四月研究生季度考核

学生：裴文溪

导师：季筱璐

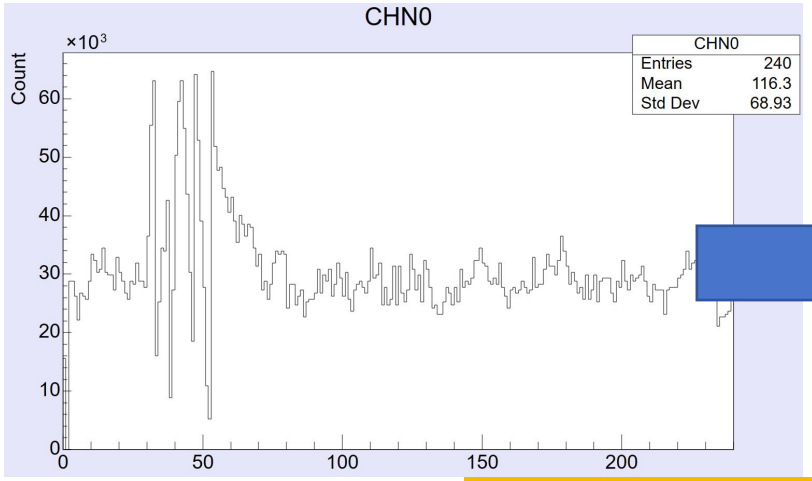
专业：计算机技术

时间：2024年04月29日

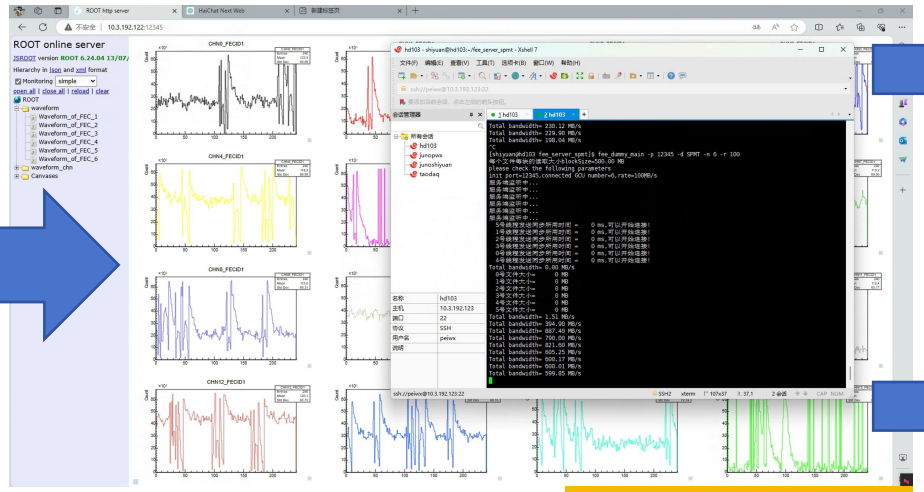
报 告 提 纲

- TAO-CD波形读出软件的完善
- 设计TAO远程值班系统及智能助手架构
  - 调研开题相关内容
  - 实验所需环境的搭建及部分功能实现
- 后续工作

波形读出软件的完善



12月测试结果



最终测试结果

- ✓ 实现抽样显示
- ✓ 已经达到6个FEC板子 同时发送100MB/s 数据
- ✓ 模拟数据源带宽满足需求
- ✓ 程序稳定运行

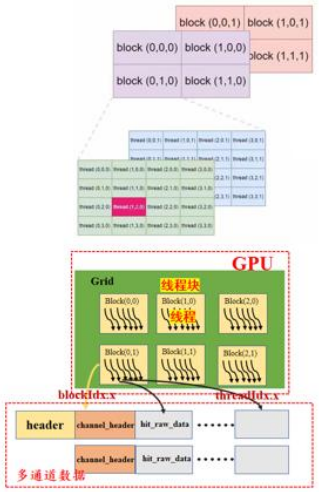
上个季度实现单个FEC板

16个通道的波形显示

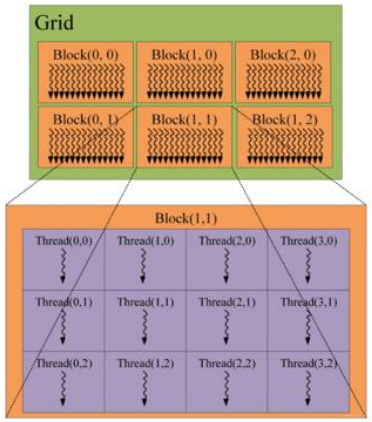
- 对TAO实验数据链有更直观和进一步的了解

GPU和CUDA相关知识的学习

1.CUDA核函数编写基础



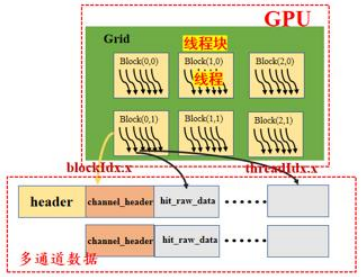
网格Grid 线程块Block 线程thread之间关系

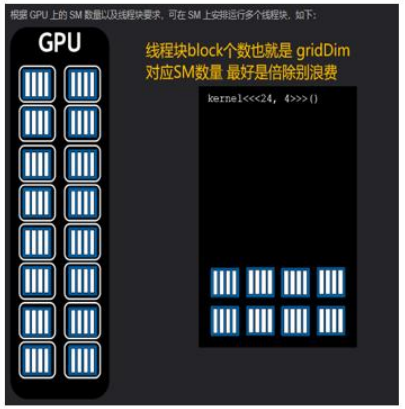


并行计算对应线程的对应元素需要对应的索引

优化思路：thread开了1024个可能会冗杂 warp可能会有限制

1.2优化思路——GPU处理多通道数据方案





目前thread开了1024个，线程可能会冗余

warp可能会对处理效率有限制

block 里面的thread 数量不是32 的倍数，

那他会把剩下的thread独立成一个warp

思路：如果能考虑到合理的架构则可以避免资源的冗余

调研工作

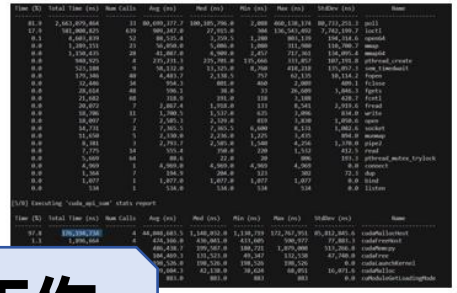
2.矩阵相乘算法——CPU/GPU程序编写以及结果对比

结果：GPU的矩阵相乘结果和CPU处理结果是在一定精度范围内相同 GPU算法编写正确

CPU方式的矩阵相乘是通过三个循环进行相乘加

GPU方式则是利用开多个线程完成同样的操作同时进行各自对应元素的乘加

nsys工具的使用



发现GPU矩阵乘法的核函数的使用时间：900us

维度是 （int）1000*1000

一般htod时间会比较长

但是推测因为我的数据量不够

htod时间不是主要耗时：300us

主要是创建cpu内存空间时间耗时比较长 但是还未深究

乘优化——GPU程序

Time(%)	Total Time(ns)	Avg(ns)	Name
56.0	890,476	890,476.0	gpu_matrix_mult(int *,int *,int *)
44.0	699,786	699,786.0	gpu_matrix_mult_shared_memory(int *,int *,int *)

结果完成了矩阵乘法的功能并且和CPU结果对比后在精度范围内相同

目前了解部分概念：

- 共享内存相关信息（分块方式）
- 按行读取数据快于按列读取数据（倒置方式）

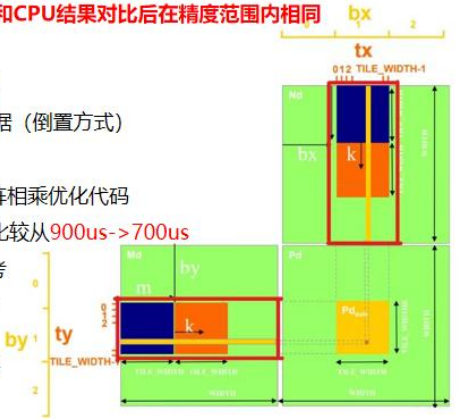
对比上面两种最后选择

是利用静态申请共享内存完成矩阵相乘优化代码

和未加入共享内存的GPU代码相比较从900us->700us

但是对于分块更好的方式还在思考

矩阵相乘加入共享内存的实现思路



调研GPU及CUDA相关内容：

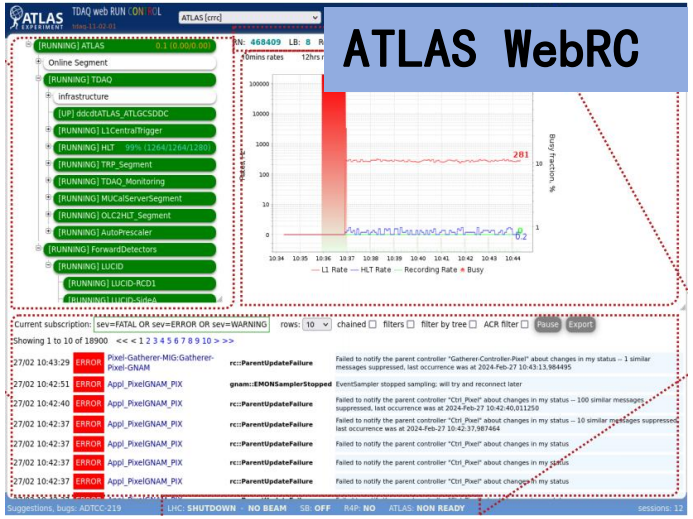
- ✓ 了解基本GPU工作原理
- ✓ 学习CUDA程序检查工具
- ✓ 矩阵相乘算法程序的优化

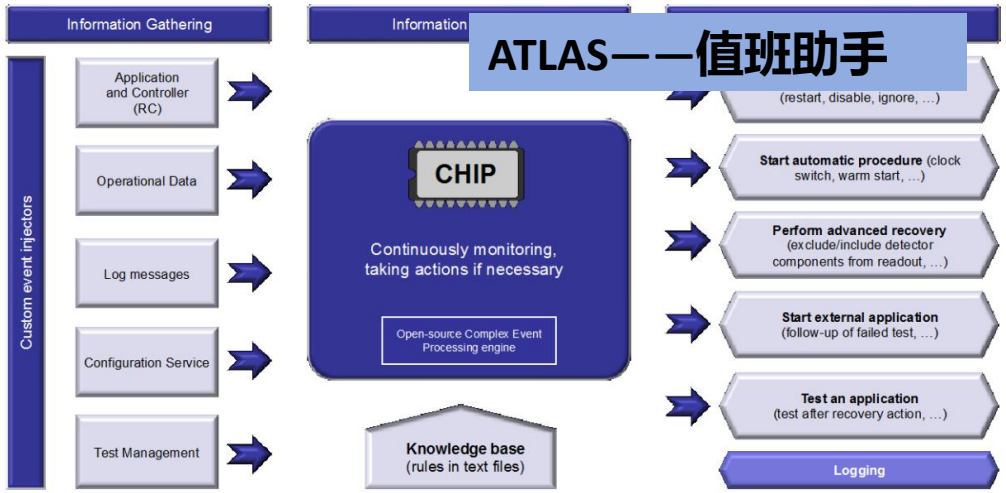
提前为TAO值班系统

智能化方案设计

打下理论基础

高能物理实验值班系统及智能化发展现状

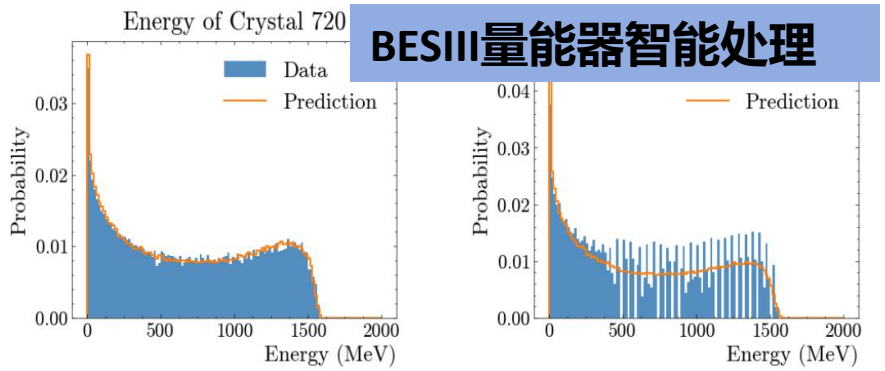




调研相关内容

通过调研发现：

- ✓ 实现实时监测
- ✓ 异常分析及告警
- ✓ 智能化方案加入



智能化远程值班系统 — 提高取数效率，保障实验高效稳定运行

TAO对值班系统的需求

TAO： 实验靠近反应堆 **现场值班NO**

远程值班 + **专家 → 监控系统、观察异常**

传统的值班系统实现

- 轮岗值班
- 繁重的纸质/电子文档信息
- 培训值班人员耗费时间及精力
- 数据量多且复杂监测难度大
- 出现系统故障解决时间长难度大
- 过度依赖专家解决故障

NOT GOOD

✓调研发现：

需求：TAO需要实现远程稳定的智能化值班系统

数据监测及告警反馈

智能化分析与应用

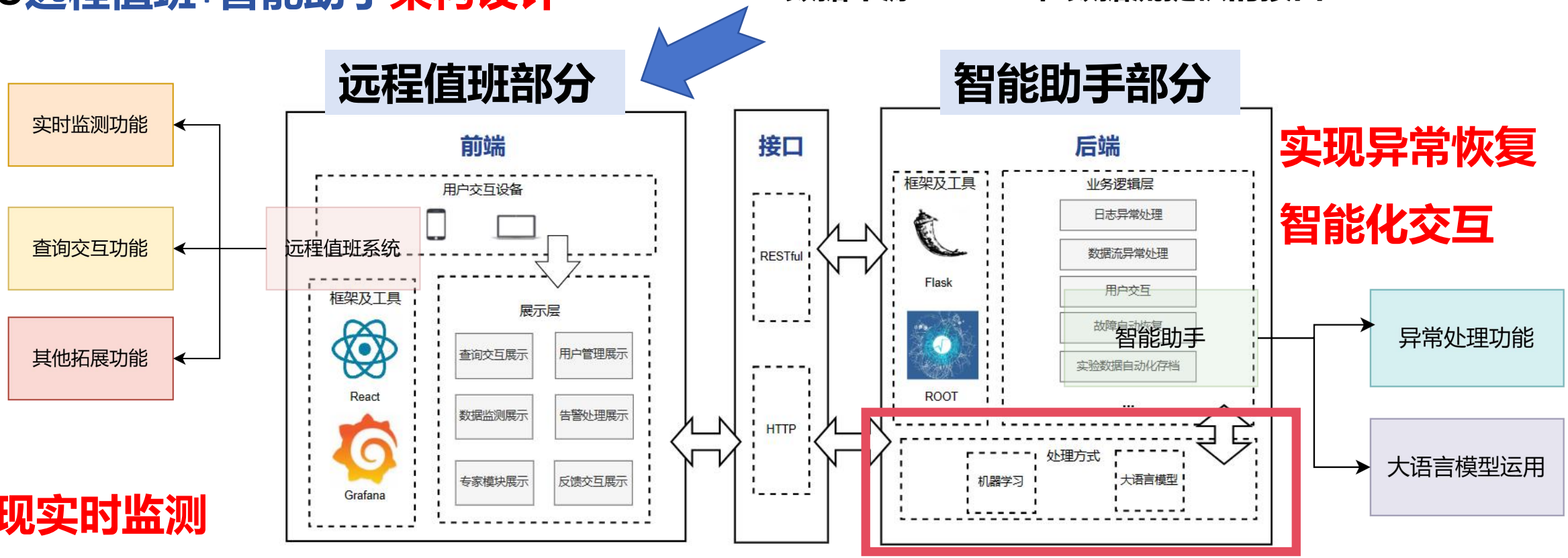
- ✓ 实现**远程**在线**监控**实验数据
- ✓ 实现实验故障及**异常**自动**告警**
- ✓ **智能化**地**分析**各类异常

为改善传统的方式取数效率低、存在风险、过度依赖专家值班经验提出了远程值班系统及智能助手

设计TAO的远程值班及智能助手的框架

TAO远程值班+智能助手架构设计

数据来源：Online和数据流提供的接口

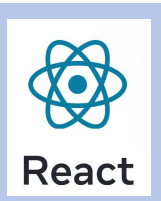
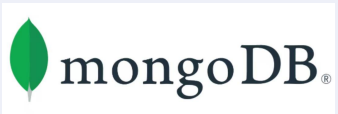


高内聚低耦合模块化设计思想

针对实验需求完成框架设计

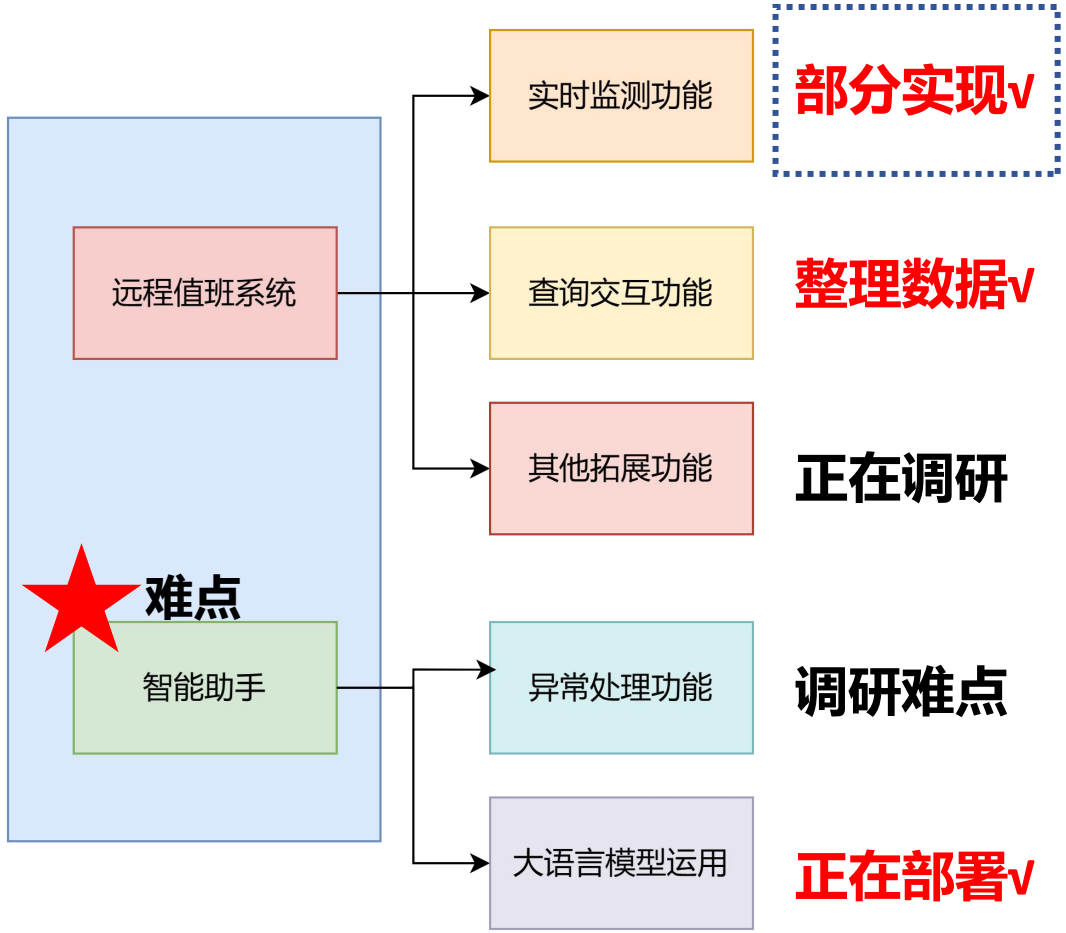
调研课题实验技术及环境搭建

- ✓ 调研各类主流框架
- ✓ 选择适合的技术栈
- ✓ 完成相应环境的搭建

TAO的介绍及需求	针对TAO的技术优点	技术选择
TAO是JUNO子实验	✓ 能够JUNO监控系统互相兼容 ✓ 需要及时响应完成实时值班效果	 前端使用React
TAO值班系统**需加入智能化技术**	✓ python语言易编写机器学习 ✓ 处理数据效率高	 后端使用Flask
TAO值班系统需**处理大量日志**	✓ mongoDB具有Bson类型 ✓ 对于日志也有高性能读写操作	 数据库使用mongoDB

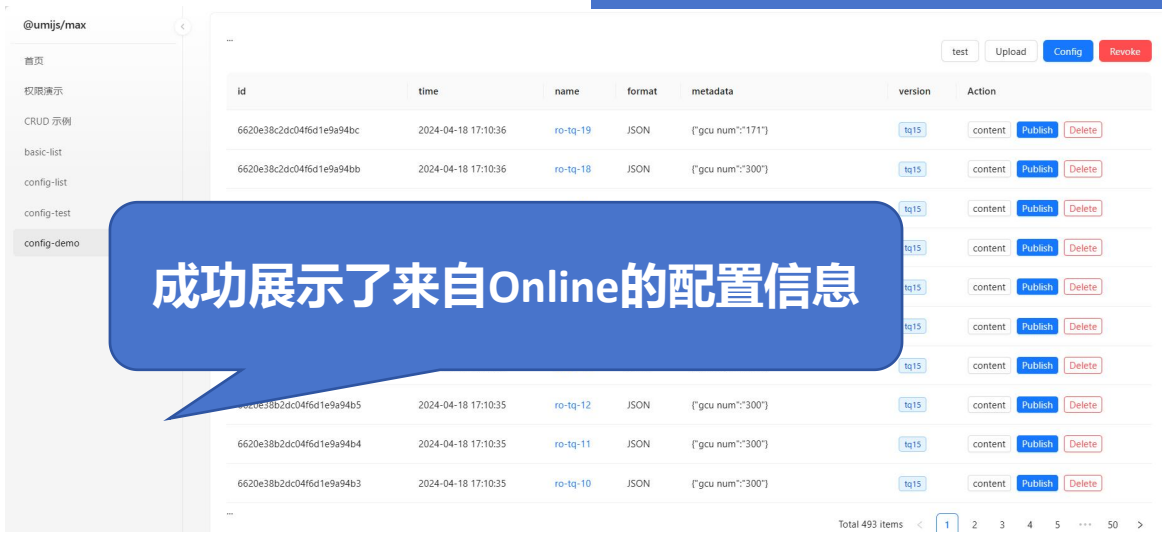
工作进展——实时监测功能

课题工作完成情况：



TAO**远程值班系统+智能助手**的功能

暂时使用JUNO的数据来源



TAO实验作为JUNO子实验

和JUNO有一样的与Online服务对接接口

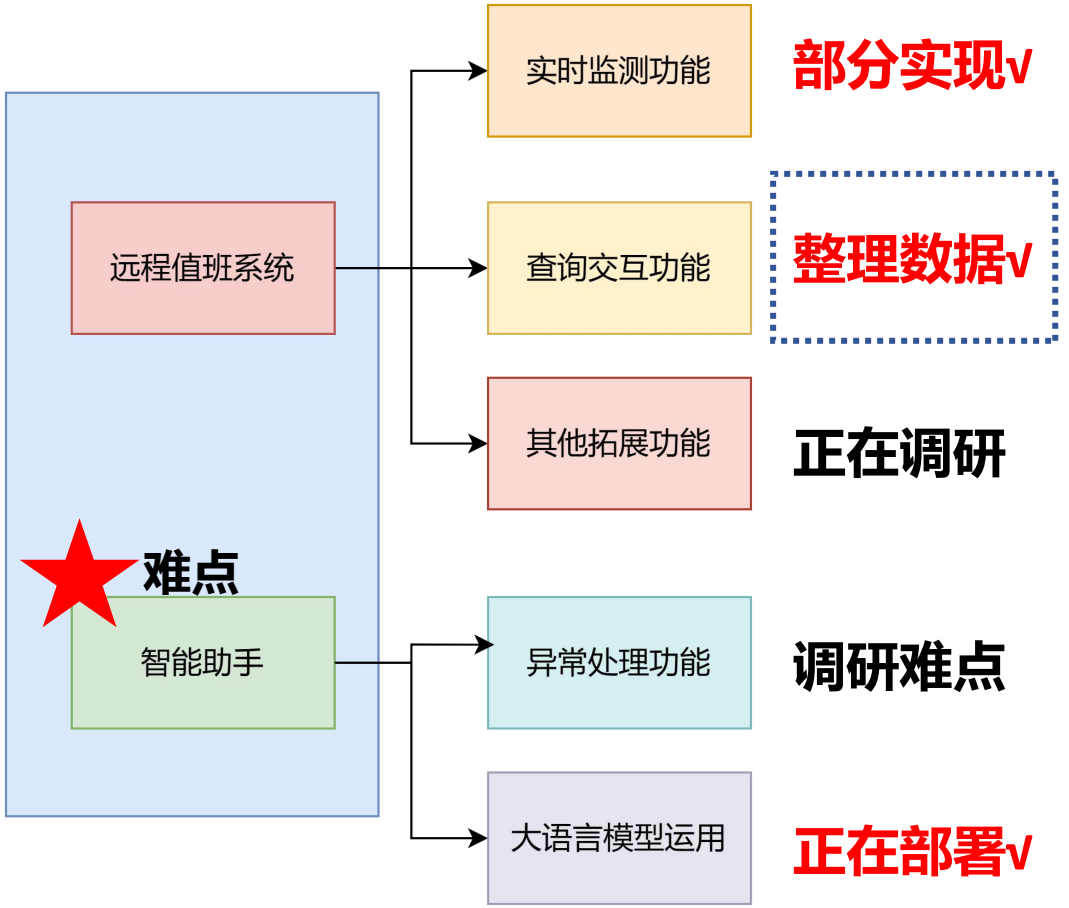
可以展示在线软件的相关信息

完成的工作：

✓ 目前实现监测功能的配置模块实时展示

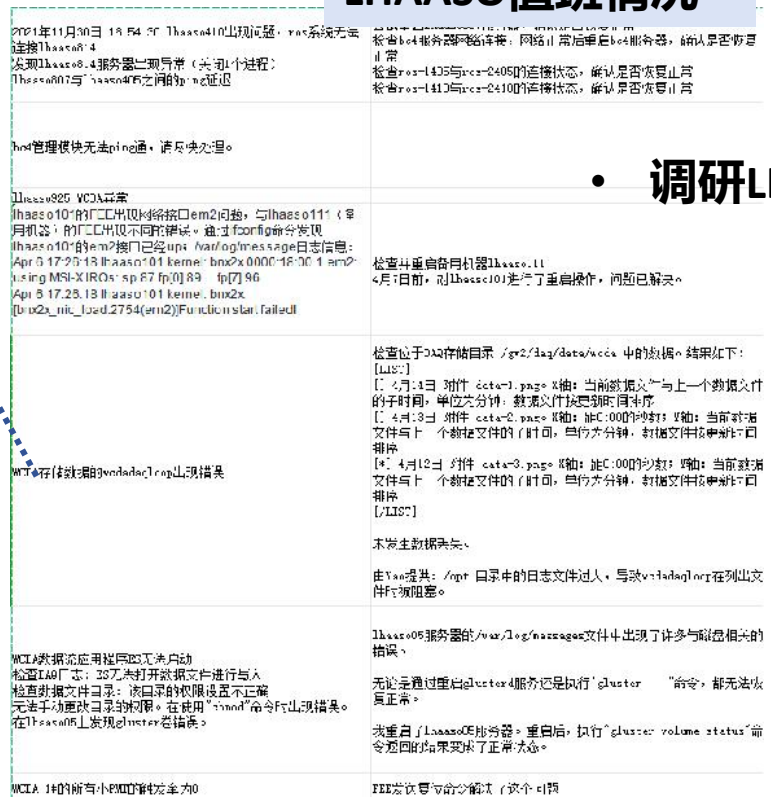
工作进展——数据预处理

课题工作完成情况:

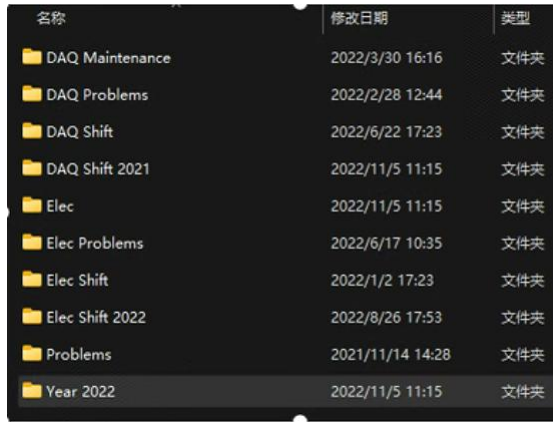


TAO**远程值班系统+智能助手的功能**

LHAASO值班情况



- **调研LHAASO值班异常管理方式**

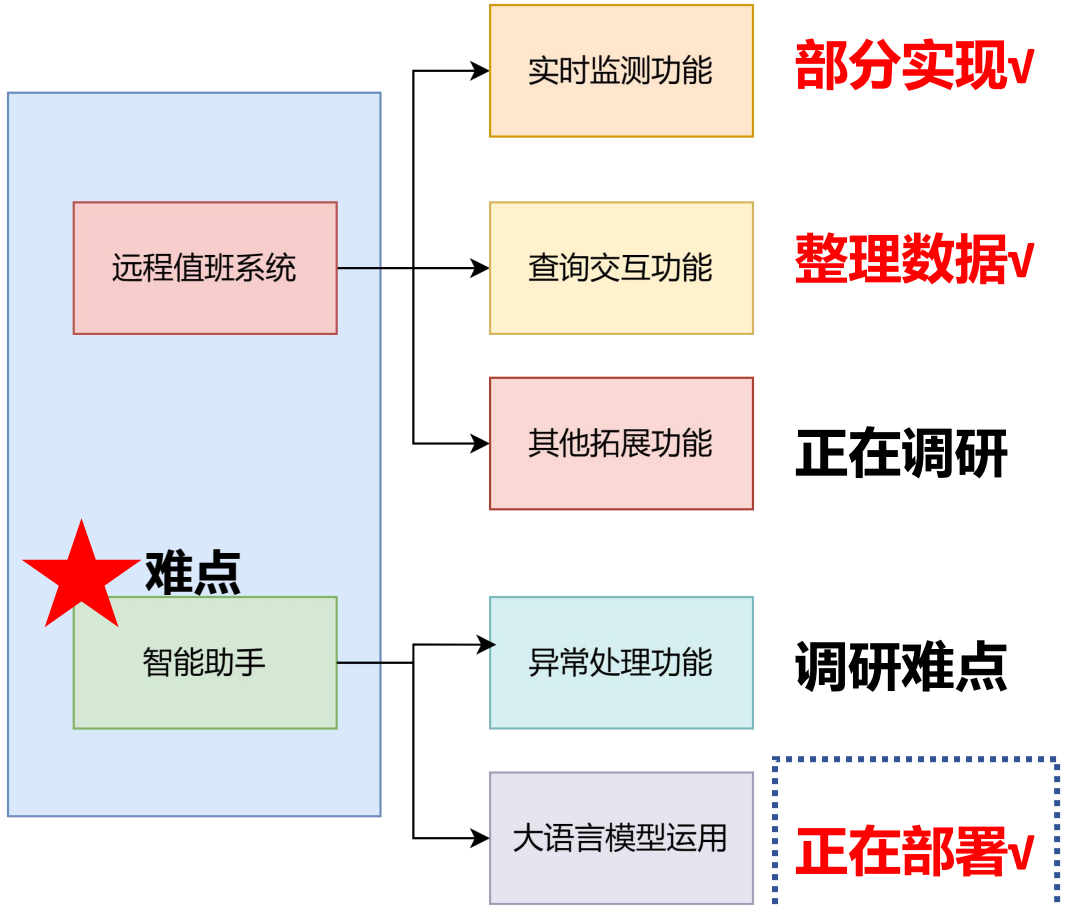


完成的工作:

- ✓ 数据库数据收集（调研LHAASO实验值班异常及对策）
- ☐ 数据库用于模型微调等后续处理

工作进展——大语言模型调研及部署

课题工作完成情况:



TAO**远程值班系统**+**智能助手**的功能

测试使用Hepai库作为交互

调研微调方式

微调模型	模型优点	模型缺点	适用场景
Fine-Tuning（x）	简单易用：直接在预训练模型上进行微调 适应性强：可以针对特定任务调整整个模型的参数。 效果显著：通常能显著提高模型在特定任务上的表现。	计算成本高：需要调整模型的大量参数。 数据需求较高：为了有效微调，通常需要较多的标注数据。	当有足够的标注数据和计算资源时，适用于大多数NLP任务。
Parameter-Efficient Fine-Tuning（PEFT）	参数高效：只修改或优化模型的一小部分参数。 节省计算资源：比完全微调需要的资源少。	可能效果有限：对于某些复杂任务，仅优化少量参数可能不足以达到最佳效果。	资源受限的情况，或者需要快速适应新任务时。
Prompt-Tuning	无需改变模型架构：通过设计任务相关的提示（prompt），引导模型生成所需的输出。 资源消耗少：不需要改变模型参数。	需要精心设计prompt：有效的prompt设计可能需要丰富的经验和实验。	快速适应新任务，尤其适用于资源有限的场景。
LoRA	参数高效：通过引入低秩矩阵来调整模型，减少需要优化的参数数量。 节省内存和计算资源。	效果可能有限：对于某些复杂任务可能无法达到完全微调的效果。 需要一定的技术知识来实现和调试。	需要参数高效调整的场景，特别是在资源有限的情况下。
P-Tuning	可解释性强：通过可训练的prompt向量进行微调。资源消耗相对较少。	需要适当的调整和实验来找到最佳配置。 可能不适用于所有类型的任务。	适用于需要提高模型解释性的任务，以及资源有限的情况。

完成的工作：

- ✓ 调研微调方式
- ✓ 大语言模型的部分环境搭建

下一季度工作计划

- 实现TAO远程值班系统的基础功能
- 在智能化处理方面继续部署及应用
- 完成其他工作

感谢各位老师，请批评指正