中国科学院高能物理研究所
Institute of High Energy Physics
Chinese Academy of Sciences

国家高能物理科学数据中心
National HEP Data Center

高能所計算中心
IHEP Computing Center

HEPS数据处理软件框架Daisy

胡誉 (huyu@ihep.ac.cn)

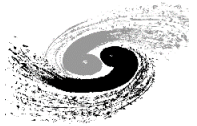
中国科学院高能物理研究所

DAISY
Data Analysis and Software Integration System

HEPS

第四届高能物理计算暑期学校 2023年08月

Outline



1. 背景介绍
2. 科学数据与软件系统的需求与挑战
3. 软件框架的架构与设计
4. 基于软件框架的科学软件开发
5. 科学软件示例
6. 总结

Outline

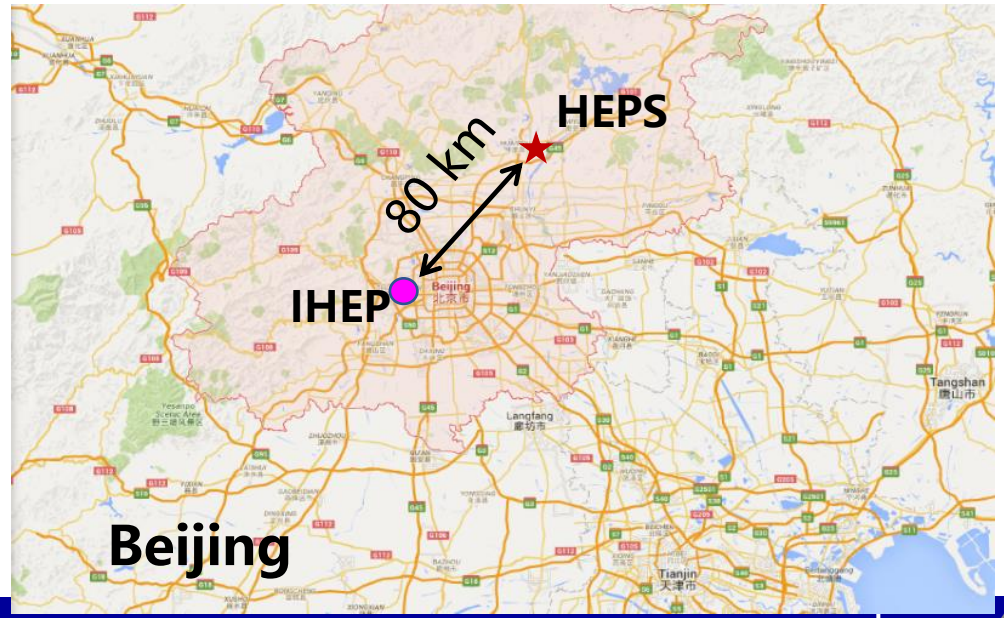


1. 背景介绍
2. 科学数据与软件系统的需求与挑战
3. 软件框架的架构与设计
4. 基于软件框架的科学软件开发
5. 科学软件示例
6. 总结

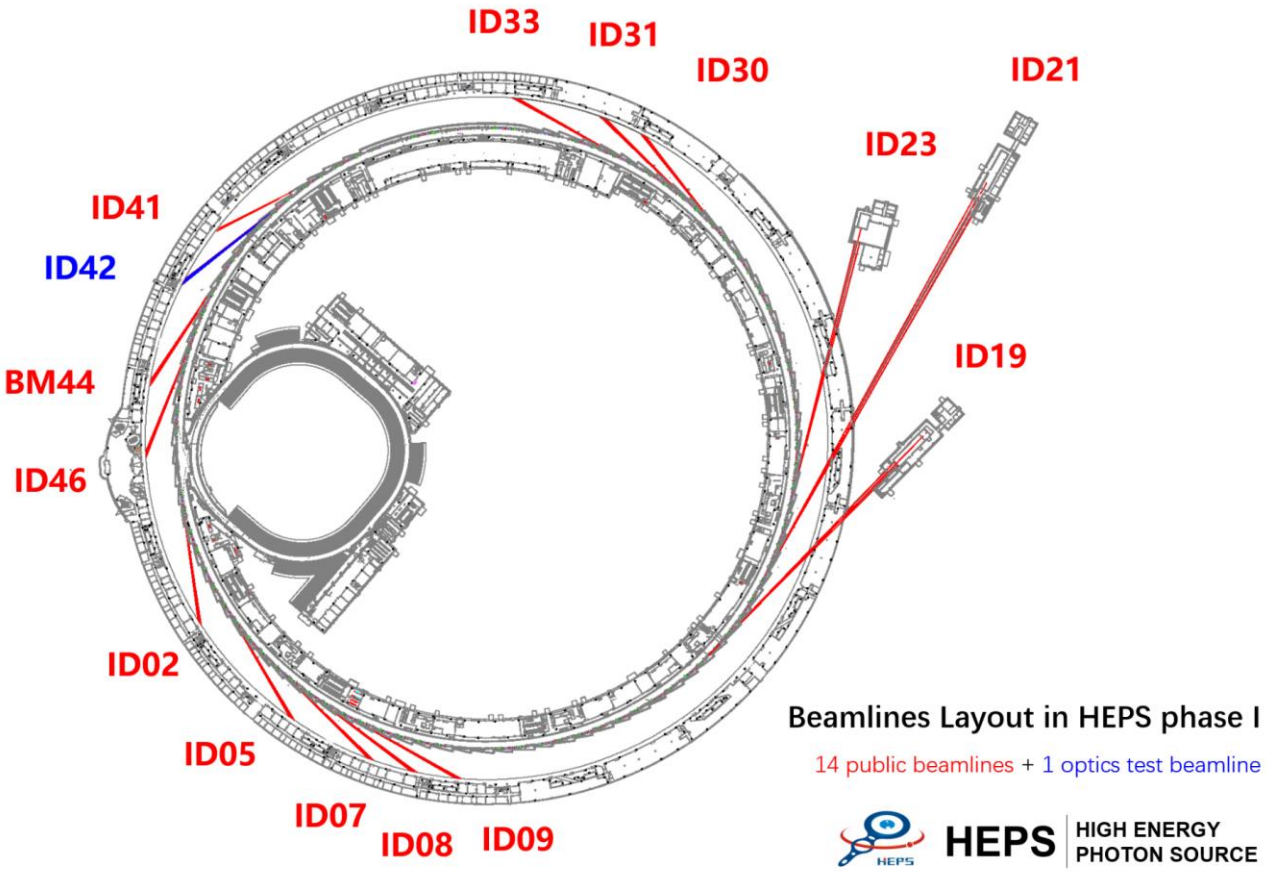
高能同步辐射光源（HEPS）

- 第四代同步辐射光源 — 高能量，高亮度，低发射度
- 将与世界上正在运行的美国 APS、欧洲 ESRF、日本 Spring 8 、德国 PETRA III 一起，构成世界五大高能同步辐射光源
- 位于北京怀柔科学城，核心装置，距离中科院高能所 80 km
- 建设周期6.5年，2018年底开始建设，将于2025年底验收投入运行

Main parameters	Unit	Value
Beam energy	GeV	6
Circumference	m	1360.4
Emittance	pm·rad	< 60
Brightness	phs/s/mm^2/mrad^2/0.1%BW	>1x10^{22}
Beam current	mA	200
Injection		Top-up



HEPS 一期光束线站

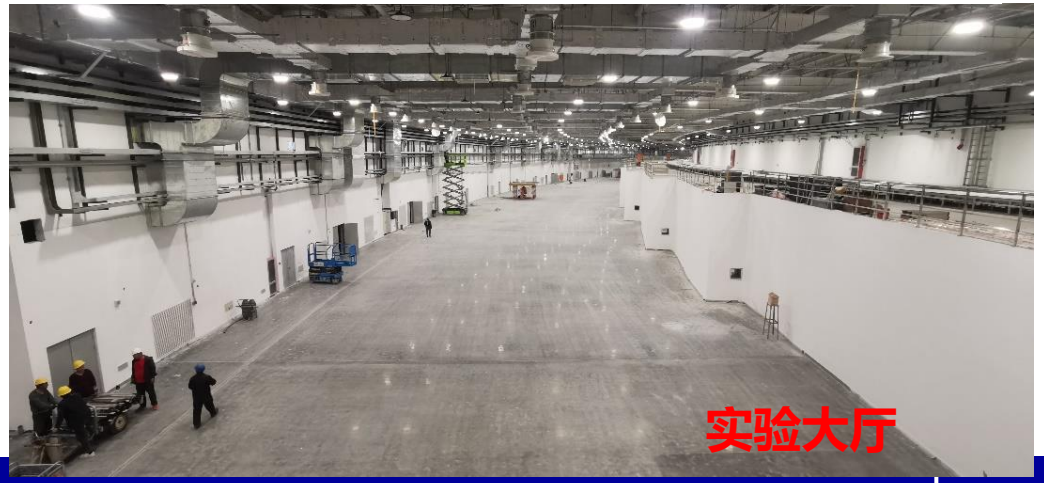


生物大分子微晶衍射线站	-ID02	ID30-	X射线显微成像线站
低维结构探针线站	-ID05	ID31-	高压线站
工程材料线站	-ID07	ID33-	高分辨谱学线站
粉光小角散射线站	-ID08	ID41-	高分辨纳米电子结构线站
硬X射线相干散射线站	-ID09	ID42-	光学测试线
硬X射线纳米探针线站	-ID19	BM44-	通用环境谱学线站
硬X射线成像线站	-ID21	ID46-	X射线吸收谱学线站
结构动力学线站	-ID23		辅助实验室

一期将建设14条公共光束线站 + 1 条光学测试线站

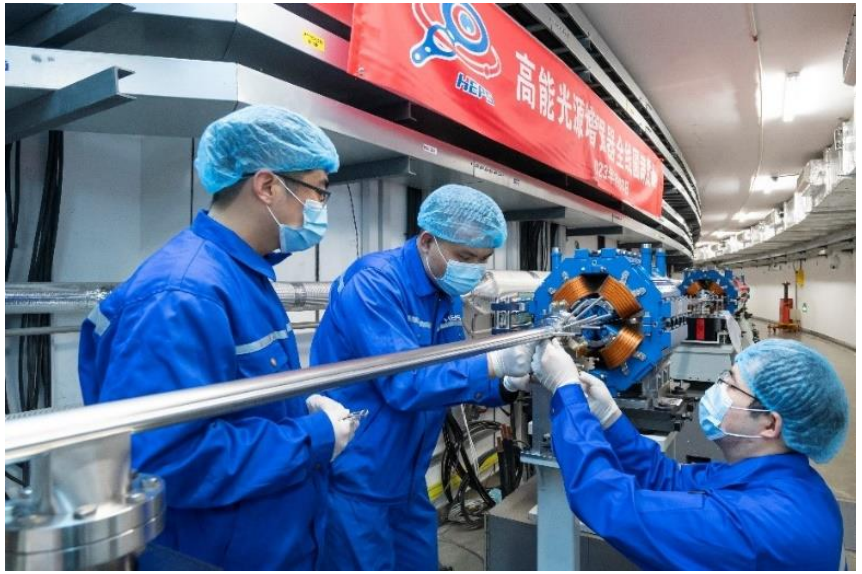
涉及成像、衍射/散射、谱学等学科

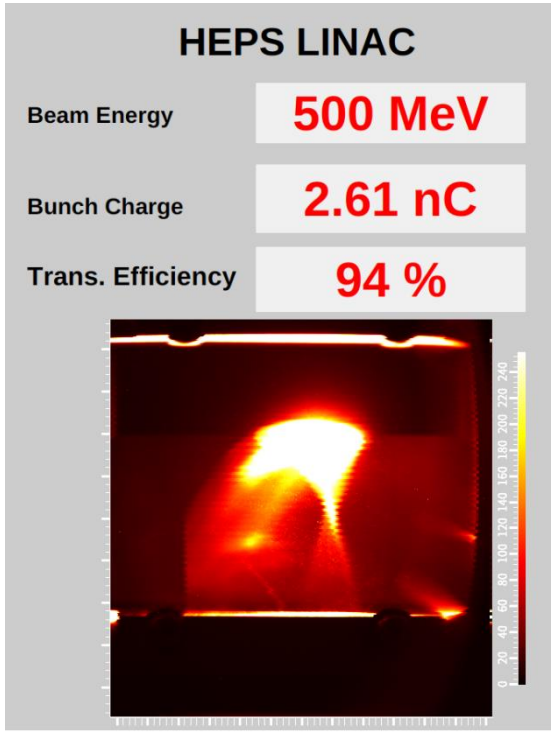
HEPS 总共可以容纳超过 90 条线站



HEPS 进展

- 目前已经完成全部土建结构施工，进入设备安装阶段
- 2023.01, HEPS 增强器全线贯通
- 2023.02, 启动储存环隧道设备安装
- 2023.03, HEPS直线加速器满能量出束，成功将第一束电子束加速到500 MeV





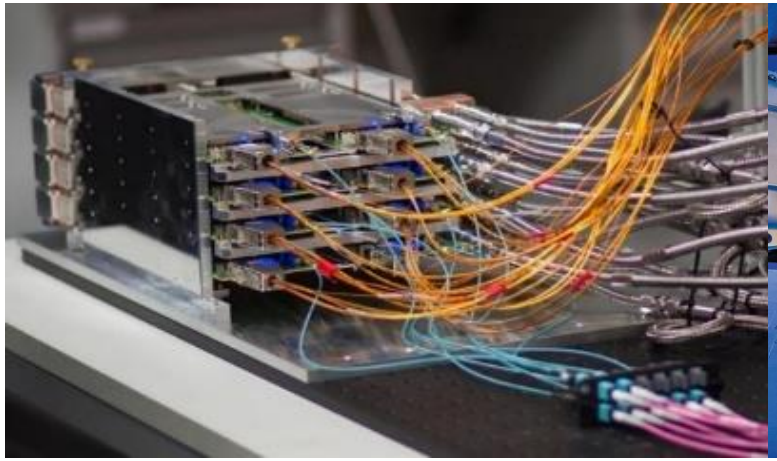
Outline

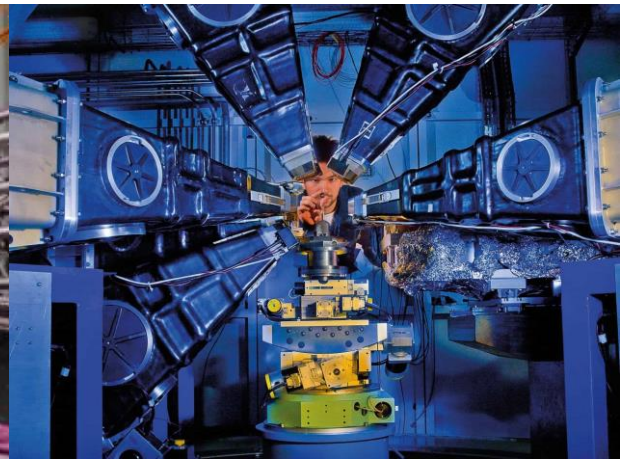


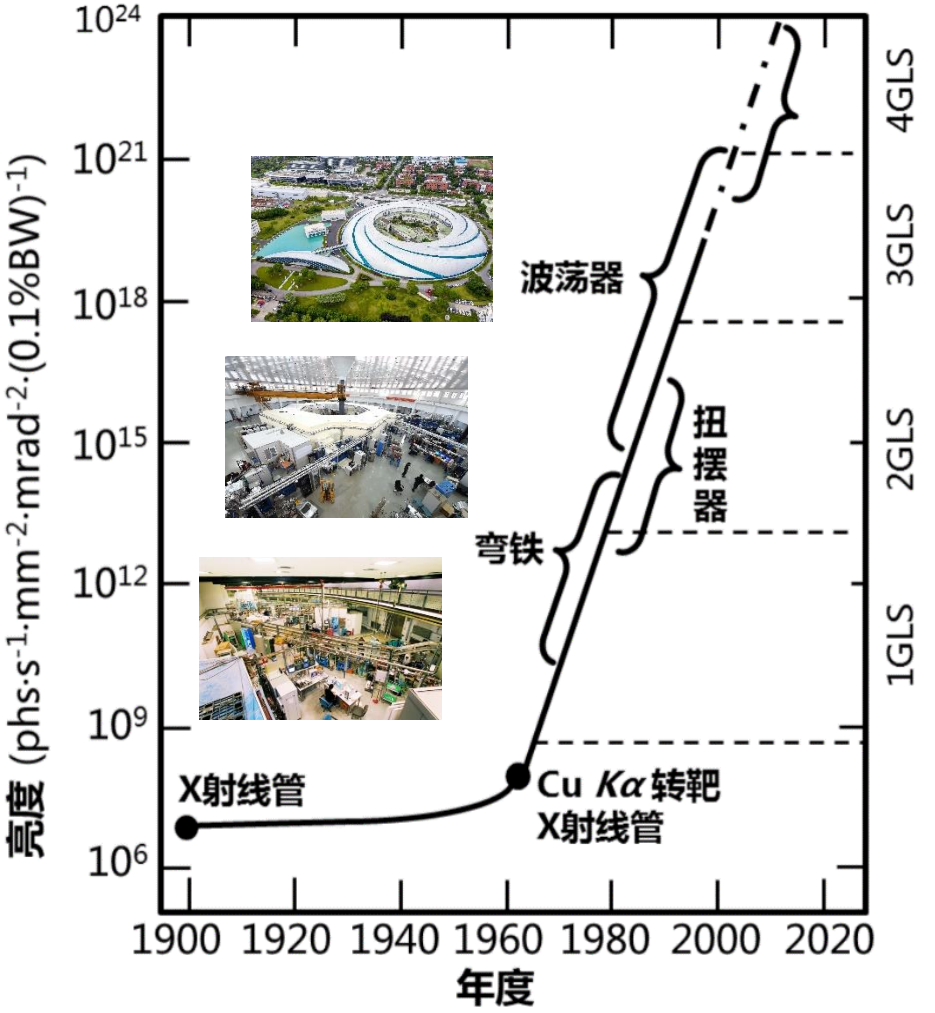
1. 背景介绍
2. **科学数据与软件系统的需求与挑战**
3. 软件框架的架构与设计
4. 基于软件框架的科学软件开发
5. 科学软件示例
6. 总结

Data Challenges @HEPS

- 四代光源亮度相对三代光源提升了 2-3 个量级
  - 在更短的时间内产生更加海量的具有更多细节信息的原始数据
- X 射线探测器能力不断提高:
  - 更宽的动态范围，更快的读出速率，更大的像素阵列
  - 更大的帧数，更高的帧率=更多的原始数据



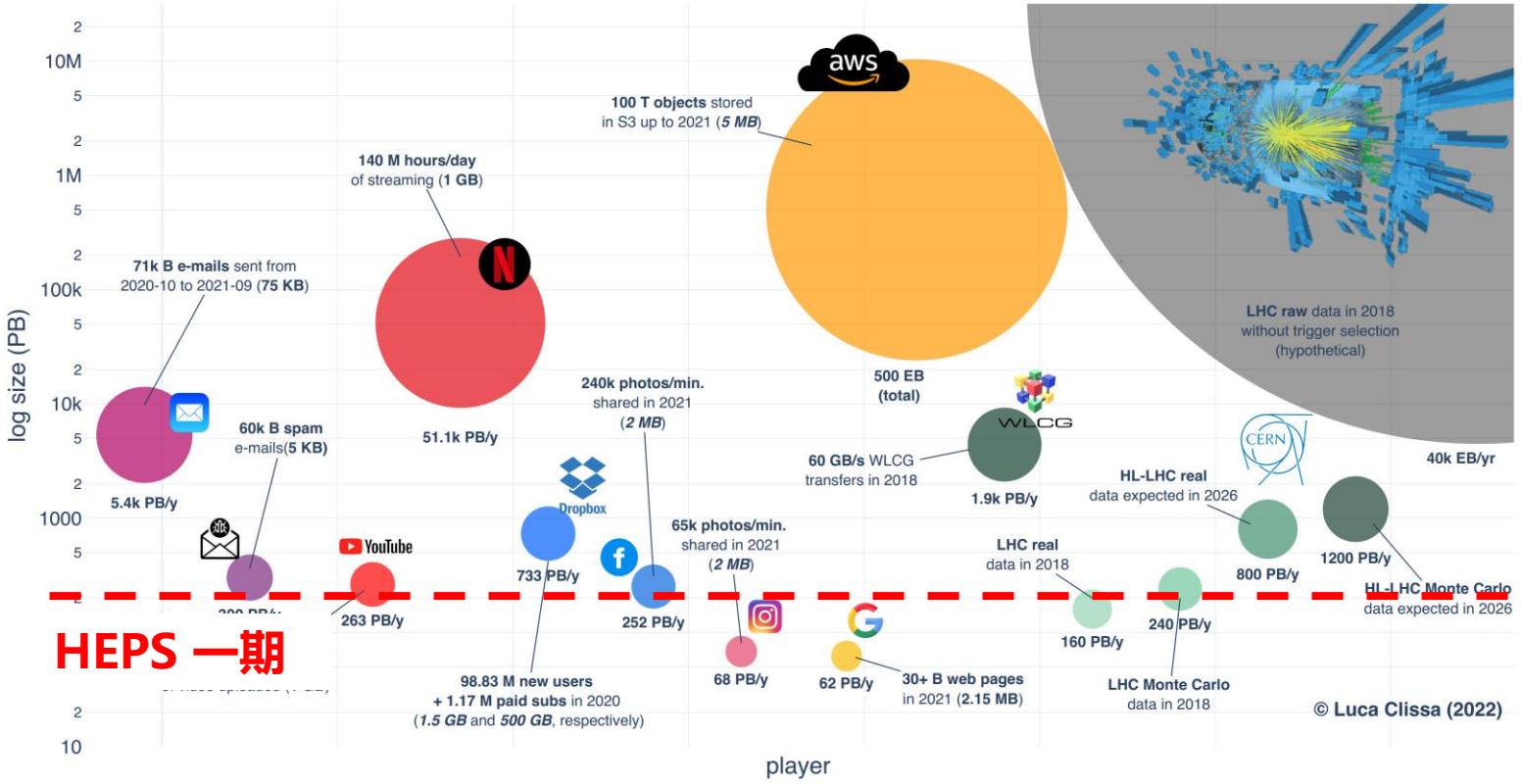




同步辐射光源的发展

Data Challenges @HEPS

- HEPS 一期(15个线站)数据产生率接近 PB/天，每年将产生超过 200PB 的数据
- HEPS 总共能容纳超过90条线站，更多的数据

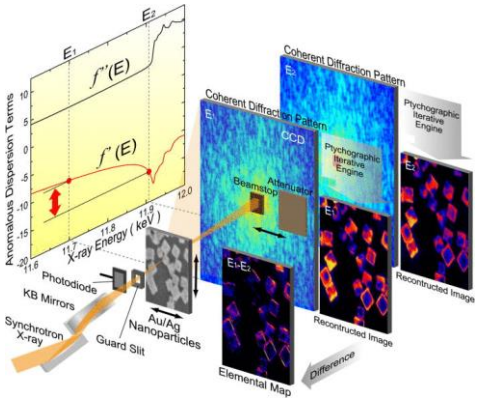


Data volume of HEPS Beamlines:

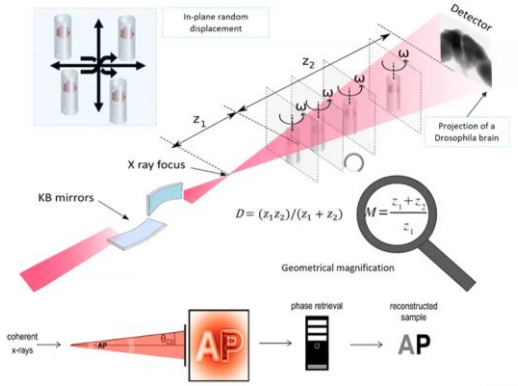
线站	峰值产生 (TB/day)	平均产生 (TB/day)
B1 工程材料	600.00	200.00
B2 硬 X 射线纳米探针	500.00	200.00
B3 结构动力学	8.00	3.00
B4 硬 X 射线相干散射	10.00	3.00
B5 高分辨谱学	10.00	1.00
B6 高压	2.00	1.00
B7 硬 X 射线成像	1000.00	250.00
B8 X 射线吸收谱学	80.00	10.00
B9 低维结构探针	20.00	5.00
BA 生物大分子微晶衍射	35.00	10.00
BB 粉光小角散射	400.00	50.00
BC 高分辨纳米电子结构	1.00	0.20
BD 通用环境谱学	10.00	1.00
BE X射线显微成像	25.00	11.20
BF 光学测试	1000.00	60.00
Total average:		**805**

Data Challenges @HEPS

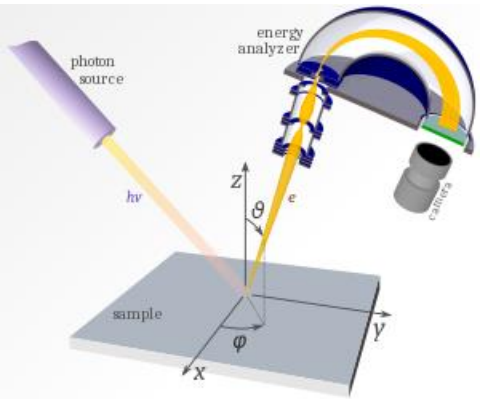
- X射线光源技术的发展，不断催生出更加复杂的新方法、新技术以及新研究领域，需要新的学科软件及算法
- 多模态实验需要结合多个样本、技术和设备的数据
- 原位和动态加载实验需要实时反馈和自主控制
- 不同的光束线站以及科学目标，其实验数据通量和容量存在巨大差异



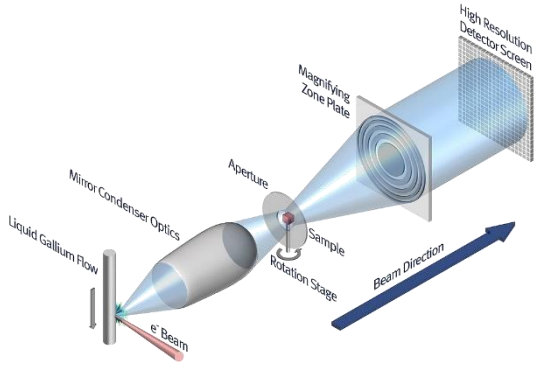
Fluorescence mapping



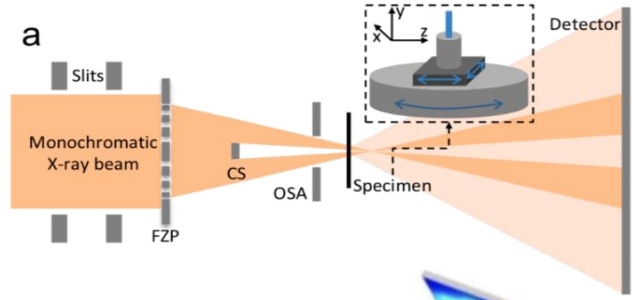
Nanoholotomography



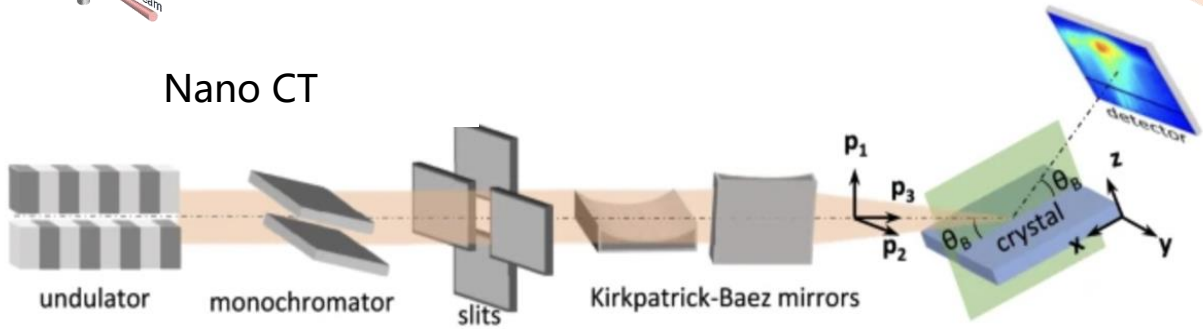
ARPES



Nano CT



Ptychography CT



Bragg ptychography

Data Challenges @HEPS

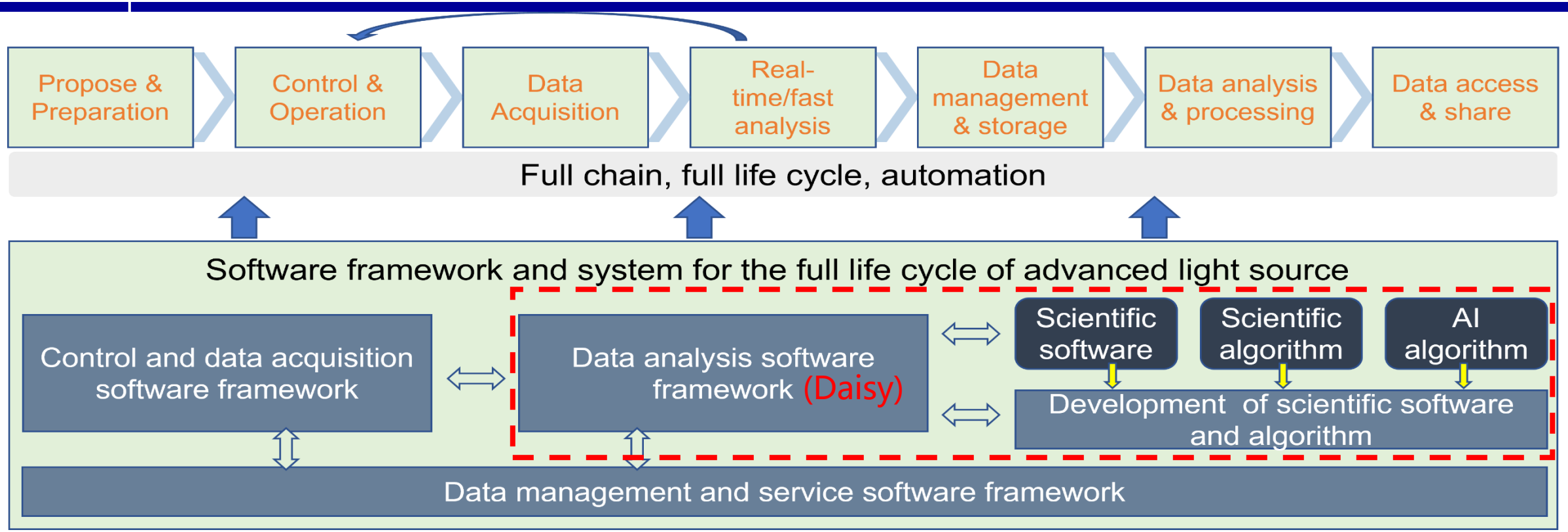
- 先进光源上大规模科学数据的分析和管理变得越来越具有挑战性
- 需要开发和集成先进的分析和管理工具
  - 提供海量科学数据的存储、组织和管理
  - 为方法学软件和算法的多样化发展提供通用的底层软件框架支持
  - 在实验过程中，进行实时数据分析和快速反馈，提供决策指导和修正实验过程，并优化数据采集
  - 在实验结束后，处理海量多模态数据，帮助用户快速完成实验数据分析、获取科研成果，加速科学发现
  - 提供可伸缩的分布式异构算力支持，满足不同科学目标不同规模的计算分析需求

Outline



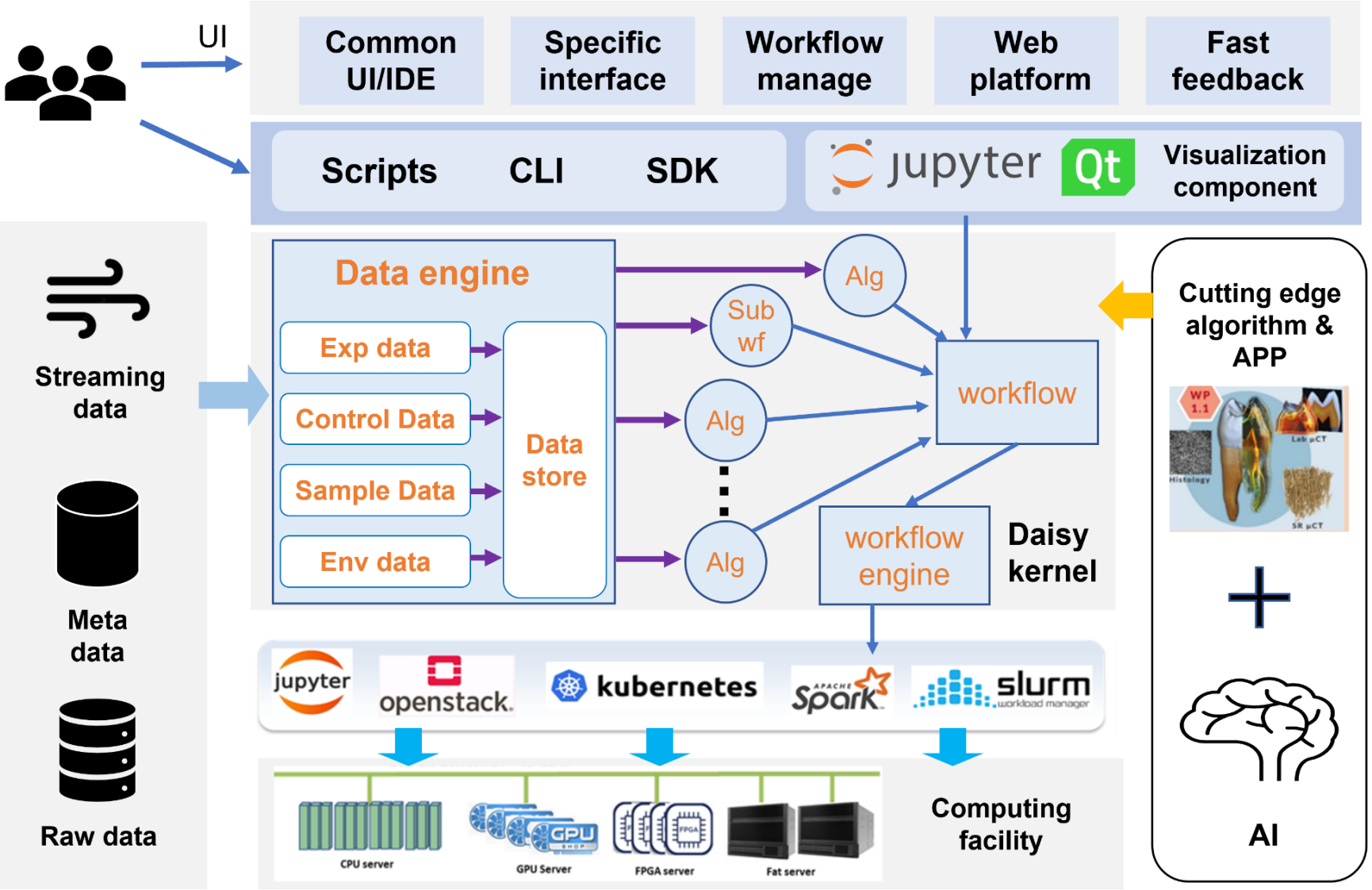
1. 背景介绍
2. 科学数据与软件系统的需求与挑战
3. **软件框架的架构与设计**
4. 基于软件框架的科学软件开发
5. 科学软件示例
6. 总结

光源全生命周期的软件系统



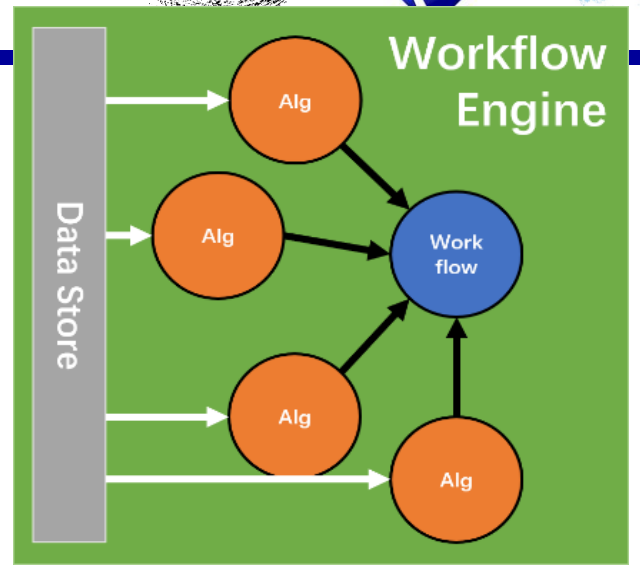
- 设计了先进光源全链条、全生命周期的软件框架和系统
- 促进先进光源软件系统全流程的智能化、自动化，实现科学数据全生命周期的跟踪和管理
- 支持发展光束线站数据分析新方法、新软件，以及已有方法学和软件的标准化框架集成
- 建立有影响力的开源软件社区，吸引并支持潜在的社区贡献者，建立大科学装置全生命周期、多设施协作、多学科融合的软件生态环境

数据处理软件框架总体架构



- 数据处理软件框架核心
- 满足新一代光源数据处理需求的衍生技术模块
  - 应对高通量数据I/O、多模态数据解析、多源数据接入的数据对象管理
  - 应对不同规模、不同通量、低延迟数据处理需求的弹性异构计算集群算力支持
  - 服务于学科方法学软件集成和发展的用户软件接口和软件开发环境支持
- 基于软件框架的学科专用应用软件以及针对灵活数据处理需求的通用工作流编排系统

数据处理软件框架核心

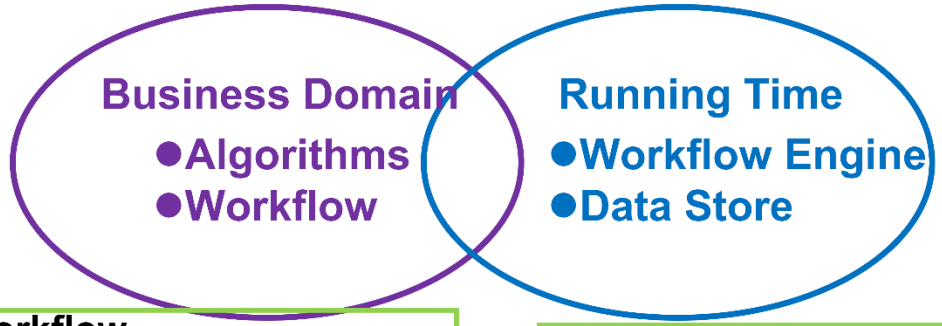


- 遵循**领域驱动设计**的概念，从领域知识中提取出与实现技术无关的**领域模型**，保证业务抽象性和独立性，并建立领域模型之间的关系，形成**领域架构**。
- 一个**领域模型**对应开发一个或多个**算法或子工作流**，对多个算法或子工作流的**有序调用**对应**领域框架的实现**。

- **算法**：框架中的最小单元，定义领域模型，具体的数据处理模块，支持第三方程序库集成。同一个算法可以有多个实体实现，相互替代。
- **工作流**：定义领域架构，通过调用一系列算法完成特定的处理分析任务，支持嵌套。
- **工作流引擎**：根据计算环境提供的计算资源，在运行时判断算法的具体执行实体，同时管理算法模块的并行分布式执行。解除了业务流程和计算环境之间的耦合。
- **数据仓库**：管理算法之间数据对象的创建和传递，使业务代码不用考虑数据对象的管理问题。

Algorithms
- Input Data Processing
- Output Data Defined

Workflow Engine
- Handle Data Store
- Running Time Management



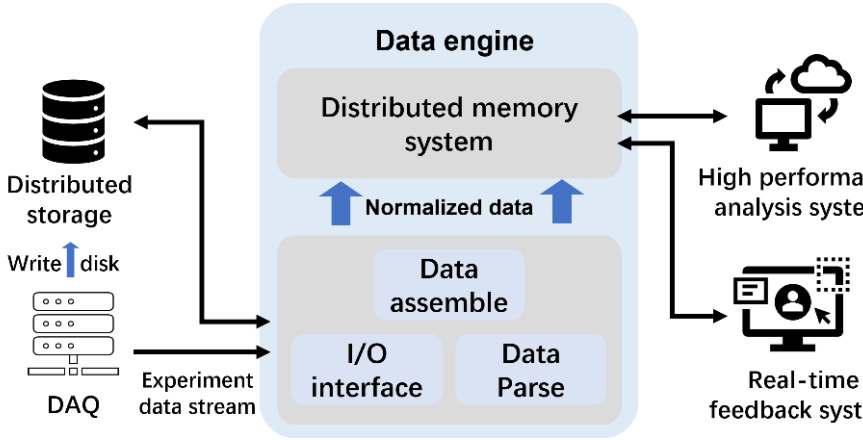
Workflow
- A sequence of Algorithm
- Workflow is also an algorithm

Data Store
- Data Object Management

高性能模块

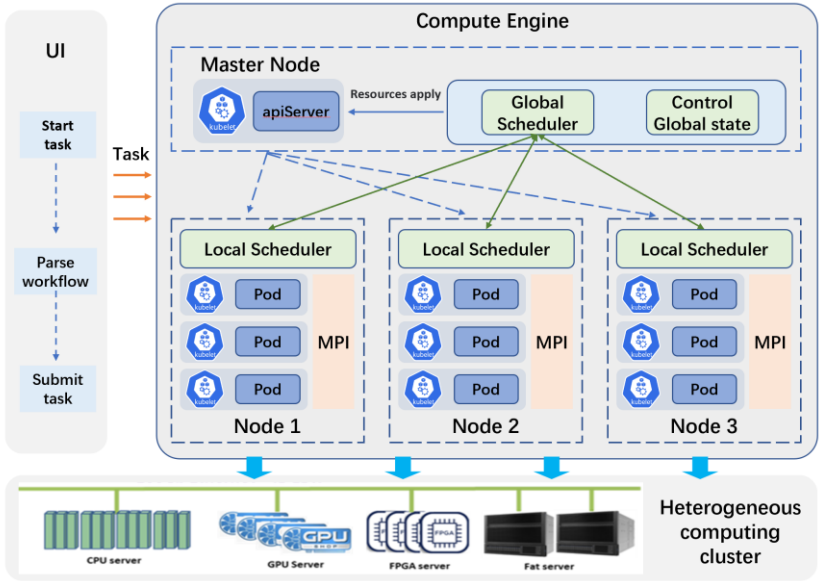
数据对象管理

- 提供统一的读写接口，屏蔽底层架构，光源数据格式，数据来源的差异性
- 支持磁盘文件和流数据的读写，实现实时、高通量的在线数据处理
- 采用异步并行、分布式内存、自适应存储参数和数据压缩等方法优化数据的读写性能



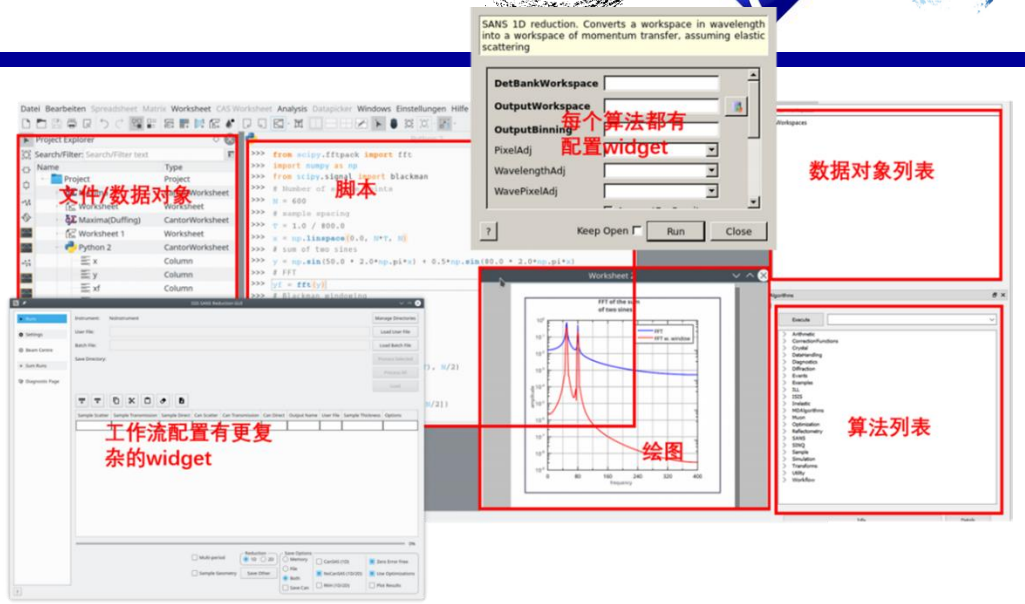
异构分布式算力支持

- 充分利用先进计算基础设施
- 针对计算热点，形成高性能数值分析计算库
- 提供统一灵活的并行编程接口API，屏蔽底层硬件资源的体系结构差异，降低数据分析软件的并行编程复杂度
- 提供分布式计算任务调度器，保证分布式任务的高可用和高效率



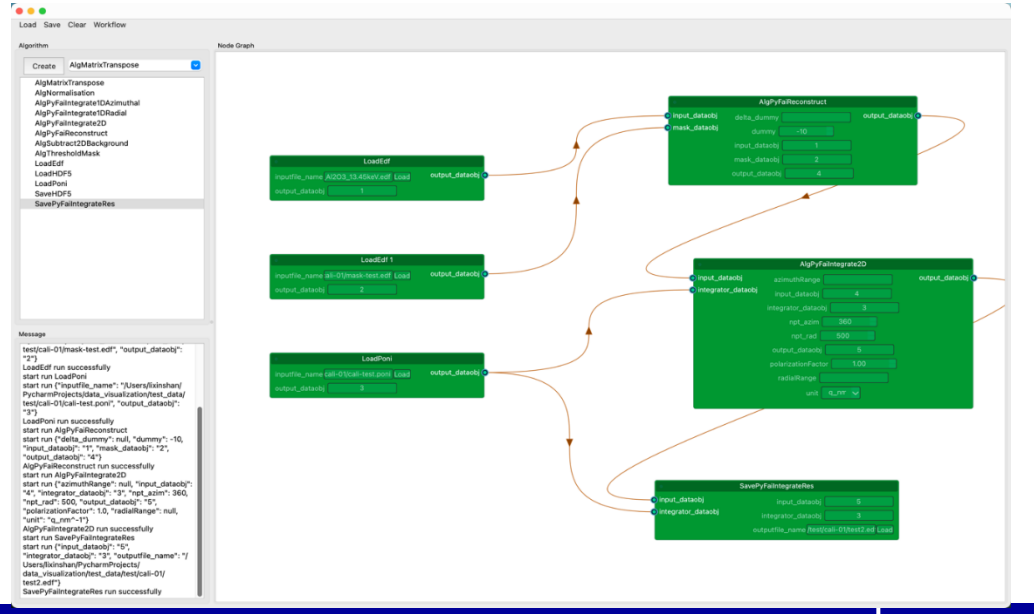
面向学科应用的用户软件接口

- **提供多种形式的用户软件接口，支持不同场景的方法学应用**
  - 数据可视化界面
  - 集成开发环境界面
  - 方法学接口界面
  - Web 数据分析平台
  - 脚本和命令行接口



- **提供多种可复用的图形化界面常见控件，支持各实验站在此基础上进行二次开发**
  - 分析绘图桌面，算法配置控件，工作流配置控件，算法/工作流对象列表，数据对象列表，IDE……

- **用于灵活通用数据处理任务的 workflow 管理系统**
  - 前端提供App和Web端的图形界面，支持交互式的工作流编排、导入、导出和运行监控等
  - 遵循 Common Workflow Language(CWL) 标准，支持CWL的解析和生成



开发用户者支持

版本控制

- Git 控制版本，Gitlab 托管项目代码、连接CI/CD

运行环境

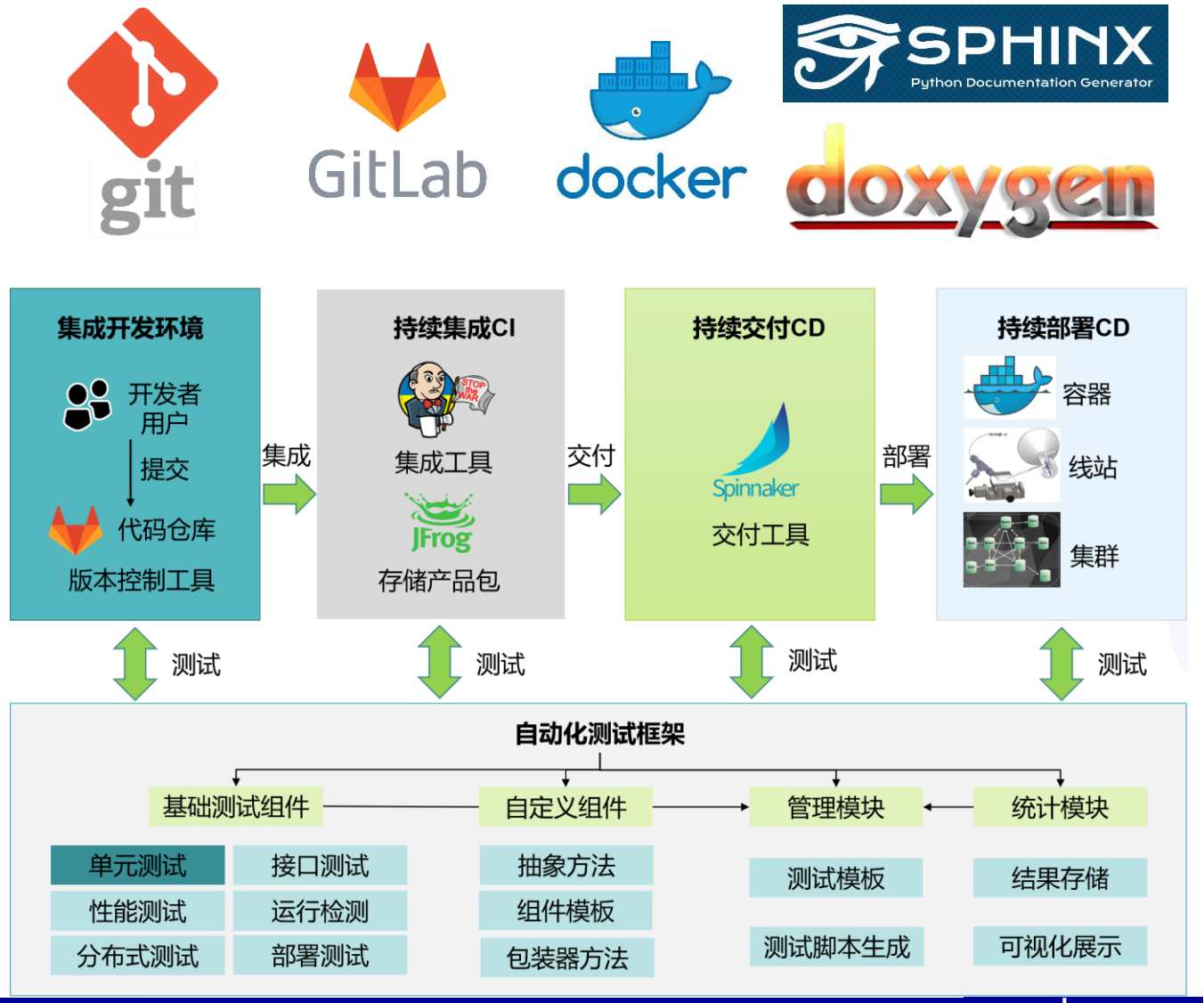
- 容器封装基础运行环境，CVMFS 部署编译好的软件

文档指南

- 用户使用文档，开发者开发指南
- Jupyterbook，Sphinx，readthedocs
- Doxygen 根据注释自动生成代码文档

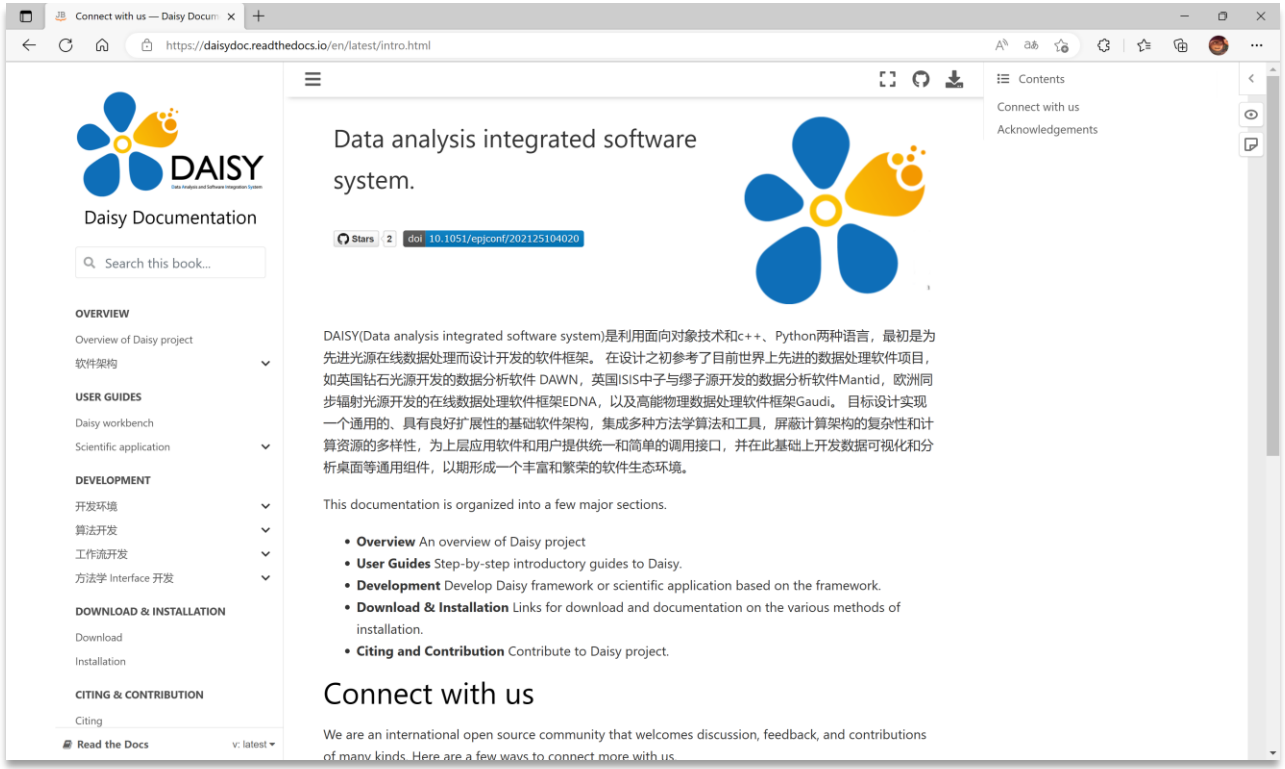
软件开发的可持续集成、部署(CI/CD)和测试框架

- 支持软件的自动化集成、编译、测试和部署，降低开发门槛
- 实现软件开发过程的标准化、自动化和智能化



研究现状

- 设计实现了通用科学数据处理基础软件框架 Daisy（Data analysis integrated software system）
- 向外提供了四类基础编程接口：
  - 算法模块和工作流模块：实现领域方法学模型和具体业务逻辑
  - 工作流引擎模块和数据仓库模块：管理软件运行时环境和数据对象
- 多种图形化应用接口：通用用户界面，领域方法学集成接口，web 应用接口、用户开发 IDE
- 代码和容器镜像开源，发布了网页版用户文档，建立了丰富的内部知识库
- 集成了十几个学科方法学软件与算法，开发了多个学科方法学应用

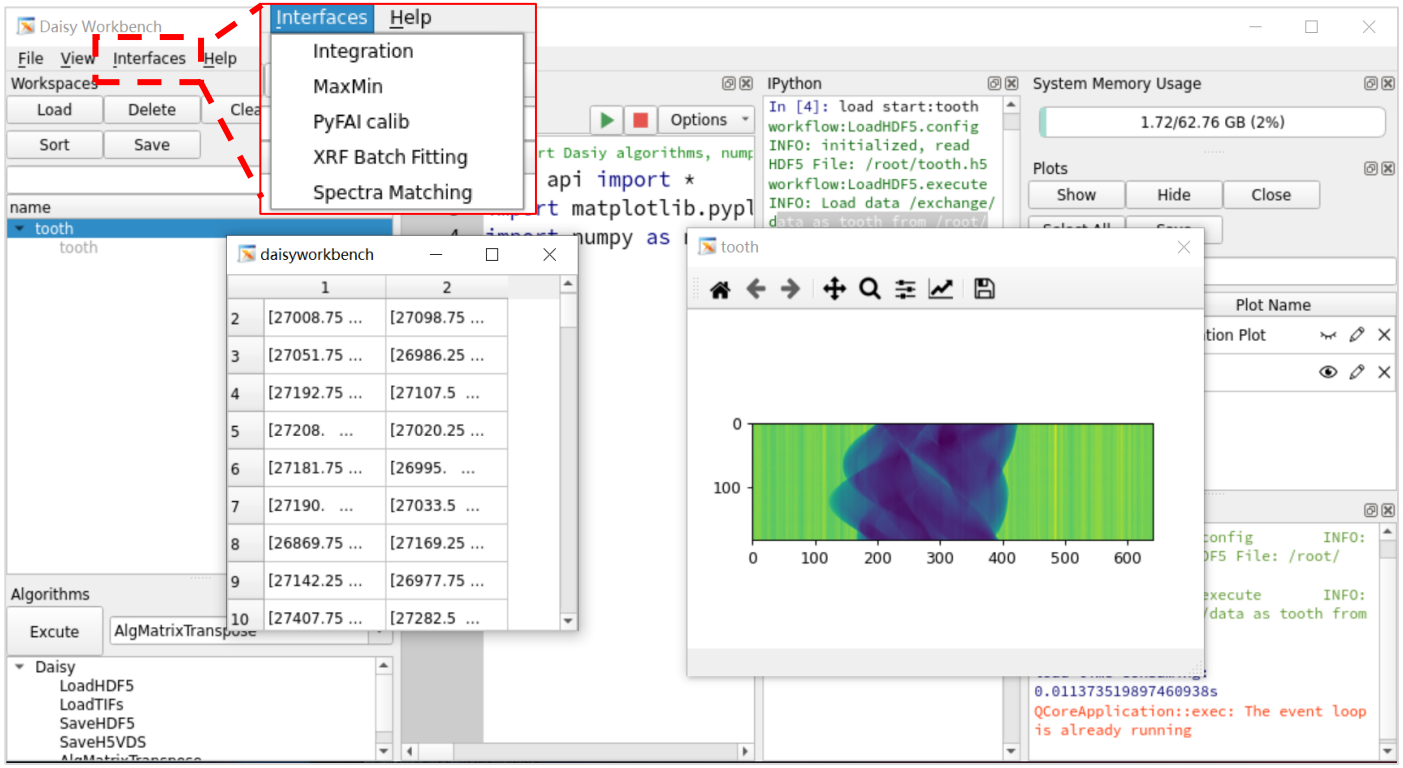


User documentation：

https://daisydoc.ihep.ac.cn/

Yu Hu et al. EPJ Web of Conferences 251, 04020 (2021).

Daisy graphical user interface



Daisy workbench:

- 通用用户界面，基于 PyQt5
- 包含数据对象列表，算法列表，数据展示/可视化，Log信息，系统内存监控，以及提供给开发者的 IDE 等模块
- 面向多样化学科方法学的专用用户界面接口

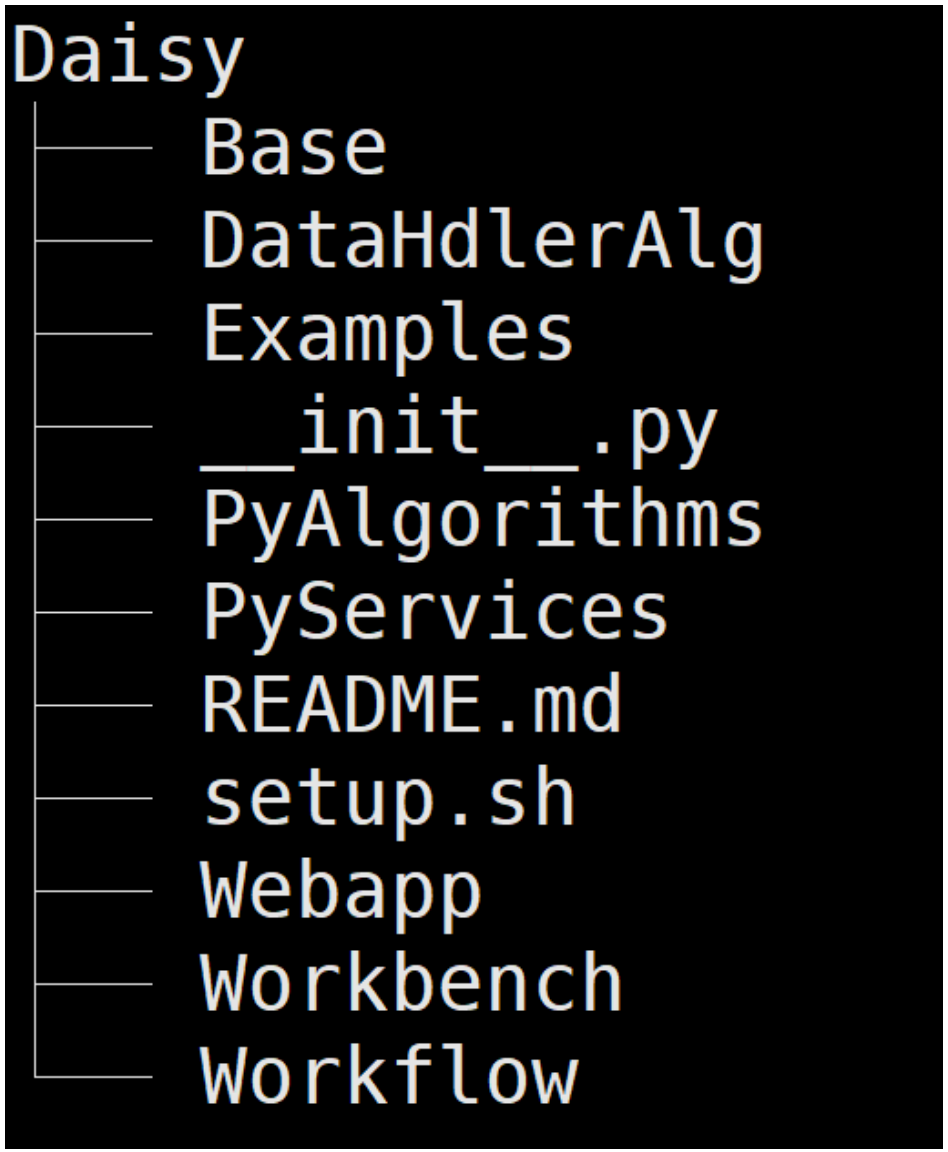
Web data analysis platform:

- 基于 Jupyterlab 生态，开箱即用
- 利用容器技术封装开发和运行时环境
- 通过K8s提供弹性可伸缩的计算资源
- 终端+用户友好界面，适合不同专业程度的用户

Outline

1. 背景介绍
2. 科学数据与软件系统的需求与挑战
3. 软件框架的架构与设计
4. **基于软件框架的科学软件开发**
5. 科学软件示例
6. 总结

Daisy代码组织结构



- **Base**：Daisy 的核心基类，包括data store, algorithm, workflow，workflow engine以及service等
- **PyAlgorithms**：领域方法学模型即数据处理逻辑的具体实现，如积分算法，转轴矫正算法等。不同学科的代码将有更细致的组织
- **Workflow**: 用户完整业务逻辑的实现，如CT图像重建，材料原子结构定量分析及建模，数据产品生成等
- **DataHdlerAlg**: 不同类型数据 I/O 的实现，如HDF5，Tiff，fits等
- **Workbench**: 通用用户界面
- **Webapp**: 基于 web 的应用软件
- **PyServices**: 外部公共服务，如条件数据库查询等
- **Examples**: 用户开发示例

方法学应用开发示例 —— 算法实现（Python）

```python
import numpy as np
import tomopy
from Daisy import DaisyAlg
class AlgTomopyRecon(DaisyAlg):
    def __init__(self, name):
        super().__init__(name)

    def initialize(self):
        self.data = self.get("DataStore").data()
        self.LogInfo("initialized, Tomopy Reconstruction")
        return True

    def execute(self, input_dataobj, theta, center, alg_type, output_dataobj):
        projs  = self.data[input_dataobj]
        thetas = self.data[theta]
        dataobj = tomopy.recon(projs, thetas, center=center, algorithm=alg_type)
        self.data[output_dataobj] = dataobj
        return True

    def finalize(self):
        self.LogInfo("finalized")
        return True
```

A

继承算法基类，实现 initialize、execute 和 finalize 三个方法。方法学的数据处理逻辑在 **execute** 中实现

一个算法一般包括一组**输入数据对象**，一组**输出数据对象**，一组**计算参数**。

从数据仓库中拿到输入数据

实现具体的数据处理逻辑

将处理结果写回数据仓库，写回的数据可被其它模块访问，或保存到输出文件

方法学应用开发示例 —— Workflow 实现（Python）

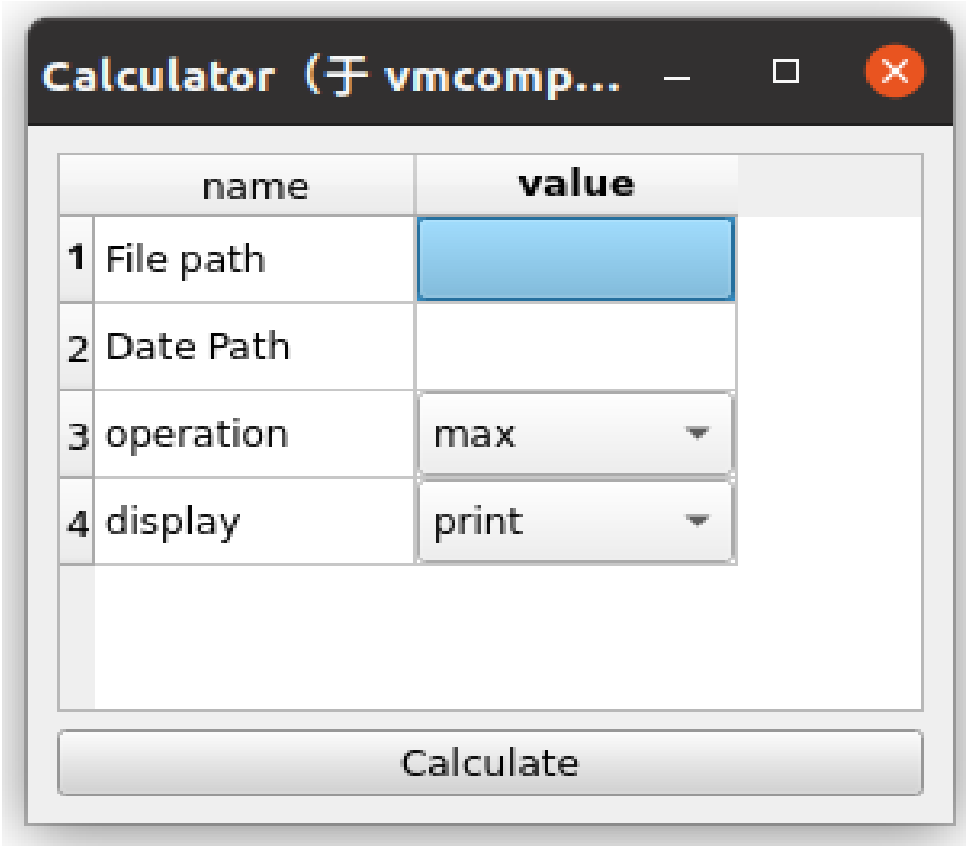
```
@Daisy.Singleton
class WorkflowCTReconstruct(Daisy.PyWorkflow):                    B
 def execute(self):
  self.engine['loadhdf5'].execute(input_path='/entry/tomo', output_dataobj='tomodata')
  self.engine['loadhdf5'].execute(input_path='/entry/dark', output_dataobj='darkdata')
  self.engine['loadhdf5'].execute(input_path='/entry/flat', output_dataobj='flatdata')
  self.engine['normalize'].execute(projs_dataobj='tomodata',darks_dataobj='darkdata',\
                                   flats_dataobj='flatdata', output_dataobj='normdata')
  self.engine['angles'].execute(input_dataobj='normdata', output_dataobj='thetas')
  self.engine['minuslog'].execute(input_dataobj='normdata', output_dataobj='mlogdata')
  self.engine['reconstruct'].execute(input_dataobj='mlogdata', theta='thetas',\
                                     center=1030, alg_type='fbp',output_dataobj='recodata')
  self.engine['savehdf5'].execute(input_dataobj='recodata',output_path='/entry/reco')

wf = WorkflowCTReconstruct('WorkflowCTReconstruct')
wf.initialize(workflow_engine='PyWorkflowEngine', \
              workflow_environment = init_dict, algorithms_cfg = cfg_dict)
wf.execute()

data =wf.data_keys()
algs =wf.algorithm_keys()

wf.finalize()
```

工作流和算法一样，实现 initialize、execute 和 finalize 三个方法。在**execute**方法中按序列调用算法或者子工作流，实现具体的业务逻辑。

工作流初始化时需要指定工作流引擎，工作流引擎通过算法名创建和调用特定算法和工作流模块，并通过数据仓库管理算法所需的输入输出数据对象。

算法对象初始化参数列表和算法运行参数可以由 JSON 对象或者 Python 字典对象表示

方法学应用开发示例 —— 方法学 interface 开发



Interface 功能：

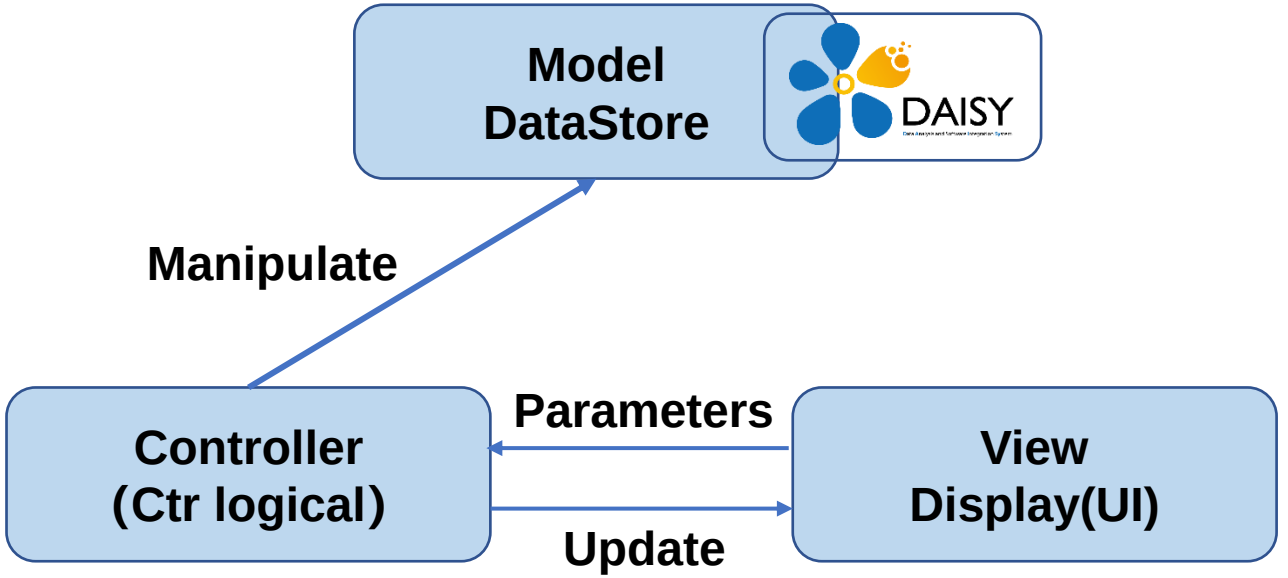
1.调用 Daisy 算法从 HDF5 文件中读取矩阵数据。

2.调用 Daisy 算法求矩阵数据的最大值或最小值。

3.打印结果。

建议开发设计模式：MVC/MVP

- Model 是业务模型，View 层是界面，Controller 用来调度 View 和 Model。
- 用户界面和业务逻辑分离，使得一个程序可以使用不同的表现形式，同一个界面下面也可以有不同的业务实现，使得代码具有可扩展性、可复用性，维护方便。
- 本例中 Daisy 的数据处理逻辑即包含在 Model。



方法学应用开发示例 —— 方法学 interface 开发



开发流程：

1. 在 Daisy/Workbench/windows/interface/ 目录下创建该 interface 的子目录，本例为 Calculator/。
2. Calculator 目录下创建必要的文件并实现相应的代码，本例的代码目录如下：

```
1 ├── CalculatorManager.py
2 ├── Controller.py
3 ├── __init__.py
4 ├── Model.py
5 └── View.py
```

CalculatorManager.py文件为该 interface 的入口程序。

CalculatorManager.py：

```python
from windows.interface.Calculator.Model import Model
from windows.interface.Calculator.View import View
from windows.interface.Calculator.Controller import Controller

"""
A wrapper class for setting the main window of the interface
"""
class CalculatorManager(QtWidgets.QMainWindow):

    def __init__(self, parent=None):
        super(CalculatorManager, self).__init__(parent)  # noqa
        self.setWindowFlag(Qt.Window)
        self.setAutoFillBackground(True)

        self.window = QtWidgets.QMainWindow()
        demo_view = View()
        demo_model = Model()
        # create controller
        self.controller = Controller(demo_view, demo_model)
        # set the view for the main window
        self.setCentralWidget(demo_view)
        self.setWindowTitle("Calculator")

    def show(self) -> None:
        super(CalculatorManager, self).show()
        self.activateWindow()
```

方法学应用开发示例 —— 方法学 interface 开发



开发流程：

1. 在 Daisy/Workbench/windows/interface/ 目录下创建该 interface 的子目录，本例为 Calculator/。
2. Calculator 目录下创建必要的文件并实现相应的代码，本例的代码目录如下：

```
1 ├── CalculatorManager.py
2 ├── Controller.py
3 ├── __init__.py
4 ├── Model.py
5 └── View.py
```

Model.py 调用Daisy 算法’ LoadHDF5’ 和 ‘MaxMin’ 实现HDF5 文件的加载以及矩阵数据极值的求解

Model.py：

```python
from api import work_flow

class Model(object):
    def __init__(self):
        # Get the algorithms from Daisy workflowengine
        self.MaxMin = work_flow.engine['MaxMin']
        self.Loadh5 = work_flow.engine['LoadHDF5']

    def CalMaxmin(self, Filepath, DataPath, operation):
        # Load matrix data from the HDF5 file
        self.Loadh5.config({'inputfile_name': Filepath})
        self.Loadh5.execute(input_path = DataPath, output_dataobj='data_h5')
        # Calculate the maximum or minimum of the matrix
        if operation == "max":
            self.MaxMin.execute(input_dataobj='data_h5',  output_dataobj='maxmin', showMin=False)
        elif operation == "min":
            self.MaxMin.execute(input_dataobj='data_h5',  output_dataobj='maxmin', showMin=True)
        # Get the result from the datastore
        self.result = work_flow.engine.datastore['maxmin']
        return self.result
```

方法学应用开发示例 —— 方法学 interface 开发



开发流程：

1. 在 Daisy/Workbench/windows/interface/ 目录下创建该 interface 的子目录，本例为 Calculator/。
2. Calculator 目录下创建必要的文件并实现相应的代码，本例的代码目录如下：

```
1 ├── CalculatorManager.py
2 ├── Controller.py
3 ├── __init__.py
4 ├── Model.py
5 └── View.py
```

View.py 添加窗口小部件实现图形化用户界面

View.py：

```
class View(QtWidgets.QDialog):
    # In PyQt signals are the first thing to be defined in a class:
    displaySignal = QtCore.pyqtSignal()
    btnSignal = QtCore.pyqtSignal()

    def __init__(self, parent=None):
        # Call QDialog's constructor
        super(View, self).__init__(parent)

        # Initialise the widgets for the view (this can also be done from Qt Creator
        self.table = QtWidgets.QTableWidget()
        self.table.setWindowTitle("MVP Demo")
        self.table.resize(600, 250)
        self.table.setRowCount(5)
        self.table.setColumnCount(2)
        self.table.setHorizontalHeaderLabels("name;value;".split(";"))

        self.FilePath = ''

        # Set display values in the widgets
        keys = ['File path', 'Date Path', 'operation',  'display', 'result']
        self.combo = {}
        self.create_File_selction(0, 1, 'File')
        self.create_combo_table(2, 1, 'operations')
        self.create_combo_table(3, 1, 'display')
        for row in range(len(keys)):
            self.set_names(keys[row], row)

        # Initialise layout of the widget and add child widgets to it
        grid = QtWidgets.QGridLayout()
        grid.addWidget(self.table)
```

方法学应用开发示例 —— 方法学 interface 开发



开发流程：

1. 在 Daisy/Workbench/windows/interface/ 目录下创建该 interface 的子目录，本例为 Calculator/。
2. Calculator 目录下创建必要的文件并实现相应的代码，本例的代码目录如下：

```
1 ├── CalculatorManager.py
2 ├── Controller.py
3 ├── __init__.py
4 ├── Model.py
5 └── View.py
```

Controller.py 连接 Model 和 View 模块，实现界面控制逻辑

Controller.py：

```python
class Controller(object):
    # Pass the view and model into the presenter
    def __init__(self, demo_view, demo_model):
        self.model = demo_model
        self.view = demo_view

        # Define the initial view
        # Note that, in the view, the drop-down could be replaced with a set of
        # tick boxes and this line would remain unchanged - an advantage of
        # decoupling the presenter and view
        self.view.set_options('operations', ['max', 'min'])
        self.view.set_options('display', ['print', 'update', 'print and update'])
        self.printToScreen = True
        self.view.hide_display()

        # Connect to the view's custom signals
        self.view.btnSignal.connect(self.handle_button)
        self.view.displaySignal.connect(self.display_update)

    # The final two methods handle the signals
    def display_update(self):
        display = self.view.get_display()
        if display == 'update':
            self.printToScreen = False
            self.view.show_display()
        elif display == 'print':
            self.printToScreen = True
            self.view.hide_display()
        else:
            self.printToScreen = True
            self.view.show_display()
```

方法学应用开发示例 —— 方法学 interface 开发

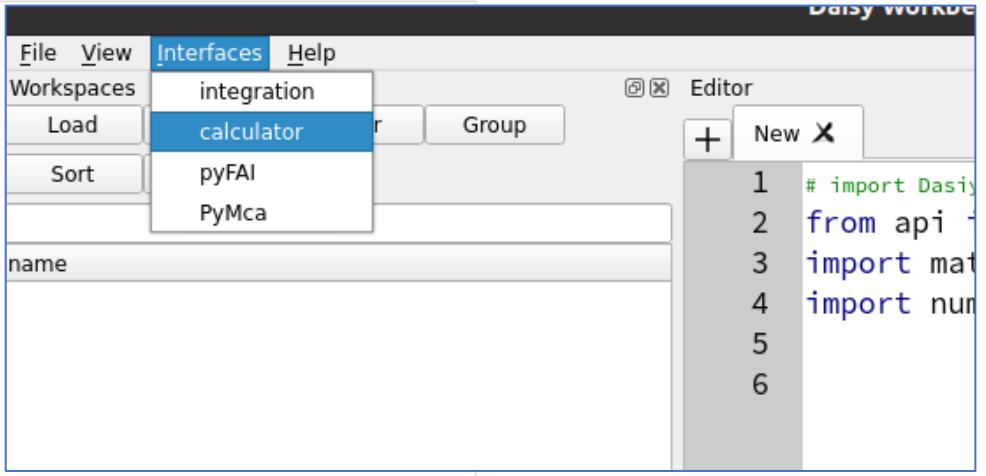
开发流程：

3. 在 Daisy/Workbench/windows/MainWindowManager.py 中主窗口的 setup 函数中进行 interface 窗口实例化。

```
from windows.interface.Integration.CalculatorManager import CalculatorManager
self.interfaces_calculator = CalculatorManager(self)
```

4. 在 create_interfaces_action 中，为该 interface 在主窗口的 interface 菜单下添加一个按钮连接到该 interface。

```
create_action(
                self,
                "calculator",
                on_triggered=self.interfaces_calculator.show,
                shortcut_context=Qt.ApplicationShortcut,
            ),

```



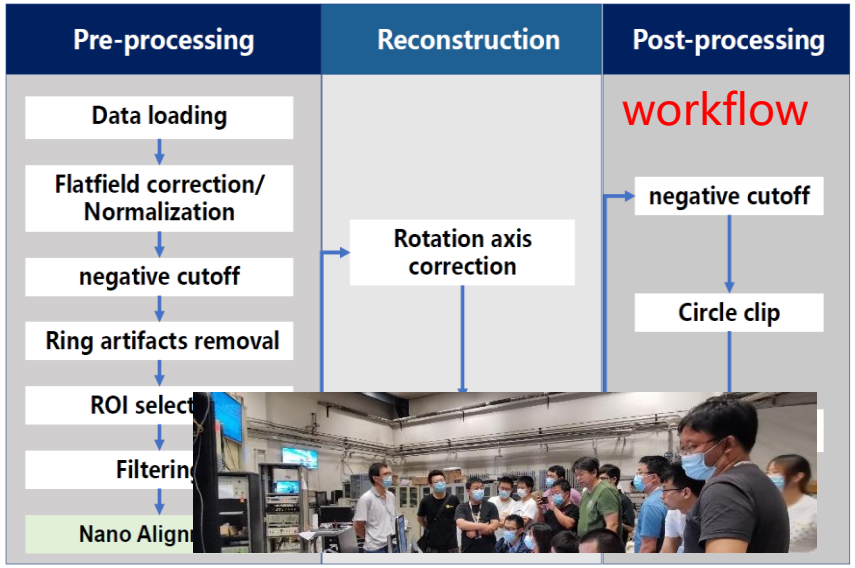
Outline

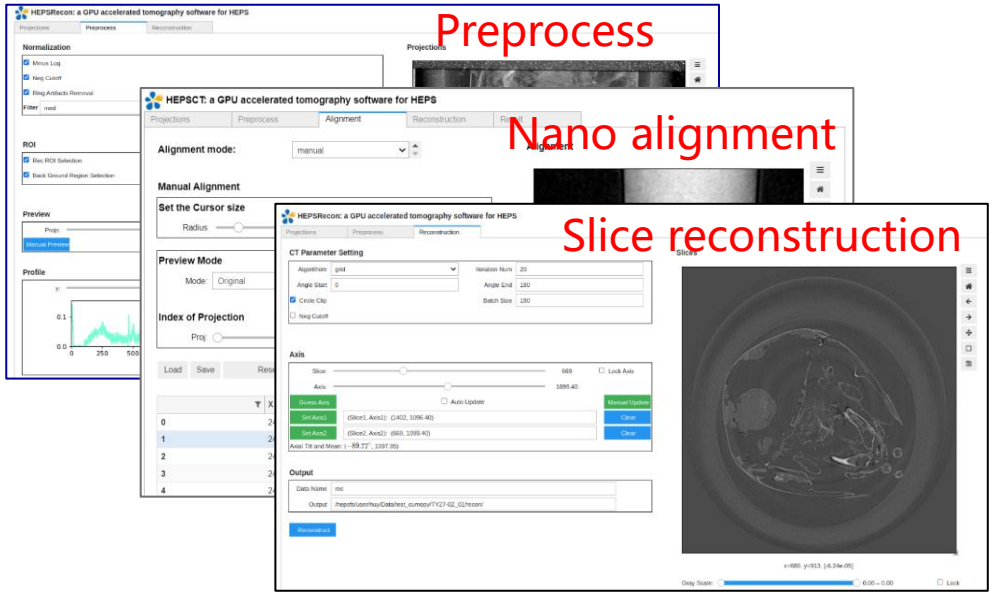
1. 背景介绍
2. 科学数据与软件系统的需求与挑战
3. 软件框架的架构与设计
4. 基于软件框架的科学软件开发
5. **科学软件示例**
6. 总结

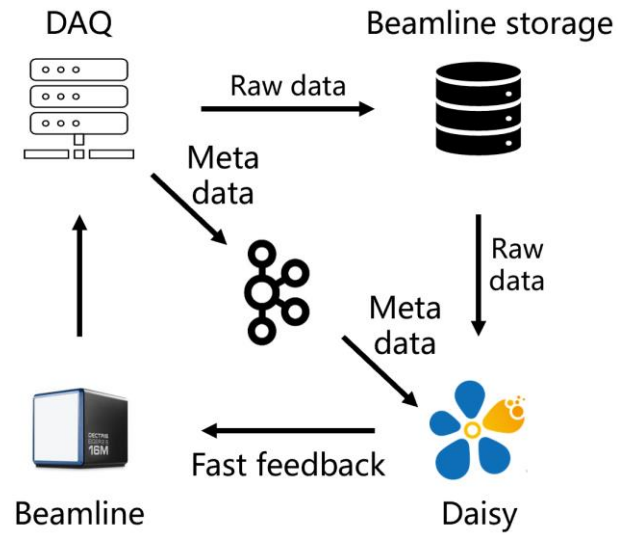
Web based application for X-ray CT

- 集成了多个CT数据处理软件： Tomopy， UFO， 及HEPS-BE自研的HEPSCT，基于 GPU 硬件加速
- 基于 web 的用户交互式应用，支持多种数据格式的显微CT和纳米CT的断层扫描重建
- 将支持HEPS多个线站(B1，B2，B4，B7，BC，BE)的成像数据处理，可以通过HEPS交互式计算平台访问使用

原始数据	重建时间
2048×2048×1442(12GB)	1min
6144×4400×1500(76GB)	10mins
10668×4402×7200(630GB)	2h

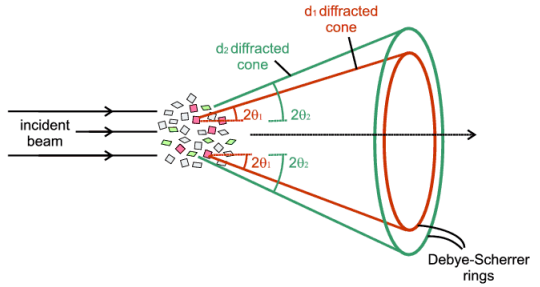


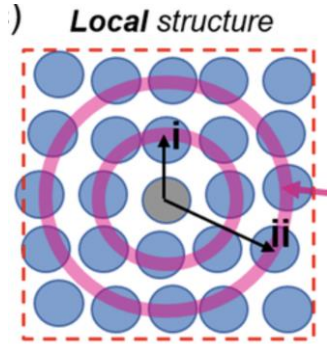


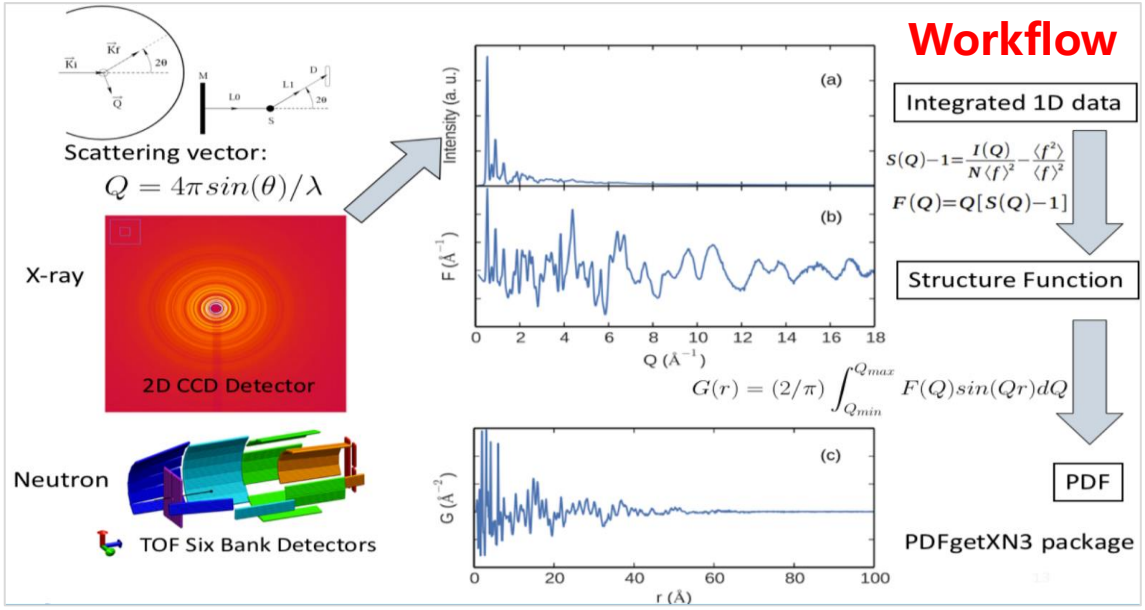


Application for Pair distribution function(PDF)

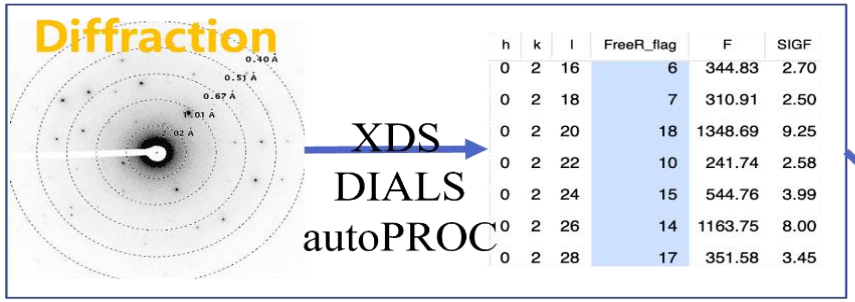
- 面向全散射实验，直接揭示实空间材料中原子间的距离
- 集成了多种衍散射科学数据处理软件， 形成 PDFHEPS 包
- 基于 web 的用户交互式 PDF 数据处理应用，提供了从衍射数据到 PDF（包括背景扣除、masking、方位角积分、PDF 转换和结果可视化等）的完整 pipeline
- 可以通过 HEPS 交互式计算平台访问使用







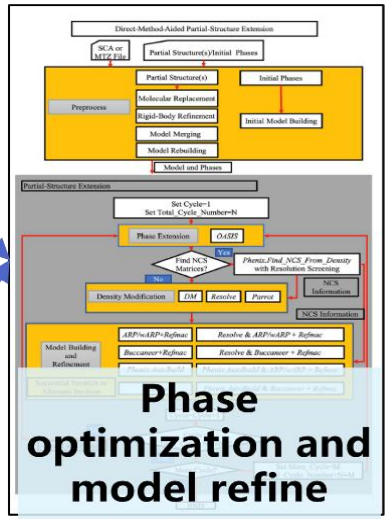
AI-based application for biological macromolecules

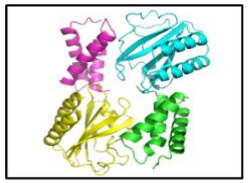


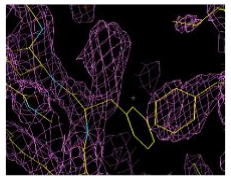
h	k	l	FreeR_flag	F	SIGF
0	2	16	6	344.83	2.70
0	2	18	7	310.91	2.50
0	2	20	18	1348.69	9.25
0	2	22	10	241.74	2.58
0	2	24	15	544.76	3.99
0	2	26	14	1163.75	8.00
0	2	28	17	351.58	3.45

Real-time data processing

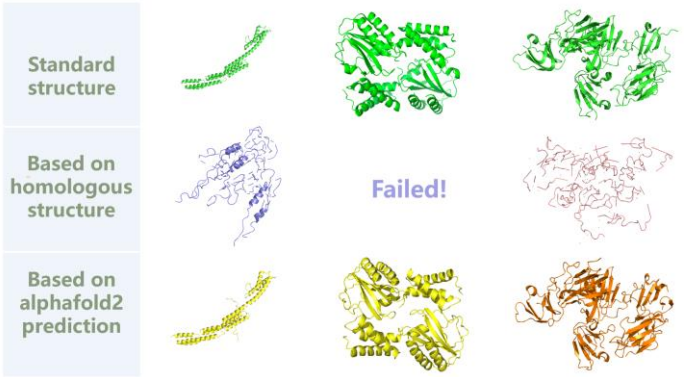
Structure prediction based on AlphaFold2







Structure truing based on AI

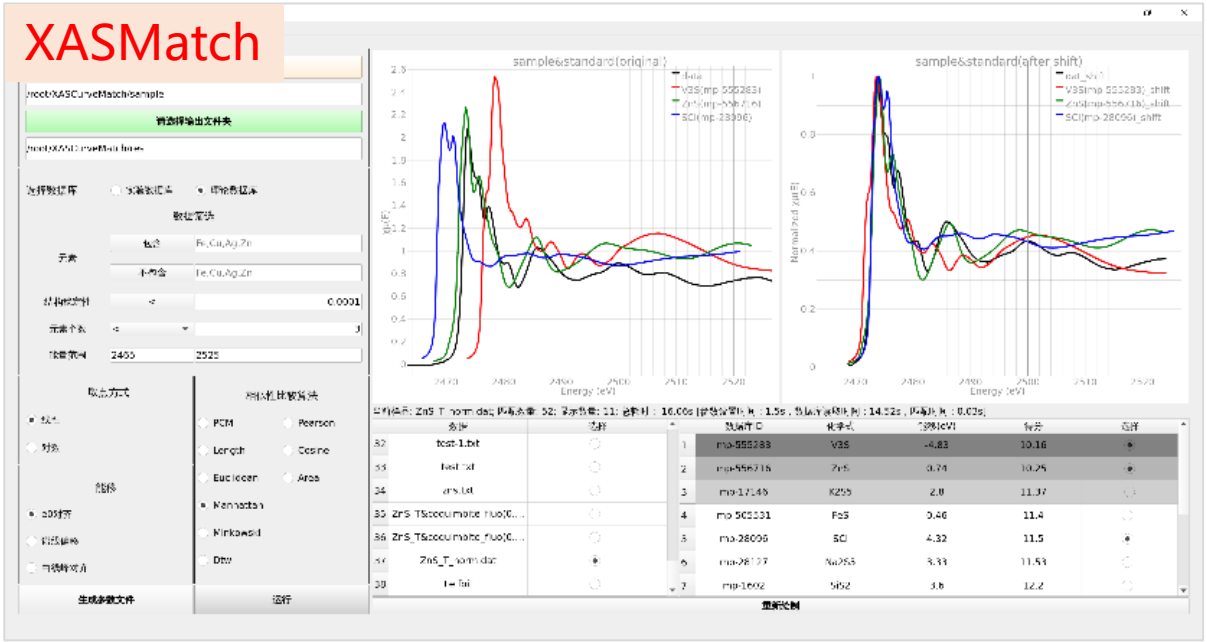


- 从衍射数据到生物大分子结构的全自动流水线
- 基于 web 提供自动化数据处理服务，实时提供数据处理状态监控和结果查询
- 结合传统算法与人工智能算法，提高了生物大分子结构解析的成功率和精确性

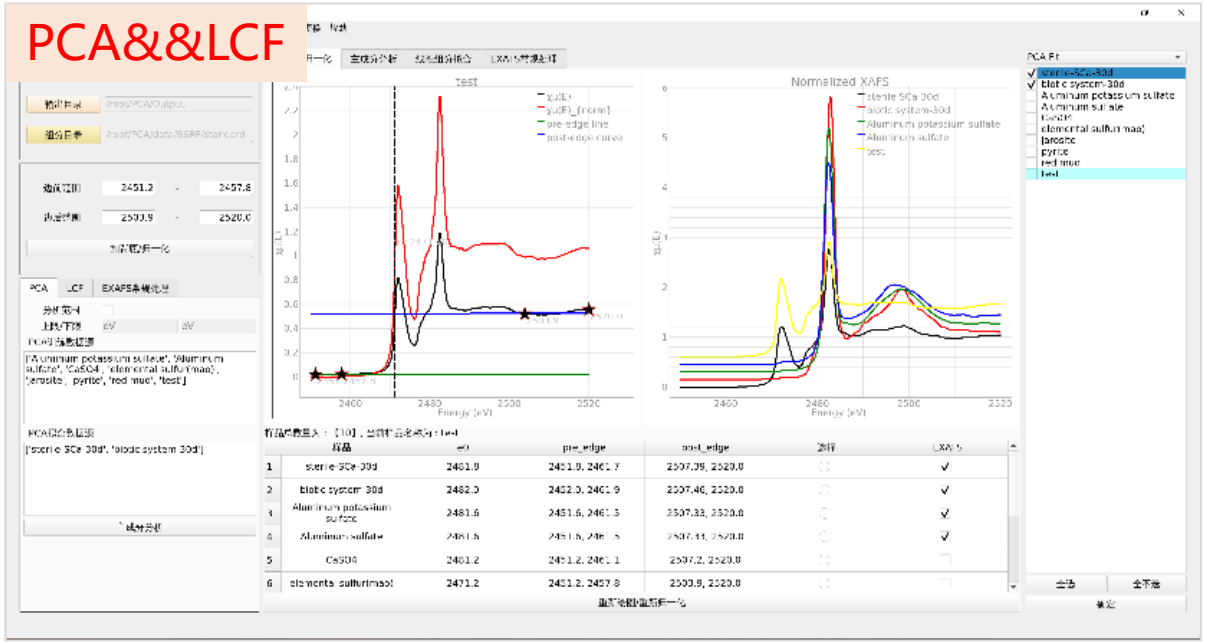
Applications for X-ray absorption spectroscopy

- 交互式谱学成分分析软件，允许用户在实验过程中快速了解样品成分
- XASMatch：对实验谱在数据库中进行快速匹配，集成多种匹配算法、能移方法，支持多种数据库输入
- PCA&&LCF：适用于催化、电池反应中相变的成分分析，支持自动化批处理、多标准谱输入
- 基于pyqt，本地客户端形式

XASMatch



PCA&&LCF



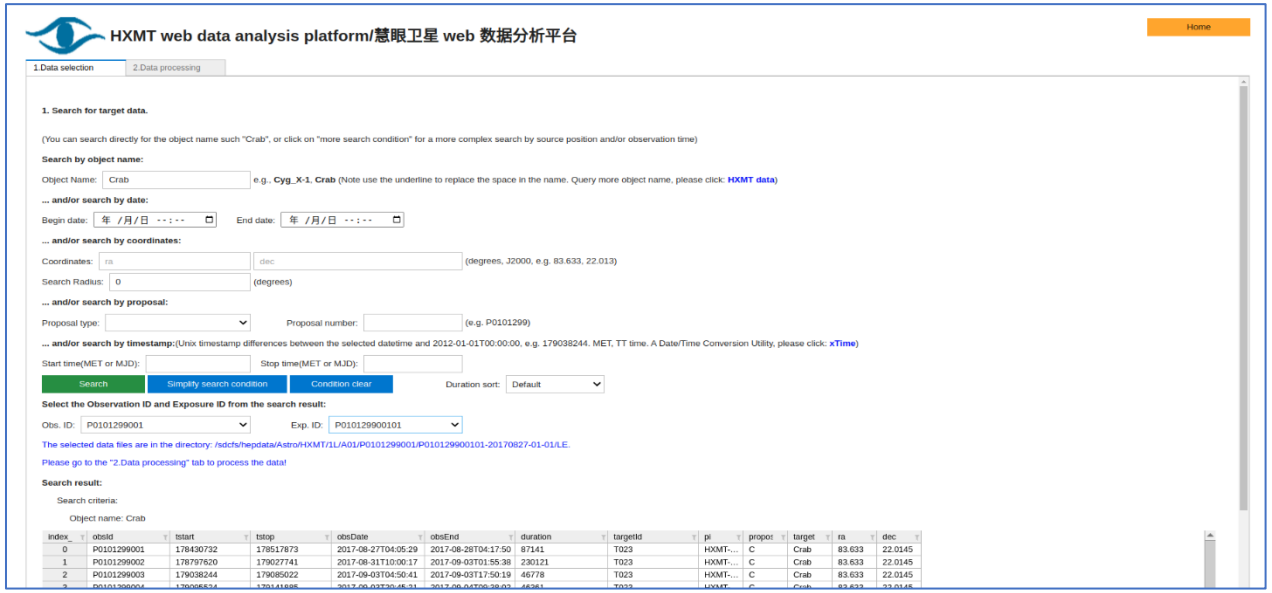
Daisy 在空间天文学的应用

可能的应用场景

- 数据处理，数据分析，数据产品生成
- 探测器模拟，观测模拟
- 整合现有软件资源，形成共性软件包

HXMT web 数据处理平台

- 基于 jupyterlab 的 HXMT web 数据分析平台
- 通过 web 浏览器为用户提供数据处理环境和服务



https://sdccompute.ihep.ac.cn/

Svom 数据产品生成软件集成

- 将Daisy软件框架应用于 **Svom** 卫星二级数据产品生成
- fits 数据文件读写算法，部分数据产品生成算法已经集成到 Daisy 框架中

eXTP 卫星数据处理软件集成

- 数据模板生成，数据分割等算法已经集成到Daisy框架中

Outline

1. 背景介绍
2. 科学数据与软件系统的需求与挑战
3. 软件框架的架构与设计
4. 基于软件框架的科学软件开发
5. 科学软件示例
6. 总结

总结

- 四代光源的建成以及探测器技术的发展，科学数据爆发式增长，带来了新的机遇与挑战
- 大数据场景下的科学发现，需要先进的数学及算法
- 多样化的应用场景及数据结构，使得学科软件及算法的发展需要底层软件框架支持
- 围绕HEPS需求，规划、设计了面向先进光源的科学数据处理分析软件框架
- 基础框架已经形成并验证，围绕软件框架，学科方法学开发在持续进行中
- 科学软件生态的发展还需要用户社群的支持与参与

https://daisydoc.ihep.ac.cn/

谢谢！