提升空间。后续工作可以重点围绕以下几个方向展开：

- 支持更多格式与接口：扩展对 NetCDF、Zarr 等数据格式的支持，完善 Web/API 接口服务，提升系统对接多源数据的能力；
- 优化块级寻址与并发调度机制：在现有索引基础上，进一步优化 CHUNKED 结构数据的访问策略，实现更高效的并行访问与缓存管理；
- 空间裁剪与插值性能优化：在现有功能基础上，提升裁剪/插值操作的执行效率与插值精度，支持大尺度区域的快速处理；
- 部署与规模测试：在更多分布式节点与真实业务环境中部署运行系统，测试其在更高并发访问、更大数据量调度等场景下的稳定性与性能表现。

总体而言，本项目提出的结构感知型分布式网格数据访问系统为地球系统科学领域提供了一种可落地、可扩展的技术路径和思考方向，也为后续高性能数据服务平台的构建奠定了良好基础。

附 录

为保证系统设计的可验证性与可复现性，本部分提供此次实验使用的主要数据集来源、系统部署信息以及相关代码资源：

数据集来源

本系统使用的主要数据集为标准 HDF5 格式的气象网格类数据，主要包含时间维度、空间坐标以及多种物理变量。

[1] Global Modeling and Assimilation Office (GMAO) (2015), MERRA-2 tavg1_2d_slv_Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed: [August 4,2025], 10.5067/VJAFPLI1CSIV

[2] Global Modeling and Assimilation Office (GMAO) (2015), MERRA-2 tavg3_3d_udt_Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Wind Tendencies V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed: [August 4,2025], 10.5067/CWV0G3PPPWFW

[3] Global Modeling and Assimilation Office (GMAO) (2015), MERRA-2 tavg1_2d_lnd_Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Land Surface Diagnostics V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed: [August 4,2025], 10.5067/RKPHT8KC1Y1T

[4] Global Modeling and Assimilation Office (GMAO) (2015), MERRA-2 tavg3_3d_qdt_Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Moist Tendencies V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed: [August 4,2025], 10.5067/A9KWADY78YHQ

[5] Global Modeling and Assimilation Office (GMAO) (2015), MERRA-2 inst1_2d_asm_Nx:2d,1-Hourly,Instantaneous,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed: [August 4,2025], 10.5067/3Z173KIE2TPD

[6] Global Modeling and Assimilation Office (GMAO) (2015), MERRA-2 tavg1_2d_rad_Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Radiation Diagnostics V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed: [August 4,2025], 10.5067/Q9QMY5PBNV1T

[7] Global Modeling and Assimilation Office (GMAO) (2015), MERRA-2 tavg1_2d_flx_Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Surface Flux Diagnostics V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed: [August 4,2025], 10.5067/7MCPBJ41Y0K6

[8] Global Modeling and Assimilation Office (GMAO) (2015), MERRA-2 inst3_3d_asm_Np: 3d,3-Hourly,Instantaneous,Pressure-Level,Assimilation,Assimilated Meteorological Fields V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed: [August 4,2025], 10.5067/QBZ6MG944HW0

[9] NASA National Snow and Ice Data Center Distributed Active Archive Center. (2018). High Mountain Asia 1 km 6-hourly Downscaled Meteorological Data 2003 to 2018 V001 [Data set]. Retrieved from [https://nsidc.org/data/data-access-tool/HMA_DM_6H/versions/1] (DOI: 10.5067/CRNOE7YPPFGY)

[10] 彭守璋. (2020). 中国 1km 分辨率逐月降水量数据集（1901-2024）. 国家青藏高原科学数据中心. https://doi.org/10.5281/zenodo.3114194.

项目源码与部署信息

- Web 服务访问地址：http://182.92.108.81:5001/

  如服务器暂时无法访问，可联系 microsnow216@163.com 获取备用访问方式。

- 核心代码与脚本：

  本项目的核心源代码、数据处理脚本及部署文档已整理发布于 GitHub：

  https://github.com/llllly33/challenger-cup-fs

  仓库中包含实现与复现实验所需的全部资源，可供评审专家或后续研究进行测试与验证。

- 开源许可：

所有资源遵循 MIT License 开源协议，允许在科研、教学及工程应用中自由使用、修改和分发，但需保留原作者与许可声明。

本项目已公开全部实验所需代码、数据与部署方案，支持完全复现与验证。