

長安大學
CHANG′AN UNIVERSITY

# 长安大学高性能计算平台

# 用户手册

长安大学信息与网络管理处

2026 年 9 月

# 目录

第一章 高性能计算平台介绍 ..... 6
1.1 平台硬件资源 ..... 6
1.2 平台软件资源 ..... 6
1.3 平台队列 ..... 11
1.3.1 CPU 通用计算队列 ..... 11
1.3.2 大内存 CPU 计算队列 ..... 11
1.3.3 GPU 计算队列 ..... 11
1.3.4 AI 计算队列 ..... 13
1.4 资源池分类 ..... 13
第二章 快速上手 ..... 14
2.1 命令行方式 ..... 14
2.2 门户方式 ..... 15
2.3 作业限制 ..... 16
2.4 helloWorld 示例 ..... 17
2.5 节点资源选择 ..... 20
2.5.1 关于 CPU 核数选择的重要说明 ..... 20
第三章 平台接入 ..... 26
3.1 命令行接入 ..... 26
3.2 门户接入 ..... 27
第四章 数据管理 ..... 28
4.1 文件传输 ..... 28
4.1.1 命令行方式 ..... 28
4.1.2 门户方式 ..... 30
4.2 数据协同 ..... 31
第五章 作业管理 ..... 31
5.1 命令行方式 ..... 31
5.1.1 编写脚本 ..... 32
5.1.2 提交作业 ..... 34

5.1.3 作业查看与终止 ..... 37
5.1.4 CPU 作业示例 ..... 37
5.1.5 GPU 作业（以 tensorflow 为例） ..... 41
5.2 门户方式 ..... 44
5.2.1 提交作业 ..... 45
5.2.2 查看作业信息 ..... 50
5.2.3 作业控制 ..... 53
**第六章** 软件应用 ..... 54
6.1 命令行方式 ..... 54
6.1.1 软件环境管理（Environment Modules） ..... 54
6.1.2 软件安装 ..... 55
6.2 门户应用 ..... 58
6.2.1 Fluent 计算模式 ..... 58
6.2.2 Fluent 交互模式 ..... 64
6.2.3 仿真设计应用 ..... 67
6.2.4 Matlab ..... 67
6.2.5 MaterialsStudio ..... 71
6.2.6 Rstudio ..... 78
**第七章** 门户常用功能 ..... 80
7.1 桌面应用夹 ..... 80
7.2 开始菜单 ..... 81
7.3 通知 ..... 82
7.4 修改密码 ..... 82
7.5 显示桌面 ..... 83
7.6 系统公告 ..... 84
7.7 我的会话 ..... 84
7.8 应用仓库 ..... 89
7.8.1 桌面快捷方式 ..... 90
7.8.2 应用申请 ..... 91
7.8.3 环境变量 ..... 91

7.8.4 工作路径 ..... 92
第八章 常用命令 ..... 93
8.1 调度常用命令说明 ..... 93
8.1.1 jsub ..... 93
8.1.2 jjobs ..... 100
8.1.3 jhosts ..... 103
8.1.4 jqueues ..... 104
8.1.5 jctrl ..... 106
8.1.6 jhostgroup ..... 116
8.1.7 jlimits ..... 117
8.1.8 jcode ..... 119
8.2 Linux 常用命令说明 ..... 121
8.2.1 基础快捷键使用 ..... 121
8.2.2 命令 ..... 121
第九章 第九章 常见问题 ..... 129
9.1 启动 JHFileClient 超时 ..... 129
9.2 mpi 程序的两个 slots 在 cu01 上、两个 slots 在 cu02 上 ..... 130
9.3 关于修改密码 ..... 130
9.4 请问这个 cpu 数量是随机分配的吗？计算速度主要是根据什么来确定？怎么感觉算的还没自己的电脑快呢？ ..... 131
9.5 cuda 的安装路径是什么？ ..... 131
9.6 平台上没有集成我想用的软件 ..... 132
9.7 ssh 无法登录 ..... 132
9.8 abaqus 引用的子程序需要调整为指定格式再引用吗 ..... 133
9.9 bashrc 设置环境变量后，提交作业提醒命令找不到 ..... 134
9.10 sudo 无法获得管理员权限 ..... 134
9.11 CPU 核数设置及使用问题 ..... 134
9.12 自己安装的软件环境设置问题 ..... 135
9.13 para 队列默认自动加载 openmpi ..... 135
9.14 GPU 显存不够用问题 ..... 135
9.15 lib64 动态链接库文件引用错误 ..... 136

9.16 文件传输户端无法使用 ..... 137
9.17 上传下载打不开的原因 ..... 137
9.18 会话应用使用出现弹窗会话应用使用出现弹窗 ..... 137
9.19 如何传文件到 ln03 笃学资源池的家目录 ..... 138
9.20 提交作业显示 You are not allowed to submit ..... 138
9.21 GPU 作业在等待和运行中循环往复 ..... 139
9.22 ai_service 作业一直等待 ..... 139
9.23 申请创新用户但是平台却显示普通用户 ..... 139
9.24 会话打不开 ..... 140
9.25 运行机器学习作业出现 glibc 相关的报错 ..... 140
9.26 指定了 ln03 和对应节点提交作业报错 ..... 140
9.27 安装 cuda 时遇到权限问题 ..... 140
9.28 下载安装客户端组件后，还是要求下载 ..... 140
9.29 用 ln01 节点想装 matlab，但是运行安装程序显示没有权限 ... 140

# 第一章 高性能计算平台介绍

## 1.1 平台硬件资源

长安大学高性能计算平台单精度总算力达 2.13PFLOPS，由 113 台计算节点及 7 台服务节点构成，具体硬件配置如下：

**计算节点**：95 台双路 CPU 计算节点、6 台大内存四路 CPU 计算节点、10 台 GPU 计算节点（含 72 张加速卡：8 张 A100 卡、44 张 A800 卡、8 张 A40 卡、4 张 RTX 3090 卡）以及 2 台 4 卡高性能显卡图形节点（A40 卡）。

**服务节点**：3 台登录节点、4 台管理节点。

集群所有节点通过 HDR InfiniBand 网络互联，并配置一套 1.3 PB 的高性能并行文件存储系统。

## 1.2 平台软件资源

平台已集成多种主流科学计算与工程仿真软件，覆盖分子动力学、量子化学、有限元分析、电磁仿真、流体力学、数学建模及高性能计算开发等多个专业领域，可支撑材料科学、生物医药、航空航天、汽车工业、电子通信等学科方向的科研计算需求。主要软件资源如下表所示。

| 软件名称 | 软件分类 | 主要用途 | 应用领域 |
| --- | --- | --- | --- |
| LAMMPS | 计算应用 | 大规模原子/分子并行模拟 | 分子动力学、材料科学 |
| VASP | 计算应用 | 第一性原理密度泛函计算 | 电子结构计算、量子力学模拟、材料模拟 |
| MATLAB | 计算应用 | 数值分析、矩阵计算、科学可视化 | 数据分析、无线通信、深度学习、图像处理、信号处理 |

| 软件名称 | 软件分类 | 主要用途 | 应用领域 |
| --- | --- | --- | --- |
| | | | 、量化金融、机器人、控制系统 |
| Intel® oneAPI Toolkits | 工具包 | 跨架构统一编程模型工具包（含编译器、MPI 等） | CPU、GPU、FPGA、加速器多架构开发 |
| GROMACS | 计算应用 | 分子动力学模拟 | 生物分子体系（蛋白质、核苷酸、糖等）、溶液或晶体模拟 |
| ABAQUS | 计算应用 | 工程模拟有限元分析 | 线性/非线性结构分析、热传导、质量扩散、声学、岩土力学 |
| LS-DYNA | 计算应用 | 非线性动力分析（显式求解为主） | 高速碰撞、爆炸、金属成型等瞬态动力学问题 |
| AUTODYN | 计算应用 | 显式有限元分析（含 Euler、Lagrange、SPH 等求解器） | 弹道学、战斗部设计、穿甲、爆轰、水下爆炸 |
| ANSYS | 计算应用 | 多用途有限元分析 | 结构、流体、电力、电磁场、碰撞等 |
| CFX | 计算应用 | 流体计算 | 航空航天、旋转机械、能源、石油化工、机械制造、汽车、生物技术、水处理、火灾安全、冶金、环保 |
| CST Studio Suite | 计算应用 | 高性能 3D 电磁仿真 | 天线、滤波器、电磁兼容（EMC/EMI）、机电效应、热效应 |
| COMSOL Multiphysics | 计算应用 | 多物理场高级数值仿真 | 科学研究、工程计算、各领域物理过程模拟 |
| FLUENT | 计算应用 | 计算流体力学 | 流体流动、传热、化学反应 |
| Gaussian | 计算应用 | 量子化学综合计算 | 能量计算、分子结构、振动频率、化学反应机理、激发能 |
| CST 2019 | 交互应用 | 3D 电磁仿真计算 | 设计、分析和优化电磁（EM）部件及系统，适用于整个 |

| 软件名称 | 软件分类 | 主要用途 | 应用领域 |
| --- | --- | --- | --- |
| | | | EM 范围内各类应用领域的电磁场求解器全部包含在 CST Studio Suite 的一个用户界面中 |
| Lin_ABAQUS | 交互应用 | 工程模拟有限元分析 | 线性/非线性结构分析、热传导、质量扩散、声学、岩土力学的前后处理 |
| Lin_CFXpost | 交互应用 | 流体计算 | 航空航天、旋转机械、能源、石油化工、机械制造、汽车、生物技术、水处理、火灾安全、冶金、环保等后处理 |
| Lin_CFXpre | 交互应用 | 流体计算 | 航空航天、旋转机械、能源、石油化工、机械制造、汽车、生物技术、水处理、火灾安全、冶金、环保等前处理 |
| Lin_Comsol | 交互应用 | 多物理场高级数值仿真 | 科学研究、工程计算、各领域物理过程模拟图形交互应用 |
| Lin_Fluent | 交互应用 | 流体力学（a40design01） | 流体流动、传热、化学反应 |
| Lin_Fluent_2 | 交互应用 | 流体力学（胖节点） | 流体流动、传热、化学反应 |
| Lin_GaussianView | 交互应用 | 量子化学综合交互应用 | 能量计算、分子结构、振动频率、化学反应机理、激发能 |
| Lin_Mechanical | 交互应用 | 结构力学分析、热分析、多物理场耦合分析、材料与接触模拟、结构优化 | 航空航天、汽车工业、能源产业、电子设备、土木工程、医疗设备、运动装备 |
| Lin_Pycharm | 交互应用 | 语言设计的集成开发环 | 通用 Python 开发、数据科学 |

| 软件名称 | 软件分类 | 主要用途 | 应用领域 |
| --- | --- | --- | --- |
| | | 境（IDE） | 与机器学习、Web 全栈开发、远程开发与部署 |
| Lin_RStudio | 交互应用 | R 语言设计的集成开发环境（IDE） | 学术科研、商业分析、生物信息学、教育培训、金融建模 |
| Lin_Terminal | 交互应用 | Linux 操作系统进行交互的核心工具 | 软件开发、系统运维、网络安全、<br>数据处理、Linux 终端 |
| Lin_VMD | 交互应用 | 生物可视化和分析而设计的<br>蛋白质、核酸、脂质双层组件等系统 | 结构生物学、生物物理学、计算机化学与材料科学、药物研发 |
| Lin_Workbench | 交互应用 | 集成化与自动化工作流、多物理场耦合分析、参数化设计与优化、协同仿真环境 | 航空航天、汽车工业、能源产业、电子与高科技、生物医学 |
| Matlab2023 交互模式 | 交互应用 | 数值分析、矩阵计算、科学可视化 （a40design01） | 数据分析、无线通信、深度学习、图像处理、信号处理、量化金融、机器人、控制系统 |
| Matlab2023 胖节点 | 交互应用 | 数值分析、矩阵计算、科学可视化 （胖节点） | 数据分析、无线通信、深度学习、图像处理、信号处理、量化金融、机器人、控制系统 |
| ParaView | 交互应用 | 大规模数据处理、交互式数据探索、自动化与批处理、多物理场与多格式兼容 | 计算流体动力学（CFD）、结构力学分析、医学成像、地球科学、能源与材料 |

| 软件名称 | 软件分类 | 主要用途 | 应用领域 |
| --- | --- | --- | --- |
| PyroSim | 交互应用 | 三维可视化建模、火灾场景定义、智能网格划分与管理、集成求解与后处理 | 建筑防火设计、交通基础设施、工业安全、消防科研与培训、人员疏散分析 |
| RFD | 交互应用 | 高级油藏数值模拟软件 | 构建精细的地质模型和进行高效的油藏模拟运算 |
| UG | 交互应用 | CAD、CAE、CAM 功能于一体的高端产品工程解决方案 | 应用领域、航空航天、模具行业、通用机械 |
| Win_MaterialsStudio | 交互应用 | 可视化建模、多尺度模拟、性质预测与分析 | 电池与能源材料、催化与表面化学、高分子与聚合物、半导体与光电材料、药物研发 |
| Win_PFC2D | 交互应用 | 微观力学机制研究、非连续介质模拟、“数值实验室”实验、二次开发与自定义 | 岩土工程、岩石力学、地质灾害、工程过程、构造地质 |
| Win_PFC3D | 交互应用 | PFC2D 的三维扩展版本，专门用于模拟三维空间中球形或多面体颗粒集合体的运动、相互作用以及宏观力学行为 | 岩土工程、岩石力学、地质灾害、工程过程、构造地质 |
| cst2024 | 交互应用 | 全频段电磁仿真、时域频域协同<br>多物理场耦合 | 通信与消费电子、航空航天与国防、汽车工业、医疗电子、能源与工业 |
| edem2022 | 交互应用 | 颗粒物料行为仿真、设备性能评估与优化、多物理场耦合分析、定制化与自动化 | 采矿与矿物加工、重型设备与农业机械、工艺制造（制药/食品/化工）、增材制造（3D 打印）、炼钢行业 |
| gmsh | 交互应用 | 参数化几何建模、高质量网格生成、自动化与二次开发、多格式兼容 | 学术研究、计算流体动力学（CFD）、结构力学分析、多物理场仿真、生物医学工程 |

**注意事项：**所有交互式应用在使用完毕后，用户必须手动注销或退出。否则，计算任务会持续占用节点资源（

含 CPU、内存及应用授权数），并持续产生计费，直至因超时被系统强制终止或管理员介入处理。

### 1.3 平台队列

平台提供多种队列，以满足不同场景的计算需求，包括 CPU 通用计算队列、CPU 大内存计算队列、GPU 计算队列和 AI 计算队列。各队列的详细配置介绍如下。

#### 1.3.1 CPU 通用计算队列

| 队列名称 | 节点 | 配置 | 操作系统 | 主要用途 | 备注 |
|---|---|---|---|---|---|
| normal | 66 台 | 64 核 / 256GB | RedHat 8.3 | 常规 CPU 计算 | |
| normal_old | 16 台 | 40 核 / 192GB | RedHat 7.8 | 仅限 Materials Studio（MS）计算 | |
| arm | 4 台 | 64 核 / 512GB | kylin-server-V10 | ARM 架构计算 | 鲲鹏服务器 |
| common | 9 台 | 64 核 / 256GB | RedHat 7.8 | 常规 CPU 计算 | |

#### 1.3.2 大内存 CPU 计算队列

| 队列名称 | 节点 | 配置 | 操作系统 | 主要用途 | 备注 |
|---|---|---|---|---|---|
| fat | fat01~04 节点，共 4 台 | 96 核 / 1TB | RedHat 8.3 | 大内存 CPU 计算 | |
| fat_old | fatold01-02 节点 | fatold01:80 核<br>fatold02:40 核 | RedHat 7.8 | 大内存 CPU 计算 | |

#### 1.3.3 GPU 计算队列

| 队列名称 | 节点 | 配置 | 操作系统 | 主要用途 | 备注 |
|---|---|---|---|---|---|

| 队列名称 | 节点 | 配置 | 操作系统 | 主要用途 | 备注 |
| --- | --- | --- | --- | --- | --- |
| gpu | a800gpu01<br>a800gpu02<br>a800gpu03<br>a800gpu04 | a800gpu01: 4 卡 A800<br>a800gpu02/03/04:<br>8 卡 A800 | a800gpu01:<br>RedHat 8.1<br>a800gpu02/03/04:<br>RedHat 8.3 | GPU 计算 | 开启 GPU（MIG）技术；高版本 GCC 作业建议使用-m 参数 |
| nonmig | A100GPU01<br>A40GPU01<br>A40GPU02 | A40GPU01/03:<br>4 卡 A40<br>A100GPU01:8 卡 A100 | A40GPU01:<br>RedHat 8.3<br>A40GPU02:<br>RedHat 8.1<br>A100GPU01:<br>RedHat 7.8 | 显存计算量大的 GPU 计算 | 未开启 MIG |
| gpu_old | RTX3090GPU01 | 4 卡 RTX 3090 | RedHat 7.8 | GPU 计算 | 开启显卡独占模式 |

**多实例 GPU（MIG）技术:**

MIG（Multi-Instance GPU）是 NVIDIA 推出的 GPU 切片技术，可将单张物理 GPU（如 A800）在硬件层面安全划分为多个独立的 GPU 实例。每个实例拥有专属的计算核心、显存和带宽资源，相互隔离、互不干扰。通过该技术，单张物理 GPU 卡能同时分配给多个任务使用，在保证性能与稳定性的前提下，提升 GPU 资源利用率。

**关于 glibc 版本的重要提示:**

1.若程序需要自行编译，且依赖登录节点系统自带的 glibc 版本，请注意不同节点的 glibc 版本差异。由于 a100gpu01 节点的 glibc 版本（2.17）较低，无法兼容运行

需要高版本 glibc 的自编译程序，请在提交作业时指定计算节点为 a40gpu01 或 a40gpu02。

2.若直接使用 Conda 环境运行程序，则不受上述限制，可以提交至 a100gpu01，或不指定节点，由调度系统自行分配。

glibc 版本 2.28：a40gpu01、a40gpu02、a800gpu02、a800gpu03、a800gpu04、a800gpu01

glibc 版本 2.17：a100gpu01、rtx3090gpu01

### 1.3.4 AI 计算队列

| 队列名称 | 节点 | 配置 | 操作系统 | 主要用途 | 备注 |
|---|---|---|---|---|---|
| ai_excel | A800GPU02、A800GPU03、A800GPU04、 | 8 卡 A800 | RedHat 8.3 | AI 计算 | 与 gpu 队列共享节点，使用方式相同 |

## 1.4 资源池分类

平台资源池分为“笃学资源池”和“创新资源池”。两个资源池的存储空间（包括家目录）及 Conda 环境完全隔离，作业无法跨池提交，环境不能混用。

两个资源池的主要差异如下表所示。

| 对比项 | 创新资源池 | 笃学资源池 |
|---|---|---|
| 用户范围 | 依托国家级或省部级项目的用户 | 无国家级、省部级项目依托的用户 |
| 可用队列 | normal、normal_old（仅 MS 计算）、arm、fat、fat_old、gpu、nonmig、ai_excel | common（CPU 计算）、gpu_old（GPU 计算） |
| 可用服务 | 支持门户所有集成软件提交作业，也可使用 SSH 命令行提交 | 计算作业仅支持 SSH 命令行提交，图形应用和 MS 可以使用门户 |
| 登录节点 | 10.131.0.202、10.131.0.203 | 10.131.0.204 |

| | | |
|---|---|---|
| 门户 | 支持门户方式 https://10.131.0.199 | |

创新资源池用户：可通过登录不同的资源池使用所有队列。

笃学资源池用户：CPU 作业请提交至 common 队列，GPU 作业请提交至 gpu_old 队列。所有作业需通过 SSH 命令行方式提交。

1.5 账号及缴费

收费办法正在推进，目前不产生费用，后续费用将通过长安大学大型仪器平台从用户账号依托项目经费中收取。CPU 作业按照核时计费、GPU 作业按照卡时计费，收费详见长安大学校级算力公共服务平台收费标准。

# 第二章 快速上手

本章旨在帮助用户快速掌握平台的基本使用方法。平台支持命令行和门户网站两种使用方式。本章还将介绍作业运行的基本限制、常见示例及注意事项，以便用户顺利开展计算任务，避免操作误区。

## 2.1 命令行方式

命令行方式是平台的基础使用方式，适用于所有用户（创新资源池和笃学资源池）。用户需通过 SSH 客户端登录平台登录节点，使用 Linux 命令进行文件管理、作业提交和查看等操作。

主要操作流程：

1. 登录平台：使用 SSH 客户端（如 Xshell、MobaXterm、Putty 等）连接到平台登录节点，命令格式为 ssh 用户名@登录节点地址。

2. 准备作业脚本：包括作业选项、环境加载和可执行命令等。

3. 提交作业：使用作业提交命令 jsub 将任务提交到计算队列。

4. 查看作业状态：使用 jjobs 命令查看作业运行状态。

5. 管理作业：使用 jctrl 命令控制作业：

6. 查看作业结果：作业完成后，输出文件默认保存在提交作业时的工作目录。

常用命令：

| 命令示例 | 功能说明 |
| --- | --- |
| `ssh username@login_node` | 登录平台 |
| `jsub job_script.sh` | 提交作业脚本 |
| `jjobs` | 查看作业状态 |
| `jctrl` | 取消指定作业 |
| jqueues | 查看队列资源状态 |

## 2.2 门户方式

门户方式是平台提供的 Web 图形化使用方式。用户通过浏览器访问平台门户网站，通过图形界面即可完成作业提交、管理和查看。

主要操作流程:

1. 访问门户：在浏览器中输入平台门户网址，使用平台账号登录。

2. 上传文件：通过门户的文件管理功能上传作业所需文件。

3. 选择软件：在门户集成的软件应用列表中选择所需软件。

4. 配置作业参数：填写作业名称、队列、计算资源（CPU 核数、内存、运行时长等）。

5. 提交作业：点击提交按钮，系统自动生成作业脚本并提交到计算队列。

6. 查看作业：在门户的作业管理界面查看作业状态、输出结果。

**注意事项:**

平台支持的浏览器版本有：Microsoft Edge107 及以上、Mozilla Firefox 115 及以上、Google Chrome 87 及以上，以及 Apple Safari（版本 15.6.1、17.4、18.2）。其他浏览器也可以访问，但可能需要手动修改浏览器的安全配置项，兼容性无法完全保证。建议优先使用 firefox 浏览器，以获得最佳兼容性及使用体验。

### 2.3 作业限制

为保障平台资源的合理使用和公平调度，系统对用户作业实施限额管理。默认限制如下。如需突破默认限制，可向平台管理员提交特殊申请。

| 限制项 | 默认限制 | 说明 |
| --- | --- | --- |
| 单用户最大同时运行作业数 | 10 个 | 同时处于运行状态的作业数量上限 |
| 单用户最大等待作业数 | 2 个 | 排队等待中的作业数量上限 |
| 单用户同时打开图形会话数 | 2 个 | 同时进行的图形界面会话数量上限 |
| 单作业默认最大运行时长 | 10 天 | 作业运行达到 10 天后自动终止 |
| normal 队列以及 cuold 队列作业最大运行时长 | 3 天 | 作业运行达到 3 天后自动终止 |
| 单作业最大可延长时间 | 60 天 | 通过特殊申请可延长的最大运行时长 |

**注意事项**:

- 超出限制的作业提交将被系统拒绝，请合理安排作业提交计划。
- 如需长时间运行的大规模计算任务，建议提前联系管理员申请调整限制。
- 图形会话长期占用资源，使用完毕后请及时关闭。用户必须手动注销或退出。否则，计算任务会持续占用节点资源（含 CPU、内存及应用授权数），并持续产生计费，直至因超时被系统强制终止或管理员介入处理。

## 2.4 helloWorld 示例

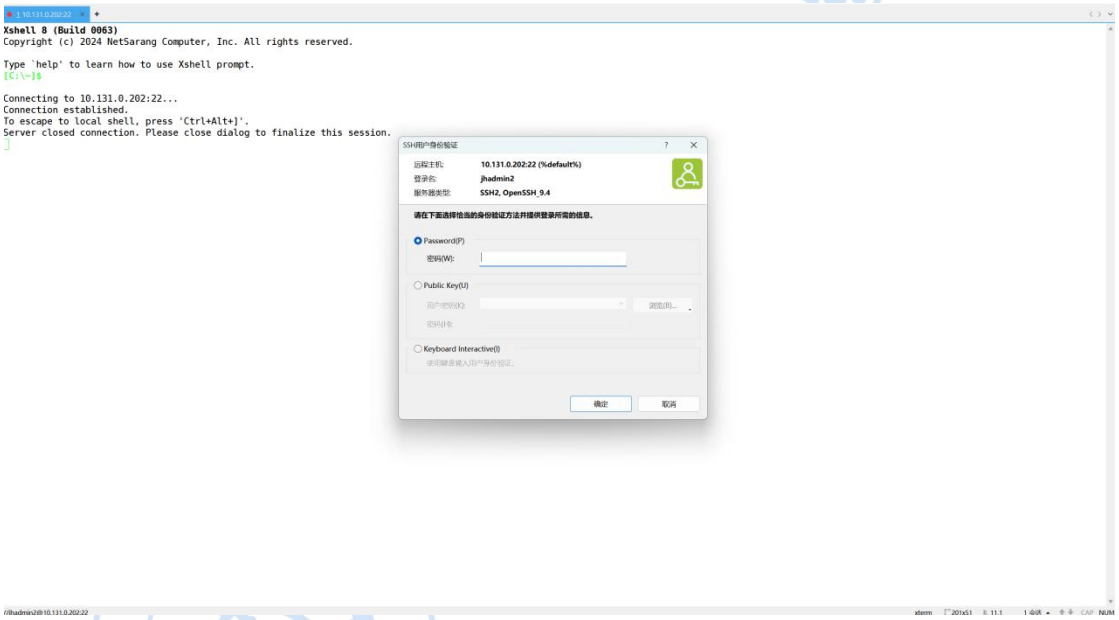
1、1.使用 ssh 客户端连接登录节点，以 Xshell 为例。




连接登录节点

2.输入密码登录。

登录账号为学工号，密码为申请账号时自行设置。登录后，当前目录即为家目录/storage/public/home。




输入密码登录

3.在当前目录下，执行“mkdir helloWorld” 命令，创建helloWorld目录。

4.进入helloWorld目录内，cd helloWorld。编写作业执行脚本submit.sh，需保证脚本有执行权限，chmod +x submit.sh。

```
#!/bin/bash
#JSUB -q common
```

```
#JSUB -n 1
#JSUB -e helloWorldError.%J
#JSUB -o helloWorldOutput.%J
#JSUB -J helloWorld
echo "hello world"
```

作业执行脚本内容说明:

echo "hello world" : 将 hello world 输出。

-q common: 指定队列为 common（注意，用户需确认自己有该队列的使用权限）。

-n 1: 指定该作业所需核数为 1 核

-e helloWorldError.%J: 错误信息输出到 helloWorldError.<作业号>的文件中

-o helloWorldOutput.%J: 标准输出信息输出到 helloWorldOutput.<作业号>的文件中

-J helloWorld : 指定作业名字 helloWorld

5.使用 jsub 命令提交作业，可看到已提交作业的作业号。

```
[100095@ln03 helloWord]$ jsub < submit.sh
Job <307540> is submitted to queue <common> and default application <default>
```

6. 通过作业号，使用 jjobs 查看作业。

```
[100095@ln03 helloWord]$ jjobs 307540
JOBID      USER     STAT  QUEUE     FROM_HOST EXEC_HOST JOB_NAME     SUBMIT_TIME    ORDER
307540     100095   DONE  common    ln03      cu074     helloWorld   May 19 08:57   -
```

查看作业

可看到，该作业的状态为“DONE”（已完成），执行队列是“common”，执行节点是“cu074”。

如果需要看作业详情可以加-l 参数，jjobs -l 作业号。

7. 使用 ls -all 可以看到目录下生成了 helloWorldError.作业号和 helloWorldOutput.作业号两个文件。使用文本编辑器或者 cat helloWorld.作业号命令查看作业输出。

```
[jhadmin2@ln01 helloWorld]$ cat hellowWorldOutput.296193
hello world


Sender: Unischeduler System <jhadmin2@ln01>
Subject: Job 296193: </storage/public/home/jhadmin2/.jhjobs/1778047449.296193.1.shell> Done

Job </storage/public/home/jhadmin2/.jhjobs/1778047449.296193.1.shell> was submitted from host <ln01> by user <jhadmin2>.
Job was executed on host(s) <8*cu032>, in queue <normal>, as user <jhadmin2>.
</storage/public/home/jhadmin2> was used as the home directory.
</storage/public/home/jhadmin2/helloWorld> was used as the working directory.
Started at 2026-05-06 14:04:09
Results reported at 2026-05-06 14:04:09

Your job looked like:

------------------------------------------------------------
# User input
/storage/public/home/jhadmin2/.jhjobs/1778047449.296193.1.shell
------------------------------------------------------------

Successfully completed.

Resource usage summary:

               Max Memory:     3.1 MB
               Max Processes:  1
```

查看作业输出

## 2.5 节点资源选择

### 2.5.1 关于 CPU 核数选择的重要说明

用户在提交作业时通过命令参数指定的 CPU 核数，仅决定了调度系统会分配多少核给该作业。而程序实际能够使用的核数，取决于程序自身的并行设计能力。

**关键原则：**申请核数应与程序实际能使用的核数相匹配。

若申请过少，则程序因得不到足够的计算资源而运行缓慢甚至出错。若申请过多，则造成大量核数闲置浪费，且因占用队列资源导致作业排队等待时间不必要地延长。

**操作建议**：用户应根据自己程序的实际并行规模进行选择。若是串行程序，则应选择 1 个核心。若是并行程序，则根据实际需要的并行核数进行选择。

用户应避免恶意或过失的资源浪费。平台管理员会实时监控作业资源使用情况。若发现用户存在过度占用资源的行为，管理员将会强制结束相关作业。

### 2.5.2 节点资源的选择

在集群上执行自己的并行作业之前，需要 CPU 核心数(-n)等参数，在某些情况下还需要选择合适的 GPU 数量。本页演示和分析不同参数情况下的作业表现，以帮助我们找到设置这些参数的最佳值。

#### 2.5.2.1 CPU 和 GPU 使用率

首先要考虑运行脚本能够利用好申请的资源。提交脚本后，用 jqueues 获取申请到的计算节点。

```
$ jqueues
```

ssh 登录到计算节点

```
$ssh NODENAME,  例如 ssh cu001
```

登录成功后，用 top -u $username(你的用户名) 命令查看 cpu 使用率。对于 gpu 卡，使用 nvidia-smi 或者 watch -n 1 nvidia-smi 动态查看 gpu 使用率。

**1.多线程 CPU 利用率**

申请资源如下:

#JSUB -n=32

#JSUB -R "span[hosts=1]"

cpu 利用率如下图所示

```
[root@mu01 nicuser]# jjobs -l 331840

Job <331840>, Job Name          , User              , Project <default>, Status
                    <RUN>, Queue <normal>, Application <abaqus>, APS Priority
                    <5000>, Command <abaqus  job=          cpus=32 int>
Fri May 29 11:51:00: Submitted from host <mu02>, CWD </storage/public/jhinno64/
                    appform/data/spooler/   .     ./   .  .    2911510031565
                    02699>, OutFile File <output.331840.txt>, 32 Processors Re
                    quested, Requested Resource <span[hosts=1]>;
Fri May 29 11:51:01: Started on 32 Hosts/Processors <32*cu056>, Execution Home
                    </storage/public/home/2025021056>, Execution CWD </stora e
                    /public/jhinno64/appform/data/spooler,                   _
                    260529115100315650269 9>, Execution user <      .    ,
Fri May 29 14:53:30: Resource usage collected.
                    The CPU time used is 344314.12 seconds.
                    IDLE RATE(cputime/slots/runtime): 0.9857
                    CPU(s) Utilization: 3167.65%
                    MEM: 1749 Mbytes; SWAP: 0 Bbytes
                    PGIDS:733700 733808; PIDS:733700 733822 733710 733786 733820
                    734074 733808

 RUNLIMIT
 14400.0 min

 SCHEDULING PARAMETERS:
             r15s      r1m      r5m     r15m       ut       pg       io       ls
 LoadSched       -        -        -        -        -        -        -        -
 LoadStop        -        -        -        -        -        -        -        -

              it      tmp     swap      mem
 LoadSched       -        -        -        -
 LoadStop        -        -        -        -
```

图中， CPU 使用率为 3167.65%，说明使用了多线程，32 个核跑满了。

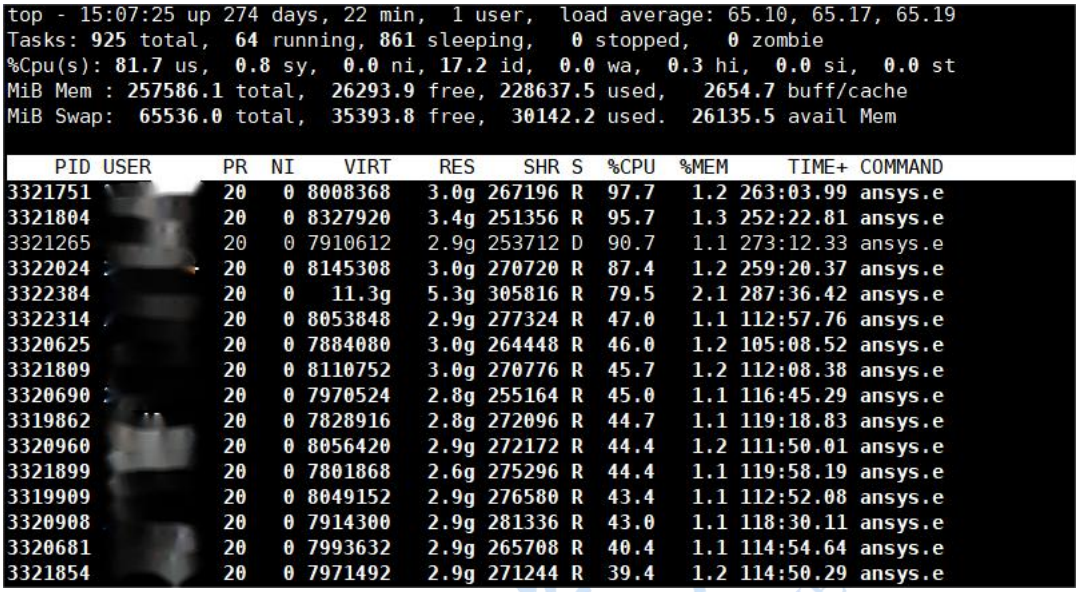
**2.并行作业的 cpu 利用率**

申请资源如下:

#JSUB -n 32

#JSUB -R "span[ptile=16]"

cpu 利用率如下图所示




图中，任务相关进程 16 个，大部分进程的利用率低于 50%，证明 CPU 利用率没有利用好。

### 3. GPU 利用率

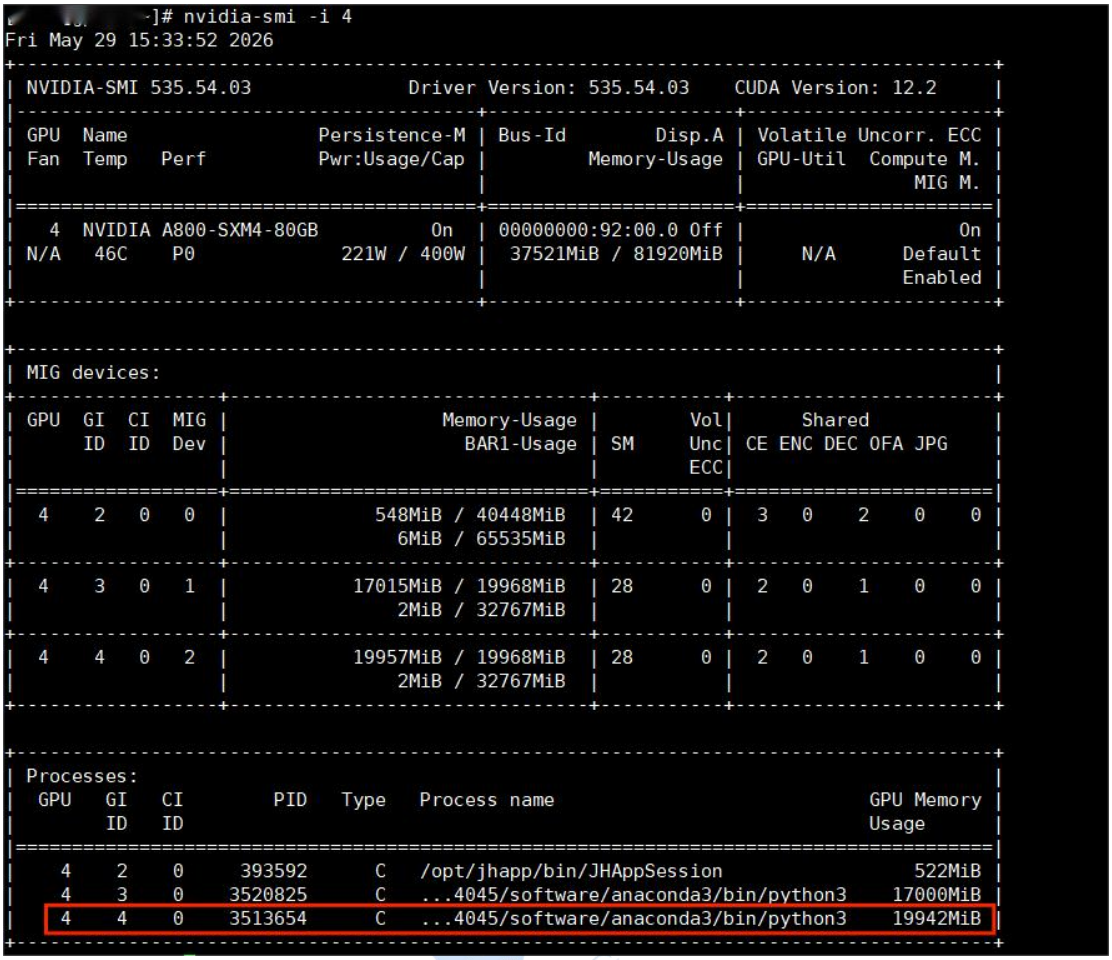
申请资源如下：

#JSUB -n 8

#JSUB -q gpu

#JSUB -gpgpu '1 mig=2'

gpu 使用率如下：




图中，脚本使用了一张 gpu 卡并且 mig 为 2g.20gb ，其中显存使用 19942MiB，gpu 利用显存利用率 99.97%。

GPU 作业在考虑多个 GPU 之前，应首先证明使用单个 GPU 时 GPU 的高利用率。 如果 GPU 利用率对于单个 GPU 来说足够高，那么你应该如上述例子构造表来分析多个 GPU 情况下作业并行效率。

## 错误示例

示例 1

```
#!/bin/bash
#JSUB -J myjob
```

```
#JSUB -n 20
#JSUB -R "span[hosts=1]"      # 每个节点的任务数
#JSUB -q normal
#JSUB -o out.%J
#JSUB -e err.%J
conda activate myenv
python demo.py
conda deactivate
```

此脚本设置一个进程，20 个 cpu，用多线程来跑 python 代码。首先需要确定代码是否支持多线程，另外据了解，python 对于多线程的支持并不太好，可以尝试用多进程并行的办法。

示例 2

```
#!/bin/bash
#JSUB -J my_cp2k_job
#JSUB -n=64
#JSUB -q normal
#JSUB -o out.%J
#JSUB -e err.%J
module load cp2k-2023.1-gcc831
mpirun -n 10 cp2k.psmp -i Ac.inp -o test.out
```

此脚本申请了 64 个 cpu，但是使用 mpirun 的时候只设置了 10 个，造成资源浪费。

示例 3

```
#!/bin/bash
#JSUB -J my_cp2k_job
#JSUB -n 64
#JSUB -R "span[ptile=16]"
#JSUB -q normal
#JSUB -o out.%J
#JSUB -e err.%J
module load cp2k-2023.1-gcc831
mpirun -n 64 cp2k.psmp -i Ac.inp -o test.out
```

此脚本使用 4 个节点跑脚本，每个节点 16 个进程(一个进程对应一个 cpu)，但建议尽量将程序运行在同一节点上(-n 64，不要设置-R "span[ptile=16]")，以提高计算性能。

# 第三章 平台接入

## 3.1 命令行接入

可通过第三方 ssh 工具（Xshell、MobaXterm、Putty 等）登录高性能计算平台。集群共三台登录节点，ssh 登录

端口为 22。登录账号为学工号，密码为申请账号时自行设置。根据不同用户资源池，登录节点配置如下。

创新资源池用户：共配置了 2 台登录节点，IP 地址分别为 10.131.0.202 和 10.131.0.203。两台节点配置完全一致，用户可任选其一登录。

笃学资源池用户：配置了 1 台登录节点，登录地址为 10.131.0.204。

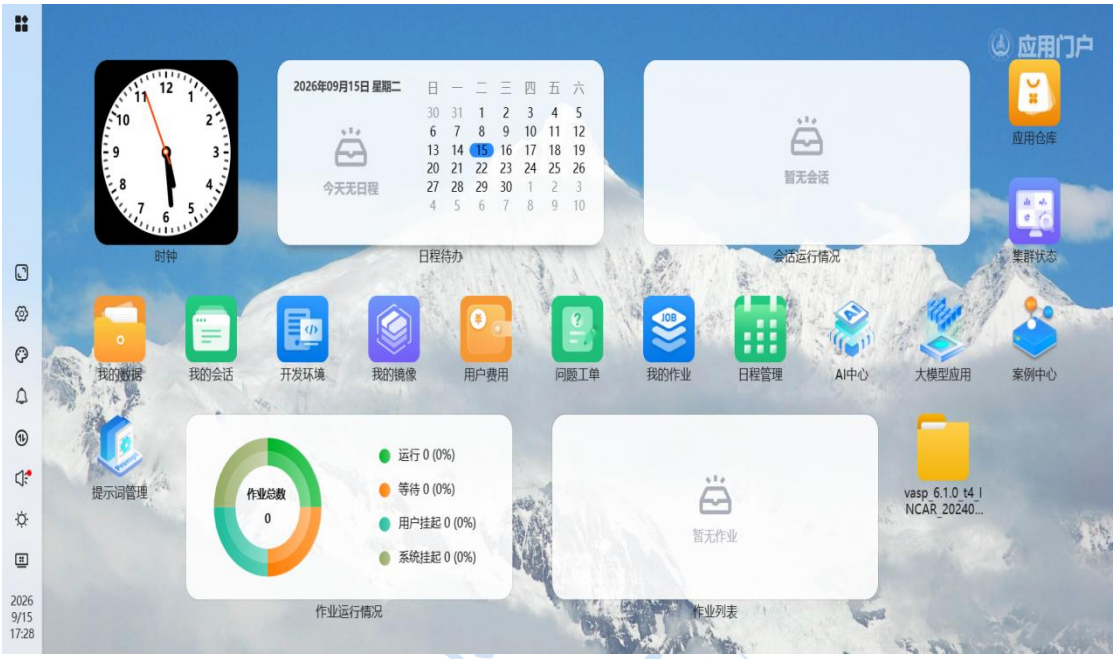
### 3.2 门户接入

用户可通过浏览器（推荐使用firefox）登录高性能计算平台门户，门户地址为https://10.131.0.199。门户登录账号和密码，与ssh登录账号和密码相同。登录页面如下图所示。




登录时，如果连续密码输入错误超过5次，账户将被自动锁定，默认锁定时长为30分钟。锁定时间结束后，账户状态将自动恢复。

登录应用门户软件后，系统将会显示用户桌面。在桌面中排列显示了用户可以使用的应用功能图标，提供了开始菜单、通知、系统公告、设置等功能入口。




用户桌面

# 第四章 数据管理

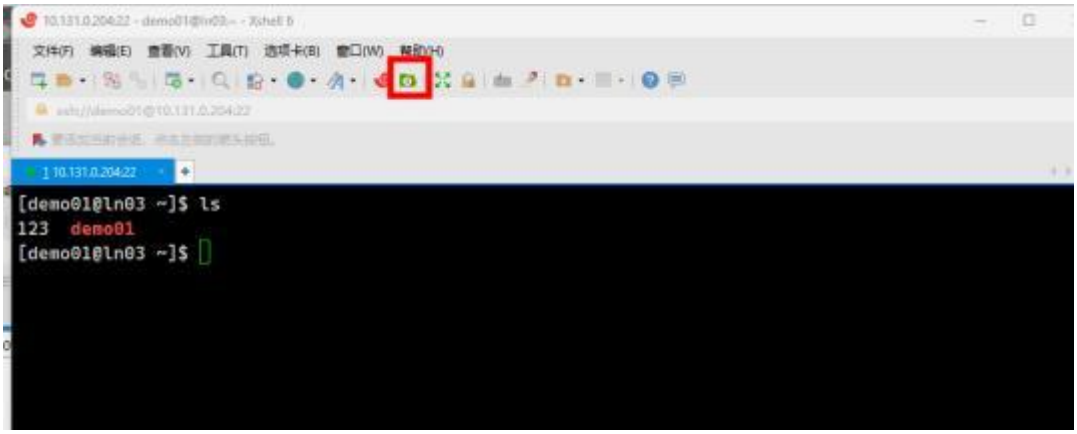
## 4.1 文件传输

### 4.1.1 命令行方式

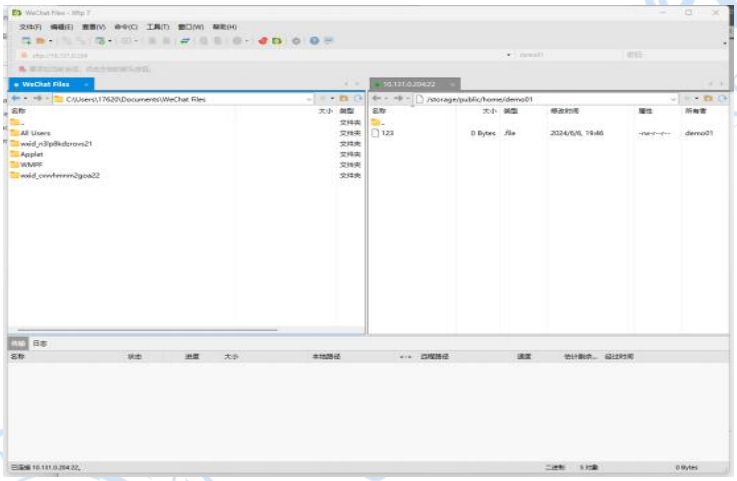
用户可通过远程拷贝（scp、rsync 等命令）或通过第三方工具 winscp、sftp 的方式进行数据传输。

以 Xshell 为例：

1.登录 Xshell 后，点击工具栏中的 xftp 图标（如下图所示，绿色图标）。




2. 打开 xftp 后，即可将本机电脑上的文件上传至平台。



在 xftp 中显示的文件，创新资源池用户看到的是平台“我的数据”中的用户家目录，笃学资源池用户看到的与家目录中 old_home 相同。

**注意事项：**

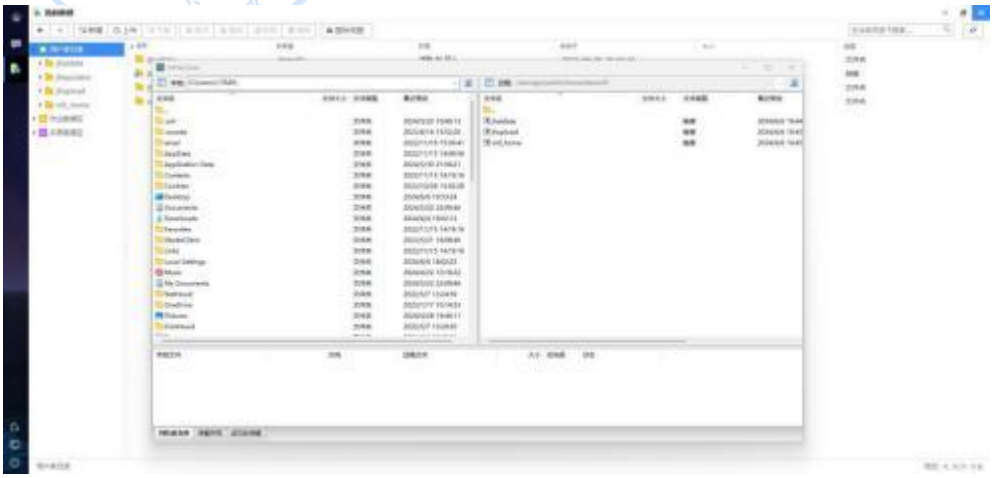
如果网络环境不稳定，导致从平台下载文件失败，建议优先使用 xftp 的方式上传和下载文件，以提高传输成功率。

### 4.1.2 门户方式

浏览器打开 https://10.131.0.199/进入高性能计算平台的登录页面，下载**客户端组件**并安装。

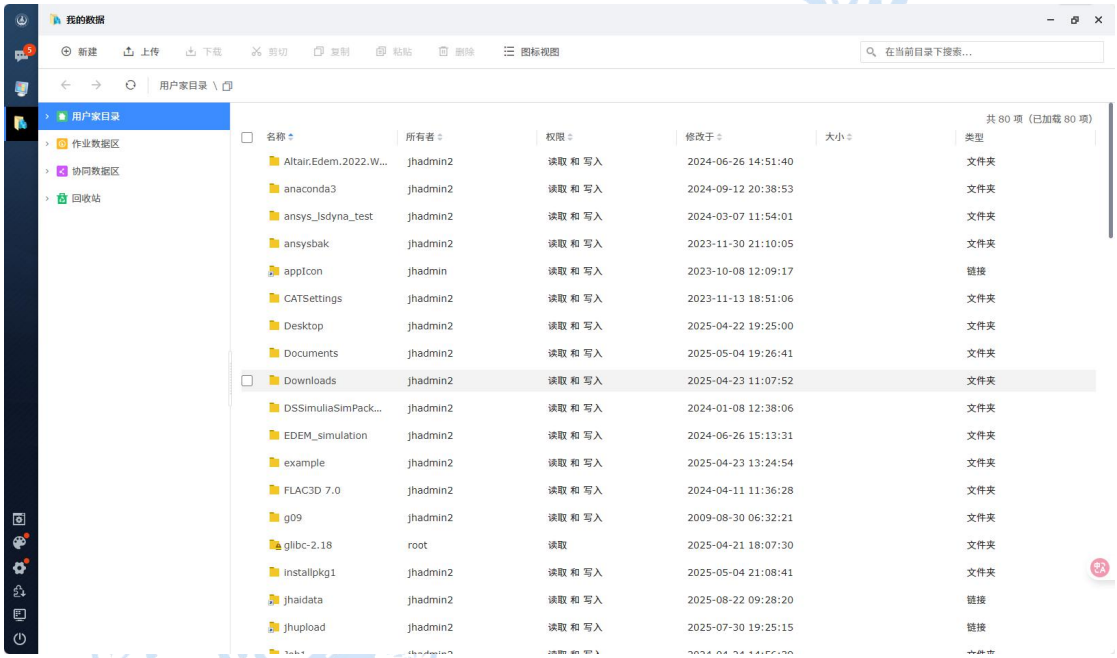



登录平台后，打开“我的数据”，在空白区域右键单击，选择上传，平台将会自动调用文件传输客户端。如下图所示，左边显示的是本机电脑的文件，右边是用户家目录。右键选择文件，点击下载或者上传，即可进行文件传输。



### 4.2 数据协同

打开“我的数据”，在左侧可看到用户家目录、作业数据区和协同数据区。用户家目录显示服务器端用户 home 目录下的文件，支持上传、打开、下载、剪切、复制、粘贴、压缩、删除等操作。作业数据区以列表形式展示用户提交作业所产生的数据。协同数据区用于用户之间共享文件，可将文件共享给指定用户或用户组。




数据目录

## 第五章 作业管理

### 5.1 命令行方式

用户可通过命令行提交和管理作业。提交作业时，通过脚本参数申请的核数，仅决定调度系统为该任务分配的资源数量，而程序实际能使用的核数取决于其自身的并行能力。因此，用户务必要根据自己的程序实际能使用多少核数进行

选择，串行程序则选择一个核心，并行程序则按照实际需要的并行规模进行选择。合理选择核数可避免资源浪费或作业排队时间过长。

作业选项（如队列名、核数等）可通过两种方式指定：在脚本中使用#JSUB（推荐），或在jsub命令行中作为参数指定。两种方式可混合使用，命令行参数优先级高于脚本中的#JSUB。基本流程为：编写脚本→jsub < 脚本文件 提交→ jjobs查看状态。

### 5.1.1 编写脚本

作业脚本是一个标准的Shell脚本，首行必须指定解释器（如 #!/bin/bash），脚本中包含作业执行命令，并可通过#JSUB注释行指定作业选项。Shell脚本格式如下：

```
#!/bin/bash
#JSUB -q <队列名>
#JSUB -n <CPU数>
#JSUB -e error.%J
#JSUB -o output.%J
#JSUB -J <作业名>
# 加载环境变量（根据需要添加）
module load <软件模块>
# 执行作业命令
```

<作业执行命令>

**参数说明:**

| 参数 | 说明 |
| --- | --- |
| #!/bin/bash | 指定脚本解释器，必须为脚本第一行 |
| #JSUB -n <核数> | 指定执行该作业所需的核数 |
| #JSUB -q <队列名> | 指定该作业运行时使用的队列。作业在资源不足时排队，资源满足时派发执行。用户可通过jqueues -u <账号>查看自己有权使用的队列。 |
| #JSUB -J <作业名> | 指定作业名，方便识别作业。 |
| #JSUB -o output.%J | 指定输出文件。这个输出指的是调度系统的输出。%J 自动替换成作业号，最终将产生一个 output.<作业号>的文件。 |
| #JSUB -e error.%J | 指定错误输出文件 |
| module load | 加载所需软件环境 |
| <作业执行命令> | 实际运行的计算命令 |

### 5.1.2 提交作业

编写好作业脚本后，使用jsub命令提交。作业选项（如队列名、核数等）既可以在脚本中通过#JSUB指定，也可以在命令行中作为jsub的参数指定。命令行参数的优先级高于脚本中的#JSUB，可用于临时覆盖脚本中的默认设置。

**作业提交命令**

*jsub [选项] ./<脚本文件>*

**常用选项:**

| 选项 | 说明 |
| --- | --- |
| -n <核 数> | 指定执行该作业所需的核数 |
| -q <队列名> | 指定该作业运行时使用的队列。作业在资源不足时排队，资源满足时派发执行。用户可通过jqueues -u <账号>查看自己有权使用的队列。 |
| -J <作业名> | 指定作业名，方便识别作业。 |
| -o <输出文件> | 指定输出文件。这个输出指的是调度系统的输出。%J 代表的是作业号，output.%J 将产生一个 output.<作业号>的文件。 |
| -e <错误文件> | 指定错误输出文件。error.%J将产生一个error.<作业号>的文件。 |

| | |
|---|---|
| -m <节点名> | 指定执行节点。通常不需要指定。 |
| -gpgpu <参数> | 指定GPU资源（详见GPU作业示例） |
| -R "span[hosts=1]" | 限制作业在单台节点上运行 |

**队列说明:**

1.队列详细介绍请参见1.3“平台队列”

2.只要队列状态为打开，用户即可向该队列提交作业

3.只有当队列为激活状态时，作业才能被派发执行

4.非并行作业将提交到默认队列中进行计算

5.每个队列都有自己的执行条件，适用于该队列中的所有作业

6.作业在满足所有执行条件之前，将一直处于等待状态

**单节点执行限制:**

如果作业需要在单台计算节点上执行（即不跨节点），请注意以下要求:

1.在提交命令中加入 -R "span[hosts=1]" 参数

2.无需通过 -m 参数指定特定节点

3. 申请的核心数不能超过队列中单台计算节点的物理核心数

**示例**

示例1：使用脚本中定义的选项（最常用）

```
[jhtest@ln01 ~]$ jsub < submit.sh
Job <12834> is submitted to queue <normal>
```

示例2：命令行覆盖脚本中的队列和核数

```
[jhtest@ln01 ~]$ jsub -q gpu -n 4 < submit.sh
Job <12835> is submitted to queue <gpu>
```

示例3：命令行指定脚本中未定义的选项

```
[jhtest@ln01 ~]$ jsub -R "span[hosts=1]" <
submit.sh
Job <12836> is submitted to queue <normal>
```

```
# 示例4：完整命令行示例（不依赖脚本中的#JSUB）
[jhtest@ln01 ~]$ jsub -q normal -n 32 -J my_job -o
output.%J -e error.%J ./my_script.sh
Job <12837> is submitted to queue <normal>
```

**使用建议**:

推荐做法：将固定的作业选项（如队列、核数）写在脚本的#JSUB中，便于保存和复用

临时调整：在命令行中指定选项，覆盖脚本中的默认值，无需修改脚本文件

### 5.1.3 作业查看与终止

作业提交成功后，系统会返回一个作业号（Job ID），用户可通过该作业号查看作业状态或终止作业。

**1.作业查看命令**

*jjobs [[-a] [-l] <作业号>]*

参数说明：

| 参数 | 说明 |
| --- | --- |
| jjobs | 显示目前正在排队或运行的作业 |
| jjobs -a | 显示最近完成的作业及正在运行或排队的作业 |
| jjobs <作业号> | 显示指定的作业状态 |
| jjobs -l <作业号> | 显示指定作业的详细信息 |

**2.作业终止命令**

*jctrl kill <作业号>*

### 5.1.4 CPU 作业示例

### 5.1.4.1 VASP 作业

1. 编写脚本

编写脚本，脚本名为 submit.sh

```
#!/bin/bash
#JSUB -q normal
#JSUB -n 32
#JSUB -e error.%J
#JSUB -o output.%J
#JSUB -J test_vasp
source /storage/public/share/jhjob_scripts/getJobHosts.sh
## forintelmpi :
module purge
module load vasp-6.1.0-intel
mpirun -machinefile $nodelist -np $ncpu -ppn 1 vasp_std
```

参数说明:

<#JSUB>: 指定作业的提交选项

<-q normal>: 指定 normal队列

<-n 32>: 指定该作业所需32核

<-m cu001>: 指定该作业运行在 cu001 节点（一般无需指定，本例中未指定）

<-e error.%J>: 错误信息输出到 error.%J 文件中

<-o output.%J>: 标准信息输出到 output.%J 文件中 <-J my_job>: 指定作业名字为 my_job

<-J test_vasp>: 指定作业名字为test_vasp

source /storage/public/share/jhjob_scripts/getJobHosts.sh 获取 cpu 个数信息（需要执行作业时传递给-np $ncpu）

module purge 清理module环境变量，避免环境变量冲突

module load vasp-6.1.0-intel 加载module环境变量

最后一行使用mpi提交vasp_std作业

2.脚本目录

脚本写好后与 vasp 算例文件放到同一目录下

```
[jhadmin2@mu01 vasp1]$ ls
INCAR  KPOINTS  POSCAR  POTCAR  submit.sh
```

3.作业提交

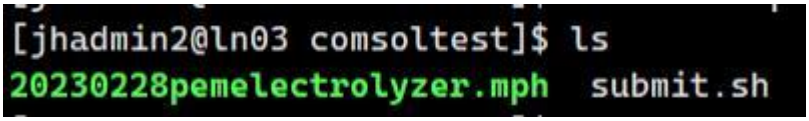
使用 jsub < submit.sh 命令提交作业

```
[jhadmin2@ln01 vasp1]$ jsub < submit.sh
Job <87234> is submitted to queue <normal> and default application <default
```

### 5.1.4.2 COMSOL 作业

1.创建目录

在家目录创建新的文件夹，将comsol算例文件与提交作业脚本submit.sh 放入该目录。




comsol测试目录

2.编写脚本

提交作业脚本submit.sh示例如下:

```
#!/bin/bash
#JSUB -q  common
#JSUB -n  1
#JSUB -e error.%J
#JSUB -o output.%J
source /storage/public/share/jhjob_scripts/getJobHosts.sh
module purge
module load comsol-6.1
comsol batch  -clustersimple  -np $ncpu  -inputfile  20230228pemelectrolyzer.mph  -batchlog batchinfo.log  -mpibootstrap ssh
```

参数解释:

<#JSUB>: 指定作业的提交选项

<-q common>: 指定 common 队列

<-n 1>: 指定该作业所需 1 核

<-m cu067>: 指定该作业运行在 cu067 节点（一般无需指定）

<-e error.%J>: 错误信息输出到 error.%J 文件中

<-o output.%J>: 标准信息输出到 output.%J 文件中

source /storage/public/share/jhjob_scripts/getJobHosts.sh 获取 cpu 个数信息（需要执行作业时传递给-np $ncpu）

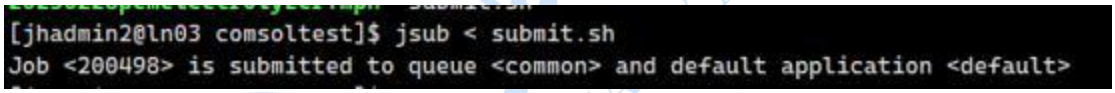
module purge 清理 module 环境变量，避免环境变量冲突

module load comsol-6.1 加载 module 环境变量

最后一行是 comsol 作业执行命令，笃学资源池用户可以根据实际情况修改参数。

3. 作业提交

脚本写好后使用 jsub < submit.sh 命令提交作业




4. 查看作业

提交完成后可以使用 jjobs 命令查看作业，目前正常运行，已经产生过程文件与结果文件。







### 5.1.5 GPU 作业（以 TensorFlow 为例）

提交 GPU 作业时，需注意目标队列是否开启了 MIG（多实例 GPU）技术，这直接影响 -gpgpu 参数的写法:

未开启 MIG 的队列（如 nonmig、gpu_old）: 直接使用整卡，只需指定卡数，格式为 <卡数>

开启 MIG 的队列（如 gpu）: 支持 GPU 切片，需指定切片数量，格式为 '<卡数> mig=<切片数>'

**MIG 功能说明**

gpu 队列开启 MIG（多实例 GPU）切片技术，可将一张物理 GPU 卡切分为多个独立实例。通过“mig=”参数指定显存大小:

| 参数 | 切片数 | 显存大小 |
|---|---|---|
| mig=1 | 1 | 10GB |
| mig=2 | 2 | 20GB |
| mig=3 | 3 | 40GB |
| mig=4 | 4 | 40GB |
| mig=7 | 7 | 80GB |

## 1. 登录

ssh进行登录，在用户家目录下搭建安装 cuda、conda 等机器学习开发环境。

2.编写脚本

进入算例文件所在目录，在此目录编写 submit.sh 脚本。

在整卡使用的情况下，脚本示例如下，其中

```
#!/bin/bash
#JSUB -q nonmig
#JSUB -gpgpu 2
#JSUB -o output.%J
#JSUB -e error.%J
source activate tensorflow
python tensorflow.py
```

参数详解:

#JSUB -q nonmig 指定 nonmig 队列（独占模式，非MIG）

#JSUB -gpgpu 2 指定gpu个数2，表示此作业可以调用2张gpu卡共160G 显存。

#JSUB -o output.%J 指定程序输出信息保存文件

#JSUB -e error.%J 指定报错信息保存文件

source activate tensorflow 加载conda 环境

Python tensorflow.py 作业运行命令

对于使用MIG技术的队列，脚本示例如下:

```
#!/bin/bash
#JSUB -q gpu
#JSUB -gpgpu '1 mig=7'
#JSUB -o output.%J
#JSUB -e error.%J
source activate tensorflow
python tensorflow.py
```

参数详解:

#JSUB -q gpu　　指定gpu 队列

#JSUB -gpgpu ′1 mig=2′　指定gpu 个数1，两块mig

3.准备提交

submit.sh 脚本编写完成后将此脚本与 tensorflow.py 程序放到同一目录下。

4.作业提交

使用 jsub < submit.sh 命令提交作业。

**注意:**

笃学资源池用户的gpu作业只能提交到gpu_old队列。

### 5.2 门户方式

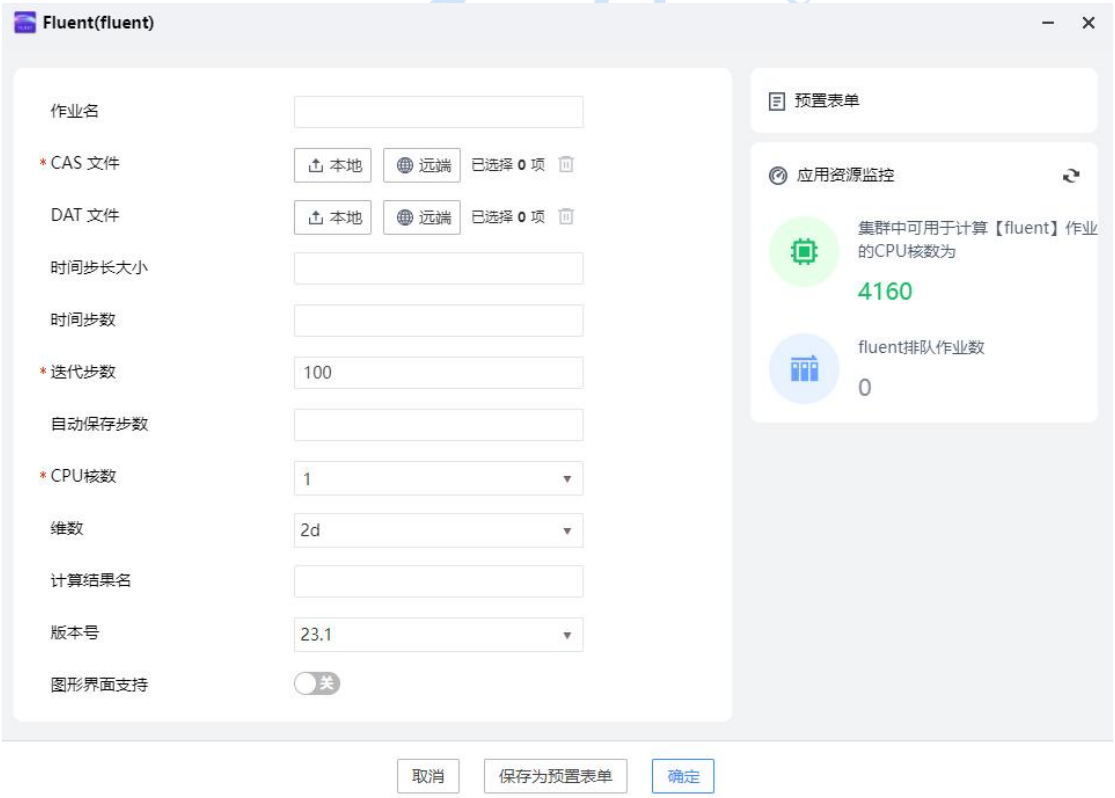
用户选择“应用仓库”，选择相应的计算软件来提交作业。在 5.2.1 中将以 Fluent 作业为例说明作业是如何提交的，其他软件的使用方法详见“第七章 软件应用”章节。

### 5.2.1 提交作业

以 Fluent 作业为例，说明作业提交流程。

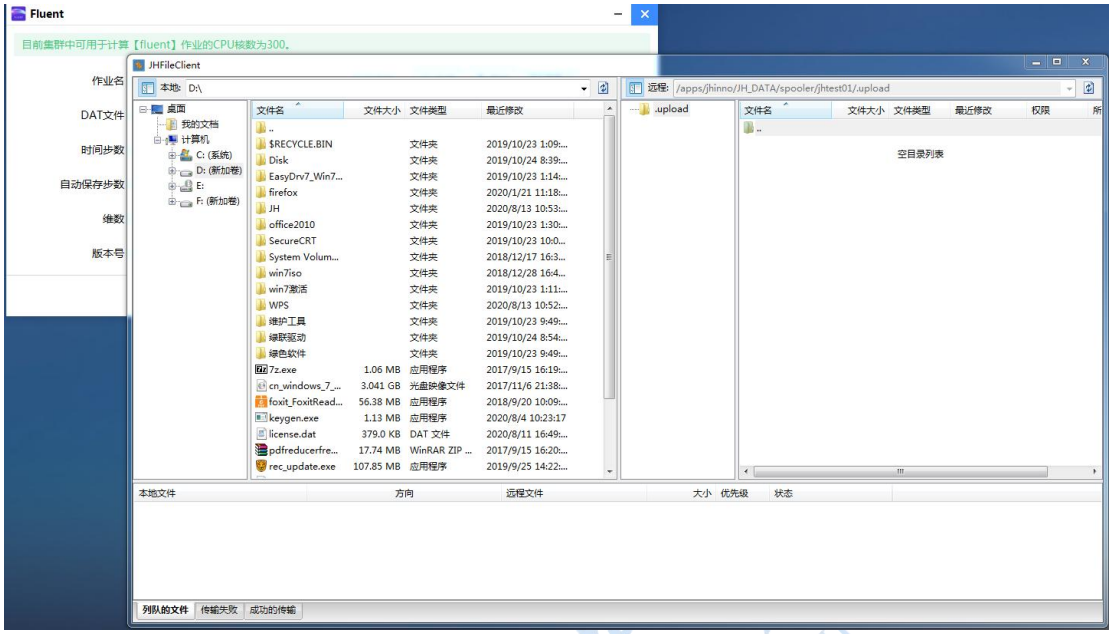
登录门户后，点击应用仓库中 Fluent 图标，系统会弹出如下提交页面。



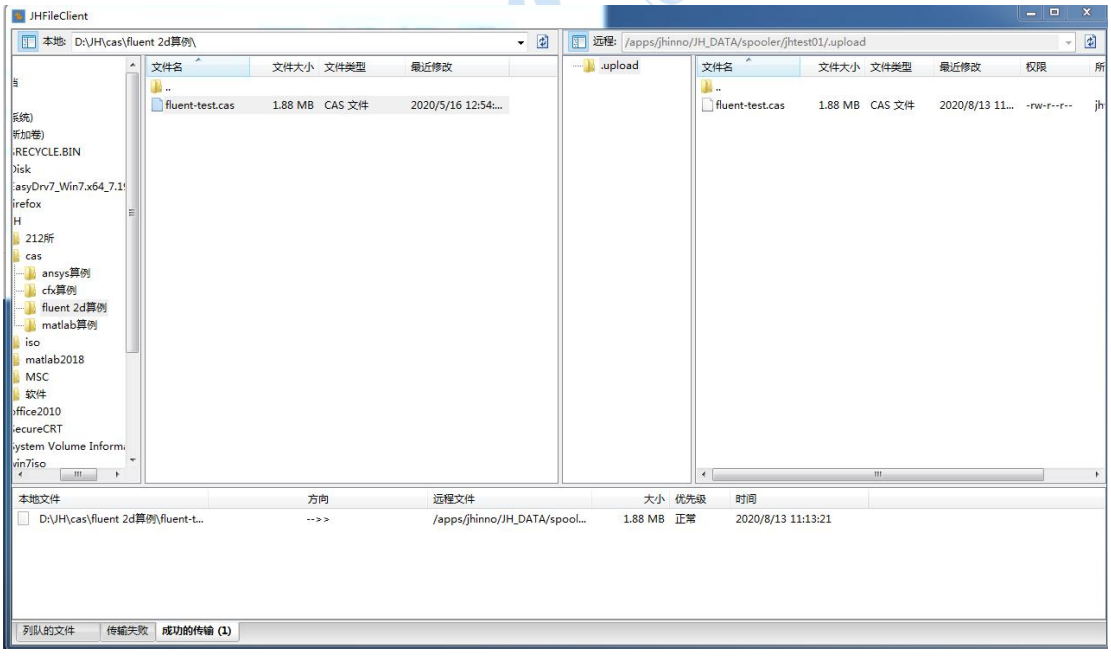




用户可以自主选择作业名称和选择 Fluent 的 CAS 算例文件。

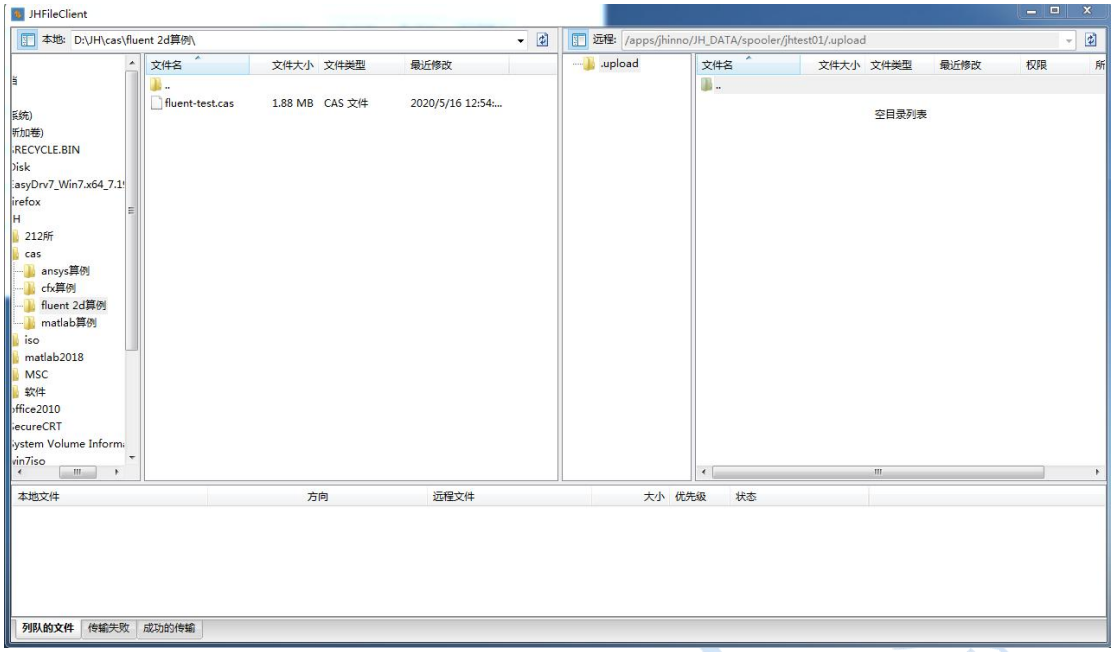
可以点击本地或服务器来选择文件。若点击“本地”，系统在安装客户端软件后会自动弹出 JHFileClient 对话框。




用户可以通过左侧的本地选择对应的目录，如选择【D:\JH\cas\fluent 2d 算例\】：

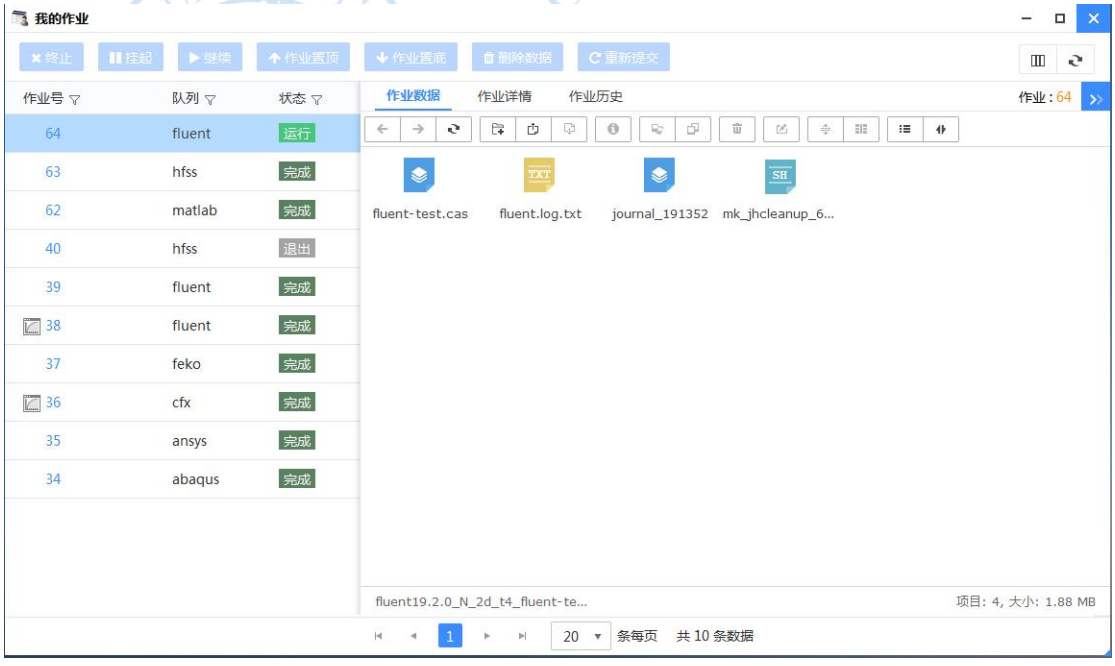



在选择好目录后，在要上传的文件处点击右键，选择上传，文件即开始上传到服务器：

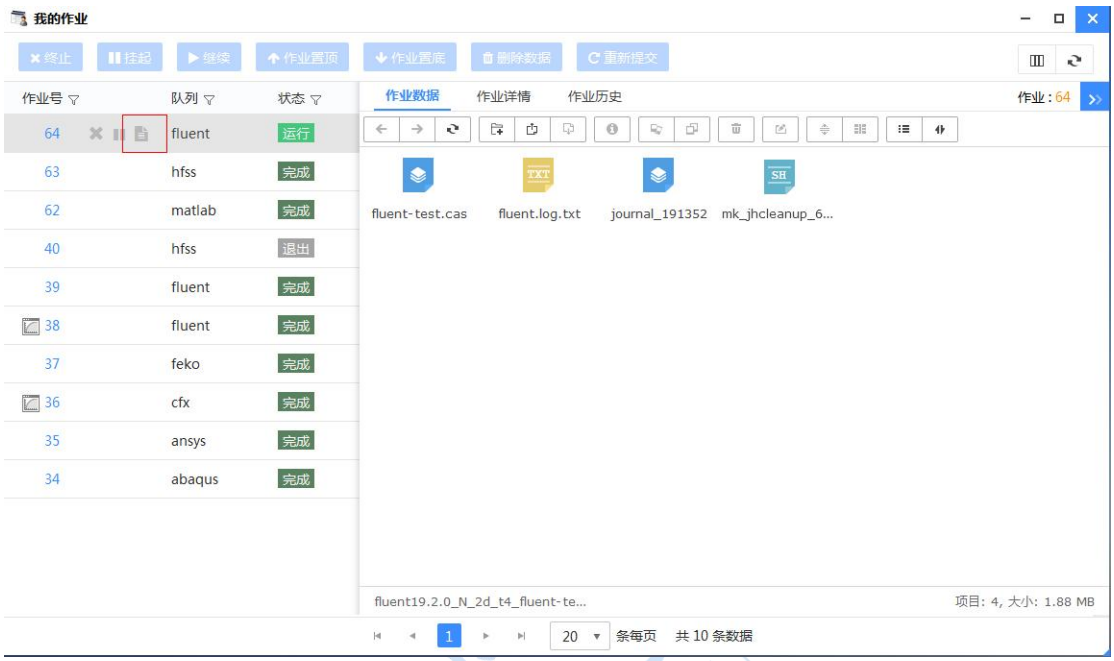



用户也可以点击“服务器端”来选择服务器端数据，进入服务器端后，选择需要计算的文件，点击“确定”。

选择 CPU 核数，填写迭代步数，选择算例类型，点击确定提交作业。系统会自动跳转到作业详细页面，如下图所示。

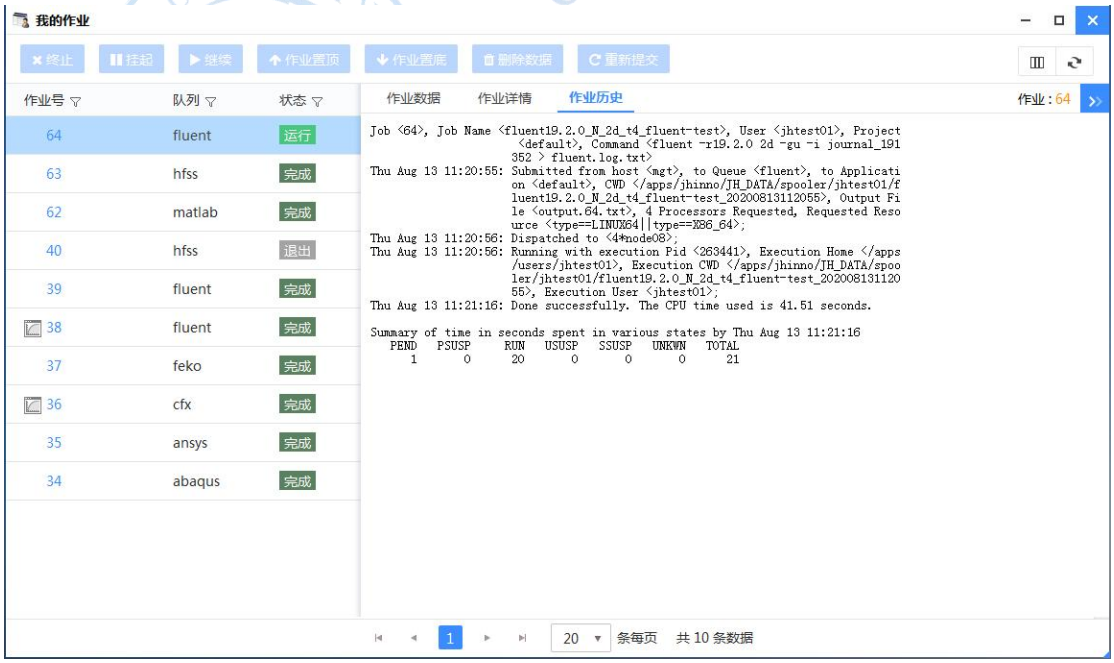


在这个页面中，我们可以点击“刷新”来刷新实时的数据情况和作业情况，同时也可以点击左侧面板中的“详细信息”进入该作业的详细页面：

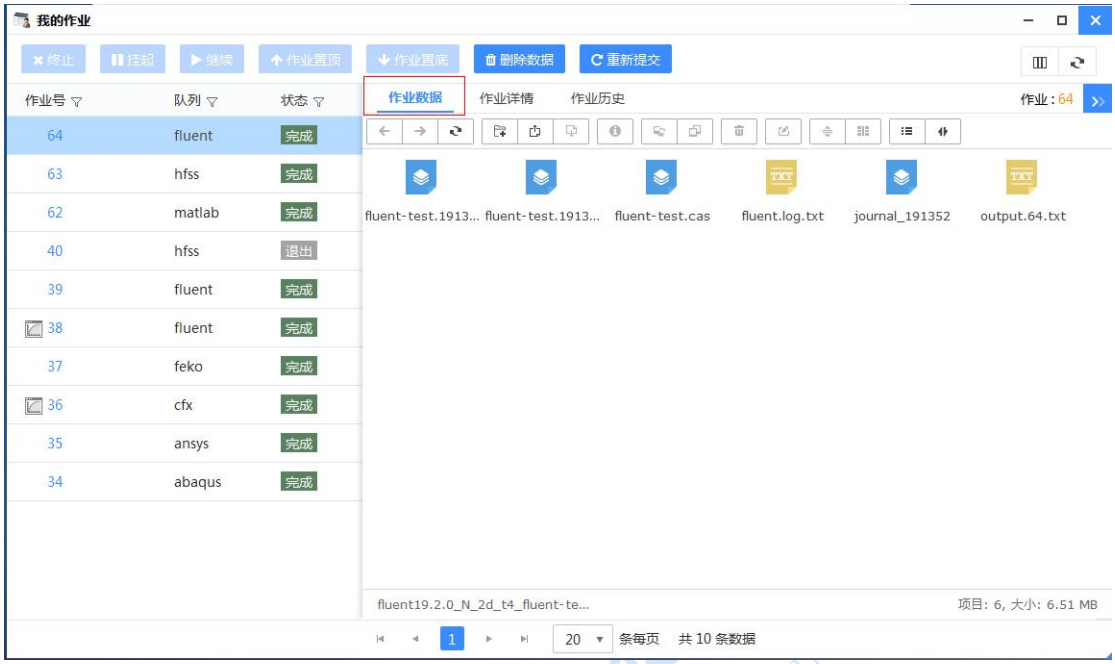



用户可以“查看作业输出”和“查看作业历史”。

点击“查看作业历史”。



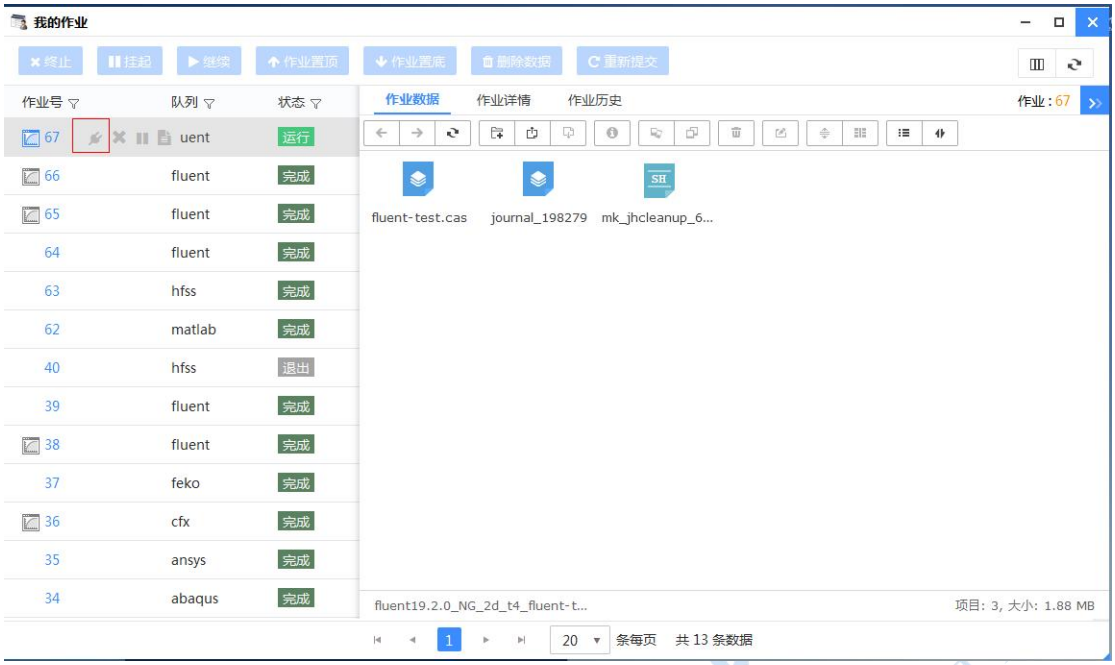
作业完成之后我们可以对作业进行后处理操作，点击作业数据按钮进入作业的数据列表页面。



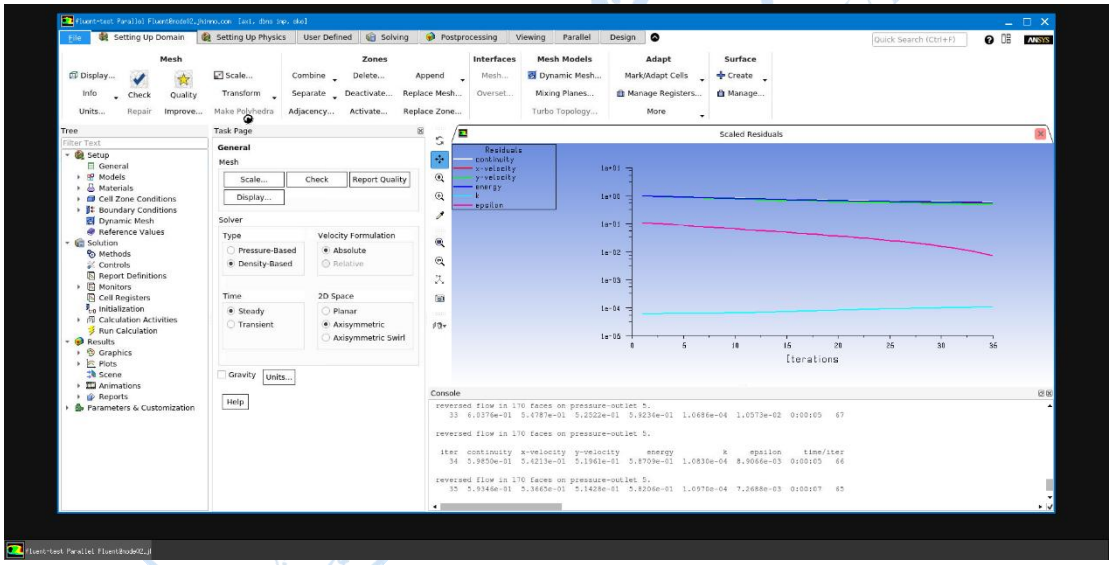

Fluent 可以实时查看作业收敛条件，如果需要在提交 Fluent 的同时查看收敛，需勾选下图的图形界面支持。




提交成功后，点击作业号右侧的打开应用图形按钮，就会自动跳转到 Fluent 计算的 Web 界面。




图形按钮




图形 web 页面

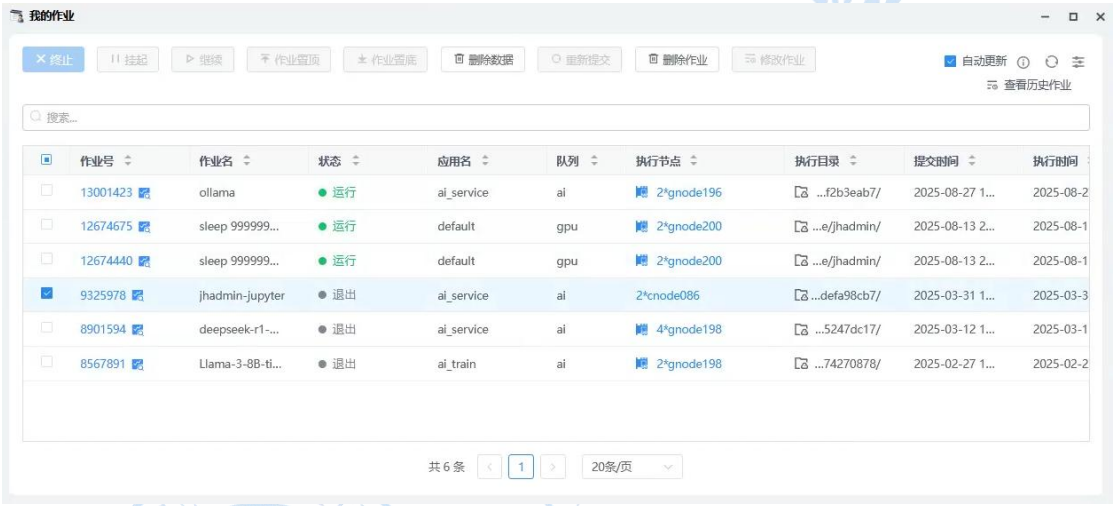
同时，我们在作业详细页面中可以看到：




## 5.2.2 查看作业信息

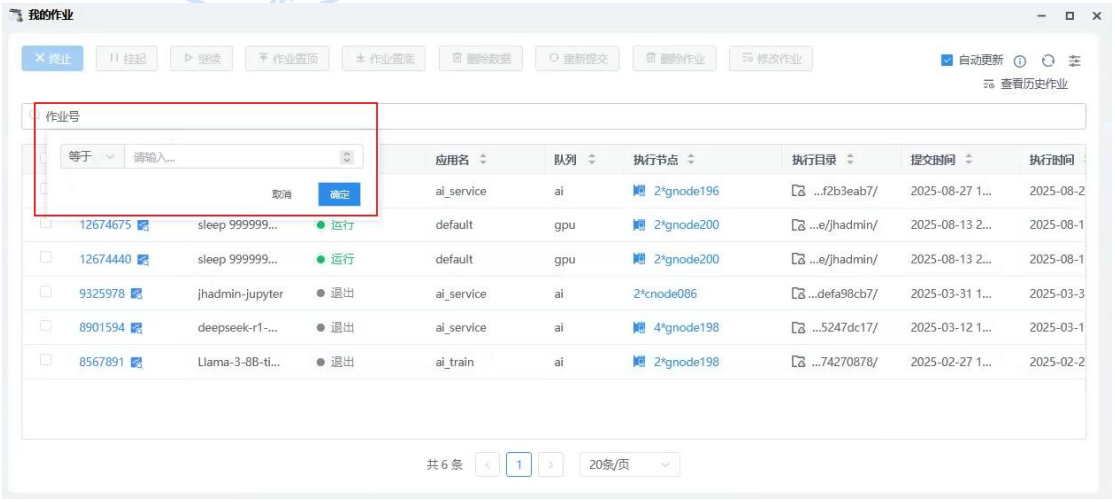
用户提交作业后，可在门户的作业管理界面查看当前及历史作业的详细信息，包括作业号、作业名称、队列、状态、提交时间、运行时长、执行节点等，并支持对作业进行挂起、恢复和终止等操作。为方便用户快速定位目标作业，门户还提供按作业号、状态、队列、提交时间、执行节点、作业名等多种维度的排序与筛选功能。

作业管理页面如图所示：

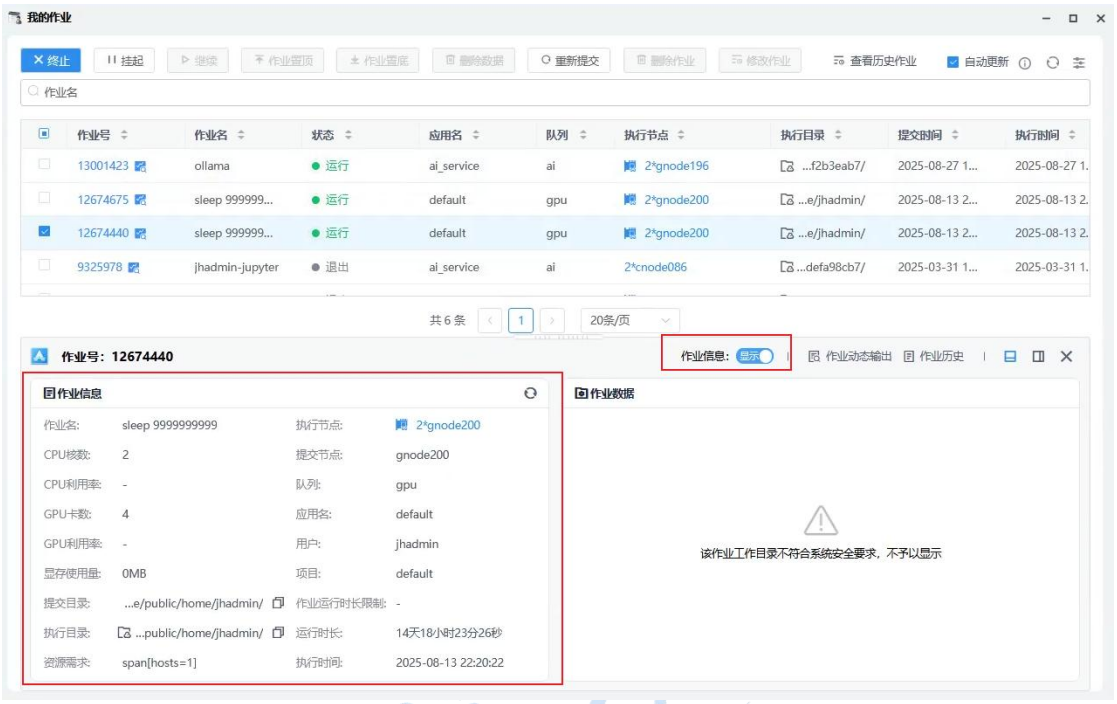



我的作业页面

作业管理页面对每一列提供了搜索，并支持个性化手动设置搜索条件。用户可以根据不同选项进行搜索。

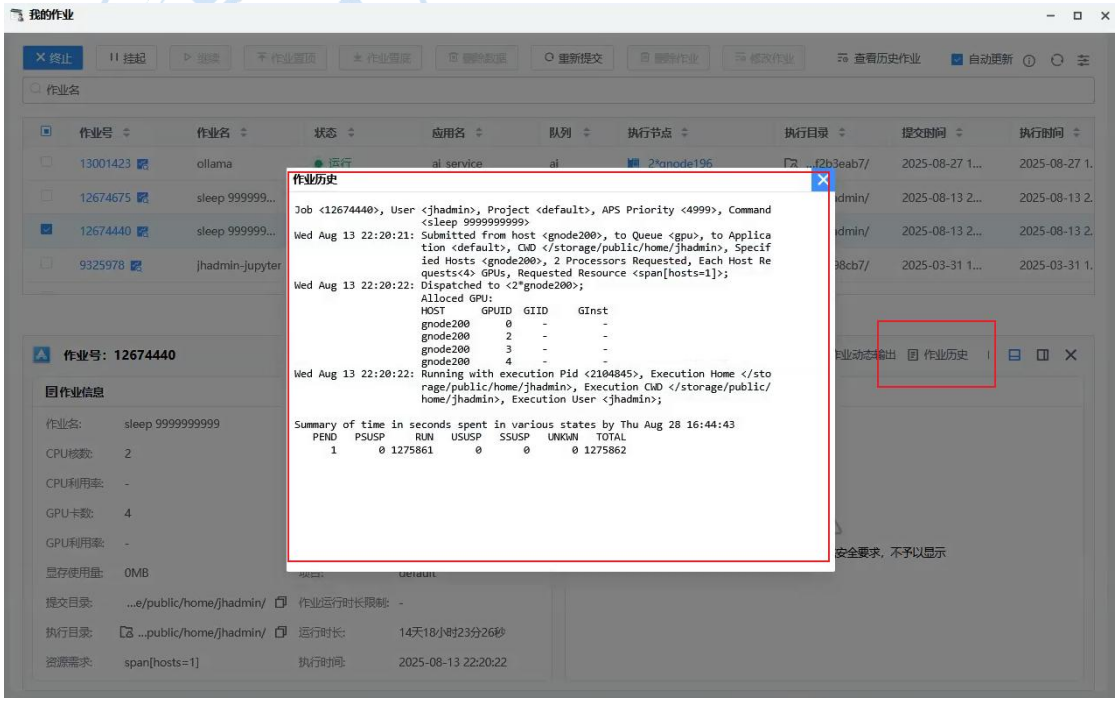


作业号查询框（我的作业页面）




作业信息页面

点击查看作业历史，可以在页面上直接看到作业的历史信息，如图所示：



作业历史信息

### 5.2.3 作业控制

用户可对作业执行挂起、继续、终止等控制操作。可在作业列表中批量选择作业，或在作业信息页面进行单作业操作。

| 操作 | 适用状态 | 操作后状态 | 说明 |
| --- | --- | --- | --- |
| 挂起 | PEND | PSUSP | 暂停正在排队或运行的作业 |
| | RUN | USUSP | |
| 继续 | PSUSP、 | PEND、 | 恢复被挂起的作业 |
| | USUSP | RUN | |
| 终止 | PEND、RUN、PSUSP、USUSP | EXIT | 结束作业 |

**状态说明**

| 状态 | 含义 |
| --- | --- |
| PEND | 作业正在排队等待资源 |
| RUN | 作业正在运行 |
| PSUSP | 排队中的作业被挂起 |
| USUSP | 运行中的作业被挂起 |
| EXIT | 作业已终止 |

# 第六章 软件应用

## 6.1 命令行方式

### 6.1.1 软件环境管理（Environment Modules）

Environment Module是一个用于管理用户软件环境的工具，可以动态加载或卸载软件模块(环境变量)，实现不同软件版本的灵活切换。

**常用命令**

| 命令 | 说明 | 示例 |
| --- | --- | --- |
| module avail | 查看系统中所有可用的软件模块 | module avail |
| module load <模块名> | 加载指定的软件模块 | module load R-4.3.0-gcc831 |
| module list | 查看当前已加载的软件模块 | module list |
| module rm <模块名> | 卸载指定的软件模块 | module rm R-4.3.0-gcc831 |
| module purge | 卸载当前所有已加载的软件模块 | module purge |

**示例**

```
# 1. 查看可用模块
module avail
```

```
# 2. 加载所需模块
module load R-4.3.0-gcc831
# 3. 确认加载成功
module list
# 4. 卸载模块（如需）
module rm R-4.3.0-gcc831
# 5. 卸载所有模块
module purge
```

**注意事项**：如果加载软件后运行时出现“xxx command not found”错误，通常是由于环境变量冲突导致。解决办法是先使用module purge命令清除冲突 的环境变量，然后再使用module load命令加载所需软件的环境变量。

**示例**

```
[jhtest@ln01 ~]$ module purge ; module load R-4.3.0-gcc831
```

### 6.1.2 软件安装

平台已预装了一些常用软件（详见软件资源列表）。如果平台没有用户所需的软件，可以自行安装在自己的家目录下。本节介绍两种常见的安装方式：编译安装和二进制包安装。

普通用户没有权限将软件安装在系统目录，因此需要将软件安装在自己的家目录下。

### 6.1.2.1 编译安装

编译安装是指从源代码开始构建软件，通常包含三个步骤：1、配置（./configure）；2、编译（make）；3、安装（make install）。

**关键说明：**

1.configure 文件是一个可执行的脚本文件，在待安装的源码目录下使用命令./configure --help 可查看所有配置选项。

2.--prefix 选项用于指定安装目录，必须配置，因为普通用户没有权限写入系统默认目录，所以必须指定安装目录在有权限写的目录。一般可在自己的家目录下新建一个目录存放编译安装的软件。

操作示例（以 gcc 为例）：

1.将 gcc 源码包上传到家目录，用 tar 命令解压

```
tar -xzf gcc-xxx.tar.gz
```

2.进入解压后的目录

```
cd gcc-xxx
```

3.配置安装目录（请将 <安装目录> 替换为实际路径）

```
./configure --prefix=/home/user/<安装目录>
```

4.编译（耗时较长）

```
make
```

5.安装

make install

提示：上述过程只是通用流程，具体软件的编译安装方法需参考软件发布的说明文档。

为确保下载的源码包完整无误，可使用官方提供的校验文件进行校验。下图为 cmake 的校验文件示例。

Download verification:

| Role | Files |
| --- | --- |
| Cryptographic Hashes | cmake-3.19.0-rc3-SHA-256.txt |
| PGP sig by EC8FEF3A7BFB4EDA | cmake-3.19.0-rc3-SHA-256.txt.asc |

### 6.1.2.2 二进制包安装

部分软件提供编译好的二进制包，无需编译即可使用。

**操作步骤:**

1.在软件发布页面选择与系统类型和版本匹配的二进制包

2.下载并上传到家目录

3.解压后即可使用

采用上述方法安装软件后，若无法直接输入命令运行，需要指定完整的路径。为了方便使用，需要将软件路径添加到环境变量中。

**配置环境变量:**

1.编辑用户配置文件 ~/.bashrc

vim ~/.bashrc

2.在文件末尾添加以下内容（/path/to 为用户安装软件目录）

```
# 添加可执行命令路径
export PATH=/path/to/bin:$PATH
# 添加动态链接库路径（如果软件包含动态库）
export LD_LIBRARY_PATH=/path/to/软件目录/lib:$LD_LIBRARY_PATH
```

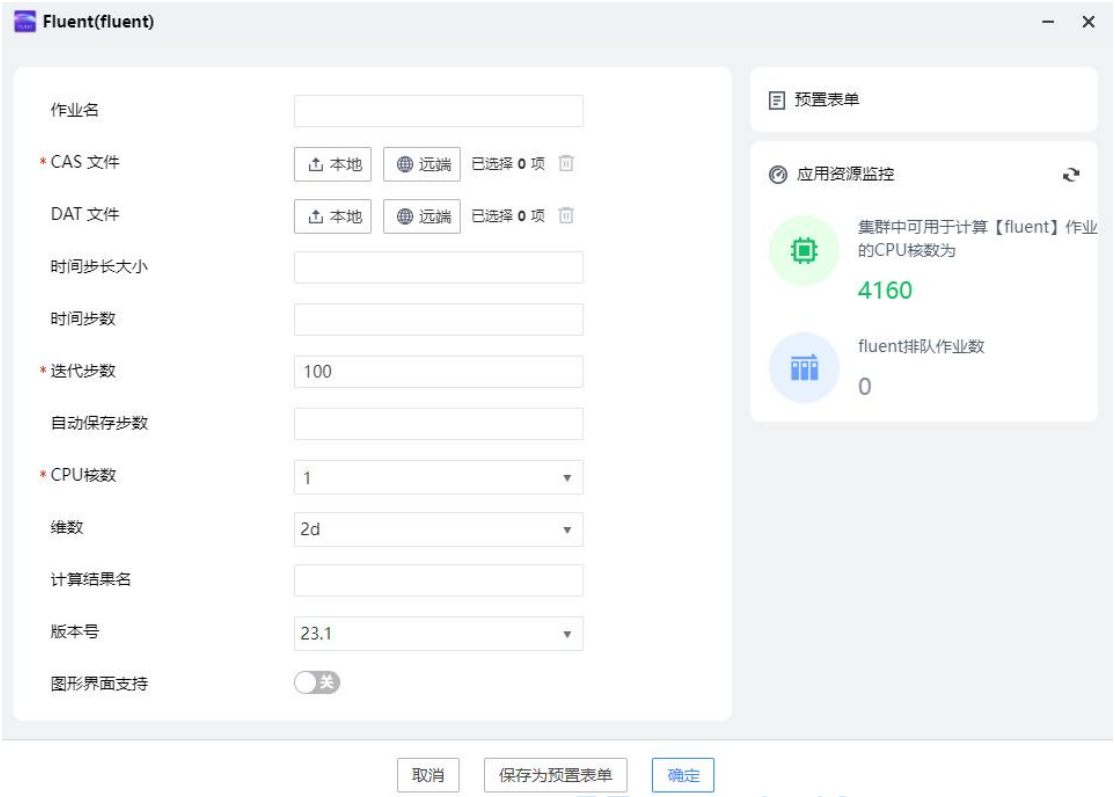
### 6.2 门户应用

门户中的应用可进入“应用仓库”查看。

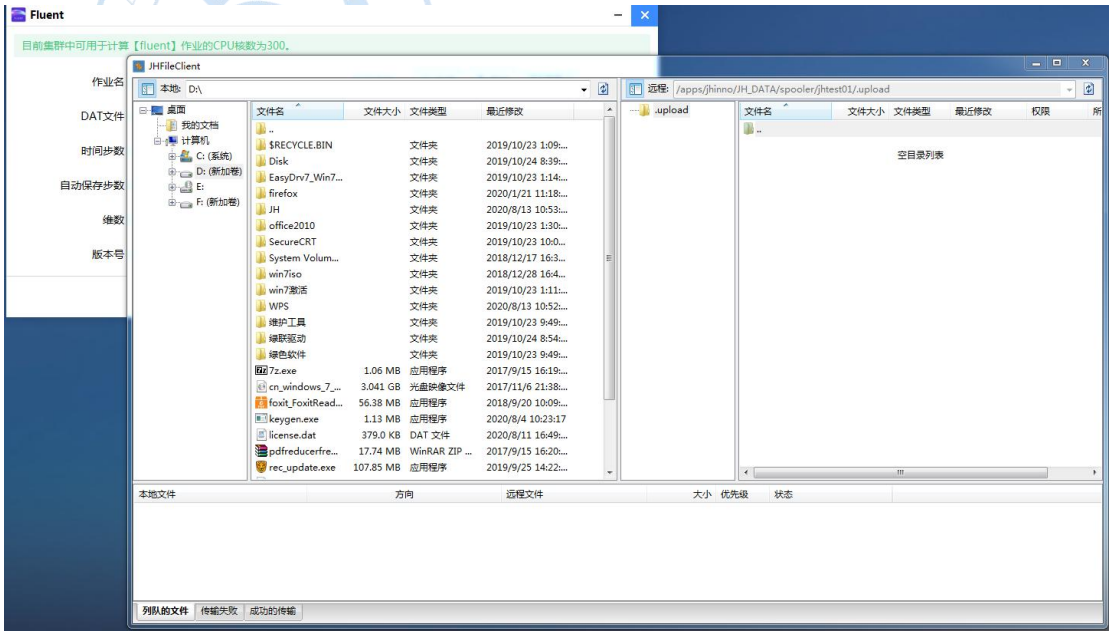
#### 6.2.1 Fluent 计算模式

登录门户后，点击应用仓库中 Fluent 图标，系统会弹出如下提交页面:

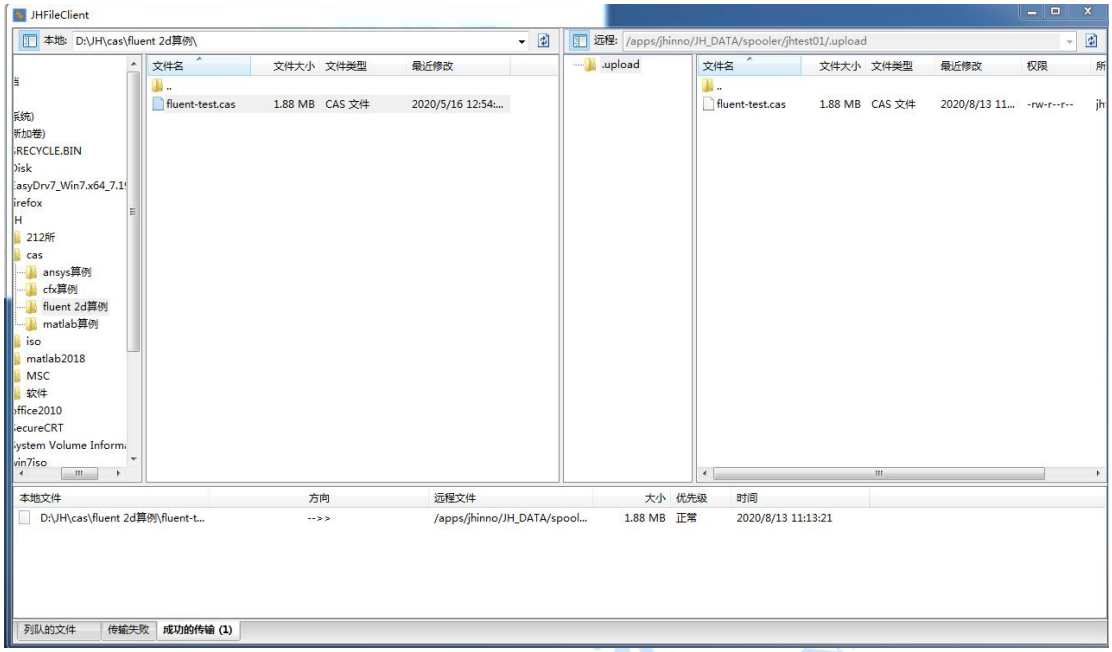




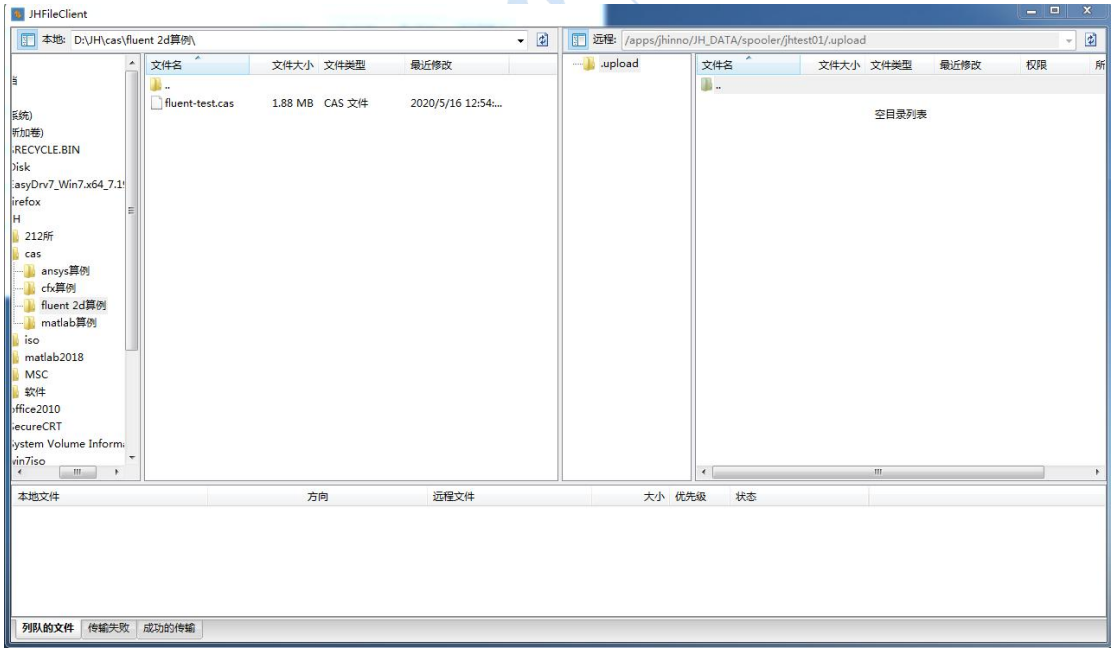

用户可以自主选择作业名称和选择 Fluent 的 CAS 算例文件。可以点击本地或服务器来选择文件。若点击“本地”，系统在安装客户端软件后会自动弹出 JHFileClient 对话框。



用户可以通过左侧的本地选择对应的目录，如选择【D:\JH\cas\fluent 2d 算例\】：

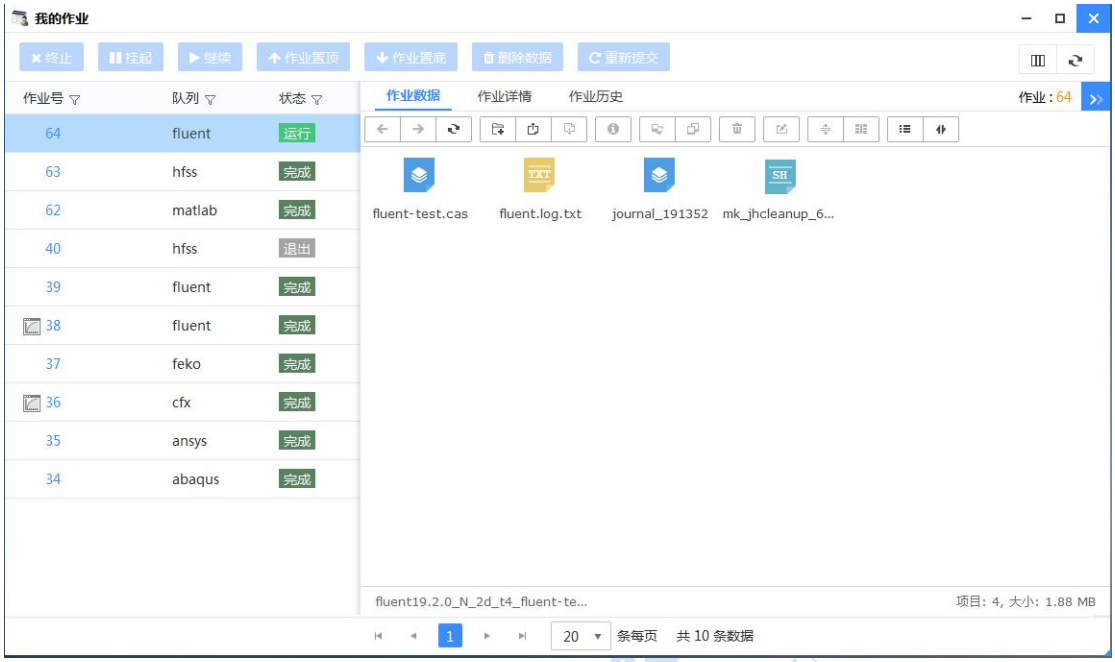



在选择好目录后，在要上传的文件处点击右键，选择上传，文件即开始上传到服务器。

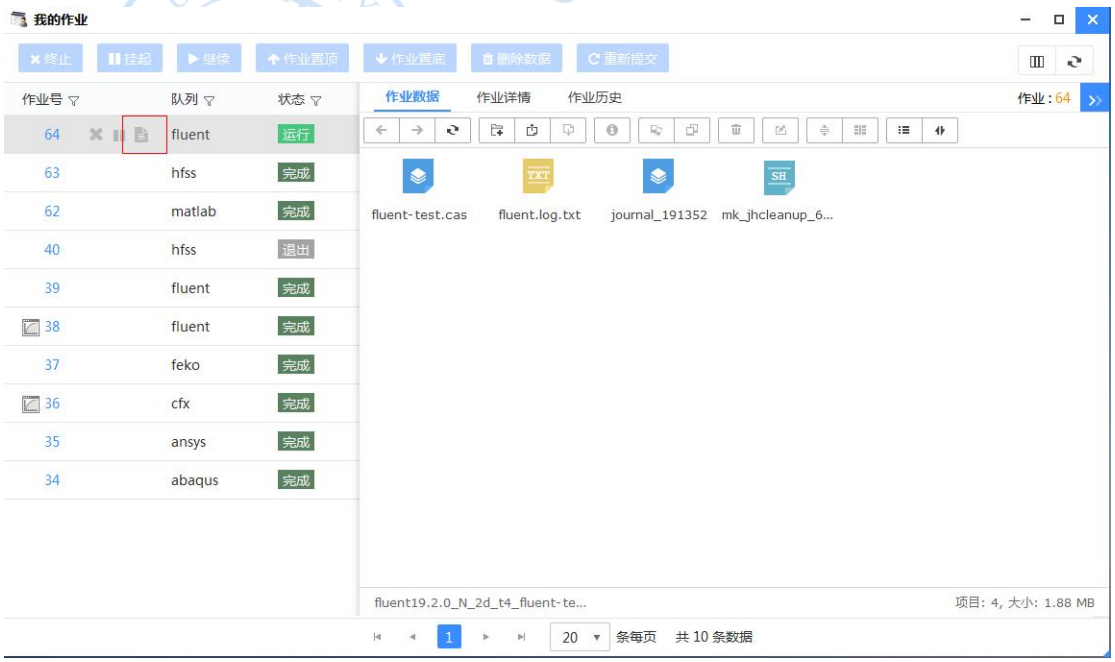



用户也可以点击“服务器端”来选择服务器端数据，进入服务器端后，选择需要计算的文件，点击“确定”。

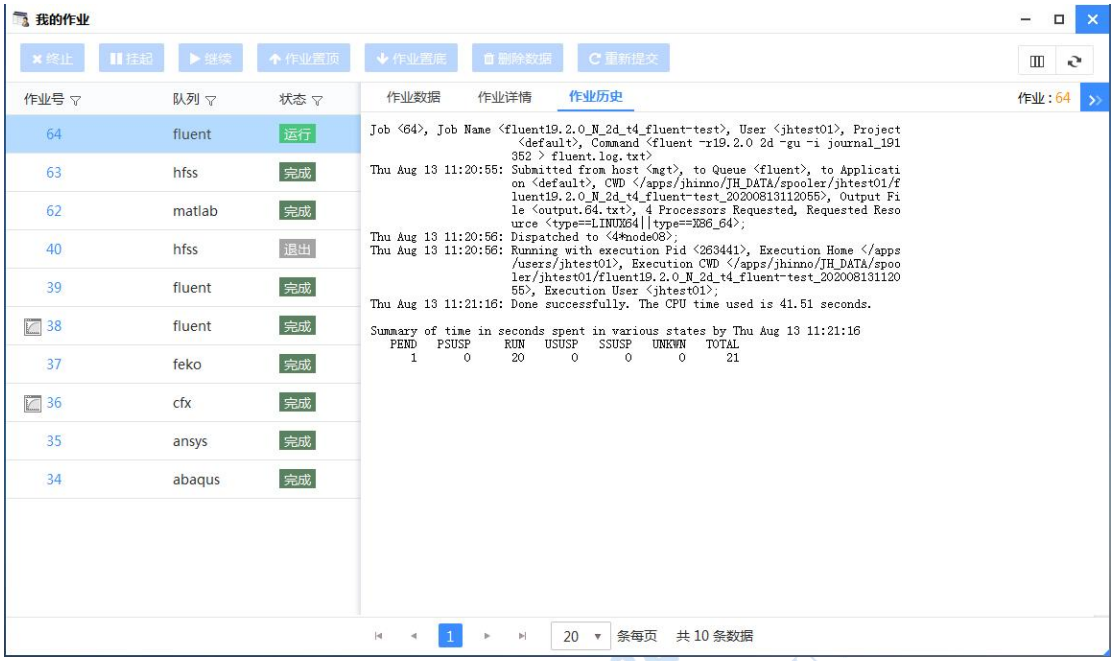
选择 CPU 核数，填写迭代步数，选择算例类型，点击确定提交作业。系统会自动跳转到作业详细页面，如下图所示。



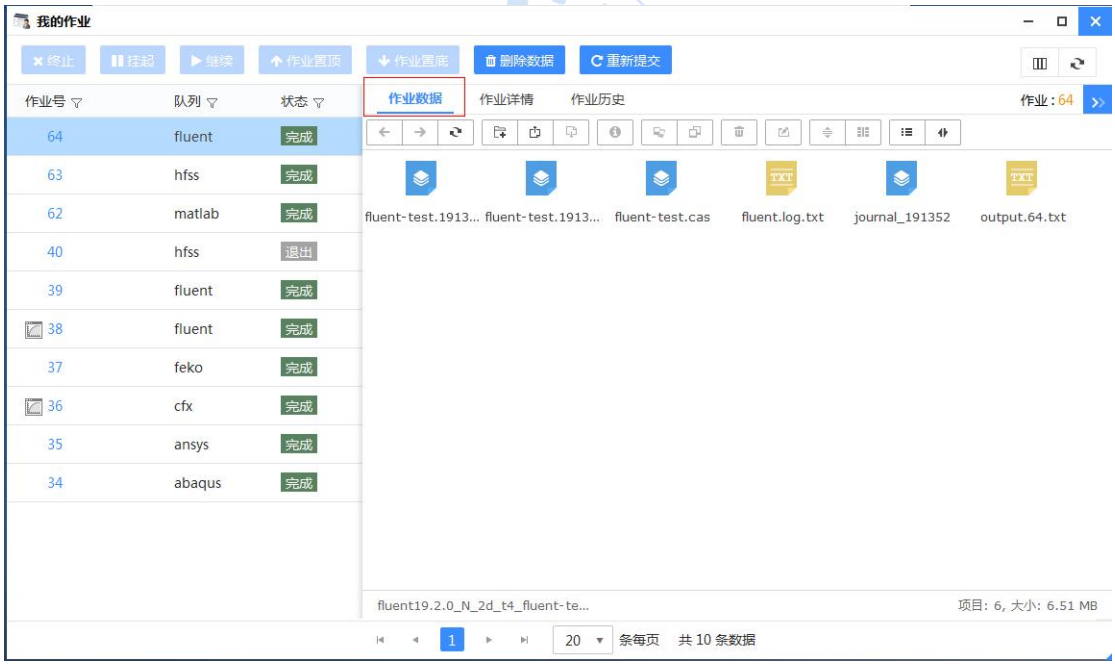

在这个页面中，我们可以点击“刷新”来刷新实时的数据情况和作业情况，同时也可以点击左侧面板中的“详细信息”进入该作业的详细页面。



用户可以“查看作业输出”和“查看作业历史”。点击“查看作业历史”。



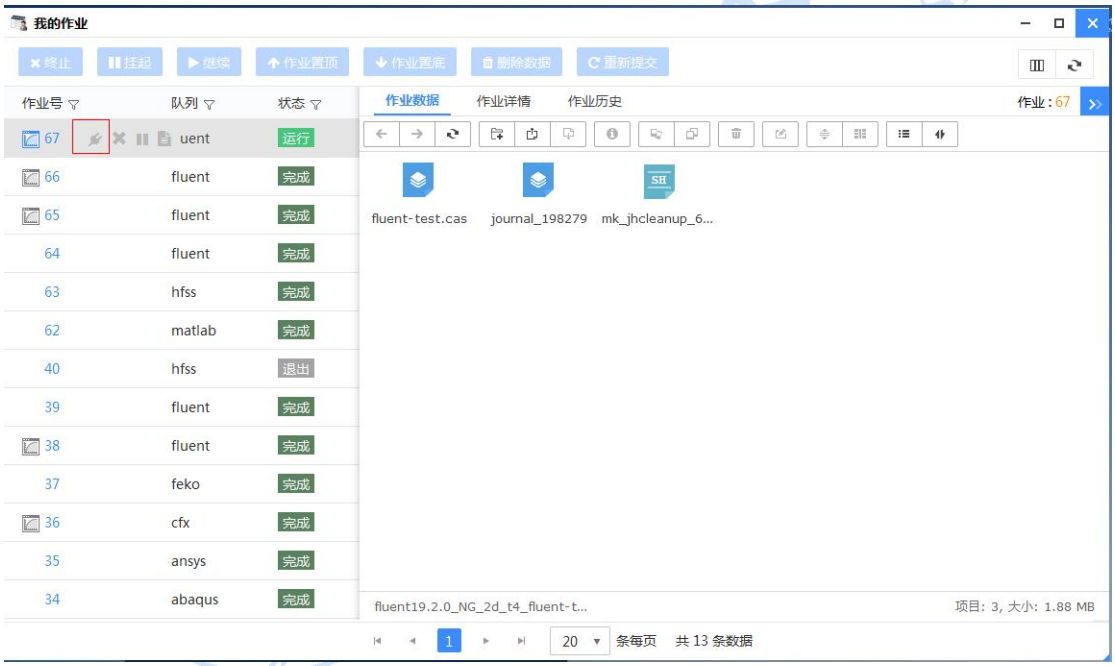

作业完成之后我们可以对作业进行后处理操作，点击作业数据按钮进入作业的数据列表页面。



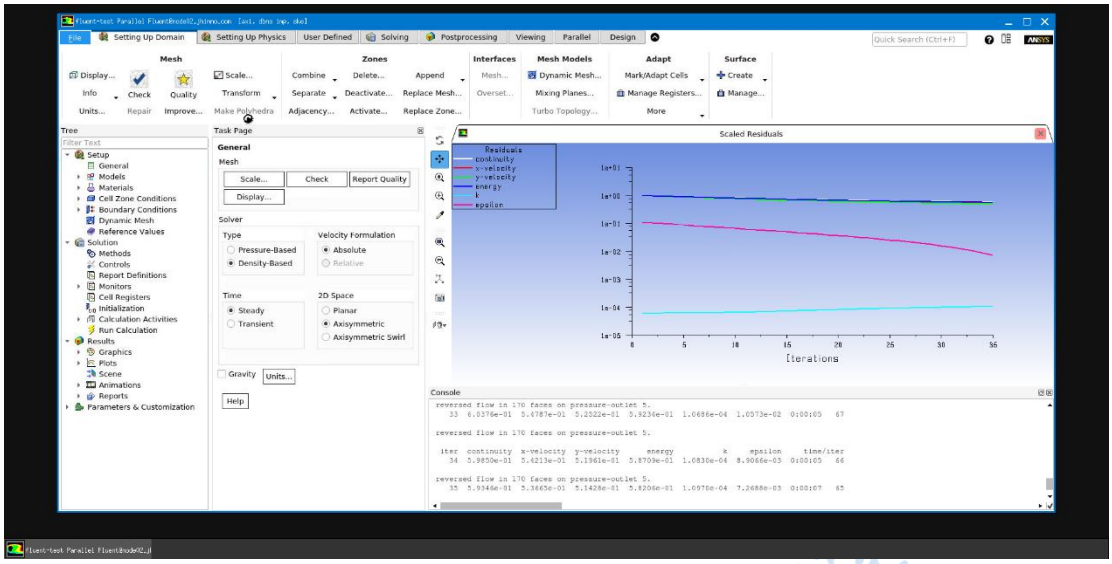

Fluent 可以实时查看作业收敛条件，如果需要在提交 Fluent 的同时查看收敛，需勾选下图的图形界面支持。




提交成功后，点击作业号右侧的打开应用图形按钮，就会自动跳转到 Fluent 计算的 Web 界面。




图形按钮



图形 web 页面

同时，我们在作业详细页面中可以看到如下页面。



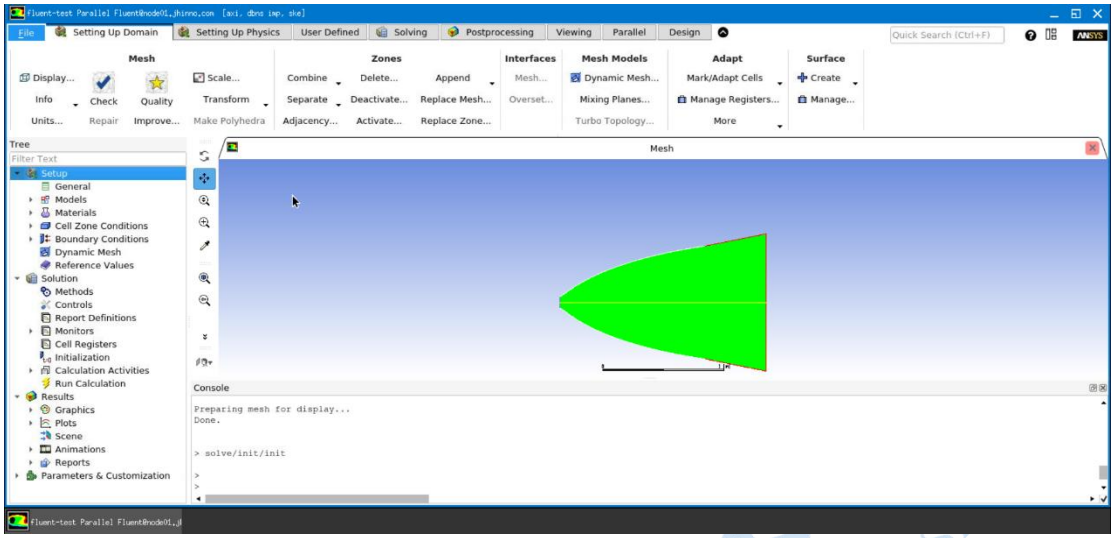

### 6.2.2 Fluent 交互模式

在应用仓库点击仿真计算的 Fluent 交互模式图标，点击之后会弹出 Fluent 交互模式的提交页面，如下

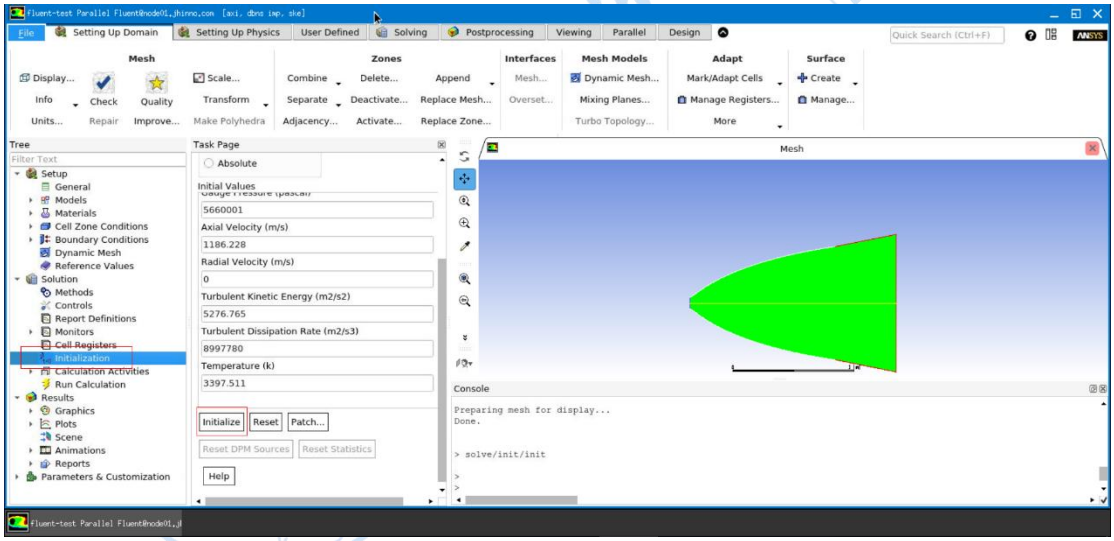



将图中各选项填写完成（其中带*的是必填项），点击确定提交作业，页面跳转到该作业的详细信息页面，在该

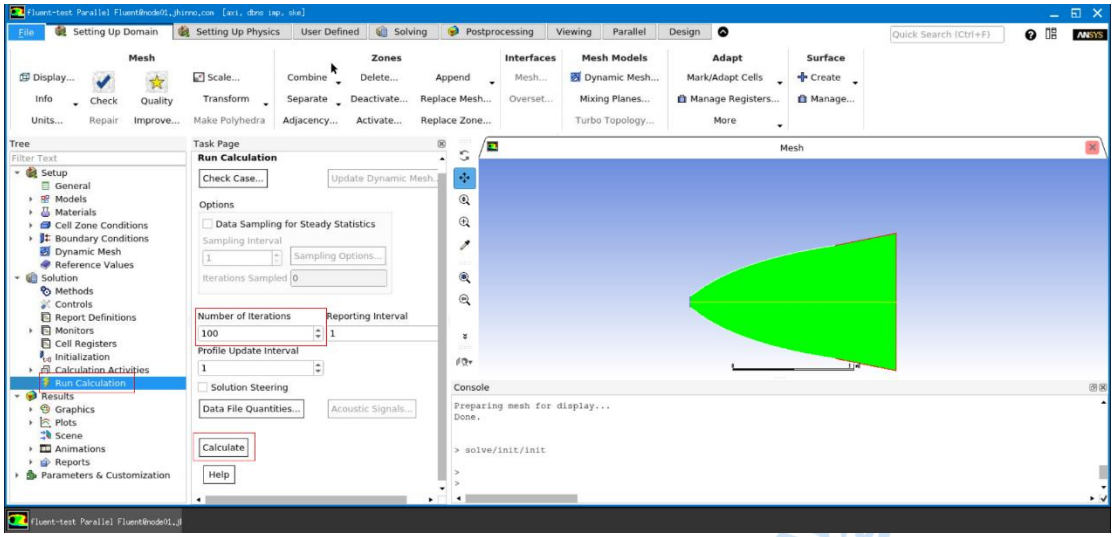
页面可以查看作业的详细信息。可以在作业信息页面打开图形界面，图形界面上可以显示 Fluent 作业的运行过程。

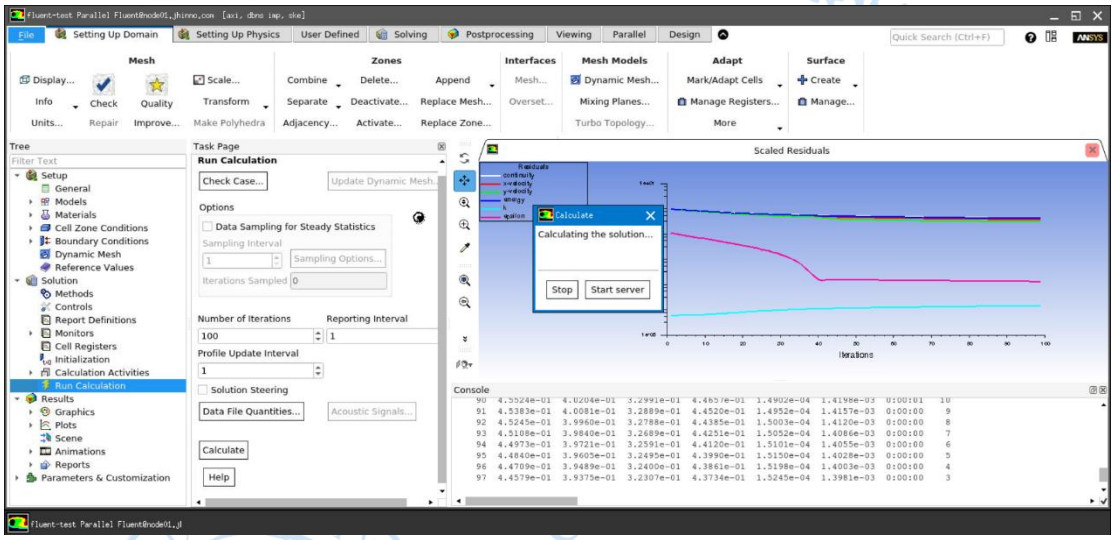


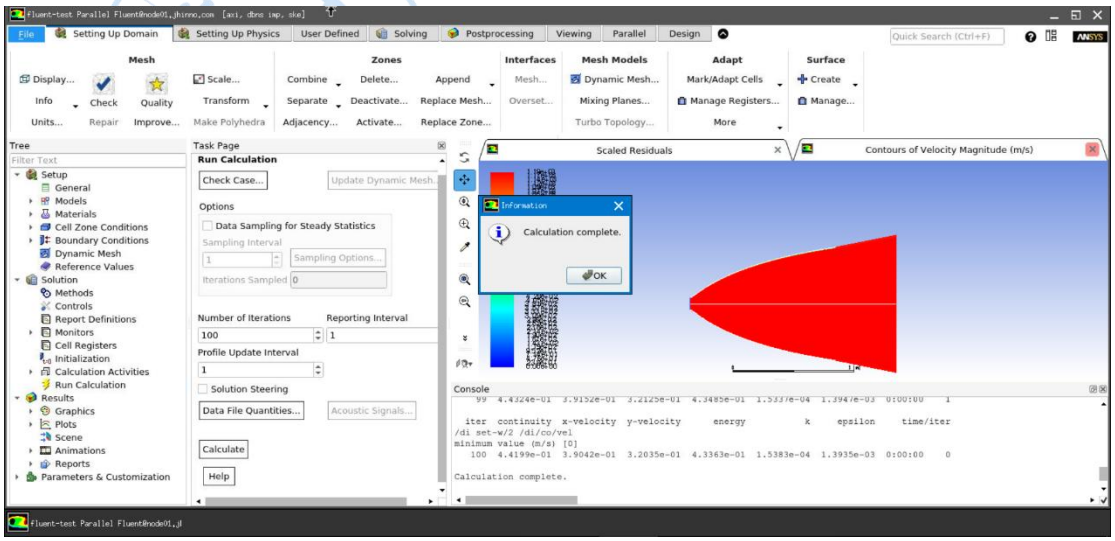

选择左列 Initialization，点击初始化。




设置好迭代步数后点击运行计算。



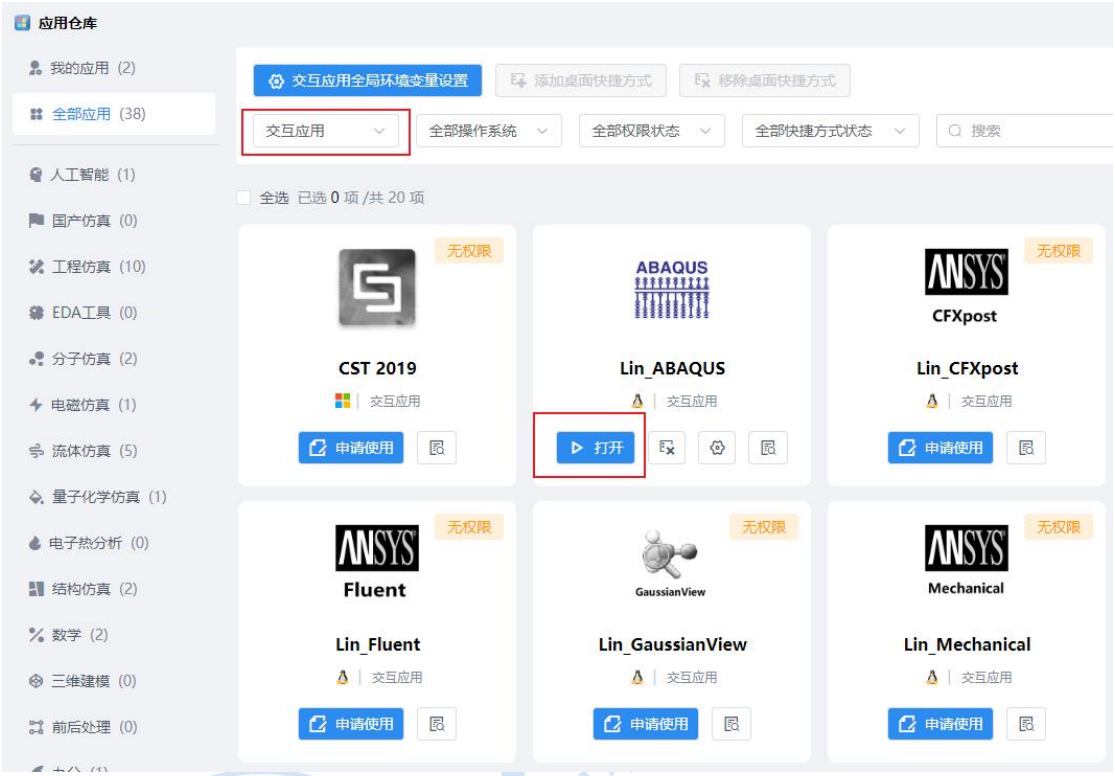




完成计算，点击右上角关闭按钮退出。

### 6.2.3 仿真设计应用

登录门户后，可在应用仓库中筛选出交互应用进行仿真设计。




点击软件图标下方的打开按钮，稍等片刻即可显示软件的程序界面，该界面也被称作会话。更多会话使用和管理请参考7.7“我的会话”小节。

### 6.2.4 Matlab

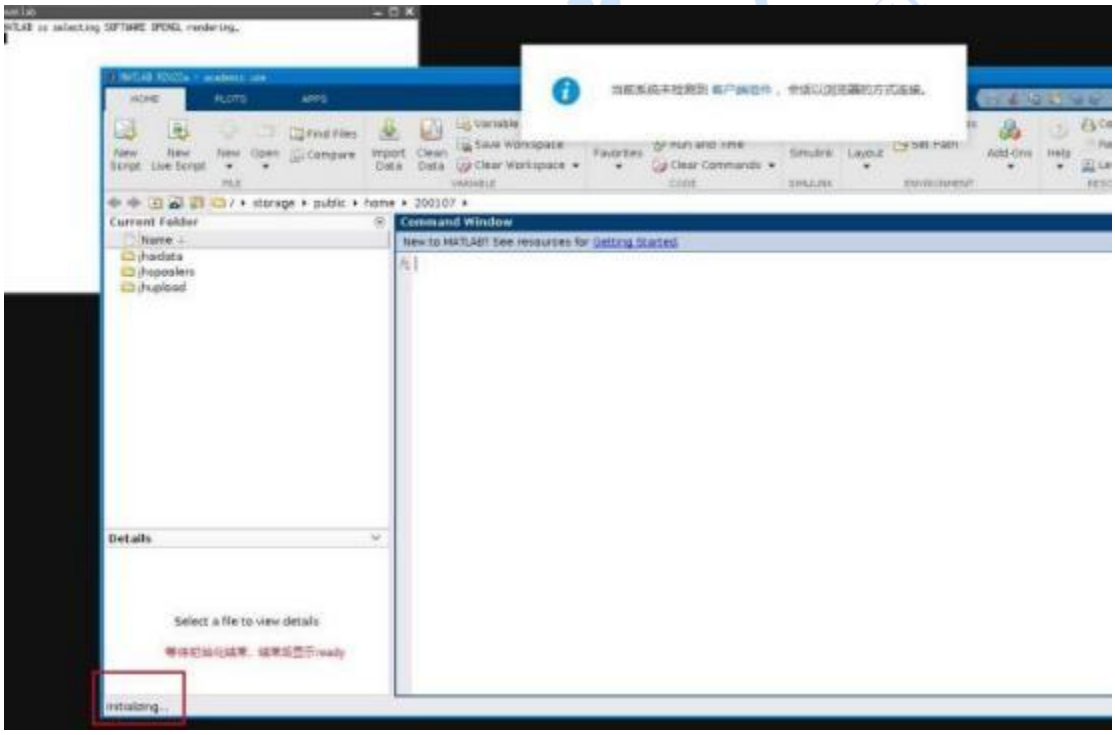
平台的 Matlab软件 （linux 版）已完成正版化部署，用户可以正常使用。首次使用需要注意：每个账号第一次使用前，需要完成Matlab集群配置设置。

配置步骤如下：

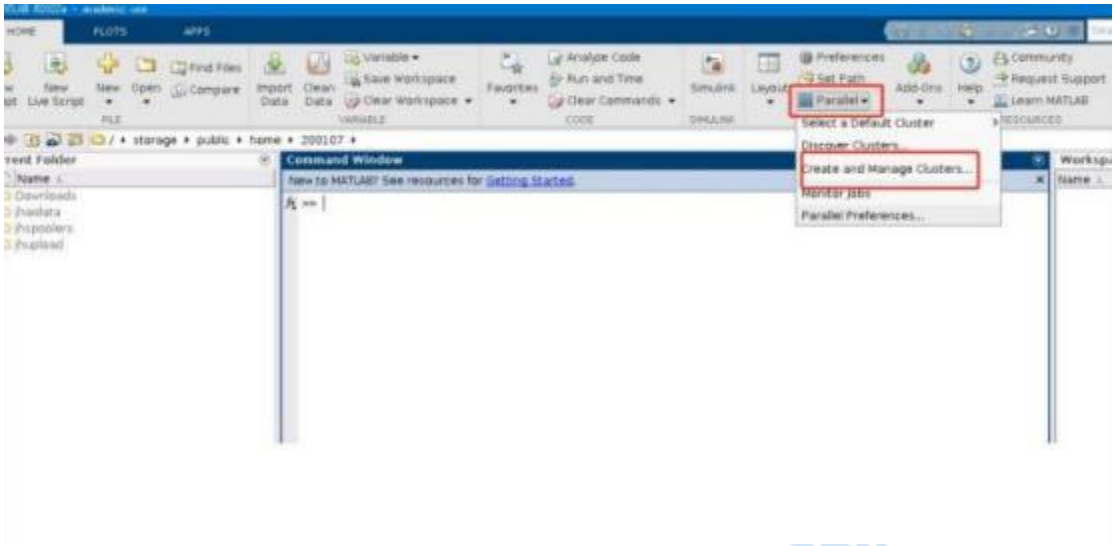
1.打开平台Matlab交互模式。



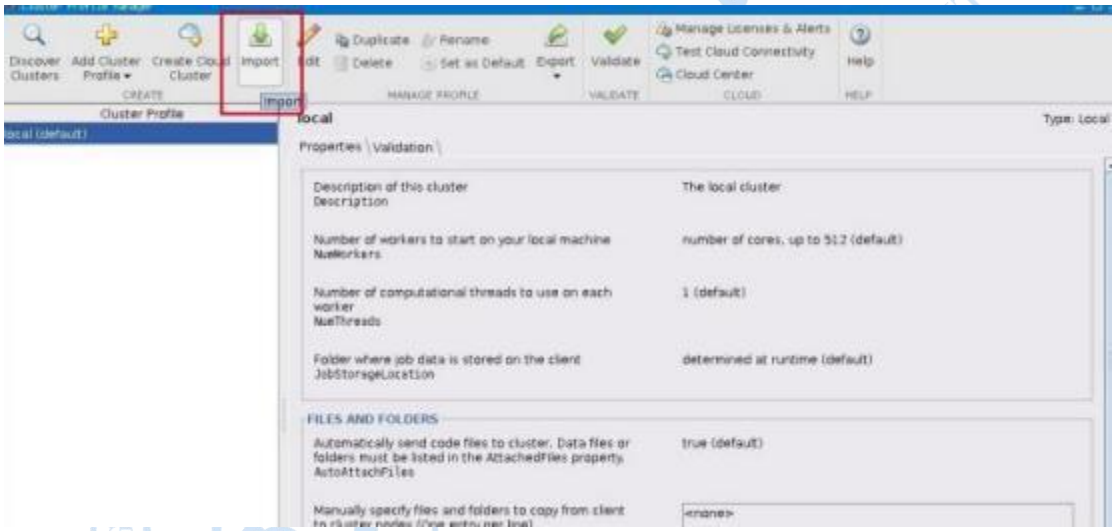

2.等待Matlab完成初始化。



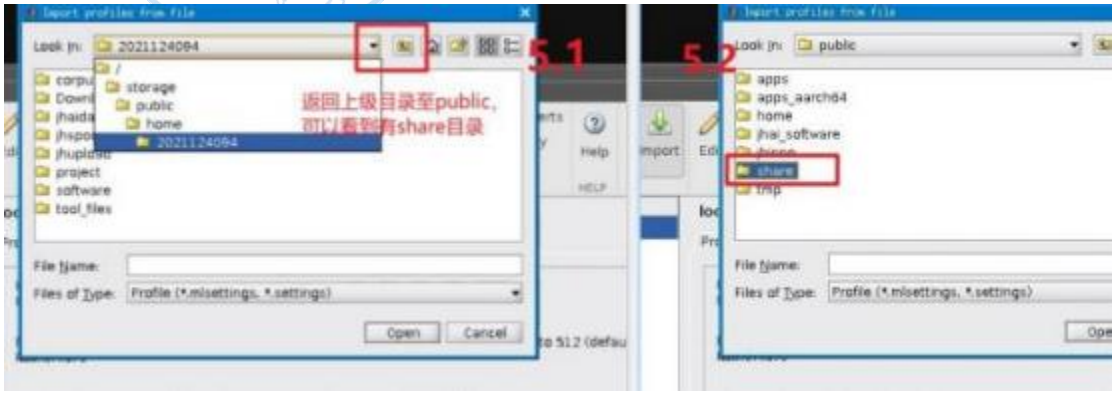

3.依次选择 “Parallel” -“Create and Manage Clusters...”。



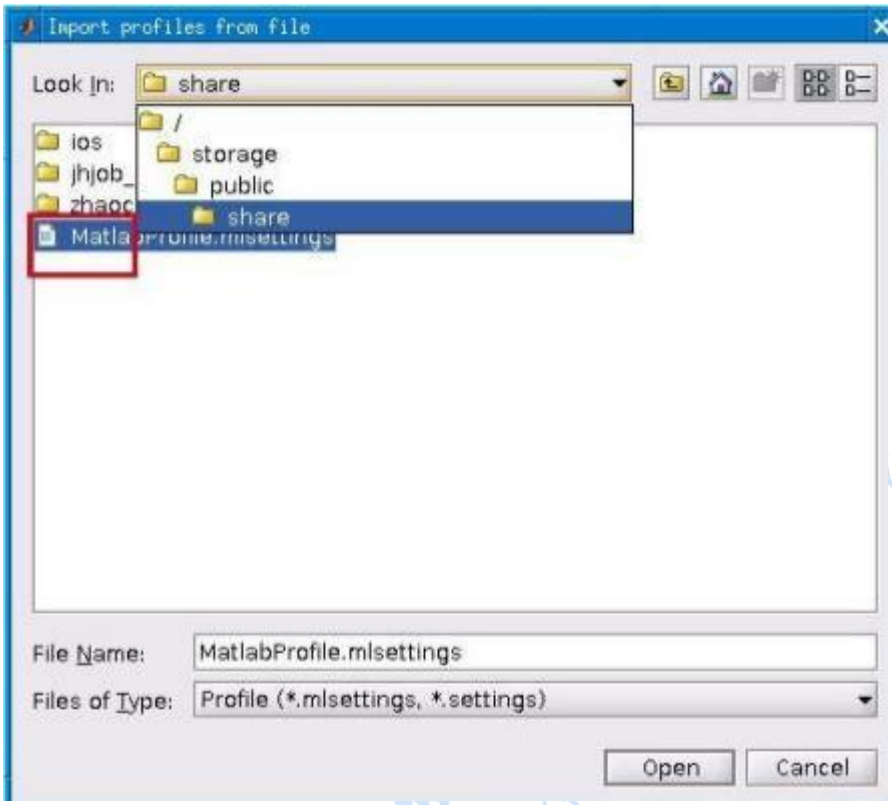

4.待页面加载完成后，单击import按钮。



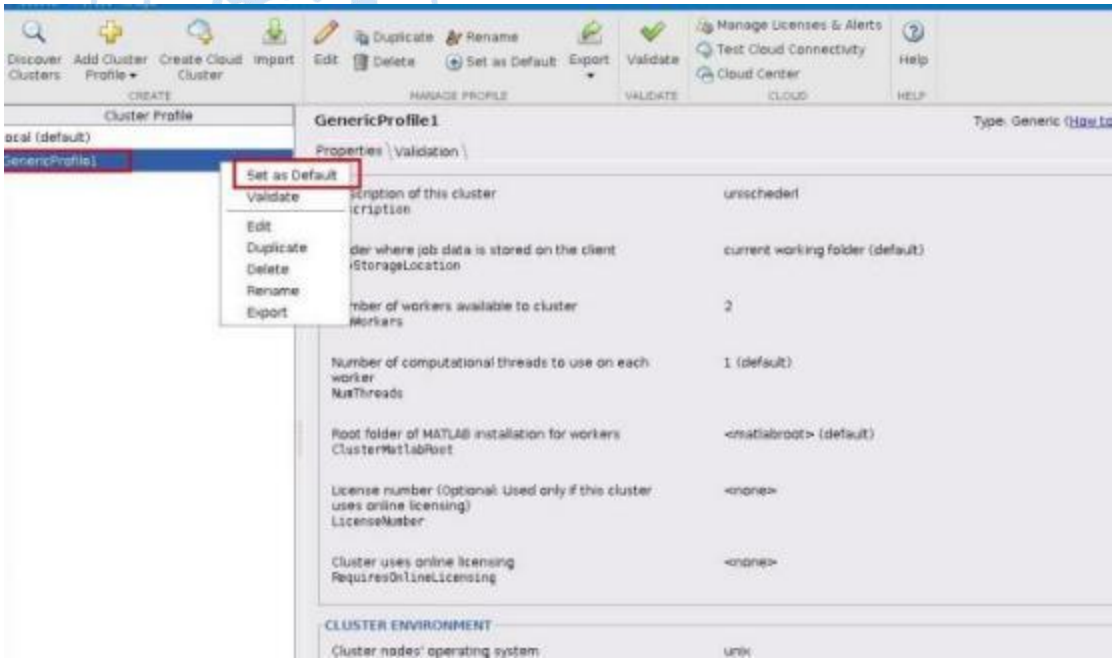

5.返回上级目录至public目录下，可以看到有share目录。



6.打开 share目录，选中文件 MatlabProfile.mlsettings 并将其打开。



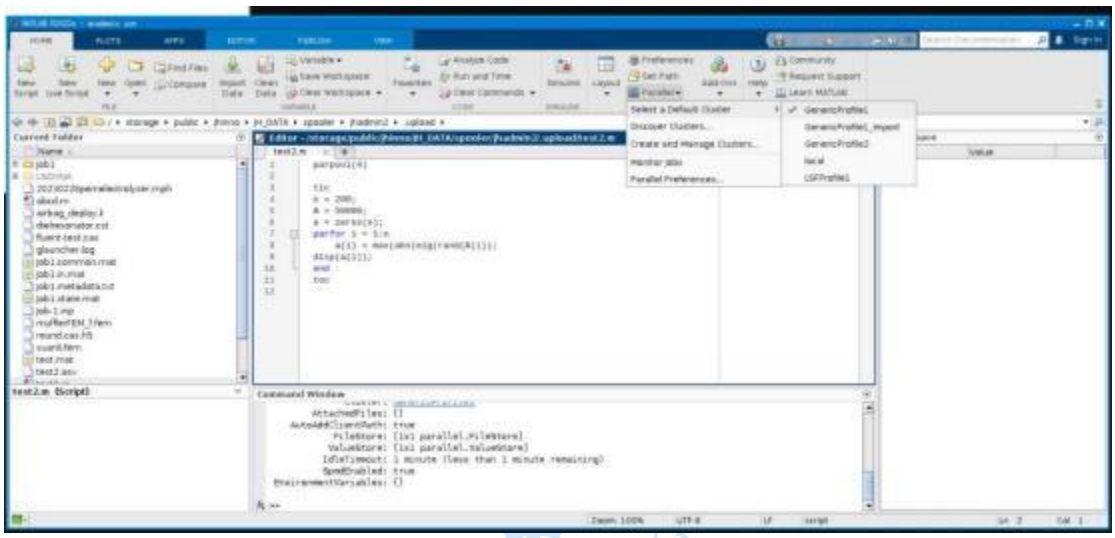

7.选中GenericProfile1，右键单击选择“Set as Default”即可。



上述设置步骤设置完成后，关闭此页面，Matlab 就可以正常使用了。配置完成后Matlab计算任务会自动分配到计算节点进行计算。

**作业示例:**

测试算例使用了 4核。



提交完成后可以在我的作业中看到提交的作业信息。



### 6.2.5 Materials Studio

#### 1.申请 ms 业务账号

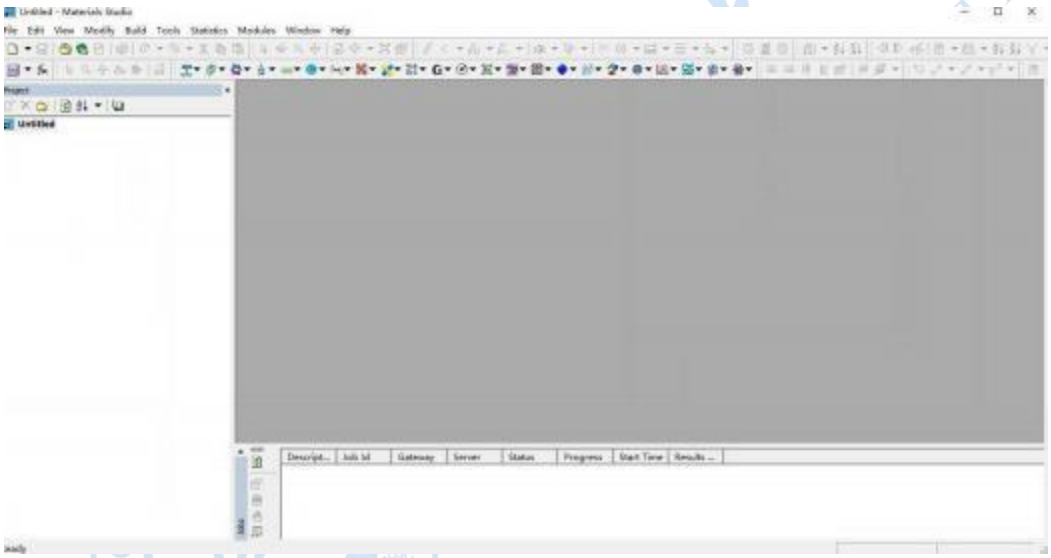
向管理员申请 ms 业务账号，用于登录提交作业。

#### 2.网关初始配置

①打开 MS windows 图形会话（需要向管理员申请审批）。首次打开时，所有选项按默认值确认即可。

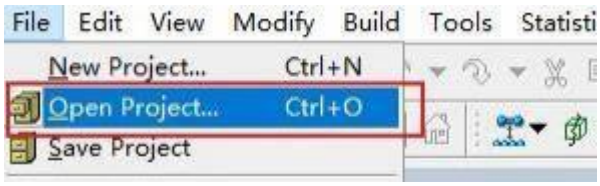


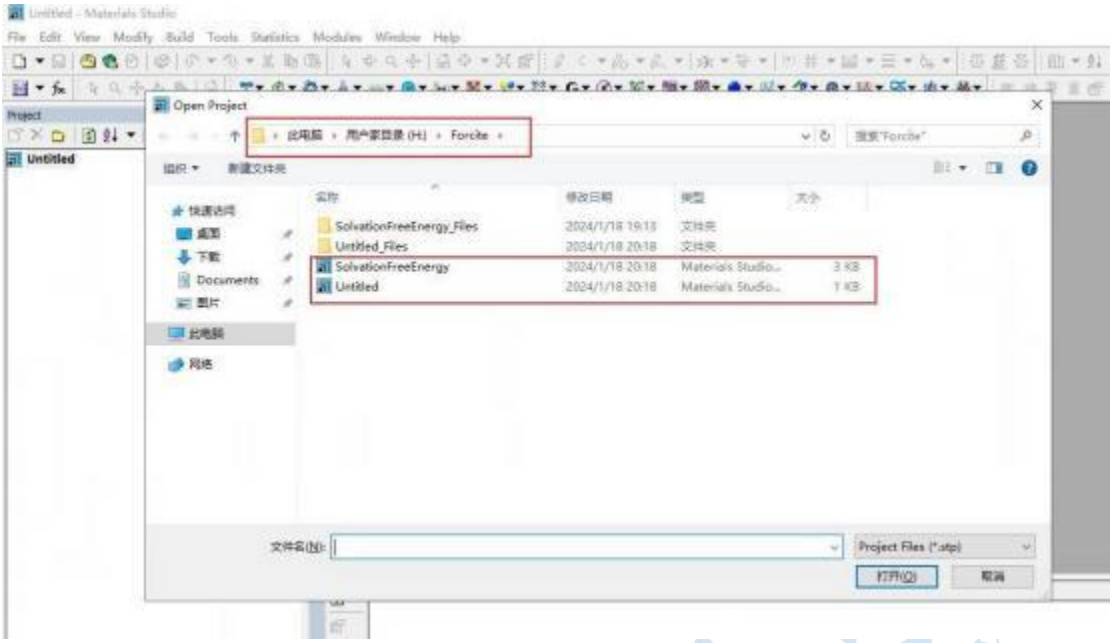

打开后软件按照默认选项确定后显示首页界面如下。

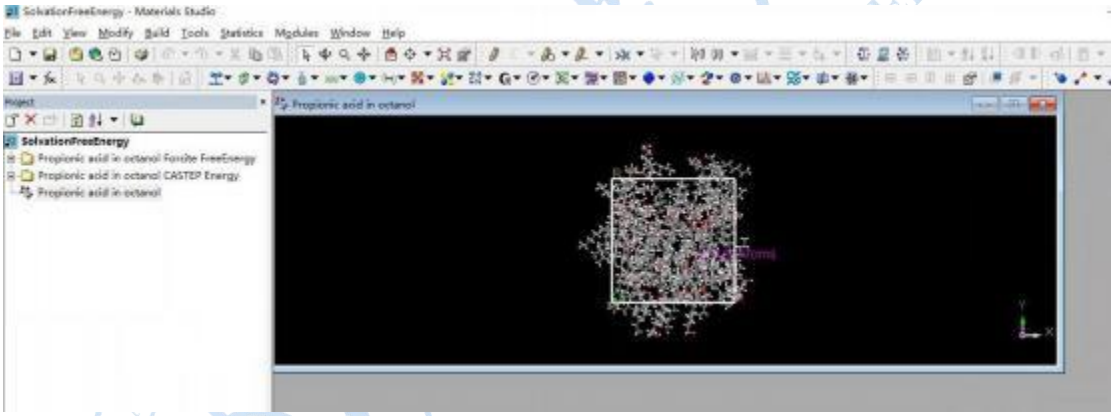


②选中计算作业

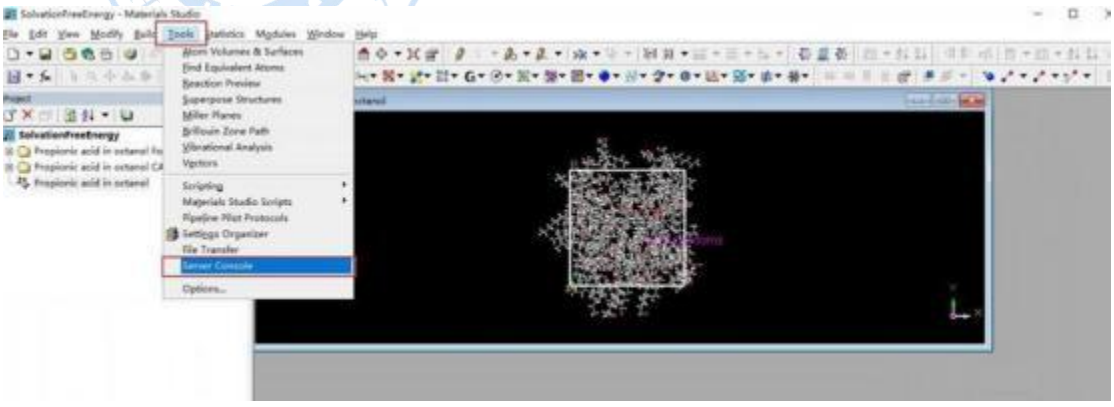
选择 Open project 打开工程，选择家目录下所需要计算的工程（以 SolvationFreeEnergy 为例）。



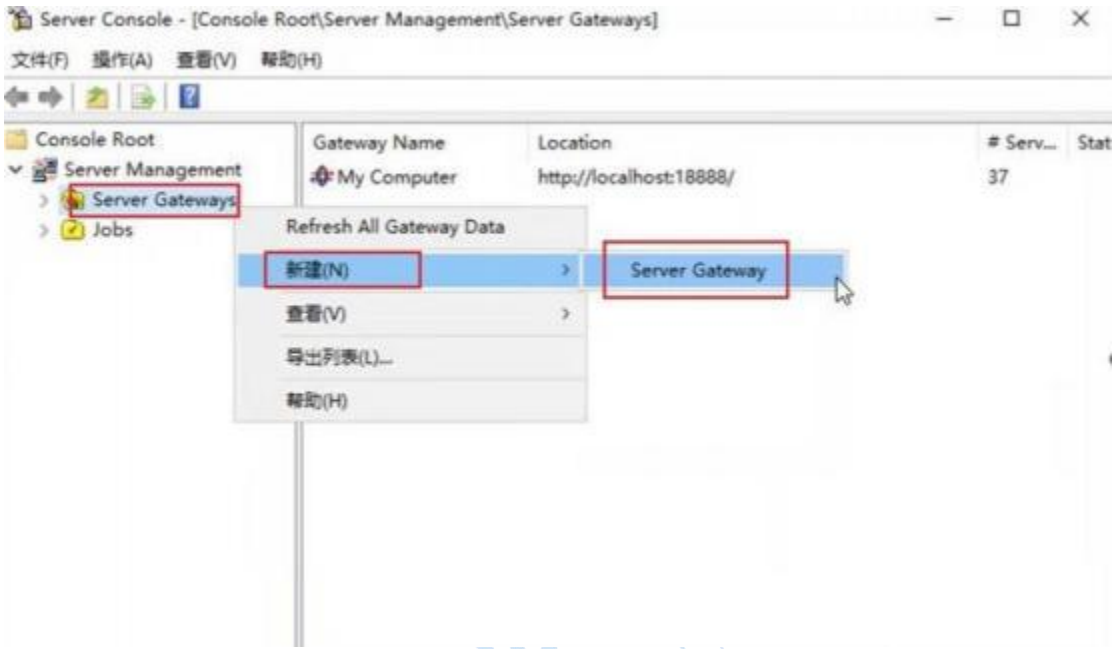


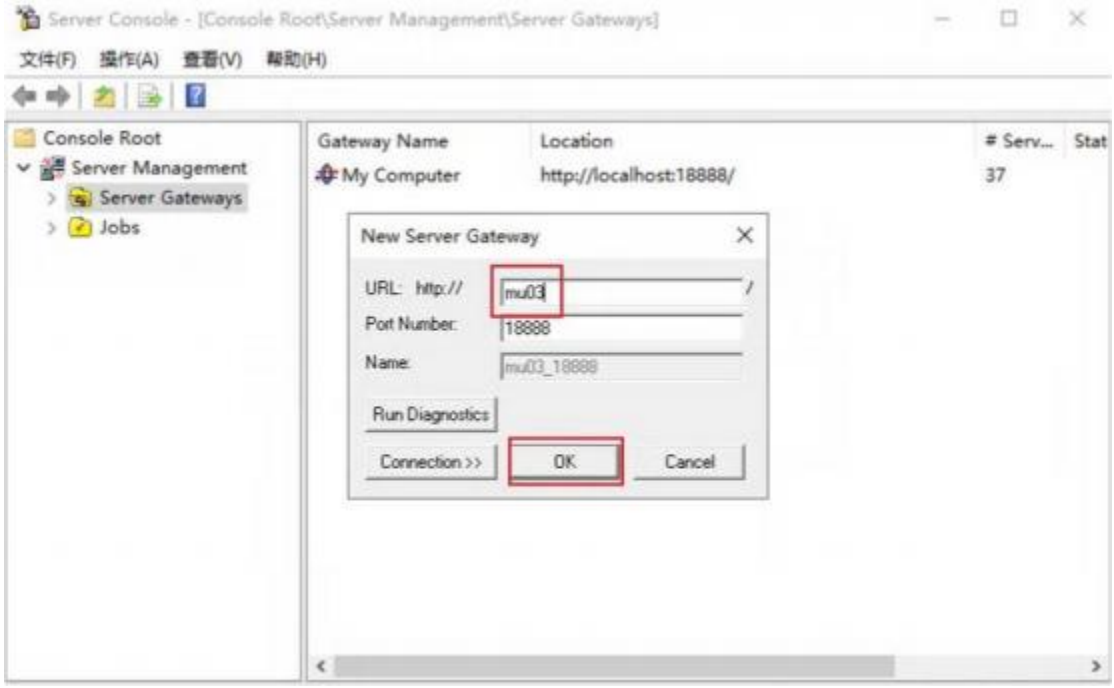
③配置ms gateway（一般情况下配置一次即可）



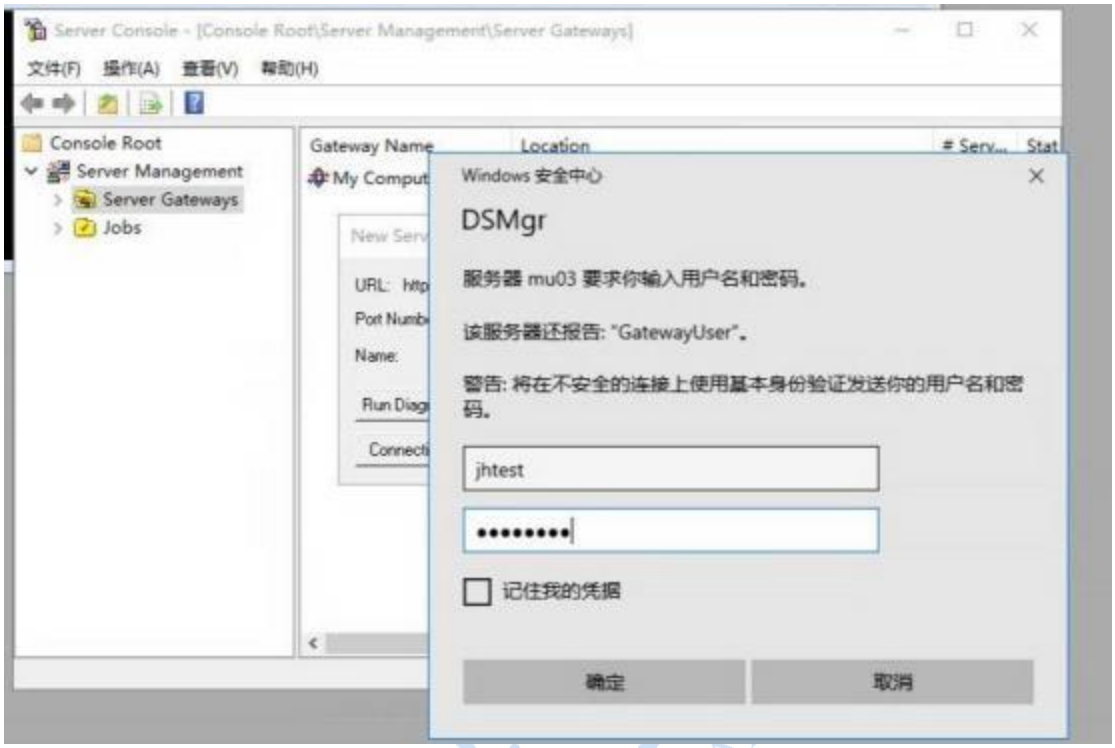

Choose Workspace Folder
Materials Studio applications require a folder to store short-lived data files that do not belong to any specific project.
OK
Please choose a suitable location for this folder:
Cancel
C:\Users\jhinternal_user_005\AppData\Local\Temp\
Browse...
Help


Server Console - [Console Root\Server Management\Server Gateways]
文件(F) 操作(A) 查看(V) 帮助(H)
Console Root
Server Management
Server Gateways
Jobs
Gateway Name
Location
# Serv...
Stat
My Computer
http://localhost:18888/
37
Refresh All Gateway Data
新建(N)
Server Gateway
查看(V)
导出列表(L)...
帮助(H)

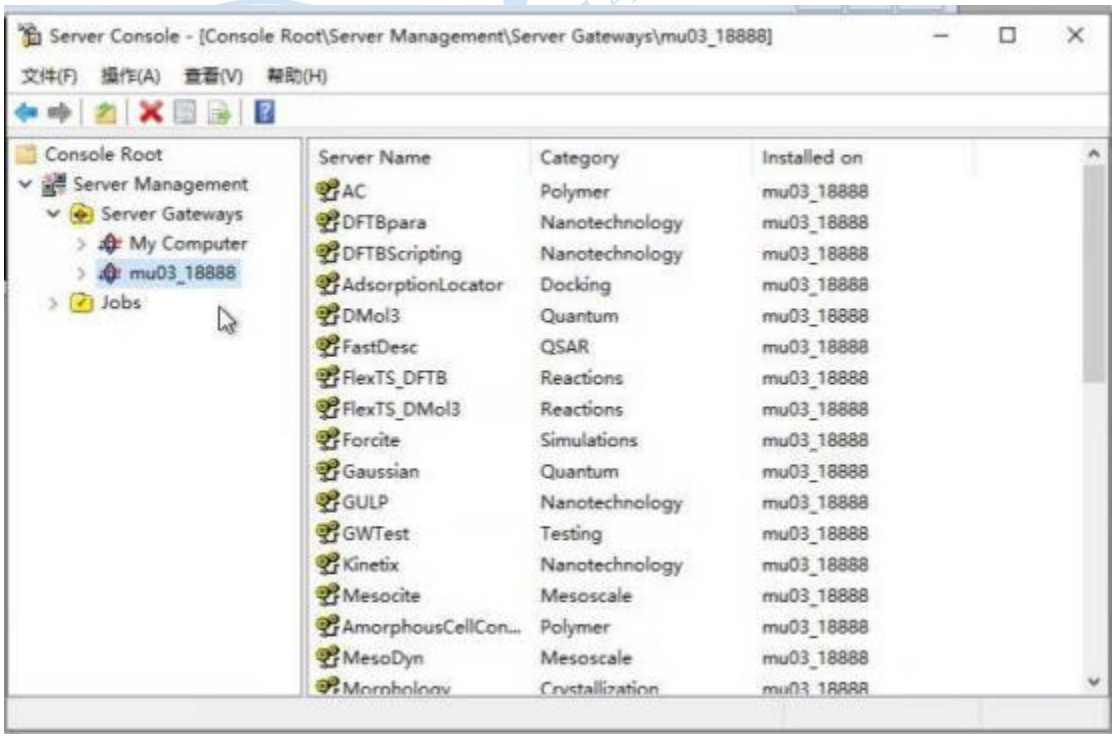
输入URL为mu03


Server Console - [Console Root\Server Management\Server Gateways]
文件(F) 操作(A) 查看(V) 帮助(H)
Console Root
Server Management
Server Gateways
Jobs
Gateway Name
Location
# Serv...
Stat
My Computer
http://localhost:18888/
37
New Server Gateway
URL: http://
mu03
/
Port Number:
18888
Name:
mu03_18888
Run Diagnostics
Connection >>
OK
Cancel

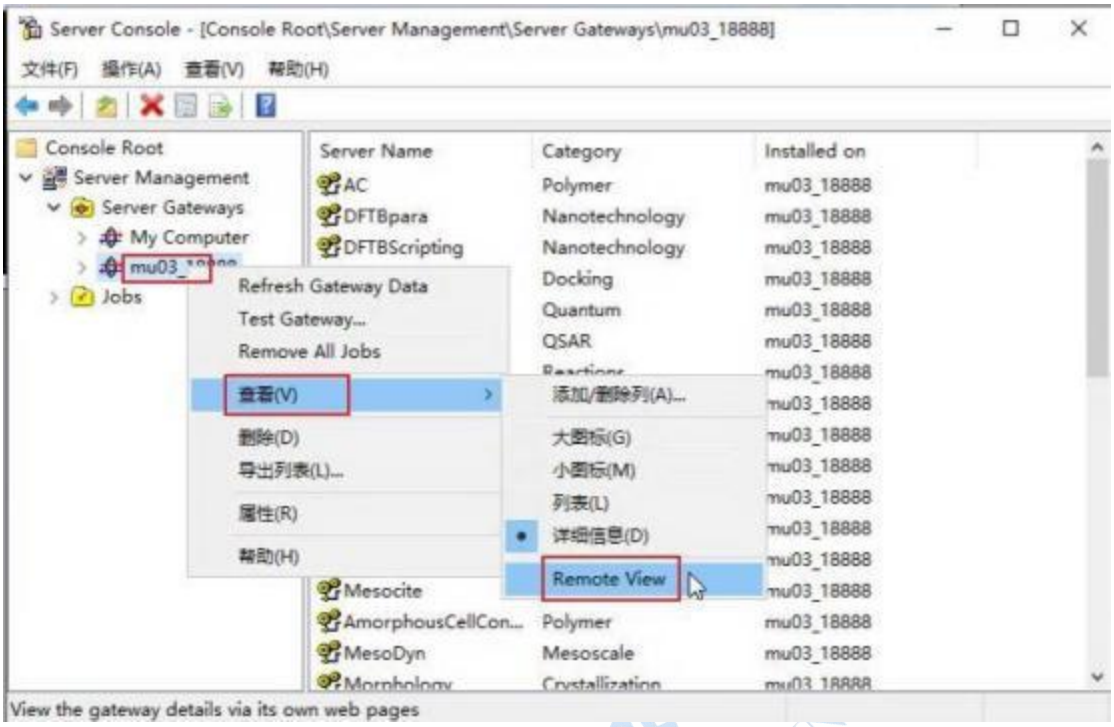
输入管理员提供的初始ms密码进行登录，用户名与平台账号相同。




如图添加成功。



修改自己的ms业务账号密码。







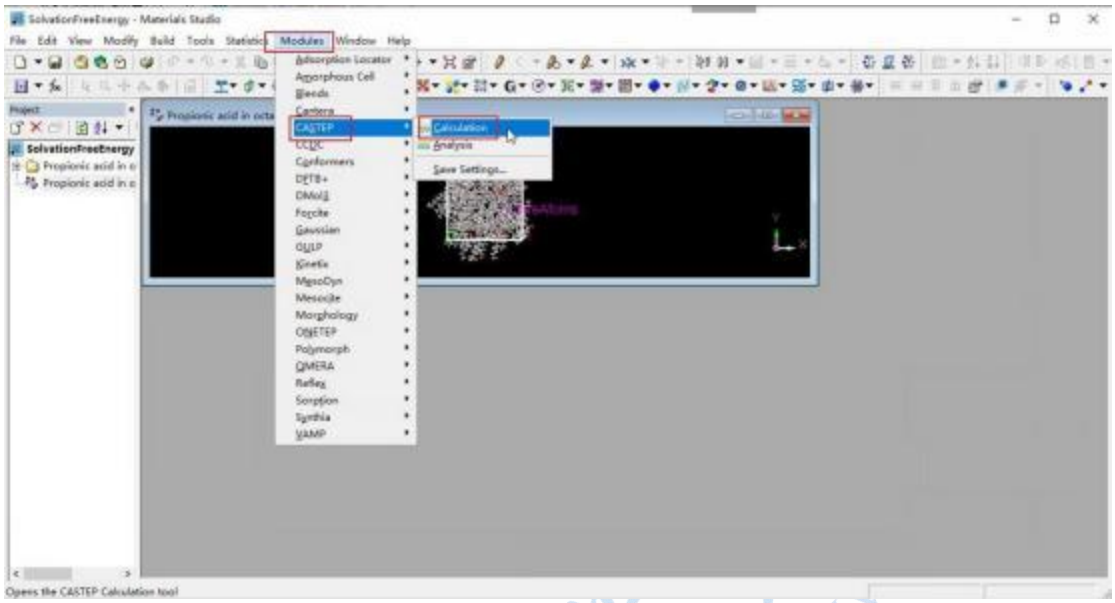
点击password修改密码。



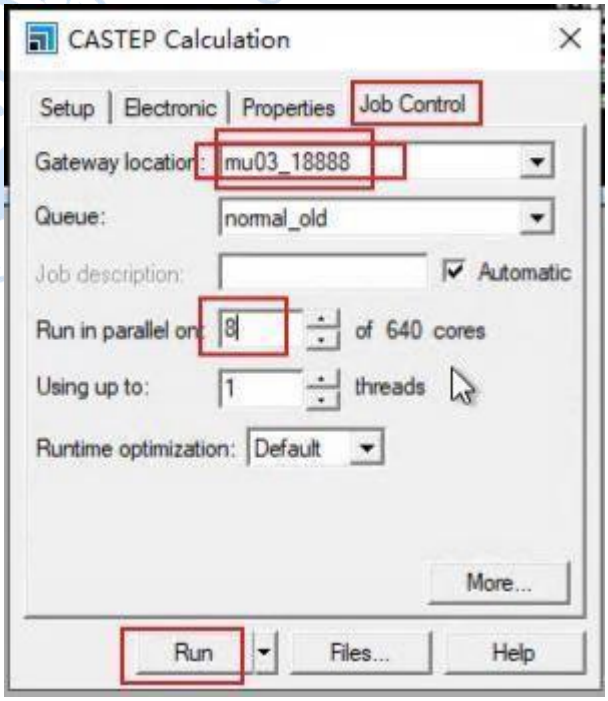

修改完成后关闭上述配置网关的窗口。

### 3. 提交计算任务

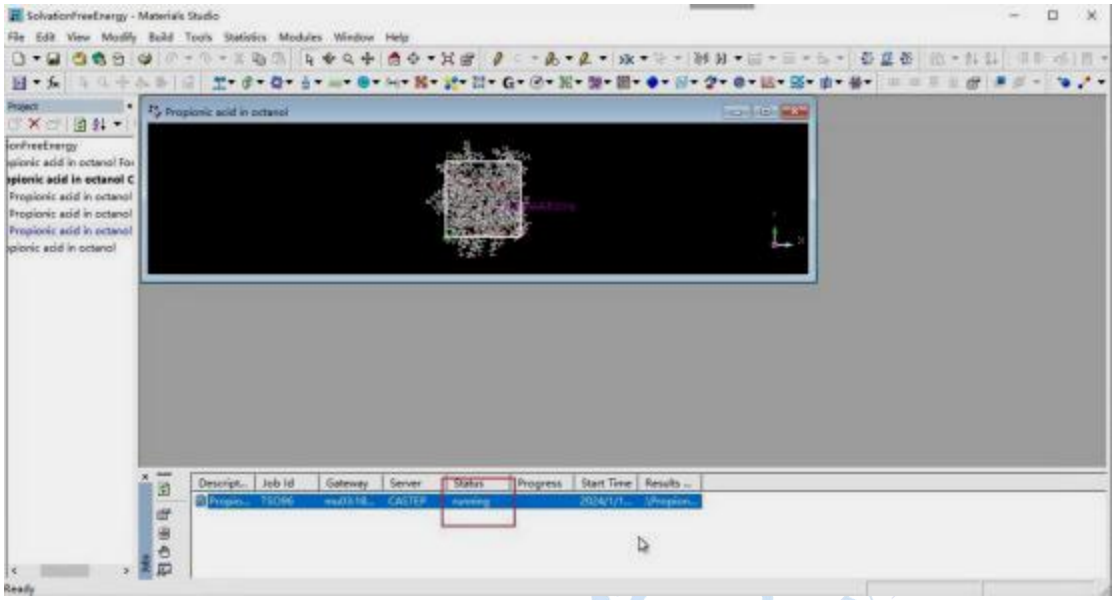
依次点击所示菜单选项。



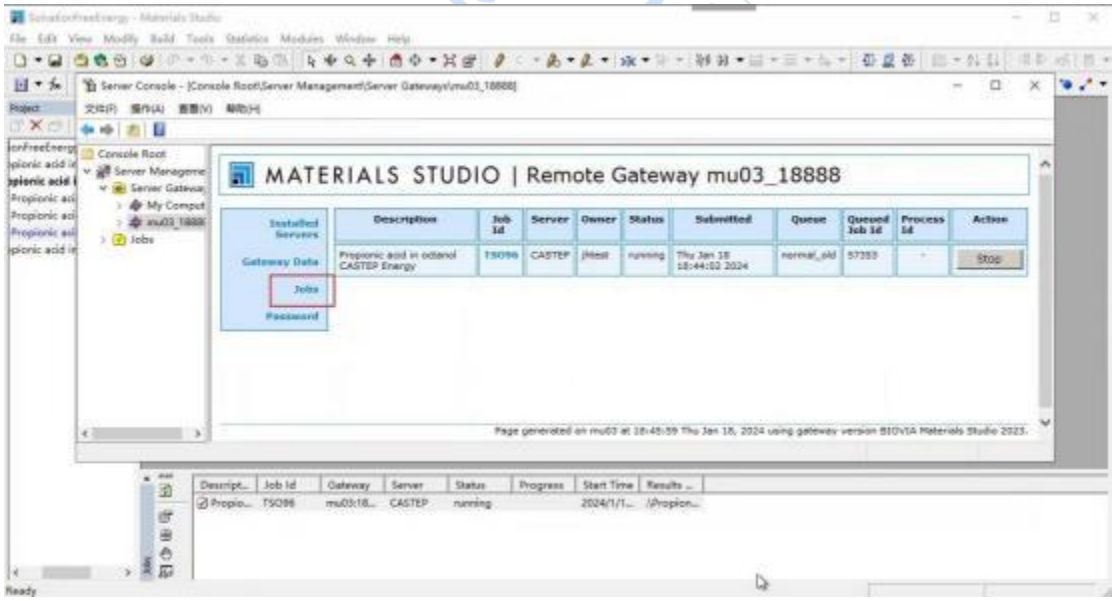

如下图所示：作业提交必须选择 mu03_18888，不能选择本地计算机。队列选择默认的 normal_old 即可，其他队列无法计算。并行核心数按需填写（单机最大 40cores）。



等待作业运行完成后，下方状态栏会显示计算信息（后台Linux 计算节点占满后显示排队）。



在网关配置界面也可以查看和管理工程计算任务。




### 6.2.6 RStudio

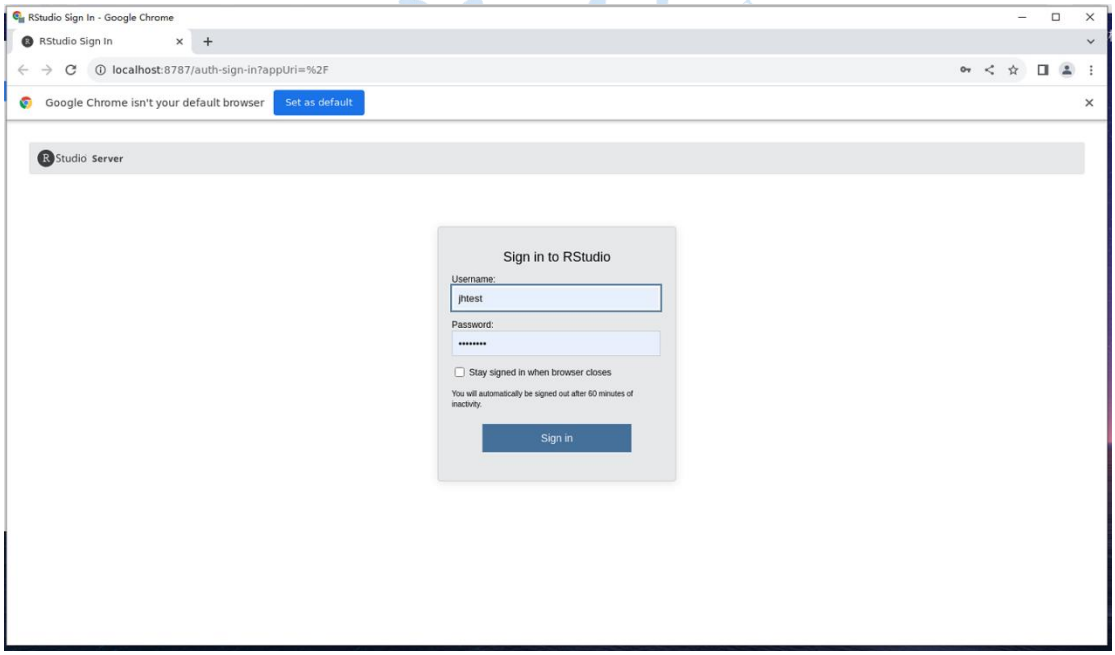
集群门户支持 Rstudio 应用，用户可通过网页方便地管理或处理 R 数据。

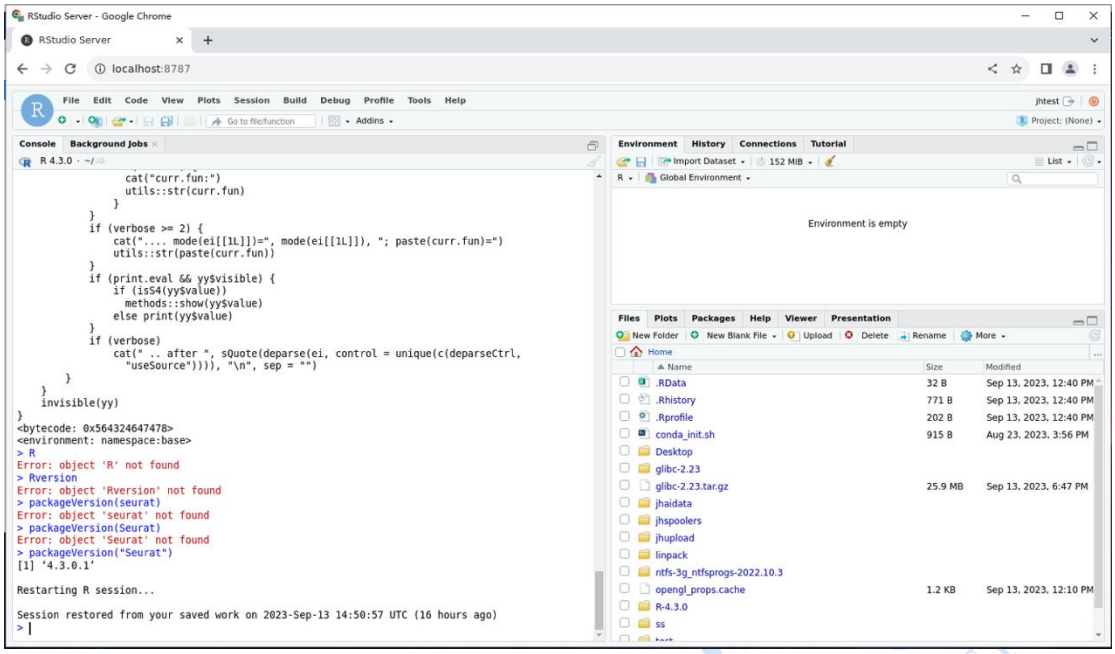
**操作步骤**

1.登录门户后打开“应用仓库”，在仓库中打开“RStudio”应用。




会弹出谷歌浏览器并默认打开Rstudio登录页面，账号和密码与平台账号信息一致。登录后即可正常使用。






2. 会话管理

在门户中“我的会话”页面中，对已经打开的 Rstudio 会话进行管理，支持包含连接、断开、注销和共享等操作。




# 第七章 门户常用功能

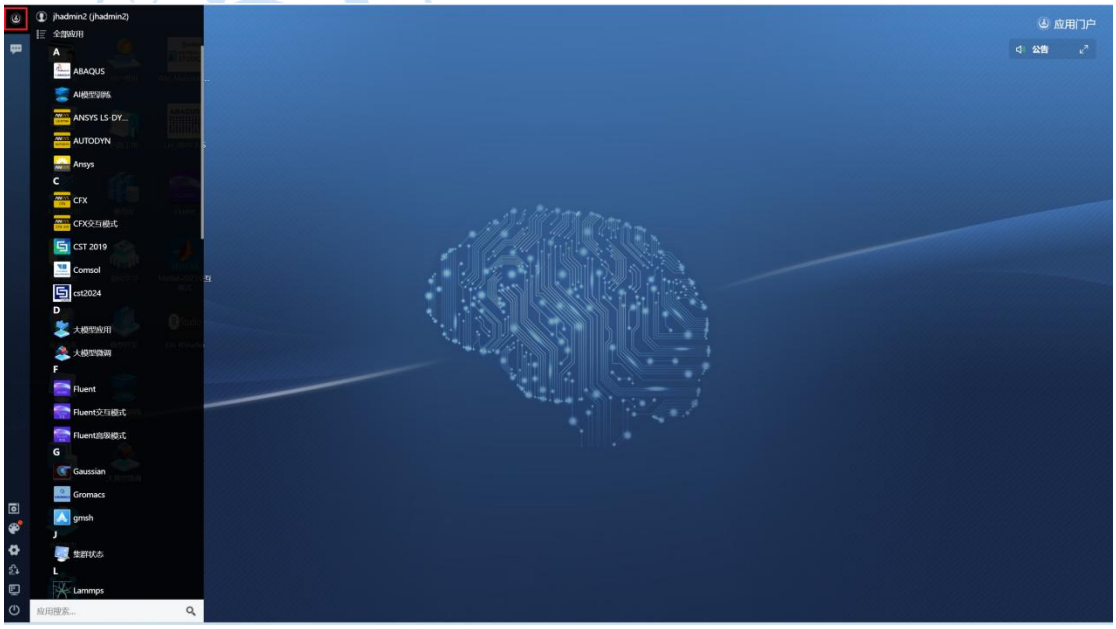
## 7.1 桌面应用夹

应用门户提供了应用夹功能，支持应用夹的新建、删除和重命名操作。

在门户桌面中，可以根据自己的工作习惯新建或删除应用夹，将一些应用分类汇总到不同应用夹，方便管理和使用。通过点击应用夹右上角的小锁（锁定应用夹），可将其固定在桌面上，不允许拖动。自定义的应用夹会作为用户的配置保存起来，每次登录后的桌面设置，依旧保留前一次登录的场景记录。

### 7.2 开始菜单

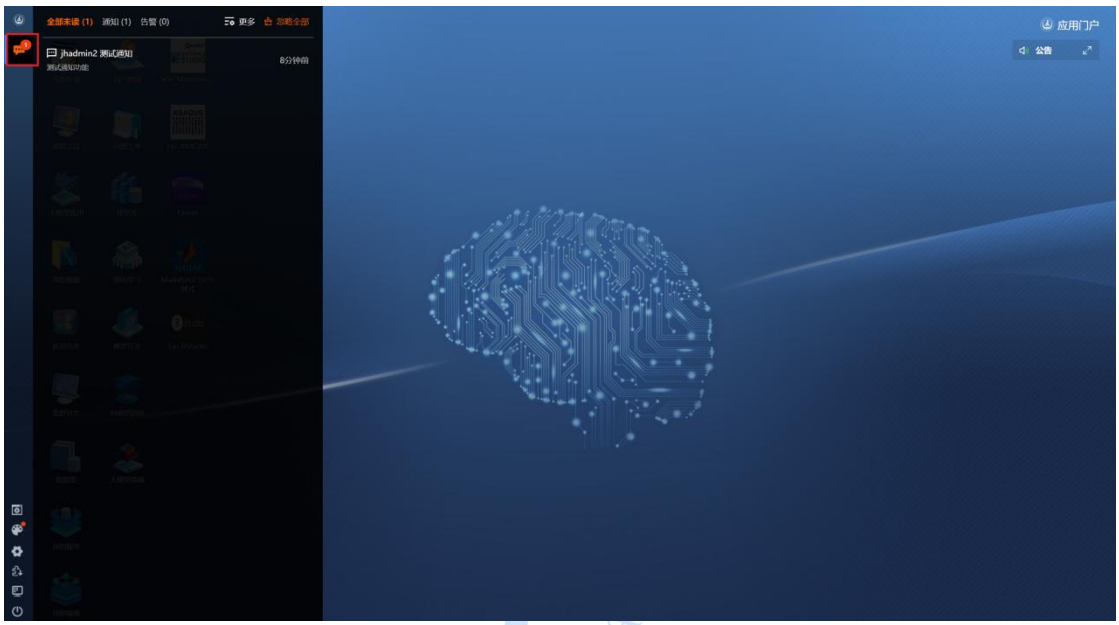
应用门户软件可以集成众多计算应用，每个应用都会以图标方式显示在桌面上，在任务栏顶部的开始菜单中，展示以首字母进行归类的所有应用，点击后可打开此应用，此外，开始菜单还提供了应用检索等功能，在“应用搜索栏”中输入信息，实时显示搜索结果。




应用门户的桌面及开始菜单

### 7.3 通知

当登录应用门户软件后，未读通知的数量，会显示在任务栏顶部的消息图标上，点击消息图标，会在桌面左侧展开未读消息列表。如下图:

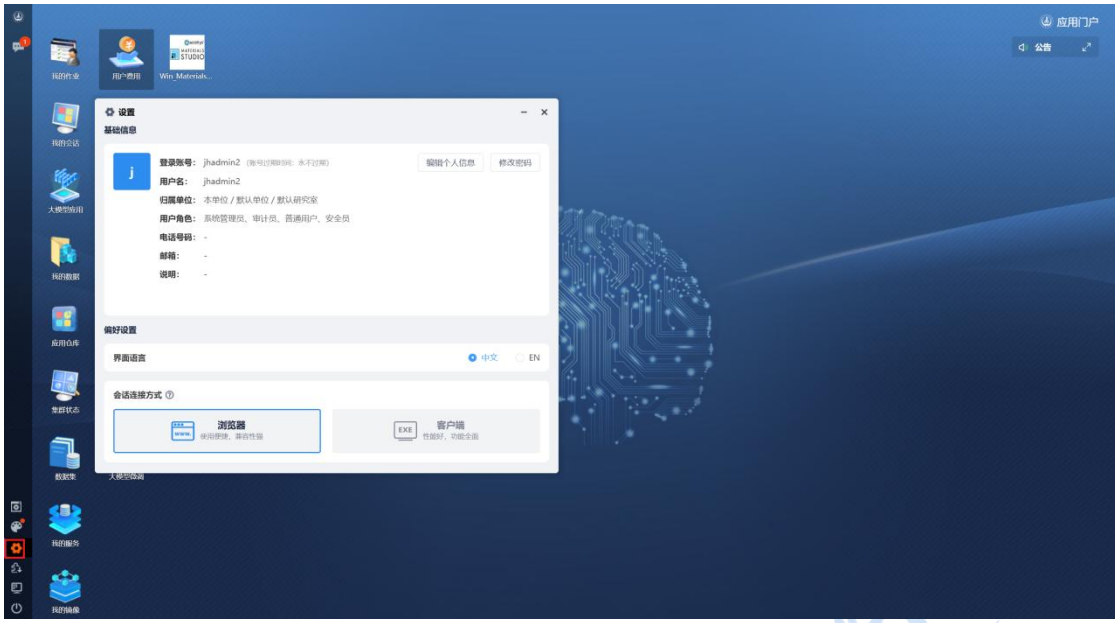



通知列表页面

在使用门户期间，如果收到消息通知，会在桌面右下角弹出消息气泡。点击消息通知列表上方的“更多”按钮，打开“通知管理”窗口。

### 7.4 修改密码

可通过任务栏中的“设置”按钮访问相关配置选项。




设置界面

“修改密码”标签页中，需要输入旧密码和新密码，完成密码修改操作。




修改密码

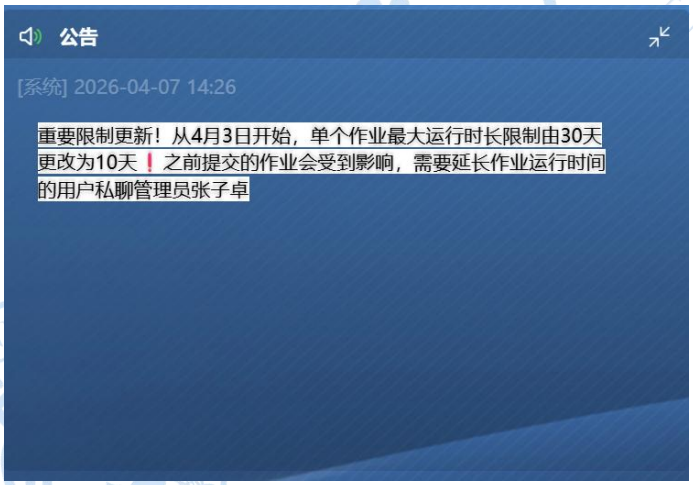
如果忘记密码无法登录平台，可通过信息门户“高性能计算平台修改密码”在线服务进行修改。

## 7.5 显示桌面

应用门户软件提供了一键返回桌面功能，当打开的页面太多，用户需要快速回到桌面进行操作时，用户可以点击任务栏上左下角的“显示桌面”按钮，将全部打开的页面窗口最小化到任务栏。将鼠标移动到任务栏中的图标上时，可通过提示信息看到当前应用的名称。

### 7.6 系统公告

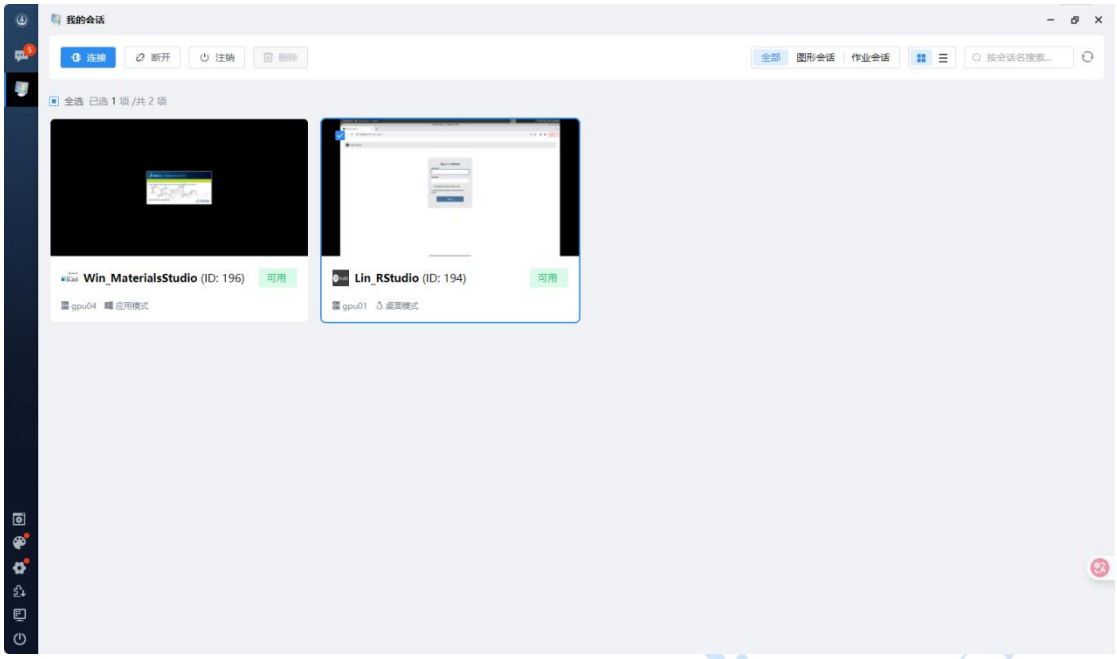
当系统管理员发布公告后，您可以在应用门户桌面中立即查看到最新的公告信息。您也可以手动点击“展开/收缩”按钮操作公告面板。




系统公告

### 7.7 我的会话

“我的会话”包含“三维图形会话”和“作业会话”两个标签页，其主要功能有查看会话详细信息，对会话进行连接、断开、注销、共享和设置密级。能查看当前用户所启动的会话，还有别人共享给自己的会话。




我的会话

在“我的会话”页面可以执行的操作有：查看会话详情以及会话的连接、注销、断开、共享、设置密级和删除。

在会话列表的会话名右侧，还提供了“连接”“断开”“注销”和“共享”快捷图标按钮，当鼠标移至任意会话所在行时会显示快捷键。

点击会话名或双击会话所在行，会从右侧滑出会话的详细信息页面。若此会话被共享，则在详细信息页面中还会显示可交互用户和可观察用户。

为了满足可视化的协作交流，在会话列表中，提供了“共享”按钮。在我的会话列表中，选中需要进行共享的会话所在行，点击“共享”按钮后，就可以调出会话共享的页面。

用户可以将自己的任意一个远程 Windows 和 Linux 或者是其他图形桌面系统同时共享给多个用户，供远程用户操作前后处理 CAD 建模或者是协同装配工作。

会话共享提供两种会话共享模式：默认共享后为观察模式，对方只能观看，不能对会话进行任何操作；参与者连接后，会话 owner 可以授权某个用户操作权限（交互模式），此用户可以直接在会话上任意操作，实现基于同一个会话的多人协同仿真计算和应用设计。

特定场景下，在给用户共享会话时，操作权可以传递给特定用户，用以操作会话，而没有操作权的用户仅能查看。

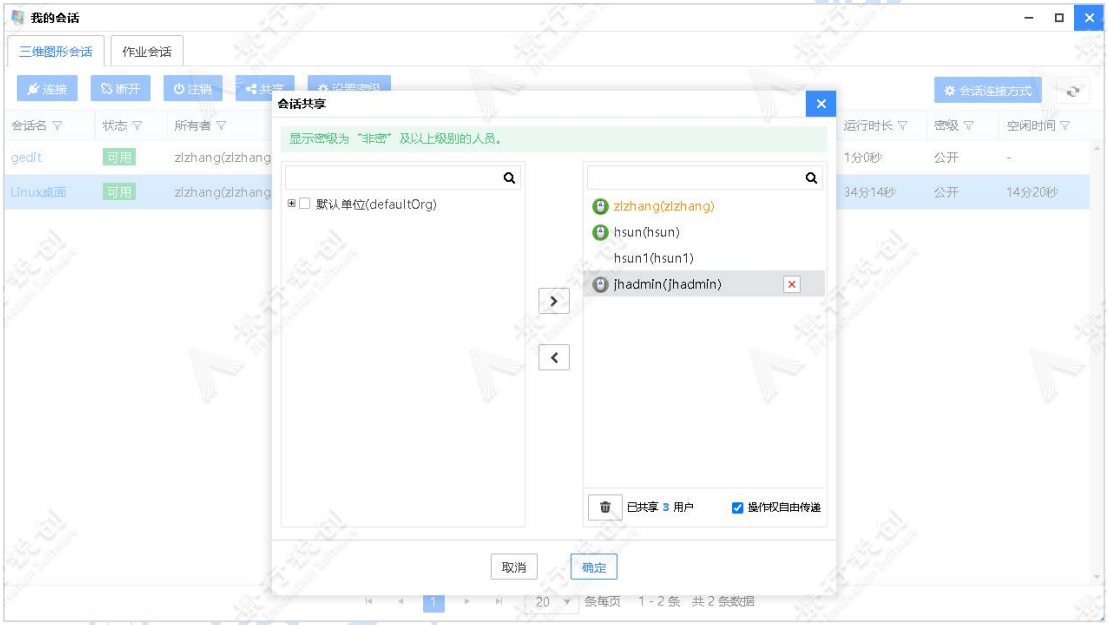
会话共享窗口的已共享列表中，当鼠标滑至用户名左侧的灰色图标上时，会有“设置操作权”的提示信息。




设置操作权的按钮

会话共享页面只能给一个用户设置操作权。点击灰色鼠标图标设置操作权，用户名前鼠标图标变为绿色，表明将会话以交互模式共享给了用户。

会话共享窗口中有“操作权自由传递”的选项，勾选此项后，拥有操作权的参与者 A 可将操作权再次传递给其他用户如参与者 B，将操作权传递后，参与者 A 将只能以观察模式连接会话。




操作权传递

在会话共享页面，可以选择单位、研究室和用户进行会话共享，共享设置成功后，共享的用户可以在我的会话列表中查看或连接此会话。在会话共享页面可以选中已共享的用户名，点击“<”或者选中用户，点击用户右侧“×”按钮来取消此用户加入会话共享，点击清空图标，可以清空加入共享的全部用户。

在我的会话应用中，点击“会话连接方式”按钮可自由选择三维图形会话和作业会话的连接方式，支持浏览器和图形客户端两种会话连接方式。在用户没有安装客户端组件的情况下，打开三维图形会话或上传下载文件时，将会自动切换为浏览器方式。




设置会话的默认连接方式

在会话列表中，如果是三维会话，当鼠标停留在会话名右侧的感叹号图标上时，会出现 tooltips 提示，例如：“执行节点 host1 的剩余内存不足 100M”。如果是带图形的 HPC 作业，只有运行节点的剩余内存达到阈值，才会加载提醒信息。




会话列表中的告警

### 7.8 应用仓库

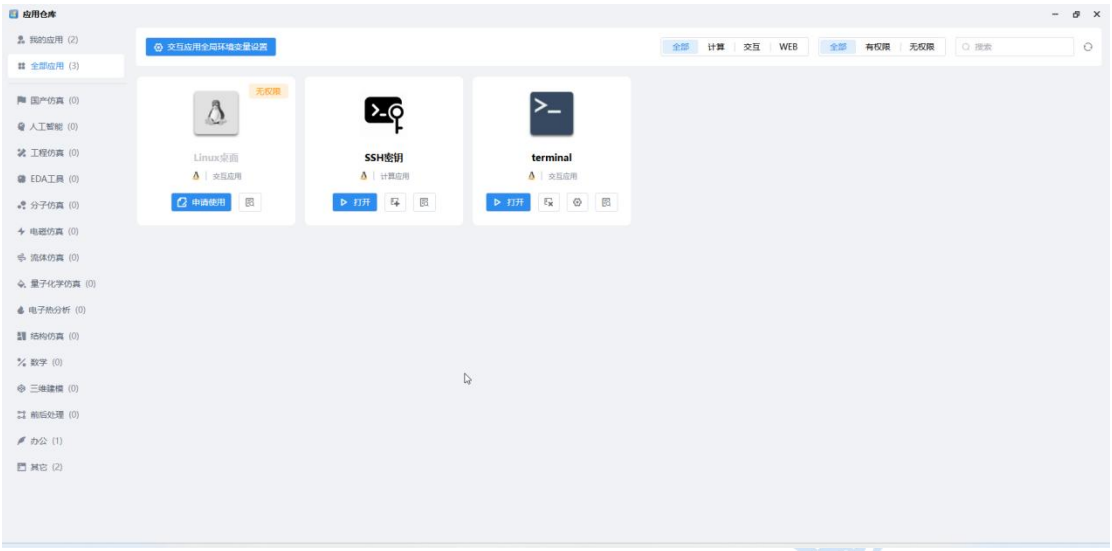
应用仓库可以对应用门户平台中所有已授权的应用进行统一的打开、添加或移除桌面快捷方式、申请、设置（配置环境变量和设置工作路径）和查看详情等操作。

“全部应用”菜单展示门户中所有上线的应用，应用卡片右上角的标签有三种应用状态：有权限、无权限、审批中。

对于有权限的应用，可以进行“打开”“设置”“查看详情”“添加或移除桌面快捷方式”

对于无权限的应用，可以进行“申请使用”和“查看详情”操作。

对于审批中的应用，可以查看详情。



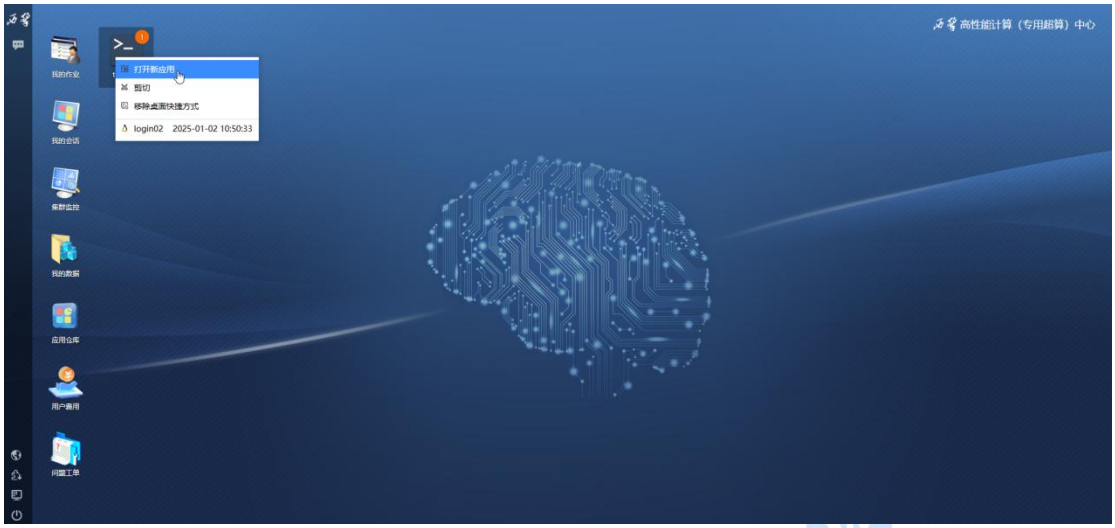

应用仓库

“我的应用”菜单中展示了授权给自己所有应用，菜单同时提供了两种分类方式“类型”和“用途”，方便用户快速地从相对应的分类选择应用。对于应用数量比较多的情况，可以通过应用状态来筛选符合条件的应用，也可以通过搜索框输入应用名称进行过滤。

如果希望查看最新的管理员审批的状态，可以点击刷新按钮，将获取到所有应用最新的状态和信息。

### 7.8.1 桌面快捷方式

管理员分配给用户的应用，默认没有桌面快捷方式，用户需要在应用卡片中点击“添加桌面快捷方式”按钮后，才能在应用门户桌面上显示。直接双击应用图标就可以启动应用，右键单击应用图标可以看到更多选项。




鼠标右键点击应用图标的菜单

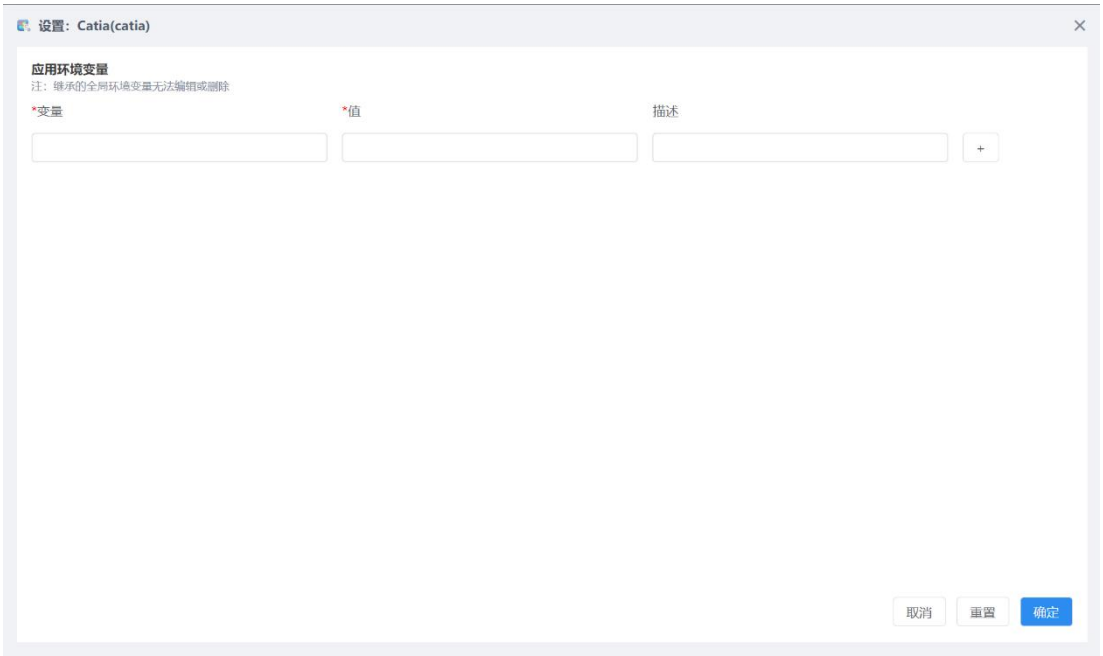
### 7.8.2 应用申请

如果管理员没有给您所需要的应用使用权限的话，则可以向管理员发出申请，同时说明申请理由。待管理员审核通过后，即可在应用仓库中使用该应用。

### 7.8.3 环境变量

如果应用需依赖特定的环境变量才能正常运行，用户可前往应用仓库，给该应用添加所需要的环境变量。如果管理员在管理门户中也设置了环境变量，则以用户设置的环境变量为准。

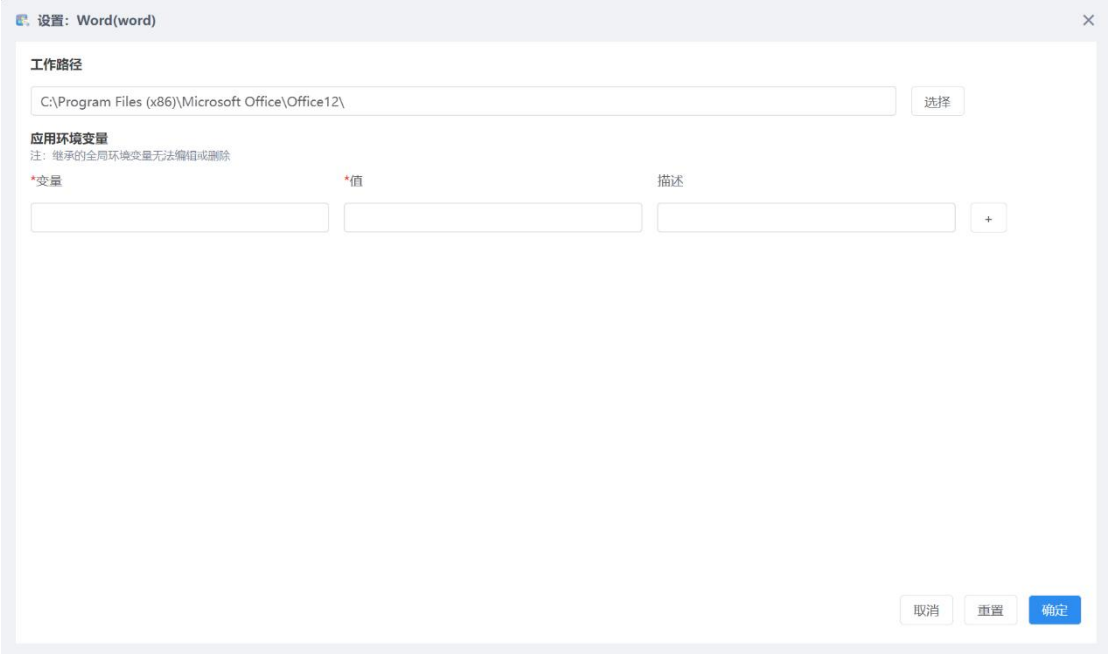
对于有权限的应用或桌面，可以直接点击卡片下方的“设置”按钮，在弹出的窗口中进行相关配置。




给应用添加环境变量

### 7.8.4 工作路径

如果您遇到某应用可以正常启动，但运行时提示找不到可执行程序的路径等错误，可以尝试修改此应用的工作路径。需要注意的是，只有管理员在应用管理中开启了“允许普通用户自定义工作路径”权限，用户才能在应用仓库的应用卡片中点击“设置”进行修改。




设置工作路径

# 第八章 常用命令

## 8.1 调度常用命令说明

### 8.1.1 jsub

■ 功能：提交作业

■ 概要：

jsub [-h] [-V] [-x] [-H] [-I] [-IX] [-Is] [-djm] [-L login_shell]

[-P project_name] [-R res_req] [-q queue_name]

[-app application_name]

[-m "host_name[+[pref_level]] | host_group[+[pref_level]]..."]

[-n min_processors[max_processors]] [-J job_name] [-Jd job_desc]

[-i in_file] [-o out_file] [-e err_file]

[-E pre_exec_command[argument...]]

[-Ep post_exec_command[argument...]]

[-cwd current_working_directory]

[-gpgpu "num [type=type1,type2...] [gmem=gmemsize] [mig=gsize]"]

[-r | rn] [-M mem_limit] [-W run_limit] [-We limit]

[-hosts "num [hselect=hselect_string]"] [-port num] [-mf file]

[-aps priority] [-f [host:]local_file operator [remote_file]]

[-command command [command ...]]

■ 选项:

➢ -h: 显示此帮助信息。

➢ -V: 打印景行资源管理与调度软件发行版本信息。

➢ -x: 指定节点以独占模式运行作业。

➢ -H: 提交一个暂时不执行的作业。

➢ -I: 提交一个交互式作业。

➢ -IX: 提交一个带 X11 转发功能的交互式作业。

➢ -Is: 提交一个服务模式的交互式作业。

➢ -djm: 提交一个分布式作业。

➢ -L login_shell: 指定 shell 登录并执行作业，支持的 shells 有:

sh、bash、csh 和 tcsh。

➢ -P project|_name: 指定提交作业的项目名称。

➢ -R res_req: 指定提交作业特定的资源需求信息。

- -q queue_name: 指定提交作业的队列。
- -app application_name: 提交作业到指定的应用程序。
- -m "host_name[+[pref_level]] | host_group[+[pref_level]]...": 指定作业可选择的执行节点或者节点组。
- -n min_processors[max_processors]: 指定处理器的最小和最大数量。
- -J job_name: 指定提交的作业名称。
- -Jd job_desc: 指定作业的描述信息。
- -i in_file: 从指定文件获取作业的标准输入，可指定环境变量。
- -o out_file: 将作业的标准输出写入指定文件，可指定环境变量。
- -e err_file: 将作业的标准错误输出写入指定文件，可指定环境变量。
- -E pre_exec_command[argument...]: 作业执行前需要运行 pre-exec。
- -Ep post_exec_command[argument...]: 作业执行结束后需要运行 post-exec。
- -cwd current_working_directory: 指定作业的工作目录，可指定环境变量。

- -gpgpu: 为作业指定 GPU 卡或 MIG 的类型和数量以及需要预留的 GPU 显存。
- -r: 指定作业可以重新运行。
- -rn: 指定作业忽略重新运行队列上的配置。
- -M mem_limit: 指定内存限制，以 MB 为单位。
- -W run_limit: 指定作业的运行时间限制，以分钟为单位。
- -We limit: 指定作业的预估运行时间，以分钟为单位。
- -hosts "num [hselect=hselect_string]": 指定 hosts number 和 hselect_string，用于 host 分配资源 hselect_string 由 num@hostgroup 或 host 组成，用逗号隔开。
- -port num: 指定作业的端口号。
- -mf file: 为作业指定使用机器配置文件。
- -aps priority: 指定作业优先级。
- -f [host:]local_file operator [remote_file]: 将本地文件或目录从 I/O 节点传输到作业执行节点的缓存目录。支持的操作符有: > < <> ><
- -command command: 指定作业命令行参数或脚本。

■ 命令示例

提交一个作业到指定节点 host1，并指定作业运行队列为 short。

```
$jsub -m host1 -q short  my_job
```

提交一个作业并指定运行节点，其中 host1 是最适合运行作业的节点，其次是 host2，最后是 host3，“+”增加的数值越高，其节点的优先级就越高。

```
$jsub -m "host1+2 host2+1 host3" my_job
```

提交一个作业并指定特定的资源需求，如果 type==NTX64 节点内存大于 50MB，作业将派发到 type==NTX64，如果 type==LINUX64 节点内存大于 100MB，作业将派发到 type==LINUX64，若这两个条件都不满足，作业将不被派发。

```
$jsub -R "select[((type==NTX64&& mem >
50)||(type==LINUX64&& mem > 100))]" my_job
```

提交一个作业并指定资源预留数量 mem 为 50MB，swap 为 100MB。

```
$jsub -R "rusage[mem=50&swap=100]" my_job
```

提交一个作业并指定运行作业最少需要 4 个 CPU 数，并且只能在一个节点上运行该作业。

```
$jsub -n 4 -R "span[hosts=1]" my_job
```

提交一个作业并指定运行作业最少需要 4 个 CPU ，并且每个节点只能为该作业提供 2 个 CPU。

```
$jsub -n 4 -R "span[ptile=2]" my_job
```

提交一个作业并指定运行作业需要的 CPU 最大数和最小数。

```
$jsub -n 2,4 my_job
```

提交一个禁止调度到 host1 上运行的作业。

```
$jsub -m ~host1 my_job
```

提交一个作业，并指定将该作业的运行信息输出保存为文件 output.sh，保存路径为该作业提交用户的家目录。

```
$jsub -o ~/output.sh my_job
```

在作业提交参数“-i/-o/-e/-cwd”中可指定路径环境变量，设置环境变量 in=/apps/dir_input 和 out=/apps/dir_output（路径可自定义），提交含有指定路径变量的作业，在作业的详细信息中可以看到路径环境变量已经生效。

```
  $ jsub -i /apps/dir_input/input.sh -o
/apps/dir_output/output.log -e /apps/dir_output/error.log -
cwd /apps/dir_output 'hhh;sleep 10;hostname'


  $ jjobs -l 59
  Job <59>, Job Name <hhh;sleep 10;hostname>, User <user1>,
Project <default>, Status <PSUSP>, Queue <normal>,
Application <default>, APS Priority <5000>, Command
<hhh;sleep 10;hostname>
  Thu Mar 31 17:22:43: Submitted from host <host1>, CWD
</apps/dir_output>, Input File <dir_input/input.sh>, Output
File <dir_output/output.log>, Error File
<dir_output/error.log>, Specified Hosts <host1>;
```

将提交节点的/files/abc.conf 文件传输至执行节点的$JH_STAGE_DIR/目录下，再将 abc.conf 文件名修改为 test.conf，再传回至提交节点的/tmp 目录下。

```
  jsub -f '/files/abc.conf > abc.conf' -f ' /tmp/ <
test.conf'  'mv $JH_STAGE_DIR/abc.conf
$JH_STAGE_DIR/test.conf'
```

注：$JH_STAGE_DIR 对应的值为管理门户->调度管理->高级配置功能页面表单中对应的配置项“Linux 节点数据缓存目录(STAGE_TOP_DIR_LINUX)”的值（此处以/tmp/stagein 为例）。

在此作业的执行节点文件系统会按照以下规则产生作业文件临时目录:

```
/tmp/stagein/.${JOB_ID}.${JOB_INDEX}.stage
```

例如:

```
/tmp/stagein/.54.0.stage/xxx
```

### 8.1.2 jjobs

- 功能: 显示作业信息。
- 概要:

jjobs [-h] [-V] [-w | -l] [-a] [-d] [-p] [-s] [-r] [-A] [-m host_name]

[-q queue_name] [-app application_name] [-u user_name | -u all]

[-P project_name] [-o "field_name[:[-][output_width]] ...

[delimiter='character']"] [-J name_spec] [-Jd desc_spec] [-env]

[-t] [-regexp]

[jobId [jobId ...]]

- 选项:
  - -h: 显示此帮助信息。
  - -V: 打印景行资源管理与调度软件发行版本信息。
  - -w: 以宽格式显示作业的详细信息。

- -l: 以长格式显示作业的详细信息。
- -a: 显示一定时间范围内所有状态的作业信息。
- -d: 显示一定时间范围内已经结束的作业信息。
- -p: 显示作业状态为 pend 的作业信息，并显示 pend 的原因。
- -s: 显示作业状态为 suspend 的作业信息，并显示 suspend 的原因。
- -r: 显示作业状态为 run 的作业信息。
- -A: 显示作业数组信息。
- -m host_name: 显示已经派发到指定节点上的作业信息。
- -q queue_name: 显示指定队列上运行作业信息。
- -app application_name: 只显示指定应用程序中的作业。
- -u user_name | -u all: 只显示指定用户/用户组提交的作业，关键字'all'指定所有用户。
- -P project_name: 显示指定项目的作业信息。
- -o: 显示自定义输出格式的作业信息。
- -J name_spec: 显示指定作业名称的作业或者作业数组的信息。
- -Jd desc_spec: 指定作业的描述信息。
- -env: 显示指定作业提交环境中的环境变量。

- -t: 显示指定 djm 作业的任务信息。
- -regexp: 在筛选条件中使用正则表达式。

■ 命令示例

显示用户 user1 的等待作业信息和作业的 PENDING REASONS。

```
$jjobs -p -u user1
```

显示节点 host1 上所有正在运行的作业信息。

```
$jjobs -r -m host1
```

显示在队列 short 上且项目名称为 my_project 的所有用户的作业信息。

```
$jjobs -q short -P my_project -u all
```

显示指定的作业详细信息。

```
$jjobs -l job_id
```

按照指定格式输出作业信息（jobid 宽度为 4，stat 宽度为 10，并右对齐，分隔符为分号）。

```
jjobs -o "jobid:4 stat:-10 from_host delimiter=';'"
```

### 8.1.3 jhosts

■ 功能：显示节点资源及负载信息。

■ 概要：

jhosts [-h] [-V] [-w | -l] [-R res_req] [-s [res_name [res_name ...]]] [host_name [host_name ...]] {stat,attrib,remove} ...

■ 选项：

➢ -h：打印命令使用方法。

➢ -V：打印景行资源管理与调度软件发行版本信息。

➢ -w：以宽格式形式显示详细信息。

➢ -l：以长格式形式显示详细信息。

➢ -R res_req：显示满足特定资源需求的节点信息。

➢ -s [res_name [res_name ...]]：查询节点上的共享资源信息，不能与其他参数一起使用。

■ 子命令 jhosts stat：

➢ jhosts stat：显示节点负载信息。

➢ -h：打印命令使用方法。

➢ -l：以长格式形式显示详细信息。

➢ -w：以宽格式形式显示详细信息。

- -n: 不换行显示主机信息。

■ 子命令 jhosts attrib:

- jhosts attrib: 显示主机及其静态资源信息。
- -h: 打印命令使用方法。
- -l: 以长格式形式显示详细信息。
- -w: 以宽格式形式显示详细信息。

■ 子命令 jhosts remove:

- jhosts remove: 从集群中删除处于宕机状态的主机信息。
- -h: 打印命令使用方法。

■ 命令示例

显示节点 host1 的状态和作业资源信息。

```
$jhosts -l host1
```

显示集群中内存大于 600MB 的节点信息。

```
$jhosts -R "select[mem>600]"
```

### 8.1.4 jqueues

■ 功能：显示集群中队列的相关信息。默认显示集群中所有队列的以下信息：队列名称、队列优先级、队列状态、作业槽信息和作业状态信息。

■ 概要：

jqueues [-h] [-V] [-w | -l] [-m host_name] [-u user_name]

[queue_name [queue_name ...]]

■ 选项：

- -h：打印命令使用方法。
- -V：打印景行资源管理与调度软件发行版本信息。
- -w：以宽格式形式显示队列信息。
- -l：以长格式形式显示队列的详细信息。
- -m host_name：显示指定节点或节点组上能够运行作业的队列信息。
- -u user_name：显示指定用户或用户组所属的队列信息。
- queue_name：显示指定队列信息。

■ 命令示例

显示节点 host1 及用户 user1 所属的队列信息。

```
$jqueues -m host1 -u user1
```

### 8.1.5 jctrl

■ 功能：主要用于作业控制，比如 kill 作业或者停止一个作业等。

■ 概要：

jctrl [-h] [-V] {stop,resume,kill,bot,top,start,requeue,peek,clean} ...

■ 选项：

➢ -h：打印命令使用方法。

➢ -V：打印景行资源管理与调度软件发行版本信息。

■ 子命令 stop（挂起作业）：

概要：

jctrl stop [-h] [-m host_name] [-q queue_name] [-u user_name | -u all]

[-J job_name] [-app application_name]

jobId | jobId[index_list] | 0 [jobId | jobId[index_list] | 0 ...]

选项：

➢ -h：打印命令使用方法。

➢ -m host_name: 仅挂起分派到指定主机或主机组的作业。

➢ -q queue_name: 仅挂起指定队列中的作业。

➢ -u user_name | -u all: 仅暂停指定用户或用户组拥有的作业，如果指定了关键字 all，表示暂停所有用户的作业。

➢ -J job_name: 仅挂起具有指定作业名称的作业。

➢ -app application_name: 仅暂停指定应用程序中的作业。

■ 子命令 resume（恢复挂起的作业）:

概要:

jctrl resume [-h] [-m host_name] [-q queue_name] [-u user_name | -u all] [-J job_name] [-app application_name] jobId | jobId[index_list] | 0 [jobId | jobId[index_list] | 0 ...]

选项:

➢ -h: 打印命令使用方法。

➢ -m host_name: 恢复指定节点或者节点组上已被挂起的作业。

➢ -q queue_name: 恢复指定队列中已被挂起的作业。

➢ -u user_name | -u all: 恢复指定用户或者所有用户已被挂起的作业。

➢ -J job_name: 恢复已被挂起的指定作业名称的作业。

➢ -app application_name: 只恢复指定应用程序中的作业。

■ 子命令 kill（发送特定信号停止一个作业）:

概要:

jctrl kill [-h] [-m host_name] [-q queue_name] [-u user_name | -u all]

[-J job_name] [-f] [-app application_name]

jobId | jobId[index_list] | 0 [jobId | jobId[index_list] | 0 ...]

选项:

➢ -h: 打印命令使用方法。

➢ -m host_name: 停止指定节点上的作业。

➢ -q queue_name: 停止指定队列上的作业。

➢ -u user_name | -u all: 停止指定用户提交的作业，若指定参数为 all，表示终止所有用户未完成的作业。

➢ -J job_name: 停止指定作业名称的作业。

➢ -f: 强制从非调度程序系统中终止作业，而无需等待作业在操作系统中终止。

➢ -app application_name: 停止指定应用程序中的作业。

■ 子命令 bot（将等待状态的单个或多个作业移动到队列尾部）:

概要:

jctrl bot [-h] [-c] [-p position] jobId | jobId[index] [jobId | jobId[index] ...]

选项:

➢ -h: 打印命令使用方法。

➢ -c: 可选，指定操作是集群级别。

➢ -p position: 可选，可以指定 position 参数以指示将作业放置在用户或集群中的哪个位置，position 是一个正整数，表示作业从用户或集群的末尾开始的目标位置。

■ 命令示例

同时对多个 PEND 状态的作业进行置底排序，-p 2 表示将等待作业置底排序到第 2 位。

```
[user1@host1 apps]$ jctrl bot -p 2 241 243 242
Job <241> has been moved to position 2 from bottom
Job <243> has been moved to position 2 from bottom
Job <242> has been moved to position 2 from bottom
[user1@host1 apps]$ jjobs
JOBID      USER     STAT  QUEUE     FROM_HOST EXEC_HOST
JOB_NAME     SUBMIT_TIME    ORDER
237        user1    RUN   normal    host1     host2
sleep 1000   Mar 30 15:40   -
238        user1    RUN   normal    host1     host2
sleep 1000   Mar 30 15:40   -
239        user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   1
241        user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   2
243        user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   3
242        user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   4
240        user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   5
```

■ 子命令 top（将等待状态的单个或多个作业移动到队列顶部）：

概要:

jctrl top [-h] [-c] [-p position] jobId | jobId[index] [jobId | jobId[index] ...]

选项:

- -h: 打印命令使用方法。
- -c: 可选，指定操作是集群级别。
- -p position: 可选，可以指定位置参数以指示作业将放置在用户或集群中的哪个位置。 position 是一个正整数，表示作业从用户或集群开始的目标位置。

■ 命令示例

同时对多个 PEND 状态的作业进行置顶排序。

```
  [user1@host1 apps]$ jjobs
  JOBID      USER     STAT  QUEUE     FROM_HOST EXEC_HOST
JOB_NAME     SUBMIT_TIME    ORDER
  237        user1    RUN   normal    host1     host2
sleep 1000   Mar 30 15:40   -
  238        user1    RUN   normal    host1     host2
sleep 1000   Mar 30 15:40   -
```

```
  239         user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   1
  240         user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   2
  241         user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   3
  242         user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   4
  243         user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   5
  [user1@host1 apps]$ jctrl top -p 1 241 242 243
  Job <241> has been moved to position 1 from top
  Job <242> has been moved to position 1 from top
  Job <243> has been moved to position 1 from top
  [user1@host1 apps]$ jjobs
  JOBID      USER     STAT  QUEUE     FROM_HOST EXEC_HOST
JOB_NAME     SUBMIT_TIME    ORDER
  237         user1    RUN   normal    host1     host2
sleep 1000   Mar 30 15:40   -
  238         user1    RUN   normal    host1     host2
sleep 1000   Mar 30 15:40   -
  243         user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   1
  242         user1    PEND  normal    host1
sleep 1000   Mar 30 15:40   2
```

```
  241         user1    PEND  normal     host1
sleep 1000    Mar 30 15:40   3
  239         user1    PEND  normal     host1
sleep 1000    Mar 30 15:40   4
  240         user1    PEND  normal     host1
sleep 1000    Mar 30 15:40   5
```

■ 子命令 start（强制运行指定节点上的等待作业）：

概要:

jctrl start [-h] [-f] -m host_name jobId | jobId[index]

选项:

➢ -h: 打印命令使用方法。

➢ -f: 允许作业运行而不会因运行窗口或挂起条件而被挂起。

➢ -m host_name: 需要，指定一个或多个运行作业的主机。

■ 子命令 requeue（终止作业并重新排队作业到运行队列）：

概要:

jctrl requeue [-h] [-u user_name | -u all] [-J job_name] [-d] [-e] [-r] [-a] [-H] 0 | jobId | jobId[index] [0 | jobId | jobId[index] ...]

选项:

- -h: 打印命令使用方法。
- -u user_name | -u all: 对指定用户或组提交的作业进行操作，如果指定关键字 all，表示对所有用户的作业进行操作。
- -J job_name: 仅对具有指定名称的作业进行操作。
- -d: 将已完成运行且处于 DONE 作业状态的作业重新排队。
- -e: 将异常终止的作业重新排队，并显示 EXIT 作业状态。
- -r: 对正在运行的作业重新排队。
- -a: 对所有作业重新排队，包括正在运行的作业、暂停作业以及具有 EXIT 或 DONE 状态的作业。
- -H: 将作业重新排队到 PSUSP 作业状态。

■ 子命令 peek（显示运行作业的执行输出）

概要:

jctrl peek [-h] [-f] [-t [task id]] jobId | jobId[index]

选项:

- -h: 打印命令使用方法。
- -f: 实时打印显示运行作业的执行输出到屏幕上。
- -t [task id]: 查询指定的任务输出。

例如:

挂起用户 user1 提交到队列 short 上的所有状态作业。

```
$jctrl stop -q short -u user1 0
```

恢复用户 user1 提交到所有队列上的未完成的作业。

```
$jctrl resume -u user1 0
```

终止所有用户提交到队列 short 上的所有作业。

```
$jctrl kill -q short -u all 0
```

重排队所有用户的所有作业。

```
$jctrl requeue -a -u all 0
```

指定一个运行作业的主机

```
$jctrl start -m hostname jobID
```

指定多个运行作业的主机

```
$jctrl start -m "hostname1 hostname2" jobID
```

■ 子命令 clean（手动清理已完成的作业）：

概要:

jctrl clean [-h] [-u user_name | -u all]

jobId | jobId[index_list] | 0 [jobId | jobId[index_list] | 0 ...]

选项:

➢ -h: 打印命令使用方法。

➢ -u: 指定用户/用户组提交的作业，关键字'all'指定所有用户。

■ 命令示例

删除用户 user1 已经完成的 jobId 为 5、6、7 的作业，也可以单独删除某个已完成的作业。

```
$jctrl clean -u user1 0
Job <5> is being cleaned
Job <6> is being cleaned
Job <7> is being cleaned
```

### 8.1.6 jhostgroup

■ 功能: 显示集群中定义的节点组信息。

■ 概要:

jhostgroup [-h] [-V] [-r] [-w] [group_name [group_name ...]]

■ 选项:

➢ -h: 打印命令使用方法。

➢ -V: 打印景行资源管理与调度软件发行版本信息。

➢ -r: 递归展开节点组信息，在展开列表中只显示节点名，不包含节点子组名称，且重复的节点名称只列出一次。

➢ -w: 以宽格式形式显示节点组信息。

➢ group_name: 显示指定节点组的信息。

### 8.1.7 jlimits

■ 功能: 主要用于显示集群中用户的 cpu 和作业的限制信息。

■ 概要:

jlimits [-h] [-V] [-s | -c] [-u user_name|user_group|all]

[-m host_name|host_group|all] [-q queue_name|all]

[-app application_name|all] [-w]

■ 选项:

➢ -h: 打印命令使用方法。

➢ -V: 打印景行资源管理与调度软件发行版本信息。

➢ -s: 显示没有指定队列、应用和节点配置的用户或用户组详细的作业限制信息。

➢ -c: 显示用户的作业限制配置信息。

➢ -u user_name|user_group|all: 显示集群中用户和用户组的相关 CPU 和作业限制信息。

➢ -m host_name|host_group|all: 显示集群中主机的相关 CPU 和作业限制信息。

➢ -q queue_name|all: 显示集群中队列的相关 CPU 和作业限制信息。

➢ -app application_name|all: 显示集群中应用的相关 CPU 和作业限制信息。

➢ -w: 宽格式显示 CPU 和作业限制信息，不截取字段。

■ 命令示例

显示集群中 user1 用户的相关 CPU 和作业限制信息。

```
$jlimits -u user1
```

显示集群中所有用户的相关 CPU 和作业限制信息。

```
$jlimits -u all
```

显示集群中主机名称是 host2 相关的 CPU 和作业限制信息。

```
$jlimits -m host2
```

显示集群中所有主机的相关 CPU 和作业限制信息。

```
$jlimits -m all
```

显示集群中队列名称是 queue1 相关的 CPU 和作业限制信息。

```
$jlimits -q queue1
```

显示集群中所有队列的相关 CPU 和作业限制信息。

```
$jlimits -q all
```

显示集群中应用名称是 app1 相关的 CPU 和作业限制信息。

```
$jlimits -app app1
```

显示集群中所有应用的相关 CPU 和作业限制信息。

```
$jlimits -app all
```

### 8.1.8 jcode

- 功能：显示错误码的释义信息。
- 概要：

jcode [-h] [-V] [-all] [-e code [code ...]]

[-r code [code ...]] [-p code [code ...]] [-zh]

- 选项:
  - -h: 打印命令使用方法。
  - -V: 打印景行资源管理与调度软件发行版本信息。
  - -all: 显示所有错误码和相应的错误码释义。
  - -e code [code ...]: 显示作业错误码和相应的错误码释义。
  - -r code [code ...]: 显示 gui 图形作业错误码和相应的错误码释义。
  - -p code [code ...]: 显示作业挂起原因错误码和相应的错误码释义。
  - -zh: 显示挂起原因的中文释义。
- 注意:

设置 export LANG=zh_CN 环境变量后，再使用 jcode 命令可查看中文错误码释义。

- 命令示例:

```
# export LANG=zh_CN
# jcode -e 10036
10036: 许可证失败
# jcode -r 2001 2002 2006 2009
2001: Gui 作业失败次数达到上限
```

2002: Gui 作业信息读取错误

2006: Gui 作业因初始化失败而退出

2009: Gui 作业申请会话失败

## 8.2 Linux 常用命令说明

### 8.2.1 基础快捷键使用

ctrl + c: 停止进程

ctrl + l: 清屏

ctrl + r: 搜索历史命令

### 8.2.2 命令

#### 8.2.2.1 ls（列出目录内容）

**功能**：列出目录内容。

**语法**：ls [选项] [目录或文件]

**常用选项**：

| 选项 | 说明 |
| --- | --- |
| -a | 显示所有文件，包括隐藏文 |
| -c | 按文件修改时间排序 |
| -t | 按修改时间排序（最近优先 |
| -h | 以 K、M、G 为单位显示文件 |
| -1 | 以长格式显示文件详细信息 |

#### 8.2.2.2 pwd（显示当前目录）

功能：显示当前目录

#### 8.2.2.3 cd（切换目录）

**功能**：改变当前工作目录。

**语法**：cd [目录]

**说明**：

- cd 快速回到用户主目录
- cd - 回到上一次所在的目录

#### 8.2.2.4 chmod（修改文件权限）

**功能**：修改文件或目录的读写执行权限。

**语法**：chmod [选项] 文件名

**权限表示**：

| 符号 | 含义 |
| --- | --- |
| r | 可读（值为 4） |
| w | 可写（值为 2） |
| x | 可执行（值为 1） |

**示例**：chmod 660 test（将 test 文件权限设为 rw-rw----）

#### 8.2.2.5 mkdir（**创建目录**）

**功能**：创建一个新目录。

**语法**：mkdir [选项] 目录名

**常用选项**：

- -p：自动创建父目录（如 mkdir -p a/b/c）

#### 8.2.2.6 cp（**复制文件或目录**）

**功能**：复制文件或目录。

**语法**：cp [选项] 源文件 目标文件

**常用选项**：

- -r：递归复制整个目录

#### 8.2.2.7 cat（**显示文件内容**）

**功能**：显示文件内容（concatenate 的缩写）。

**语法**：cat [选项] 文件名

**常用选项**：

| 选项 | 说明 |
| --- | --- |
| -n | 显示行号 |
| -b | 显示行号，但空白行不编号 |
| -s | 将连续多行空白压缩为一行 |

#### 8.2.2.8 mv（移动或重命名文件）

**功能**：移动文件或目录，也可用于重命名。

**语法**：mv [选项] 源文件 目标文件

#### 8.2.2.9 rm（删除文件或目录）

**功能**：删除一个或多个文件或目录。

**语法**：rm [选项] 文件...

**常用选项**：

| 选项 | 说明 |
| --- | --- |
| -f | 强制删除，不提示确认 |
| -r | 递归删除目录及其内容 |

#### 8.2.2.10 find（查找文件）

**功能**：在指定目录下查找文件。

**语法**：find [路径] [选项] [操作]

**示例**：find ~/ -name "*.log"（在家目录下查找所有.log 后缀的文件）

### 8.2.2.11 more（分屏显示内容）

**功能**：分屏显示文件内容。

**语法**：more [选项] [文件名]

**常用选项**：

| 选项 | 说明 |
| --- | --- |
| -p | 清屏后显示内容 |
| -+num | 从第 num 行开始显示 |
| -s | 将多行空白压缩为一行 |

**示例**：cat aa.txt | more（通过管道分屏显示）

### 8.2.2.12 grep（搜索字符串）

**功能**：在文件中搜索指定的字符串

**语法**：grep [选项] "字符串" [文件名]

**常用选项**：

| 选项 | 说明 |
| --- | --- |
| -i | 忽略大小写 |
| -n | 显示行号 |

| -v | 显示不匹配的行 |
|---|---|

**示例**：grep -E "exe|dll" aa.txt

### 8.2.2.13 gcc （C 语言编译器）

**功能**：编译 C 语言源代码。

**语法**：gcc [选项] [参数]

**常用选项**：

| 选项 | 说明 |
|---|---|
| -o | 指定输出文件名 |
| -E | 仅执行预处理 |
| -S | 将 C 代码转换为汇编代码 |
| -c | 仅编译，不链接 |
| -Wall | 显示所有警告信息 |

**示例**：gcc test.c -o test（将 test.c 编译为可执行文件 test）

### 8.2.2.14 tar（打包与解包）

**功能**：为文件或目录创建档案（打包/压缩/解压）。

**语法**：tar [选项] [文件或目录]

**常用选项**：

| 选项 | 说明 |
| --- | --- |
| -c | 创建新的打包文件 |
| -x | 解包文件 |
| -v | 显示执行过程 |
| -f | 指定打包文件名 |
| -z | 通过 gzip 压缩或解压 |
| -j | 通过 bzip2 压缩或解压 |
| -C | 指定解压目标目录 |

**示例:**

```
# 打包
tar -cvf log.tar log2012.log          # 仅打包，不压缩
tar -zcvf log.tar.gz log2012.log      # 打包并 gzip 压缩
tar -jcvf log.tar.bz2 log2012.log     # 打包并 bzip2 压缩
# 解压
tar -zxvf /opt/soft/test/log.tar.gz  # 解压 gzip 包
```

### 8.2.2.15 vim 使用说明

Vim 共有三种模式：命令模式、输入模式、底线命令模式。

#### 1. 命令模式

启动 Vim 后默认进入命令模式。此时键盘输入被识别为命令，而非字符。

**常用命令:**

| 命令 | 说明 |
| --- | --- |
| i | 切换到输入模式 |
| x | 删除光标所在处的字符 |
| : | 切换到底线命令模式 |

#### 2. 输入模式

在命令模式下按 i 进入输入模式，可正常编辑文本。

**常用按键:**

| 按键 | 说明 |
| --- | --- |
| 字符键 | 输入字符 |
| ENTER | 换行 |
| BACK SPACE | 删除光标前一个字符 |
| DEL | 删除光标后一个字符 |

| | |
|---|---|
| 方向键 | 移动光标 |
| HOME/END | 移动到行首/行尾 |
| Page Up/Page Down | 上下翻页 |
| ESC | 退出输入模式，返回命令模式 |

### 3. 底线命令模式

在命令模式下按：进入底线命令模式，可执行保存、退出等操作。

**常用命令:**

| 命令 | 说明 |
|---|---|
| :q | 退出 |
| :w | 保存 |
| :wq | 保存并退出 |
| :q! | 强制退出（不保存） |

按 ESC 键可退出底线命令模式。

## 第九章 第九章 常见问题

### 9.1 启动 JHFileClient 超时

高算平台上传文件时如果遇到“启动 JHFileClient 超时”的问题，可以尝试:

检查 JHFileClient 的安装路径，路径中不要存在中文;

或改用 xshell 上传文件。

### 9.2 mpi 程序的两个 slots 在 cu01 上、两个 slots 在 cu02 上

我想要我的 mpi 程序的两个 slots 在 cu01 上、两个 slots 在 cu02 上，我应当怎样指定呢？

```
[jhadmin@mu03 ~]$ jsub -q normal -n 4 -m "cu15 cu16" -R "span[ptile=2]" sleep 30
Job <73385> is submitted to queue <normal>
[jhadmin@mu03 ~]$ jjobs
JOBID      USER     STAT  QUEUE     FROM_HOST EXEC_HOST JOB_NAME     SUBMIT_TIME
56503      jhadmin  RUN   views_li* mu01      cu01      _job         Nov 03 16:19
73098      jhadmin  RUN   views_li* mu01      cu17      _job         Mar 19 18:39
73385      jhadmin  PEND  normal    mu03                sleep 30     Mar 22 11:05
[jhadmin@mu03 ~]$ jjobs
JOBID      USER     STAT  QUEUE     FROM_HOST EXEC_HOST JOB_NAME     SUBMIT_TIME
56503      jhadmin  RUN   views_li* mu01      cu01      _job         Nov 03 16:19
73098      jhadmin  RUN   views_li* mu01      cu17      _job         Mar 19 18:39
73385      jhadmin  RUN   normal    mu03      2*cu16    sleep 30     Mar 22 11:05
                                    2*cu15
```

添加-m 和-R [ptile=n]参数就可以实现这个情况

ptile=n 表示每个主机应当给作业分配 n 个处理器

### 9.3 关于修改密码

1. 在网页修改，如图:




注意：超算不能用 passwd 修改登录密码是因为 passwd 修改的是 linux 系统下，但是我们的用户认证和密码都是在 windows 服务器的域控里，passwd 修改不到，只能用网页修改

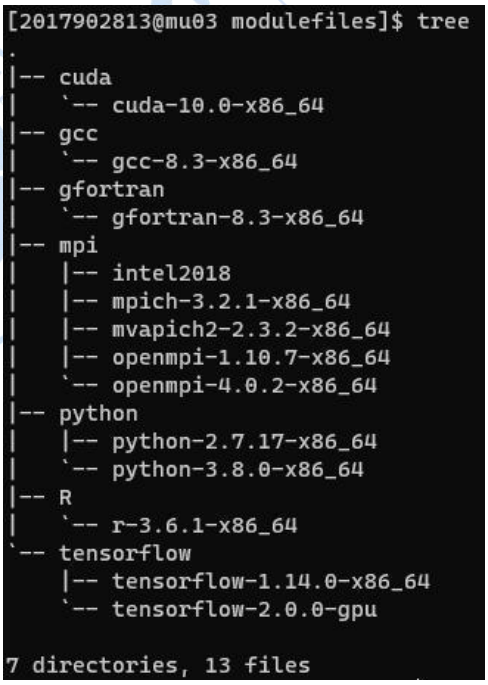
2.如果忘记旧密码可以通过“一网通办”-“高性能计算平台密码修改”在线服务或联系管理员修改密码。

**9.4 请问这个 cpu 数量是随机分配的吗？计算速度主要是根据什么来确定？怎么感觉算得还没自己的电脑快呢？**

CPU 数使用-n 指定。可以检查一下自己的程序是否能使用多核心。核并不是越多越快，核数增多通信量增大反而会增加运算时间。多核是需要自己本身编写的多线程程序才能真正发挥最大优势

**9.5 cuda 的安装路径是什么？**

各个模块应该都可以去/etc/modulefiles 里面找配置信息.



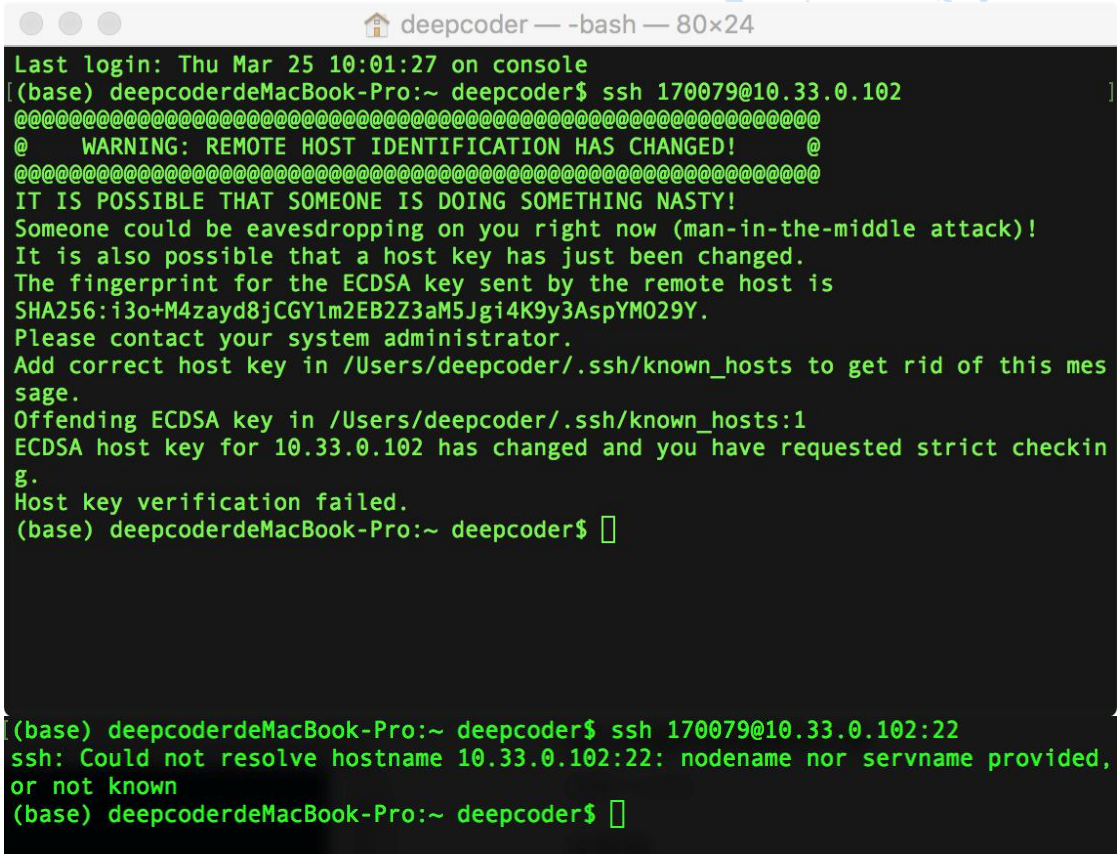

cuda 是装在 gpu 节点的，mu03 可以看到路径，但是运行的话要提交到 gpu 节点去运行

### 9.6 平台上没有集成我想用的软件

用户可以离线安装在自己的家目录下。在高算平台配置安装软件，对新手来说是不知道如何下手。我校张军老师做了一个手册：大家可以参考

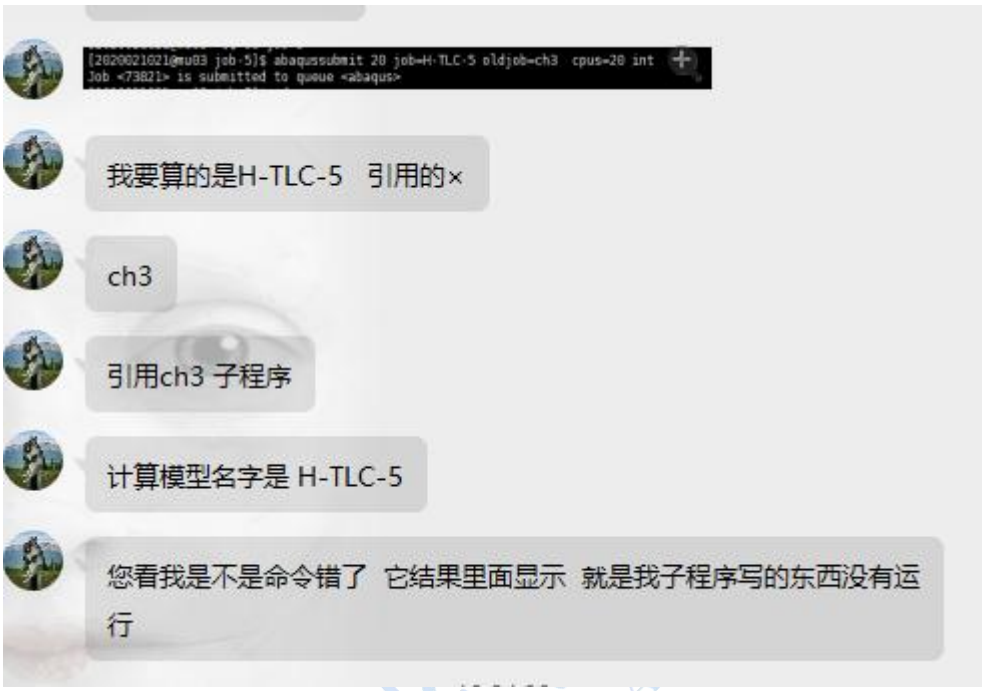
https://blog.csdn.net/jaz/article/details/156393312?sharetype=blogdetail&sharerId=156393312&sharerefer=PC&sharesource=jaz&spm=1011.2480.3001.8118

### 9.7 以前可以登录，现在 ssh 无法登录


deepcoder — -bash — 80×24
Last login: Thu Mar 25 10:01:27 on console
(base) deepcoderdeMacBook-Pro:~ deepcoder$ ssh 170079@10.33.0.102
@@@

## 9.8 Abaqus 引用的子程序需要调整为指定格式再引用吗

```
[2020021021@mu03 job-5]$ abaqussubmit 20 job=H-TLC-5 oldjob=ch3  cpus=20 int
Job <73821> is submitted to queue <abaqus>
```




解决方法：

修改了提交命令

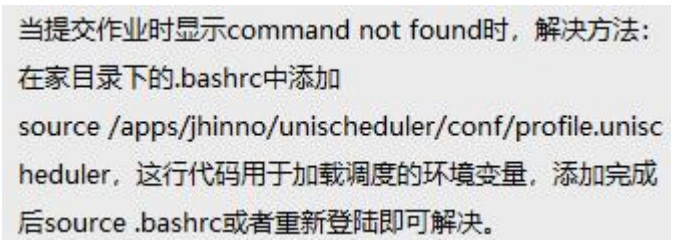



第一种方法：
1. 建立工作目录/ ]"
2. 将 ABAQUS 安装目录\6.4-pr11\site 下的 aba_param_dp.inc 或 aba_param_sp.inc 拷贝到工作目录，并改名为 aba_param.inc；
3. 将编译的 fortran 程序拷贝到工作目录；
4. 将.obj 文件拷贝到工作目录；
5. 建立好输入文件.inp；
6. 运行 abaqus job=inp_name user=fortran name即可。

第二种方法：
在 Job 模块里，创建工作，在 Edit Job对话框中选择 General 选项卡，在 User subroutine file 中点击 Select 按钮，从弹出对话框中选择你要调用的子程序文件（后缀为.for 或.f）。

```
[jhadmin@mu03 job-3]$ abaqussubmit 10 job=H-TLC-1 user=ch3 cpus=10 int
```

### 9.9 bashrc 设置环境变量后，提交作业提醒命令找不到




遇到这种情况，在脚本中再次手动添加命令的绝对路径，再次尝试，如下所示:

```
export  R_PATH=/data/users/CHDHPC/2020035003/R-4.0.3/bin

Rscript ~/test.R
```

bin 目录下存在 Rscript 运行命令，在脚本第一行继续声明环境变量，即可解决上述问题。

### 9.10 sudo 无法获得管理员权限

对于用户家目录下所有文件夹及其文件，用户都自身持有 root 权限，无需再获得 sudo 权限，但请注意，数据无价，谨慎删除。

### 9.11 CPU 核数设置及使用问题

有同学提问提交作业的时候用了 8 个 CPU 核，CPU 利用率是 150%左右，是不是最高能达到 800%?

解答：举个例子，你核数是 8 个，你如果在程序（一个进程）里有 8 个线程，那么这个利用率就将会是 800%，

因为每个核跑一个线程，但是如果你的线程是 1，那也只是一个核在使用。

问题：如果我占用了 6 个 CPU，别人可以使用这 6 个 CPU 吗，还是只能我一个人使用？

解答：你提交作业，分配给你，就一直跑到程序结束。

### 9.12 自己安装的软件环境设置问题

解答：自己安装软件后，请务必设置家目录下的环境变量。

如果软件要使用 GPU，那就提交到 GPU 节点。如果只是普通的数据处理，那就提交到 Normal 节点即可。但是自己搭建的环境，不要提交作业的 TensorFlow 队列中。

### 9.13 para 队列默认自动加载 openmpi

若要使用 regcmMPI，则需要提交到 qgpu 队列

### 9.14 GPU 显存不够用问题

GPU 节点使用 nvidia-smi 命令，查看 GPU 各个编号的显存使用。JSUB 一般默认提交到 0 号，如果训练任务显示显存不够用，在 python 代码第一行设置

```
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "2"# 0, 1, 2, ...
```

重新提交任务即可！

### 9.15 lib64 动态链接库文件引用错误

报错信息一般类似于:

ImportError: /lib64/libstdc++.so.6: version `CXXABI_1.3.9' not found

解决方案为:

在 bashrc 配置文件中配置添加动态链接库路径，此处以添加 anaconda 中已有的链接库为例:

在用户环境下配置环境，首先打开配置文件:

vi ~/.bashrc

点击 insert 将光标移动到 export 的位置下一行，输入下列一行配置

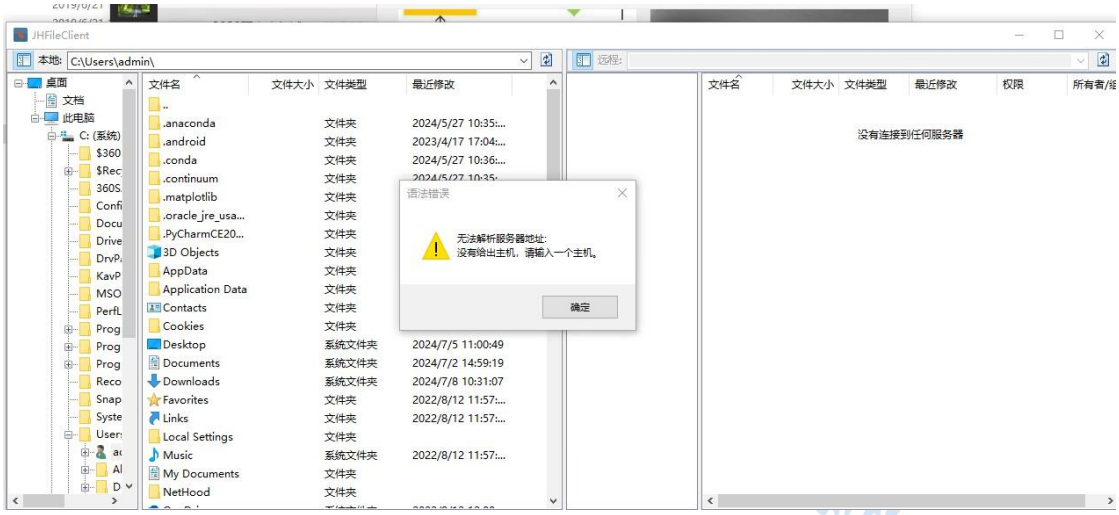
export LD_LIBRARY_PATH="/data/users/CHDHPC/学号/software/anaconda3/lib:$LD_LIBRARY_PATH"

点击返回键 Esc，输入: wq 保存退出

输入下列命令，激活环境变量

source ~/.bashrc

## 9.16 文件传输客户端无法使用




出现类似问题大概率是杀毒软件没配置软件白名单，客户端启动程序被误删除了。

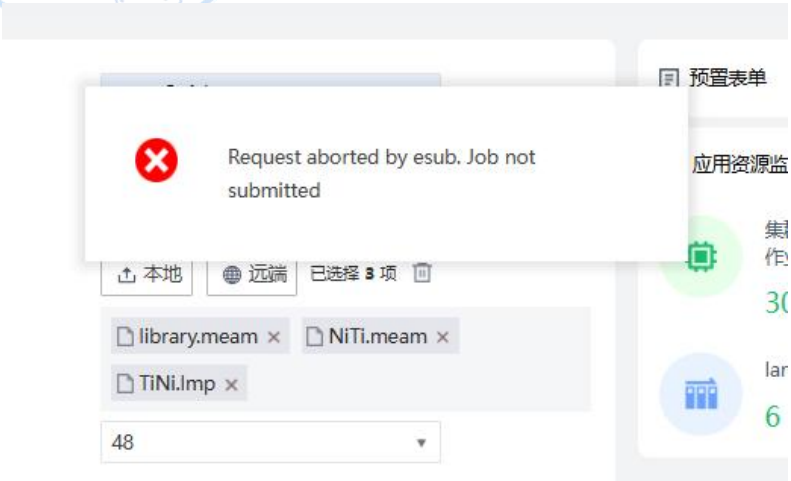
需要检查杀毒软件白名单，并将客户端加入进去，重新安装客户端即可正常使用。

## 9.17 上传下载打不开的原因

1）浏览器的版本过老。

2）JH客户端是否安装，版本是否正确。

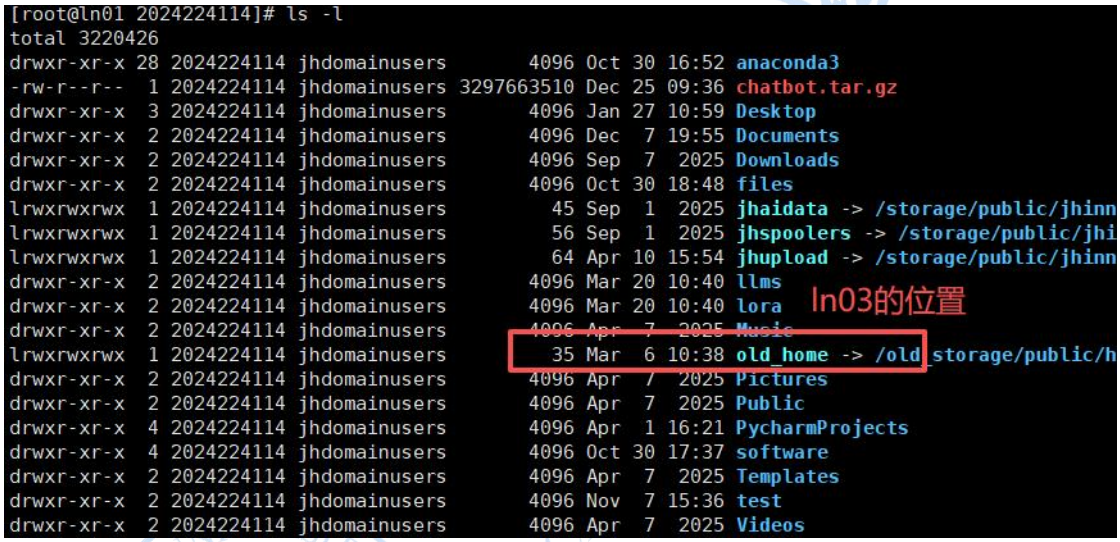
## 9.18 会话应用使用出现弹窗



目前门户上集成的软件只支持创新用户使用，笃学用户目前可以使用对应作业脚本进行作业提交。

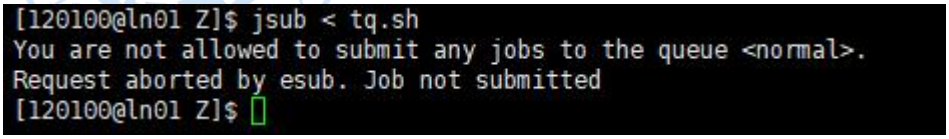
### 9.19 如何传文件到 ln03 笃学资源池的家目录

先通过网页端我的数据文件夹传文件，再通过 xshell 登录到 ln01 节点后，输入 ls -l 可以看到 old_home 文件夹。可以通过 mv 或者 cp 命令将文件移动过去。




参考命令为：mv 源文件夹 目标路径

### 9.20 提交作业显示 You are not allowed to submit



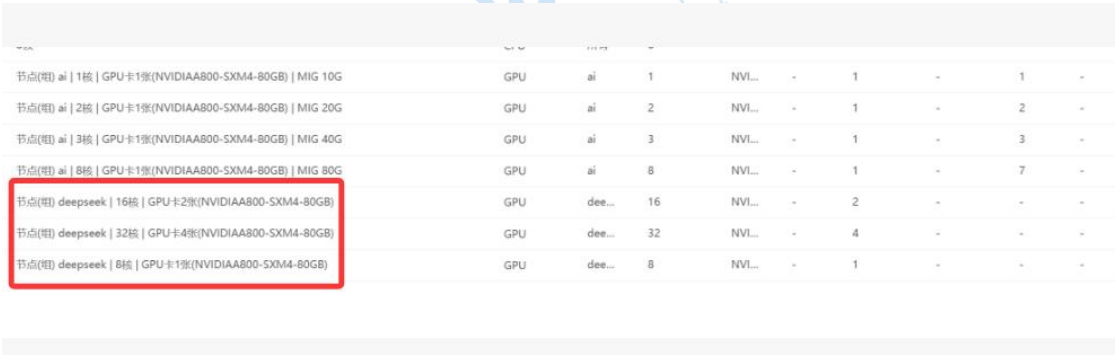

说明目前是以笃学用户的身份提交到创新用户专用节点了，群文件里有笃学用户可以提交的节点，比如 normal_old、common。切换到这些节点再提交试试。

### 9.21 GPU 作业在等待和运行中循环往复

该节点存在部分作业 mig 运行异常，导致无法被回收，只有等目前该节点作业运行完成后结束并重置。对于用户，可以在脚本里指定到其他节点运行作业，但是作业可能仍需等待。

造成这个问题的原因通常是有人使用了没有分配给自己的 gpu 卡，那个卡创建不了 mig 导致，只能把那个非法使用 gpu 的进程杀掉。所以大家不能在程序里面写死使用某一个 gpu ID 创建 mig，GPU 分配是调度分配使用，会合理安排的。

### 9.22 ai_service 作业一直等待




请不要选择 DeepSeek 组，该组目前部署了大模型，是无法排到用户 ai 作业的。

目前已将 DeepSeek 组选项隐藏。

### 9.23 申请创新用户但是平台却显示普通用户

用户只能在自己那里看到用户角色，这个只有管理员和普通用户之分。只有用户级别才分创新用户和笃学用户。

可以通过如下方法判断自己到底是不是创新用户：1. 在ln01提交测试代码交到创新用户资源池节点； 2. 直接问群里的管理员

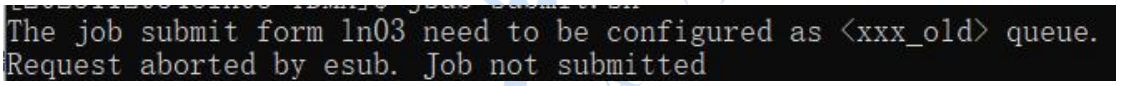
### 9.24 会话打不开

目前平台对会话数做了处理，一个用户最多只能打开两个会话，第三个会话会处于等待状态。

### 9.25 运行机器学习作业出现glibc相关的报错

nonmig队列的a100gpu01的glibc版本较低，无法兼容运行此类程序。请将作业指定到节点a40gpu01或者a40gpu02。具体各服务器glibc版本请查看使用手册。

### 9.26 指定了ln03和对应节点提交作业报错




这种情况是作业脚本有误，提交的队列不是对应资源池的队列。

### 9.27 安装cuda时遇到权限问题

安装的cuda环境应该装在自己的家目录，指定参数装在自己的目录。

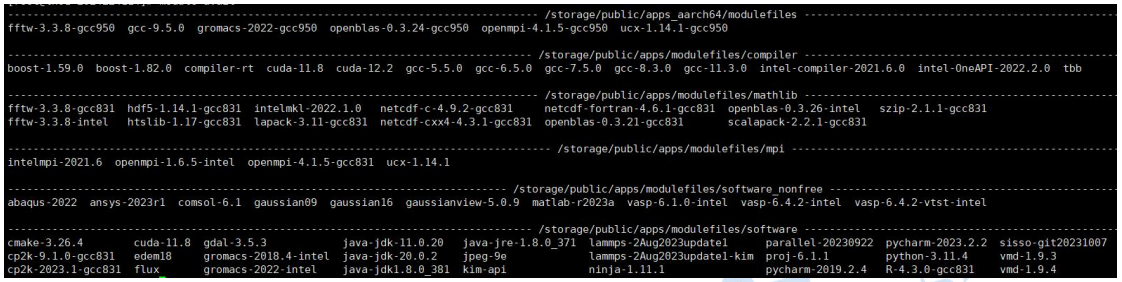
### 9.28 下载安装客户端组件后，还是要求下载

电脑的安全防护软件阻止了客户端导致其无法打开，请关闭后台杀毒软件再尝试安装。

### 9.29 用ln01节点想装MATLAB，但是运行安装程序显

**示没有权限**

平台安装任何应用都需要管理员权限，直接使用平台软件或者输入 module avail 查看自己需要的模块，再通过命令提交即可。请注意 1n03 和 1n01 可用的模块有区别，下图是 1n01 可用的。




比如我需要使用 vasp，可参考脚本为：

```
#!/bin/bash
#JSUB -q normal
#JSUB -n 32
#JSUB -e error.%J
#JSUB -o output.%J
#JSUB -J test_vasp
source /storage/public/share/jhjob_scripts/getJobHosts.sh
## forintelmpi :
module purge
module load vasp-6.1.0-intel
mpirun -machinefile $nodelist -np $ncpu -ppn 1 vasp_std
```