西北农林科技大学

高性能计算平台调度系统使用手册

2024 年 4 月 10 日

网络与教育技术中心

高性能计算平台

目　录

第一章 高算平台介绍

“一网通办”或者 “微信企业号” 搜索 “高算平台开户申请”。按提示办理开户。

西北农林科技大学新高性能计算平台由 2 台登录节点、2 台管理节点、195 台双路计算节点、7 台大内存四路计算节点、3 台 8 卡 GPU 计算节点组成，集群采用 HDR Infiniband 网络，并配置一套 10PB 统一命名空间的高性能并行文件存储系统。配置信息如下：

序号	设备类型	配　置
1	计算节点服务器	CPU 配置：英特尔® 至强® Platinum 8358（32C，2.6GHZ） 内存配置：512GB （16* 32G DDR4） 操作系统：RHEL8.3 glibc2.28
2	胖节点服务器	CPU 配置：英特尔® 至强® Platinum 8360（24C，3.0GHZ） 内存配置：3TB （48* 64G DDR4） 操作系统：RHEL8.3 glibc2.28
3	GPU 节点服务器	CPU 配置：英特尔® 至强® Platinum 8358（32C，2.6GHZ） 内存配置：1TB （32* 32G DDR4） GPU 配置：8* NVIDIA A800-80GB 操作系统：RHEL8.3 glibc2.28

4	登录服务器	CPU 配置：英特尔® 至强® Platinum 6326（16C，2.9GHZ） 内存配置：512GB（16* 32G DDR4） GPU 配置：1*NVIDIA A10 操作系统：RHEL8.3 glibc2.28
5	管理服务器	CPU 配置：英特尔® 至强® Platinum 6326（16C，2.9GHZ） 内存配置：16* 32G DDR4 操作系统：RHEL8.3 glibc2.28
6	计算网络	QM8790 200G IB 交换机
7	并行存储	高性能并行文件存储系统（10PB）

第二章 命令行使用方式

2.1. 平台登录方式

2.1.1 通过 ssh 登录

用户可通过第三方 ssh 工具（xshell、putty 等）登录高算平台。集群配置了 2 台登录节点，ssh 登录端口为 22，两台登录节点的 IP 地址分别为 172.16.115.106 和 172.16.115.107。两台登录节点没有任何区别。

2.1.2 通过 web 门户登录

用户可通过浏览器（推荐 firefox）登录景行集群门户，门户地址为 https://172.16.115.108，登录用户名和密码与 ssh 登录的账户信息一致，如下

图所示：

用户输入与 ssh 登录信息一致的账户和密码后即可登录门户页面，可通过网页的方式对自己的数据、作业进行管理。

2.2. 数据上传方式

用户可通过远程拷贝（scp、rsync 等命令）或通过第三方工具 winscp、sftp 的方式上传/下载数据。

2.3. 景行资源管理与调度软件简介

云计算资源管理：景行资源管理与调度软件可以将网络上多台异构类型的计算机资源整合为一个集群应用服务平台。应用程序资源的调用不再只局限于个人工作站的资源，也不需要为了满足个性化的资源需求而手动改动应用程序，只需要设置一些简单的应用脚本和命令就能够使用应用服务平台上的计算资源。景行资源管理与调度软件还可以根据节点主机的负载条件和应用程序的资源需求，从整个集群应用服务平台中选择最合适的计算节点。

执行作业：对于景行资源管理与调度软件管理的应用服务系统，远程和本机执行作业的行为一致。对于用户而言是开放和透明的，即使是最复杂的终端

交互控制作业，也如同作业是在本地执行似的。当作业获得所需要的合适的软硬件资源，或当应用服务平台的系统负载较轻时，景行资源管理与调度软件会根据预定的策略自动执行作业，并可以根据资源负载情况，对作业进行完全控制。如：挂起作业和恢复作业运行。景行资源管理与调度软件能够以交互式作业或批处理作业的形式执行串行或并行的应用程序。当作业在一组资源负载较轻或空闲的资源节点上执行时，作业的执行效率会大大提高。

控制系统资源的访问：对于景行资源管理与调度软件管理的应用服务系统，系统管理员可以轻易地控制资源的访问，例如：

- 谁可以提交作业，这些作业可以使用哪些主机；
- 某个用户或者某个用户组最多可以同时执行多少个作业，最多可以使用多少种应用资源；
- 提交到指定队列的作业的资源限制；
- 每个计算单元可执行作业的时间窗口；
- 在某种负载条件下指定的计算单元可以接收作业或者挂起一些低优先级的作业。
- 可对提交的用户作业，进行运行内存的限制等。

资源和作业记账：景行资源管理与调度软件提供了资源和作业记账的机制。这些信息可以帮助管理员分析资源的使用情况和作业的执行情况，以及系统在一天或者一周内的负载情况，同时帮助管理员确定是否有资源过载情况发生，为合理化系统配置或扩展和升级系统提供详尽的数据支持。

应用：绝大多数的应用都可以通过景行资源管理与调度软件的接口访问由其管理的应用服务系统。不需要直接与之发生交互，也不需要为了使用由景行

资源管理与调度软件管理的应用服务器系统而刻意的修改程序。目前，由景行资源管理与调度软件管理的应用服务系统，可支持大多数的 Linux 或者 Windows 的命令和第三方应用程序。

容错：一旦有计算资源请求，景行资源管理与调度软件会通过内置高效的调度策略寻找可用计算资源，并保证计算资源的请求被及时的派发后执行。在应用服务系统中，只要还有一台服务器在运行，景行资源管理与调度软件就能够继续接收计算资源请求。如果计算资源请求执行失败，景行资源管理与调度软件就会主动的把该请求重新派发到另一个可满足资源需求的可用服务器上执行。景行资源管理与调度软件将整个应用服务系统的状态保存在文件数据库中。只要该事务日志文件可访问，景行资源管理与调度软件就能保证执行所有的计算资源请求。并可为该事务日志文件设置镜像备份，以足够保证当主文件服务器停止工作时，景行资源管理与调度软件可以根据镜像的事务日志文件继续执行作业的操作，为整个集群应用服务系统提供了执行作业的容错能力。

异构系统的支持：景行资源管理与调度软件是连接不同操作系统的中心枢纽。其通用的架构使得景行资源管理与调度软件非常容易支持多种类型的操作系统。它不仅支持 Linux 和 Windows 操作系统，还可支持 Linux 和 Windows 之间的相互操作。

并行处理：景行资源管理与调度软件支持 MPI（Message Passing Interface）。它既是集群资源的分配者又是管理者，并可为每个并行作业模块寻找最佳适合的主机。

调度策略：景行资源管理与调度软件提供了快速、高效的调度策略，保证了受其管理的应用服务系统的正常运行。用户可以根据不同的需要使用不同的

策略，例如，用户可以设置队列级的公平共享调度策略，控制和管理对应用资源的需求冲突。景行资源管理与调度软件还内置了许多其它队列级别的调度策略，如最基本的先来先服务、抢占式和独占式策略。

资源预留：是指对某个作业或者队列强制预留资源。资源预留保证了正在运行的作业有足够的可用资源（通过对资源提前占位，可以很好的解决作业运行过程中因需求资源动态的变化而不够的问题）。

作业记账：应用服务系统记录作业的大量信息，比如说：

- 提交节点和执行节点。
- 提交、派发、执行作业和结束作业时间；
- 执行作业的资源开销；
- CPU 时间、作业整体周转时间和自然时间等；

所有这些数据都存储在一个作业记账文件中。

计费时间为作业开始执行到作业结束时间，排队时间不计费。

作业数组：作业数组延伸了作业的概念，作业从单个输入文件、单例执行的应用程序延伸为多个并行文件、多例执行的应用程序。许多现实世界的问题，如渲染一场动画或者是在批处理数据转换时，需要多次输入不同的数据参数来多次执行同一队列应用程序。使用景行资源管理与调度软件的作业数组功能，可以允许用户提交单个数组作业，而该数组作业可以使用不同的数据参数来多次执行同一应用程序。

共享资源：共享资源是指由景行资源管理与调度软件管理的集群中所有节点上的可用资源，这些资源可以在节点组之间共享。如应用软件的浮动许可证就是一个典型的可共享资源，集群中的所有节点都可以通过自动获取到的浮动

许可证来执行该软件。景行资源管理与调度软件保证了在作业派发到各执行节点前获得有效的许可证后执行作业，从而使得该浮动许可证资源在各执行节点间得到合理化的应用。

并行作业的处理器预留：在同一个景行资源管理与调度软件管理的集群中执行并行和普通应用程序时，因为普通作业只需要一个 CPU 而并行作业程序则需要等待多个空闲的 CPU，所以并行作业程序所需要的 CPU 总会被普通应用程序先占用。并行作业处理器的预留功能，允许并行作业在排队期间将所需的空闲处理器（作业槽 slots）预留一段时间而不被其它的普通作业使用。

Job Starter：每一个景行资源管理与调度软件队列都可以配置一个 Job Starter。Job Starter 是一个脚本或者是可执行程序，用来创建作业执行的环境。通过 Job Starter，景行资源管理与调度软件管理员可以自定义作业执行的环境。例如：

- 配置作业的输入/输出缓存和重定向
- 配置执行仿真作业的运行环境

可配置的作业控制方式：可通过景行资源管理与调度软件的作业控制，改变作业在集群系统中的运行状态。通常情况下，作业先是进入 PEND 状态，然后进入 RUN 状态，作业执行完成后显示为 DONE 状态。有时会在作业的生命周期内，被系统挂起进入 SSUSP 状态，或者是被用户挂起而进入 USUSP 状态。景行资源管理与调度软件给管理员提供了配置控制作业时所触发的一系列动作，当作业状态改变时，这些自定义的触发动作将会被执行。

基于数据库的调度框架：景行资源管理与调度软件提供了开放的基于数据库的调度框架。用户可以根据该框架的要求，定义设置调度策略，从而更高效

地利用集群的资源，实现对调度策略的深度定制。

GPU 调度：景行资源管理与调度软件提供了 GPU 调度功能，该功能会自动检测节点 GPU 信息，并将 GPU 信息管理起来，用于调度使用。GPU 调度支持两种模式，分别是基础 GPU 调度和 BIND GPU 调度，基础 GPU 调度是用户可以将 GPU 定义为资源，再写一个用来收集该自定义资源的 ELIM 脚本，这样就可以将 GPU 作为一种资源来调度。BIND GPU 调度是以基础 GPU 调度为基础，添加了给作业绑定 GPU 的功能，使作业独占被分配到的 GPU。

2.4. 景行资源管理与调度命令行使用方式

作业提交命令行说明：

```
jsub -n <CPU 数> -q <队列名> [-J <作业名> -o output.%J -e error.%J] ./commandline
```

-n <CPU 数> 指定执行该作业所需的 CPU 数

-q <队列名> 指定该作业运行时使用的队列,所有的作业都会被提交到队列里在资源不足时排队，在资源满足时派发并执行。队列由管理员分配使用，所以用户在提交作业时确认下自己是否有权限使用该队列(jqueues -u <自己账户名>来列出自己可用的队列)。

-J <作业名> 指定作业名,方便辨识作业

-o output.%J 指定输出文件，这个输出指的是调度系统的输出，如果作业的输出没有重定向到应用自己的 log 文件里也会输出在这个文件里。%J 表示的作业号，及最终会产生一个 output.<作业号>的文件。

作业查看

jjobs [[-a] [-l] <作业号>]

jjobs 不使用参数，显示目前正在排队或运行的作业

```
[jhtest@lnode251 linpack]$ jjobs
```

```
JOBID       USER       STAT   QUEUE      FROM_HOST  EXEC_HOST  JOB_NAME      SUBMIT_TIME     ORD
ER
12829       jhtest     RUN    normal     lnode251   64*cnode*  *_test_1node  Jul 04 14:24    -
```

jjobs -a　显示最近完成的作业及正在运行或排队的作业

```
[jhtest@lnode251 linpack]$ jjobs
JOBID       USER       STAT   QUEUE      FROM_HOST  EXEC_HOST  JOB_NAME      SUBMIT_TIME     ORD
ER
10572       jhtest     DONE   normal     lnode251   64*cnode*  *_test_1node  Jul 02 21:37    -
12829       jhtest     RUN    normal     lnode251   64*cnode*  *_test_1node  Jul 04 14:24    -
```

jjobs <作业号>　　显示指定的作业状态

```
[jhtest@lnode251 2nodes]$ jjobs 12829
JOBID       USER       STAT   QUEUE      FROM_HOST  EXEC_HOST  JOB_NAME      SUBMIT_TIME     ORD
ER
12829       jhtest     RUN    normal     lnode251   64*cnode*  *_test_1node  Jul 04 14:24    -
```

jjobs -l <作业号>　显示指定作业的详细信息

```
[jhtest@lnode251 2nodes]$ jjobs -l 12829
Job <12829>, Job Name <hpl_test_1node>, User <jhtest>, Project <default>, Statu
                     s <RUN>, Queue <normal>, Application <default>, APS Priori
                     ty <5000>, Command <#!/bin/bash;#JSUB -q normal;#JSUB -n 6
                     4;#JSUB -e error.%J;#JSUB -o output.%J;#JSUB -J hpl_test_1
                     node;###>>> The following content is generally not modifie
                     d. <<<###;ncpu=`echo $JH_HOSTS | awk '{print $2}'`;###>>>
                     The above content is generally not modified.       <<<###;##
                     # apps run command:;module load intel-OneAPI-2022.2.0;EXEC
                     =/storage/public/home/jhtest/linpack/xhpl_intel64_dynamic;
                     OMP_NUM_THREADS=32 mpirun -np 2 $EXEC>
Tue Jul 04 14:24:25: Submitted from host <lnode251>, CWD </storage/public/home/
                     jhtest/linpack/1node>, Output File <output.12829>, Error F
                     ile <error.12829>, 64 Processors Requested;
```

```
Tue Jul 04 14:24:25: Started on 64 Hosts/Processors <64*cnode122>, Execution Ho
                     me </storage/public/home/jhtest>, Execution CWD </storage/
                     public/home/jhtest/linpack/1node>, Execution user <jhtest>
                     ;
Tue Jul 04 14:29:20: Resource usage collected.
                     The CPU time used is 4183.74 seconds.
                     IDLE_RATE(cputime/slots/runtime): 0.2267
                     MEM:  476 Gbytes; SWAP: 0 Bbytes
                     PGIDs:1249876 1249961 1249965 1249966; PIDs:1249876
                     1249877 1249880 1249944 1249955 1249960 1249961 1249965
                     1249966


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


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

作业终止

jctrl kill <作业号>

```
[jhtest@lnode251 ~]$ jctrl kill 12829

Job <12829> is being terminated
```

2.4.1 **加载指定工具**

Module 是一个用于管理用户不同应用的工具，可以动态的加载、移除所安

装的应用。工具的 modulefiles 文件编写完毕后，module 工具就会识别，此时使用 module avail,即可显示所有可用模块：

```
[jhtest@lnode251 ~]$ module avail

------------ /storage/public/apps/modulefiles/compiler -----------------

compiler-rt  go-1.20.5  intel-compiler-2021.6.0  intel-OneAPI-2022.2.0  tbb

------------- /storage/public/apps/modulefiles/mathlib -----------------

fftw-3.3.8-gcc831  fftw-3.3.8-intel  gsl-2.7-gcc831  htslib-1.17-gcc831  intelmkl

-2022.1.0  openblas-0.3.21-gcc831  scalapack-2.2.1-gcc831

--------------- /storage/public/apps/modulefiles/mpi -------------------

intelmpi-2021.6  openmpi-4.1.5-gcc831  ucx-1.14.1

------------ /storage/public/apps/modulefiles/software -----------------

AdmixTools-7.0.2          beast-2.7.4                cmake-3.26.4

eig-7.2.1                 hic-pro-2.11.4             lastz-1.04.15

panacus-0.2.1_py310  samtools-1.9
```

使用 module load 可以加载模块，如 module load R-4.3.0-gcc831，就可以加载 R 工具：

```
[jhtest@lnode251 ~]$ module load R-4.3.0-gcc831

Loading R - 4.3.0 : /storage/public/apps/software/R/R-4.3.0/
```

查看是否加载成功：

```
[jhtest@lnode251 ~]$ which R

/storage/public/apps/software/R/R-4.3.0/bin/R
```

此时使用 module list 可以查看目前已经加载的工具：

```
[jhtest@lnode251 ~]$ module list
Currently Loaded Modulefiles:
 1) R-4.3.0-gcc831
```

使用 module rm 就可以取消加载工具，如 module rm R-4.3.0-gcc831 ，可以取消加载 R 工具：

```
[jhtest@lnode251 ~]$ module rm R-4.3.0-gcc831
Removing R - 4.3.0
Use 'module list' to view any remaining dependent modules.
```

如果想取消所有工具的加载，使用 module purge 命令。

注意，如果加载某个软件后，在实际运行时报错提示“xxx command not found”，这可能是由于环境变量冲突导致，需使用 module purge 命令清除冲突的环境变量，然后再使用 module load 命令加载某个软件的环境变量，命令举例如下：

```
[jhtest@lnode251 ~]$ module purge ; module load R-4.3.0-gcc831
```

2.4.2 提交通用作业

向一个队列提交作业，只要队列的状态是打开的，用户就可以向该队列提交。只有当队列是激活状态时，作业才能被派发执行。如果一个作业不属于并行作业，则提交到默认的队列中进行计算。

提交命令：

```
[jhtest@lnode251 ~]$ jsub -q normal -n 1 -m cnode1 -e error.%J -o output.%J -J my_job .
/test.sh
```

```
Job <12833> is submitted to queue <normal> and default application <default>
```

参数详解：

<jsub>：提交作业

<-q normal>：指定 normal 队列

<-n 1>：指定该作业所需 1 核

<-m cu01>：指定该作业运行在 cnode1 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>：正确信息输出到 output.%J 文件中

<-J my_job>：指定作业名字为 my_job

<test.sh>：执行的脚本

直到作业执行的所有条件都满足之前，作业一直处于等待状态。每个队列都有它自己的执行条件，该条件应用于队列中所有的作业。当用户提交作业时，可以指定其它附加的条件。

注意：如果待提交的作业需要在单台计算节点上执行（即不跨节点执行），则需在提交命令中加入“-R " span[hosts=1]"”关键字，无需通过“-m”参数指定特定节点，且申请的核心数不能大于队列中单台计算节点的物理核心数。命令举例如下：

```
[jhtest@lnode251 ~]$ jsub -q normal -n 64 -R " span[hosts=1]" -e error.%J -o output.%J -
J my_job ./test.sh
```

2.4.3 **提交 R 并行作业**

cd 到测试文件所在目录后，当前目录下存在 test.R（r 脚本）

```
[jhtest@lnode251 ~]$ jsub -n 64 -q normal -e error.%J -o output.%J 'module l
```

```
oad R-4.3.0-gcc831 ; Rscript ./test.R'
```

2.4.4 提交 mpi 并行作业

cd 到测试文件所在的目录后，当前目录下存在 test 脚本 (脚本中为 mpi 命令）；

在 test 脚本中 mpirun 或 mpiexec 命令行的上一行添加 mpi 环境变量加载命令；

集群中提供了两种 mpi，分别为 Intel 编译器套件中的 IntelMPI 和开源的 OpenMPI，可分别通过如下命令加载：

```
[jhtest@lnode251 ~]$ module load openmpi-4.1.5-gcc831

[jhtest@lnode251 ~]$ module load intelmpi-2021.6
```

提交作业：

```
[jhtest@lnode251 ~]$ jsub -n 64 -q normal -e error.%J -o output.%J ./test
```

2.4.5 提交 tensorflow 作业

cd 到测试文件所在的目录后，当前目录下存在 test.py

提交到 cpu 队列

```
[jhtest@lnode251 ~]$ jsub -q normal -e error.%J -o output.%J test.py
```

提交到 gpu 队列

```
[jhtest@lnode251 ~]$ jsub -q gpu -e error.%J -o output.%J test.py
```

2.4.6 提交 gpu 通用作业

作业提交到 gpu 队列，举例如下：

```
[jhtest@lnode251 ~]$ jsub -q gpu -n 2 -gpgpu "1 mig=1" -e error.%J -o outp
ut.%J test.py
```

重要参数说明：

- **-n 2** ： 可选，表示申请 2 个 CPU；
- **-q gpu** ： 必选，无需修改，表示作业在 gpu 队列中的节点执行；
- **-gpgpu "1 mig=1"** ： 必选，根据作业需求修改，第一个 **1** 表示申请 **1** 个 GPU 卡，第二个 **1** 表示申请 gpu 类型为 **1** 的 gpu（9728MiB 显存），具体 gpu 类型说明如下：

GPU 类型（int）	GPU 显存（MiB）	SM 计算单元（int）
1	9728	14
2	19968	28
3	40448	42
4	40448	56
7	81228	98

单个 A800GPU 支持虚拟为 7 个 GPU 类型（mig）为 1 的 GPU 实例（显存 9728MiB），也支持虚拟为 1 个 GPU 类型为 7 的 GPU 实例（整张卡，即显存为 81228MiB）。请根据作业规模评估所需的 GPU 数量和显存大小，切勿浪费，一般作业 GPU 类型为 1 即可。

作业需要两个 mig 类型为 3 的 GPU 卡资源的提交参数应为： -gpgpu “2 mig=3”；

作业需要一整张 GPU 卡资源的提交参数应为： -gpgpu “1 mig=7”

2.4.7 提交 docker 通用作业

作业提交到 docker 队列，举例如下：

```
[jhtest@lnode251 ~]$ jsub -q docker -n 2 -e error.%J -o output.%J testDocker.sh
```

其中 testDocker.sh 为作业执行脚本，脚本中包含了 docker 作业执行的命令，脚本示例如下：

```
#!/bin/bash
docker run --rm \
-v ${HOME}/data:/data \
-w /data \
jhinno/pytorch:py37-1.10.0 \
your_command
```

docker 运行命令需参考 docker 相关手册，上述脚本中的-v 指定挂载路径（主目录下/data 挂载到 docker 中的/data），-w 指定命令执行路径（命令执行路径为/data），jhinno/pytorch:py37-1.10.0 为 docker 镜像（命令空间：jhinno，镜像名称：pytorch，镜像版本：py37-1.10.0），docker 镜像可通过 web 门户中的镜像仓库进行管理，具体可参考本平台的人工智能使用手册。

注意：脚本中的 your_command 中如果有与运行核心数的相关参数，则核心数参数不能大于作业申请的核心数。

第三章 WEB 使用方式

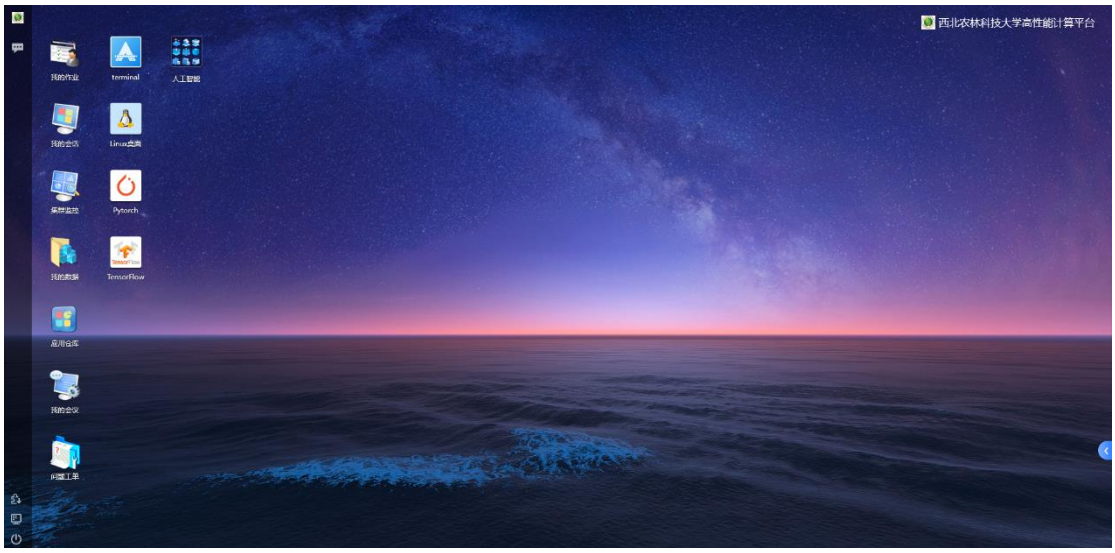
3.1 文件传输工具安装介绍

jhfileclient 主要是用来做文件的上传、下载。可以上传几十 G 的大文件，并且可以实现断点续传。访问 https://172.16.115.108 进入高算平台的登录页面，在页面上下载客户端组件并安装。

3.3 登录

访问 https://172.16.115.108 进入高算平台的登录页面，如下：

输入用户名密码登录，登录进去的主页面如下：

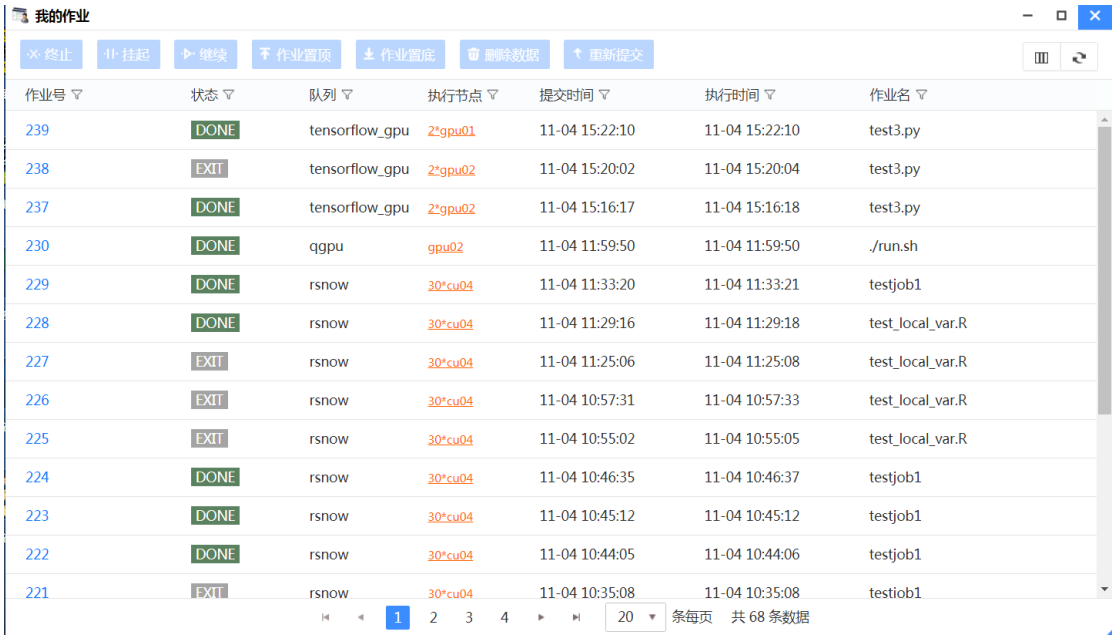


3.4 作业管理使用

作业管理主要是指查看作业信息、挂起作业、继续作业、终止作业四个功能。还可以按作业号、状态、队列、提交时间、执行节点、作业名对作业进行排序。

管理员可以在作业管理页面查看到所有用户的作业，并对这些作业进行操作。而其他用户仅能在该页面上查看到自己提交的作业。

作业管理页面如图所示：



我的作业页面

作业管理页面上显示了以下元素：

（1） 作业号：默认显示；

（2） 状态：默认显示，其中作业的状态包含有：RUN、PEND、UNKWN、UNKNOW、PSUSP、USUSP、SSUSP、ZOMBI、DONE、EXIT；

（3） 队列：默认显示，显示作业运行的队列名称；

（4） 执行节点：默认显示；

（5） 提交时间：默认显示；

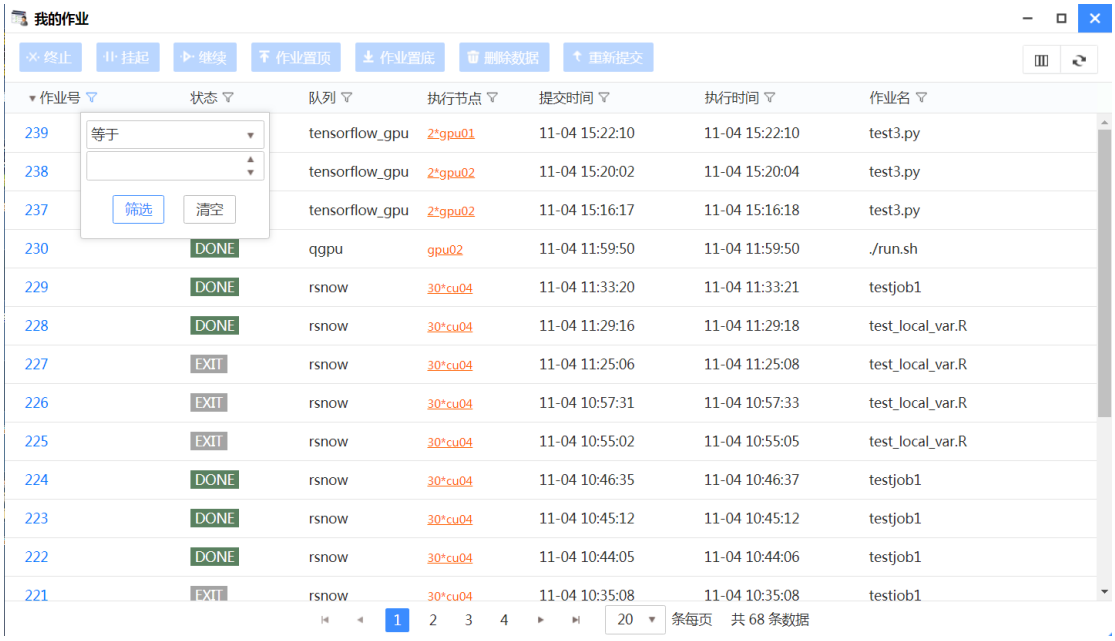
（6） 执行时间：显示的是作业执行的时间点；

（7） 作业名：显示的是作业的名称

作业管理页面对每一列提供了过滤设置，并支持个性化手动设置过滤条件。用户可以通过在每一列右边的漏斗状图标，就可以进行每一列元素的过滤。这些过滤条件会自动保存起来，下次访问该页面的时候，会执行自己设置的过滤条。

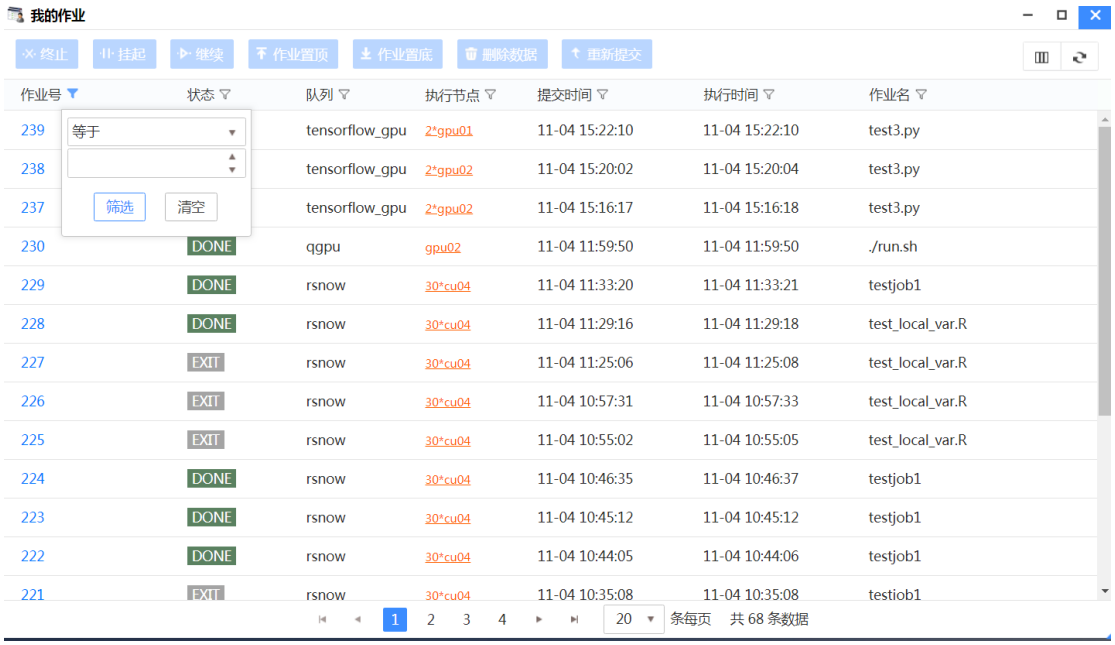
其中每一列的过滤条件有：

（1） 作业号：提供等于、大于、和小于三个过滤条件。过滤展开框如图所示：



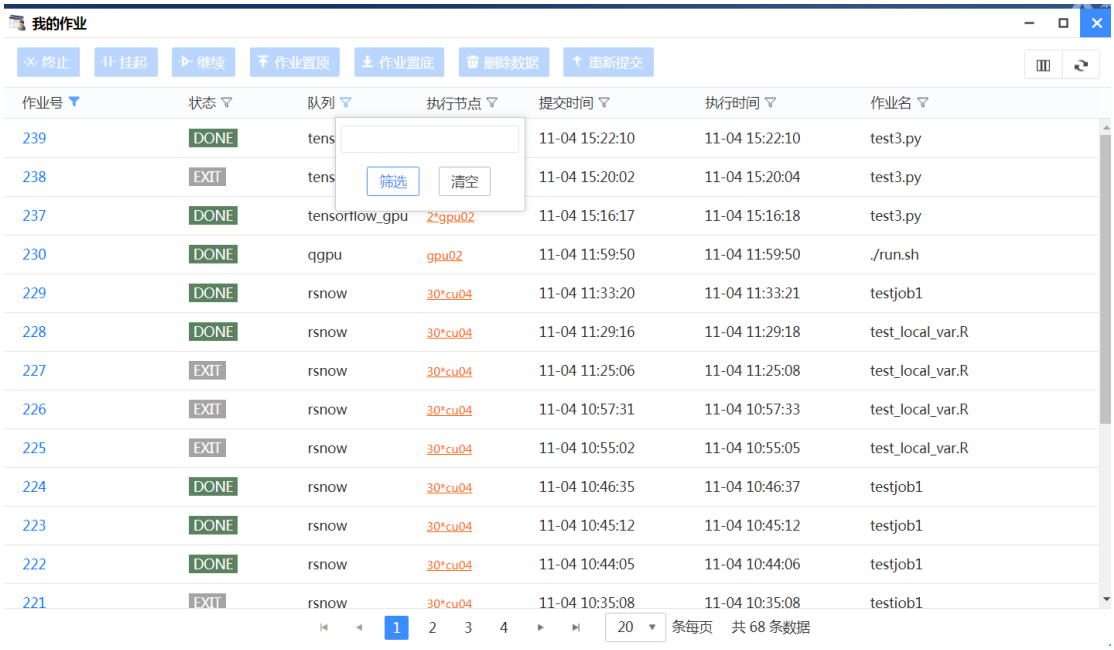
作业号过滤框（我的作业页面）

（2） 状态：提供等于、大于和小于三个过滤条件，并提供一个选择框选择作业号与两个过滤条件组合使用。过滤展开框如图所示：



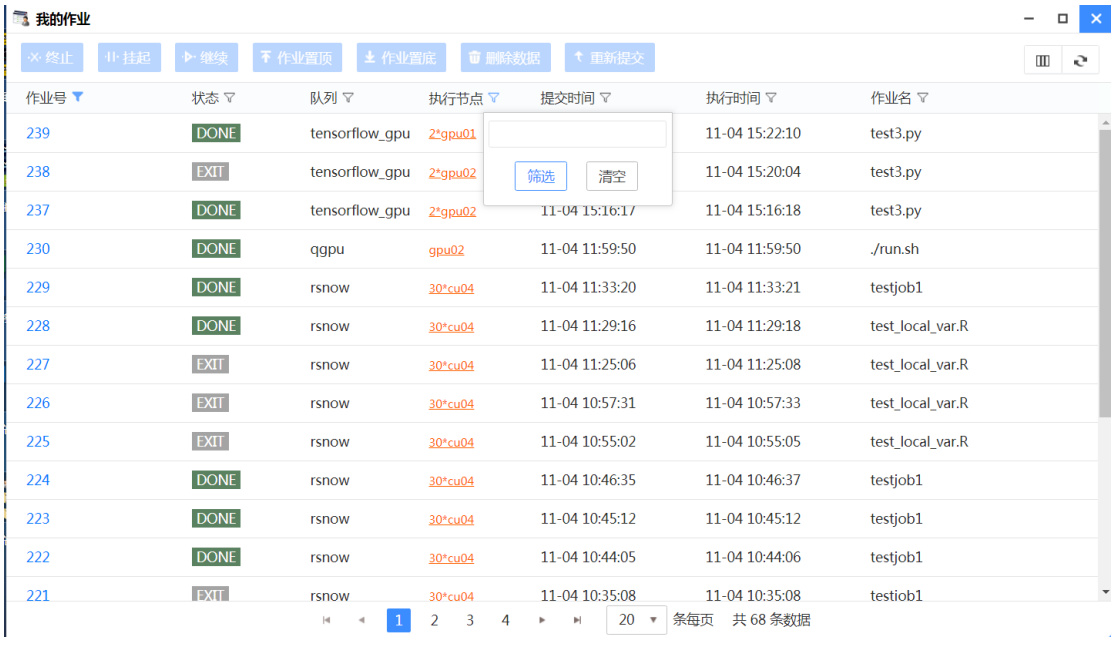
状态过滤框（我的作业页面）

（3） 队列：提供筛选的过滤条件。过滤展开框如图所示：



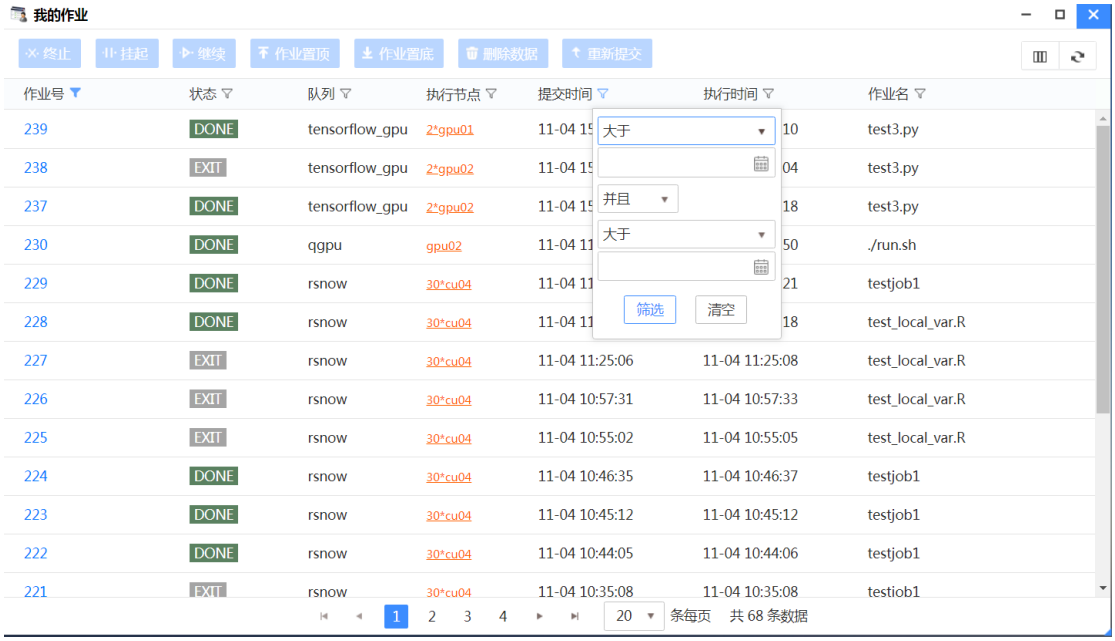
队列过滤框（我的作业页面）

（4） 执行节点：提供过滤设置，过滤展开框如图所示：



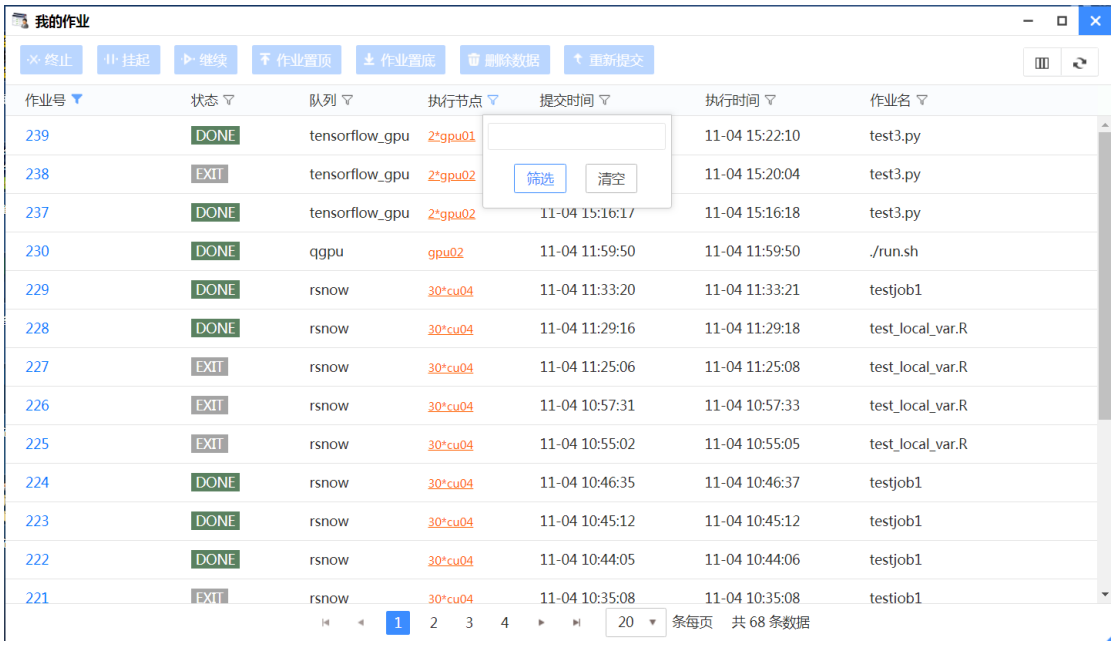
节点过滤框（我的作业页面）

（5） 提交时间：作业的提交时间的过滤条件。过滤展开框如图所示：



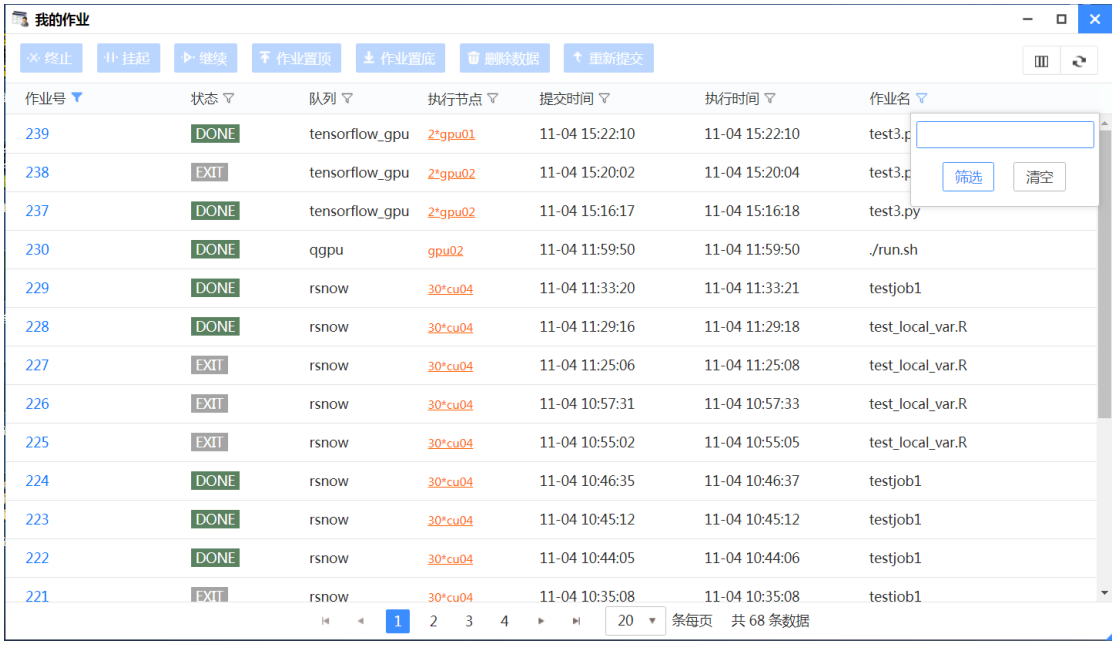
提交时间过滤框（我的作业页面）

（6） 执行时间：作业的执行时间的过滤条件。过滤展开框如图所示：



执行时间过滤框（我的作业页面）

（7） 作业名：作业名的过滤条件。过滤展开框如图所示：

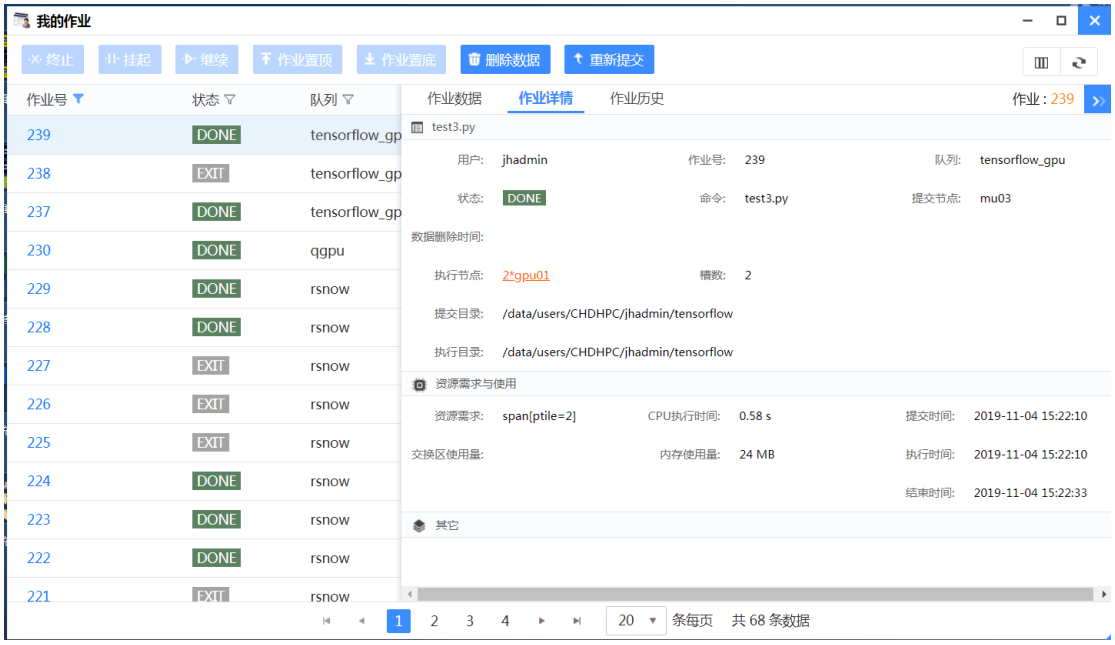


作业名过滤框（我的作业页面）

下面将详细介绍作业管理的四个主要功能：查看作业信息、挂起作业、继续作业、终止作业

3.5.1 查看作业信息

作业信息主要包括用户、作业号、队列、状态、命令、提交节点、执行节点、提交目录、执行目录、作业槽数，提交时间、执行时间、结束时间、CPU 执行时间、作业历史信息、作业输出等。作业信息页面如图所示：



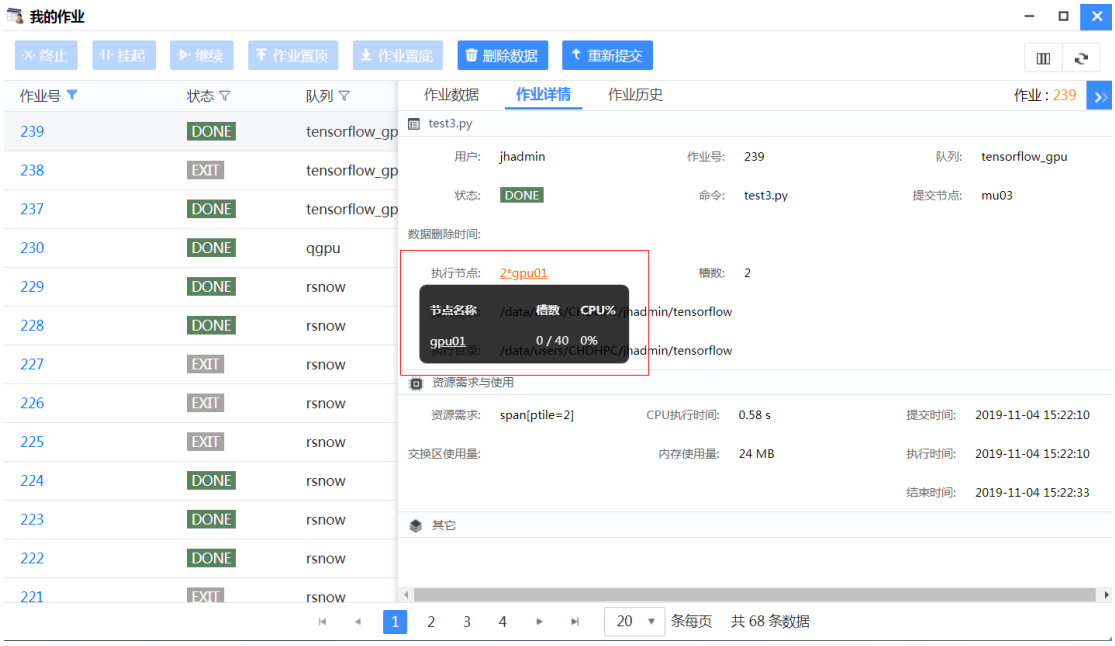
作业信息页面

作业详细信息页面统计了作业的资源需求与使用，以及作业的执行情况。详细信息页面统计了以下几项：

- 作业号
- 作业名
- 用户：作业的执行用户。
- 队列：作业的执行队列
- 项目：作业的项目名
- 状态：作业的实时状态
- 命令：作业的执行命令
- 提交节点/提交目录
- 执行节点/执行目录
- 作业槽数：若是 RUN 状态的作业，该参数指的是作业占用的槽数，若是 PEND 状态即为作业执行所需要的槽数。

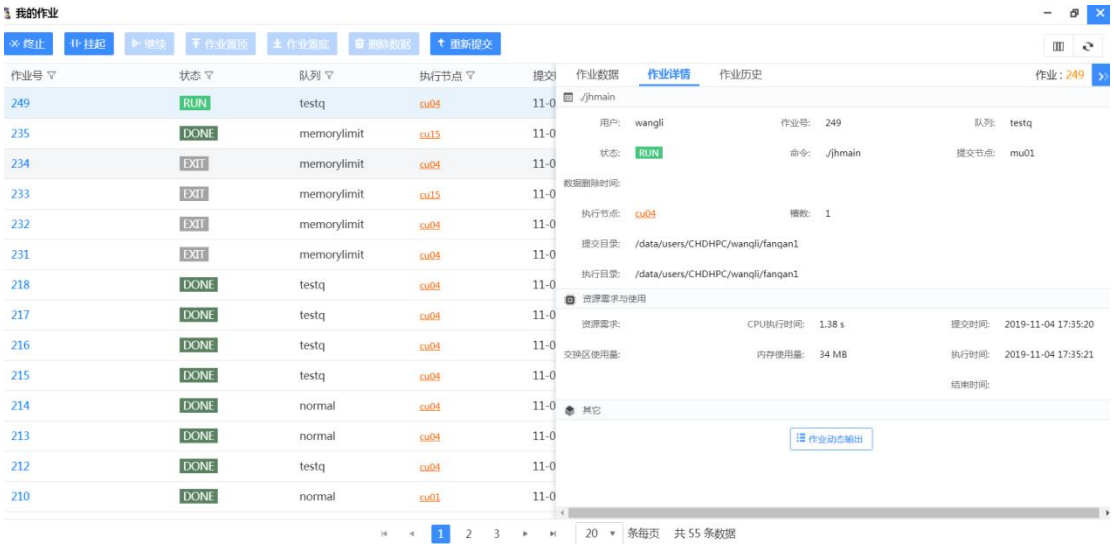
- 资源需求：作业提交的请求资源串。
- CPU 执行时间：作业执行完成后所使用的 CPU 时间。在作业执行完成后才显示。
- 内存/交换区使用量：作业执行完成后所使用的内存和交换区使用。在作业执行完成后才显示。
- 提交/执行/结束时间

点击执行节点，会显示该节点的机器状态，如图所示：



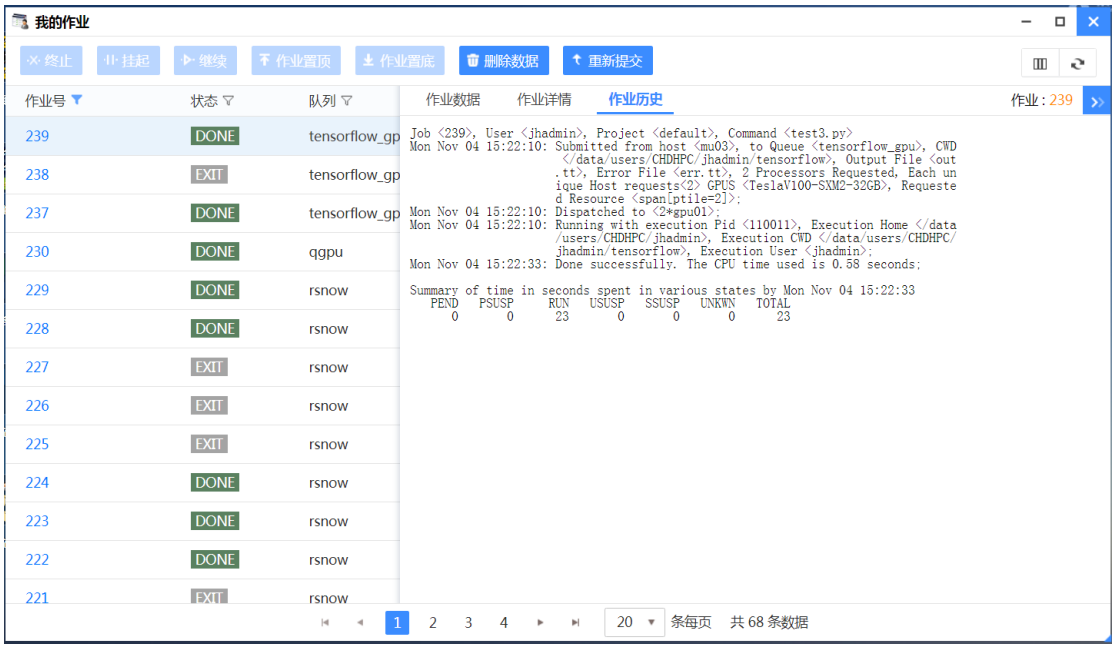
作业提交节点信息

点击查看作业动态输出，可以在页面上直接看到作业的运行输出信息（只有在作业运行时才有输出），如图所示：



作业输出信息

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

3.5.2 挂起作业

仅能对 PEND、RUN 状态的作业执行挂起操作。可以同时选择一个或多个作业进行挂起操作，也可以在作业信息页面对作业进行挂起操作。其中 PEND 状态的作业挂起后状态变成 PSUSP，RUN 状态的作业挂起后状态变成 USUSP。

3.5.3 继续作业

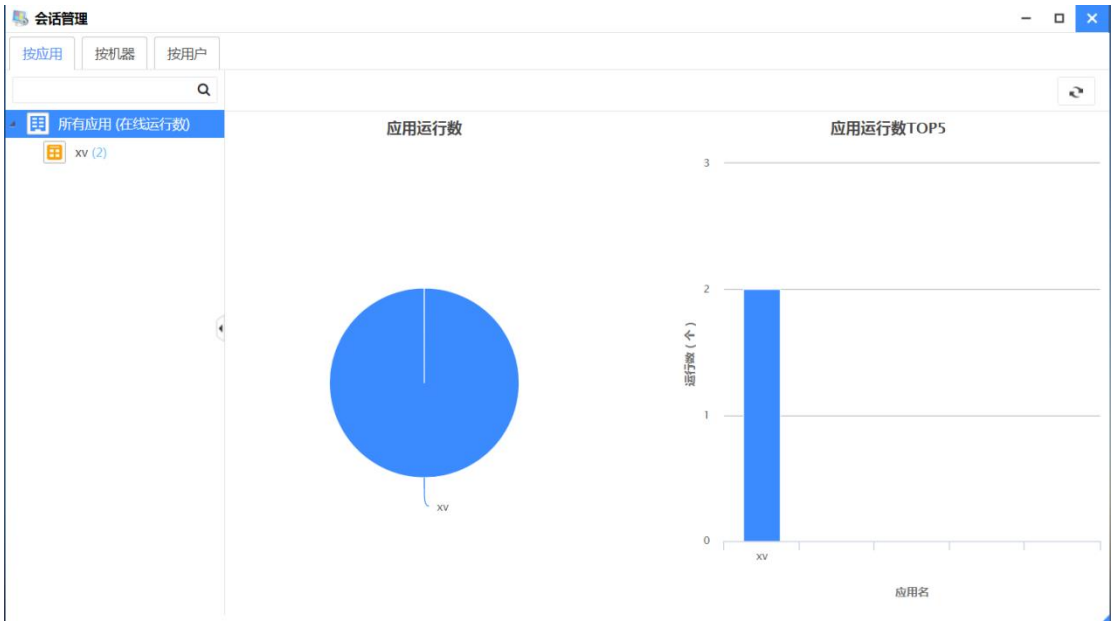
仅能对 PSUSP、USUSP 状态的作业进行唤醒操作，使挂起的作业可以继续运行。可以同时选择一个或多个作业进行继续操作，也可以在作业信息页面对作业进行继续操作。其中 PSUSP 状态的作业继续后状态变成 PEND，USUSP 状态的作业继续后状态变成 RUN。

3.5.4 终止作业

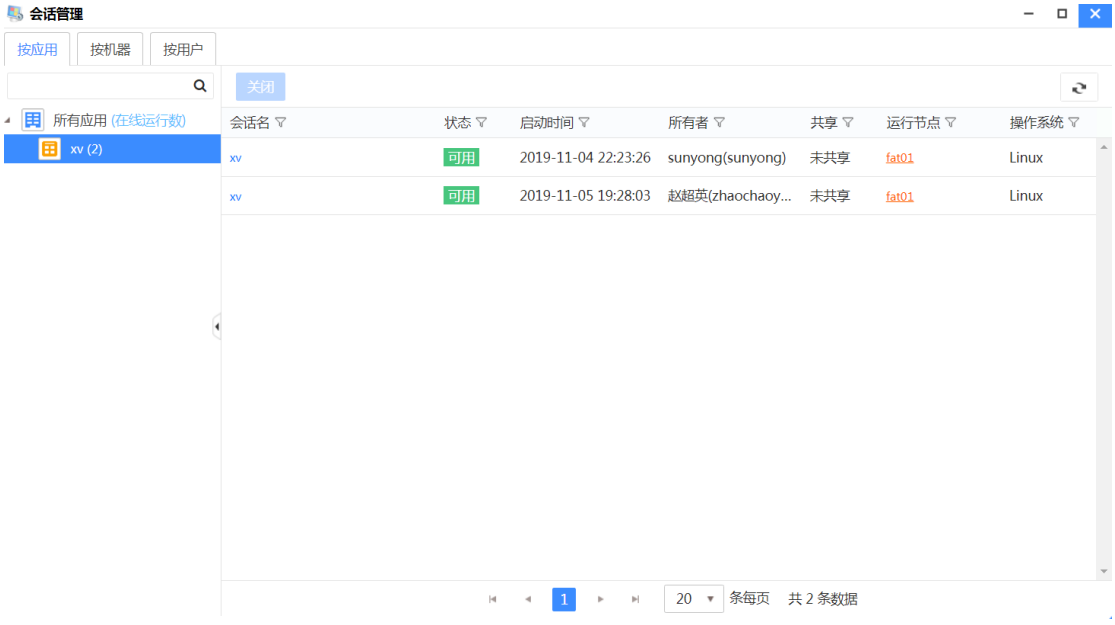
仅能对 PEND、RUN、PSUSP、USUSP 状态的作业进行终止操作。可以同时选择一个或多个作业进行终止操作，也可以在作业信息页面对作业进行终止操作。其中，对作业进行终止后作业状态变成 EXIT。

3.5 会话管理

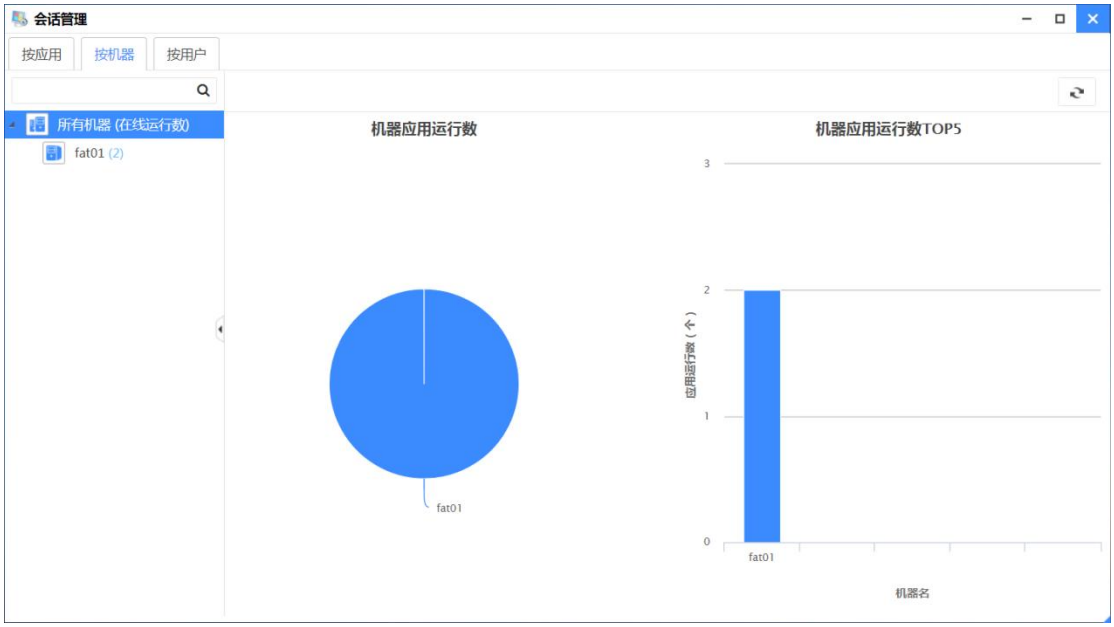
管理员可以直接访问“会话管理”中的作业会话，对系统中所有用户的作业会话进行查看并且操作。管理员点击“系统管理”图标，进入系统管理页面。在系统管理页面点击“会话管理”进入会话管理页面。如图：



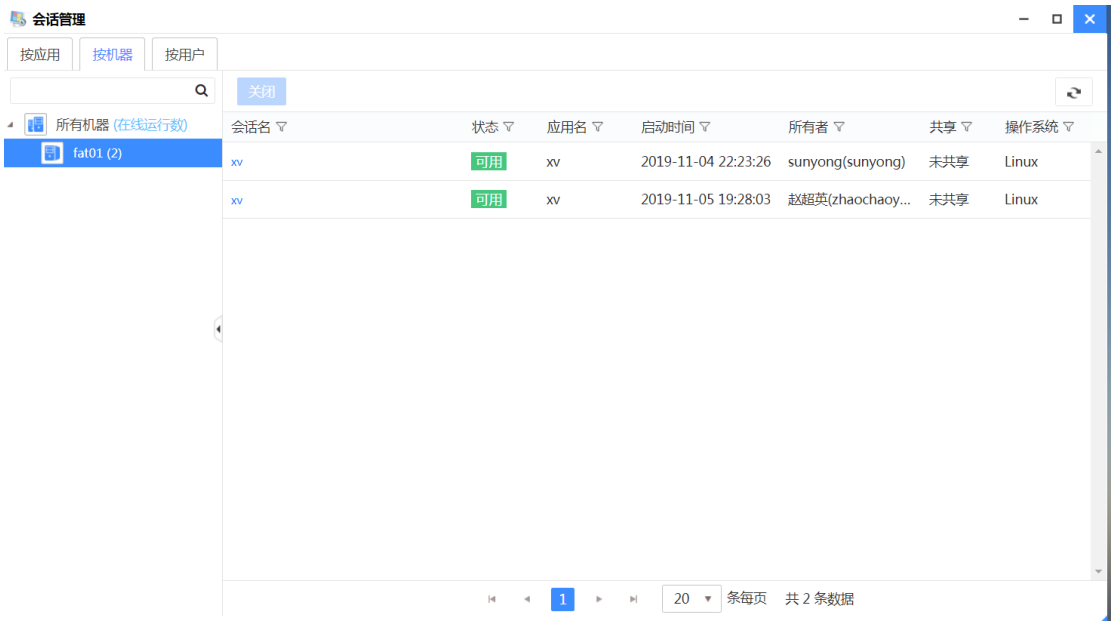
管理员的会话列表分为按应用、按机器和按用户。其中“按应用”的会话管理将所有用户的会话按应用进行分类。点击“按应用”，首页会出现饼状图和柱状图，饼状图统计了所有应用运行数和比例，柱状图统计了应用运行数 TOP5。在左边以 tree 状列举了所有启动的会话类别，并在每个类别后的“()”中注有该种应用运行的数量。点击每种种类，页面右边会切换成该种应用的会话列表，提供了关闭的功能给管理员。管理员可以强制关闭所有用户的会话应用。



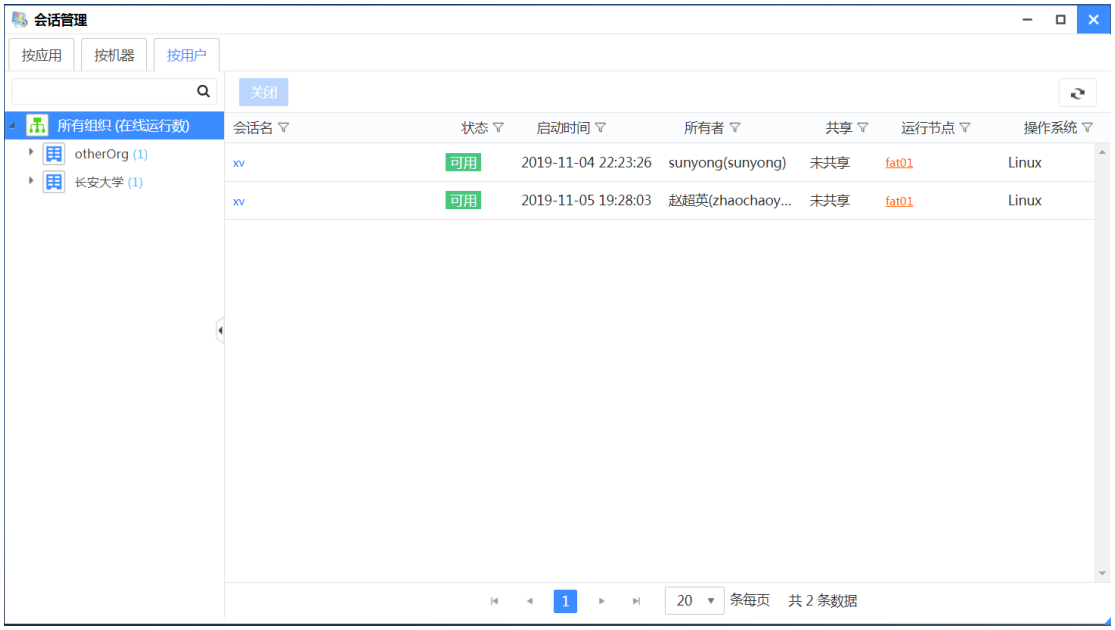
其中按机器的会话管理将所有用户的会话按运行机器进行分类。点击“按机器”，首页会出现饼状图和柱状图，饼状图统计了所有机器上的应用运行数，柱状图统计了机器应用运行数 TOP5。在左边以 tree 状列举了所有会话的运行节点，并在每个类别后的“()”中注有运行的数量。点击每个机器名称，页面右边会切换成该机器目前运行的会话列表，提供了关闭的功能给管理员。管理员可以强制关闭所有用户的会话应用。



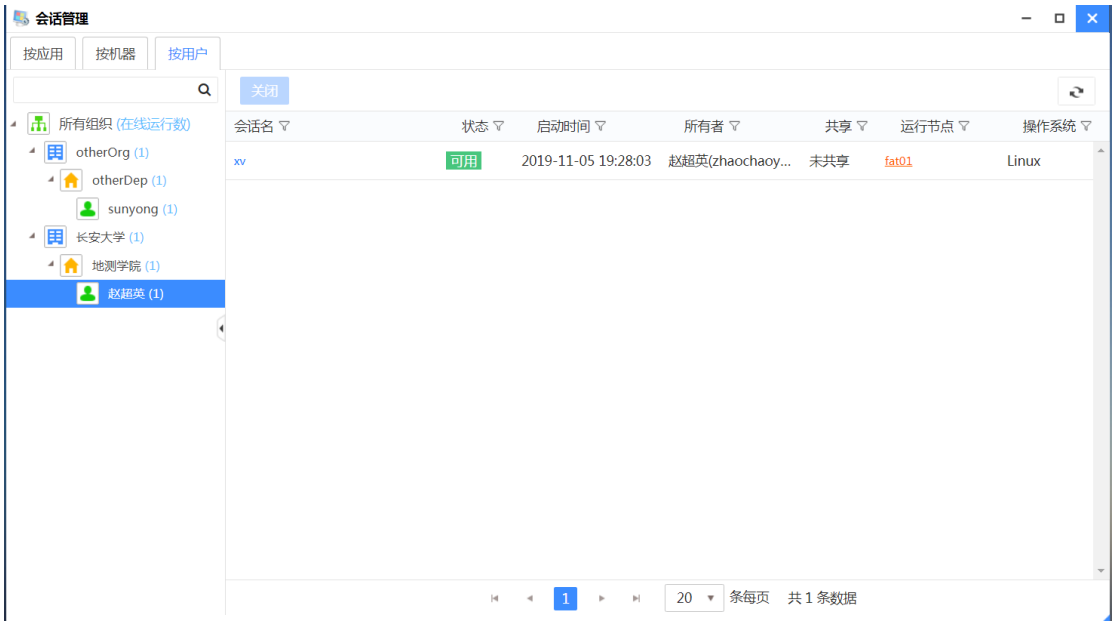
点击左半边的机器名，以列表形式统计了该机器上的所有应用，管理员可以将该应用关闭。



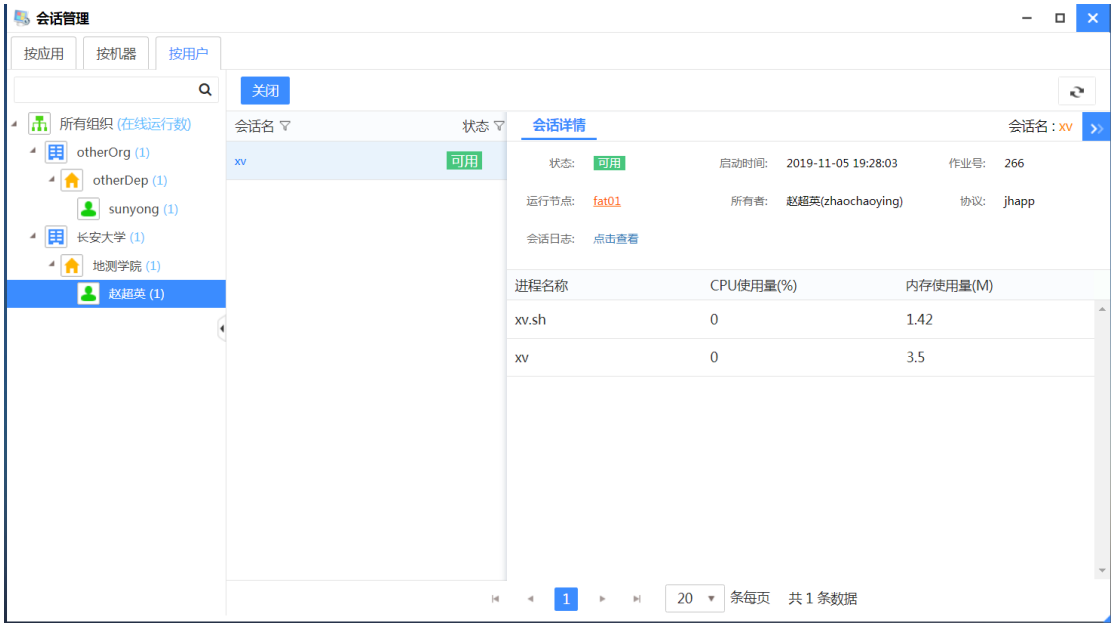
其中按用户的会话管理将所有用户的会话按组织机构进行了分类。点击“按用户”，直接用列表显示所有组织用户的会话列表。在左边以 tree 状列举了所有组织单位，并在每个类别后的“（）”中注有会话运行的数量。点击组织机构的名称，页面右边会切换成该每个组织机构的会话列表，提供了关闭和刷新的功能给管理员。管理员可以强制关闭所有用户的会话应用。



点击左半边的用户名，以列表形式统计了该用户的所有应用，管理员可以将该应用关闭。



点击会话名，或者双击会话所在行，打开会话桌面的详细信息页面：



3.6 数据管理

云端数据管理主要是对数据列表、云端家目录、云端工作区中的数据进行管理。对数据的操作主要有查看数据详细信息、删除数据、对数据列表进行排序。对文件的主要操作有新建文件夹、新建文本文件、上传文件、打开文件、下载文件、选择文件、查看文件信息、剪切文件、复制文件、粘贴文件、删除文件、重

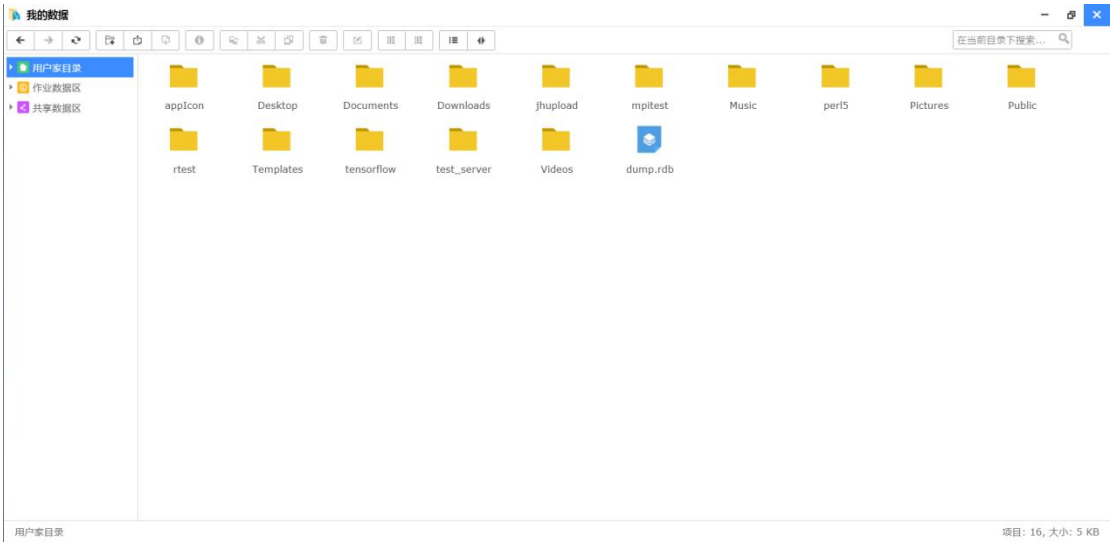
命名文件、对文件进行前后处理、对文件进行排序。

下面将详细介绍数据管理的主要三项内容：

- **用户家目录**
- **作业数据区**
- **共享数据区**

3.7.1 用户家目录

用户家目录页面主要显示了用户家目录下的文件。如图所示：

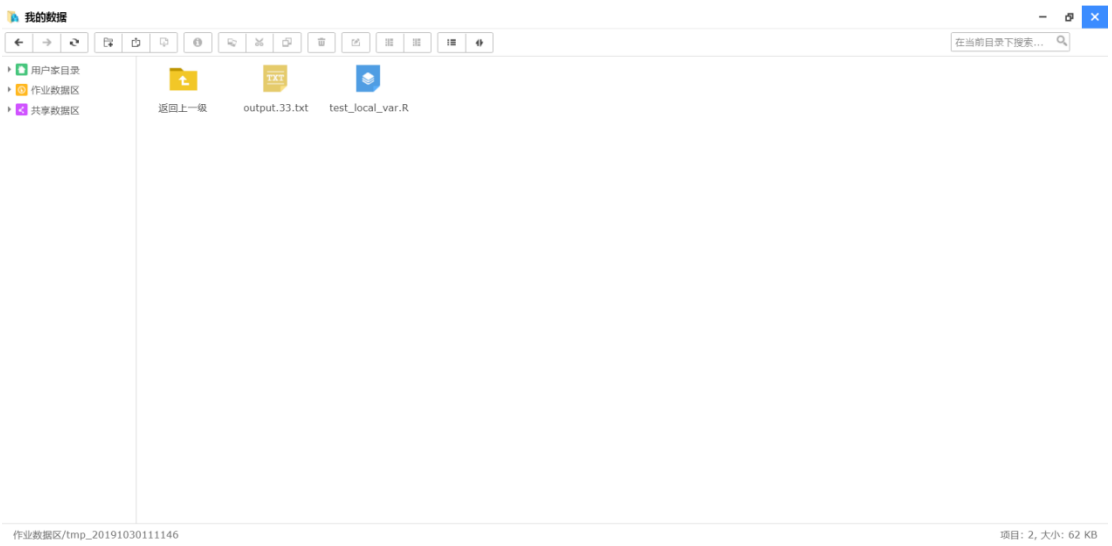


数据列表页面

云端家目录中列出的是服务器端用户 home 目录下（/data/users/CHDHPC/）的文件，可以对文件进行上传、打开、下载、剪切、复制、粘贴、压缩、删除等操作。

3.7.2 作业数据区

作业数据区页面主要显示了列表的形式显示了用户提交作业所产生的数据。如图所示：



作业数据区页面

作业数据区显示的是提交作业产生的数据信息，可以同时选择一个或多个数据进行删除操作，也可以根据作业数据名、项目、创建时间、删除时间对数据列表进行排序。点击作业数据目录名可以进入数据信息页面，在该页面可以查看数据的详细信息。

3.7.3 共享数据区

为了方便用户之间进行文件共享，作业数据区提供了数据共享的功能，用户可以选择某个用户或某个组进行数据共享。用户登录门户后， 点击“我的数据”应用图标， 然后切换到共享数据区目录，即可访问共享数据区页面。

可以通过共享组管理， 查看、新建、修改、删除共享组等功能。

在共享数据区右键新建共享组。

- **添加共享组**

添加共享组时， 共享组名称需要手动填写， 不能为空。

共享组成员需要从左侧的单位、 研究室、 用户目录树中选取， 可通过点击单位与研究室的向下箭头展开后进行选取操作， 支持键盘的 ctrl 和 shift 多选。

支持模糊查找用户功能， 缺省状态下可以查找所有用户。

左侧用户目录树中不显示当前创建共享组的用户， 创建者默认为共享组的一员。

左侧用户选择完毕后， 可以点击“添加” 按钮向右侧容器中加入选择的用户；也可以点击选择右侧已加入的用户， 将其移除出分享参与者的队列。

当页面信息填写完毕后， 点击页面右下角的“确定” 按钮， 创建共享组。

- **修改共享组**

创建者对自己建立的共享组具有修改权限， 参与者不能修改他人创建的共享组。 修改共享组， 只能是更改当前组的成员， 不能修改共享组的名称。

- **删除共享组**

创建者可以删除自己创建的共享组， 参与者不能删除他人创建的共享组。 删除操作会删除此共享组的数据库记录以及共享资源和目录。

3.7 注销

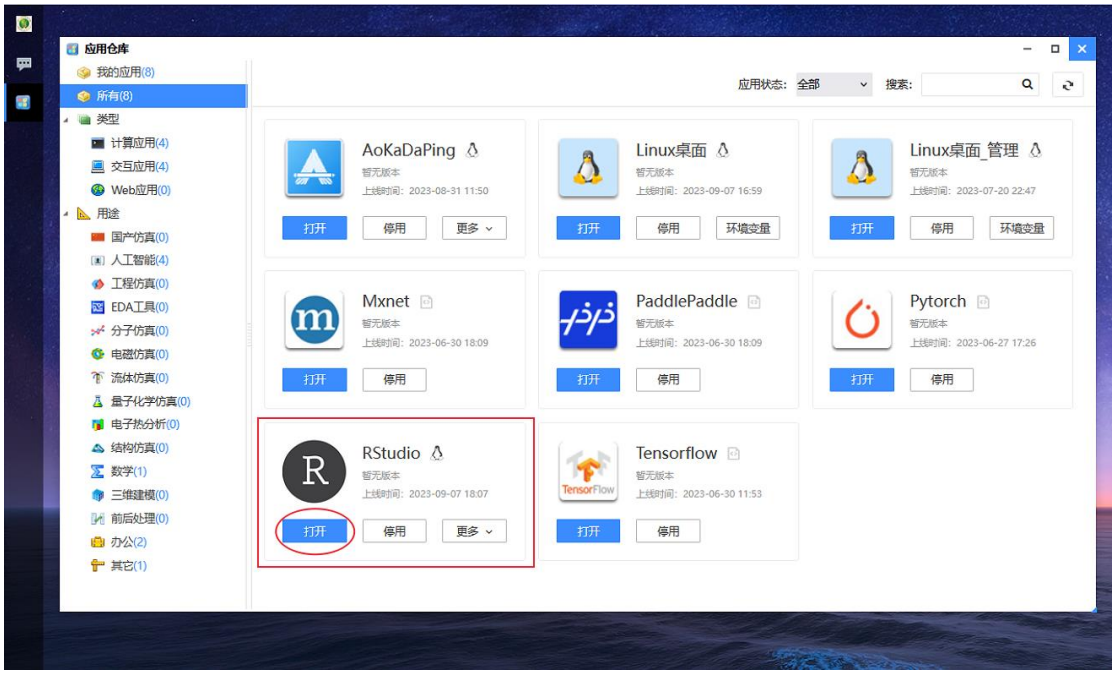
点击左下角的退出按钮可以退出当前登录用户。

3.8 Rstudio 使用

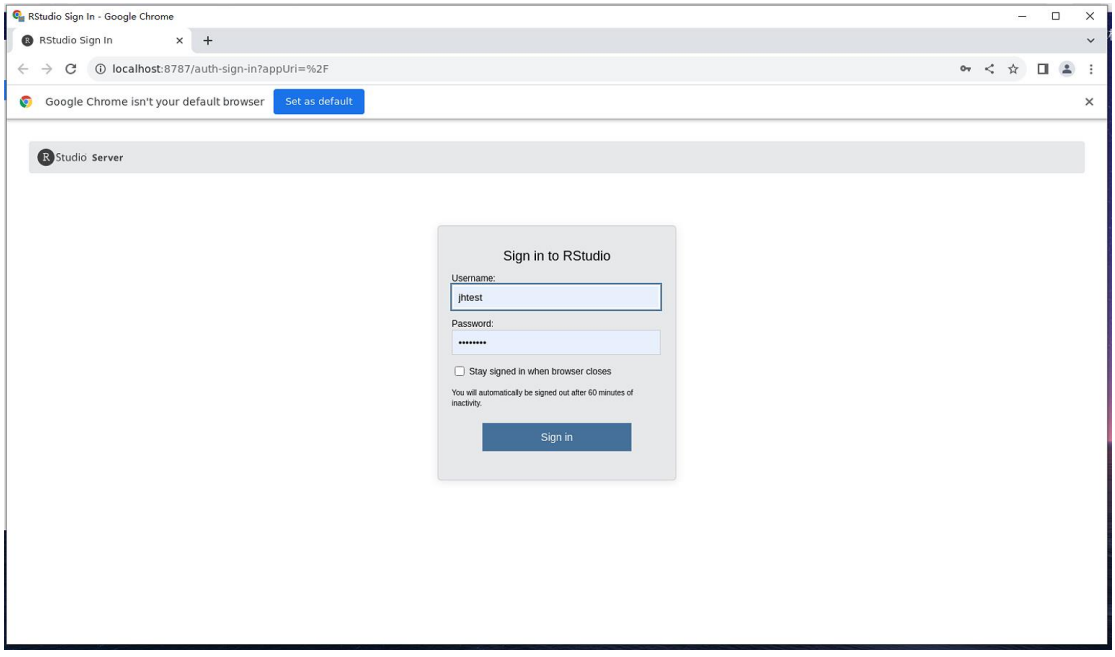
集群门户支持 Rstudio 应用，可以通过网页方便管理或处理 R 数据，具体操作步骤如下：

打开和登录

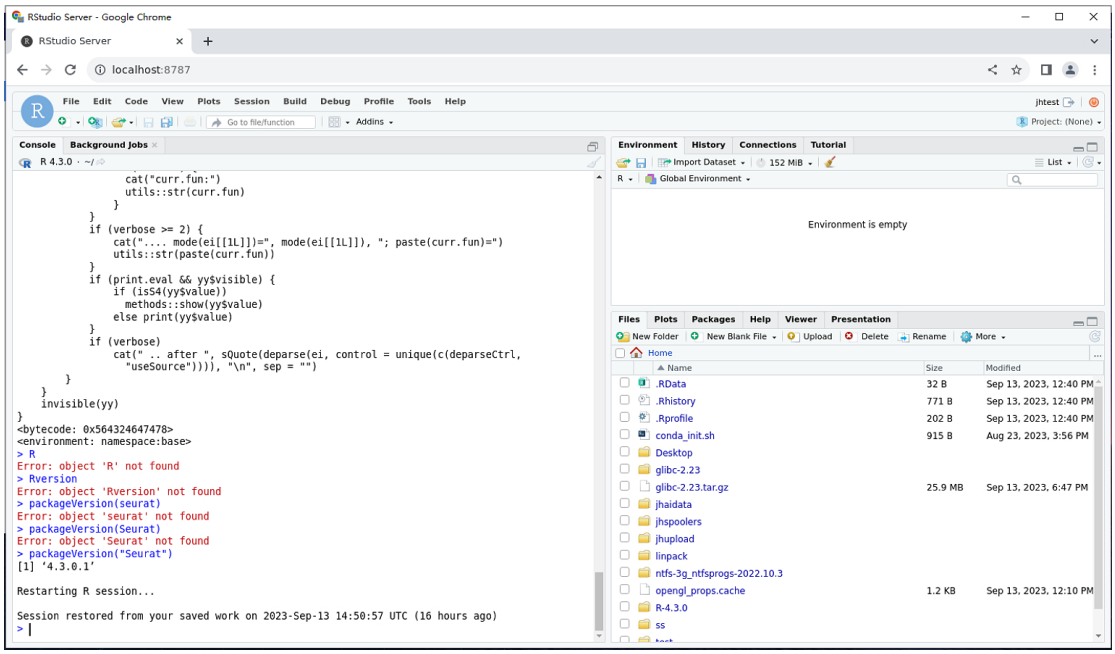
登录门户后打开“应用仓库”，在仓库中打开“RStudio”应用：



之后会弹出谷歌浏览器并默认打开 Rstudio 登录页面，账号和密码与集群中的信息一致：

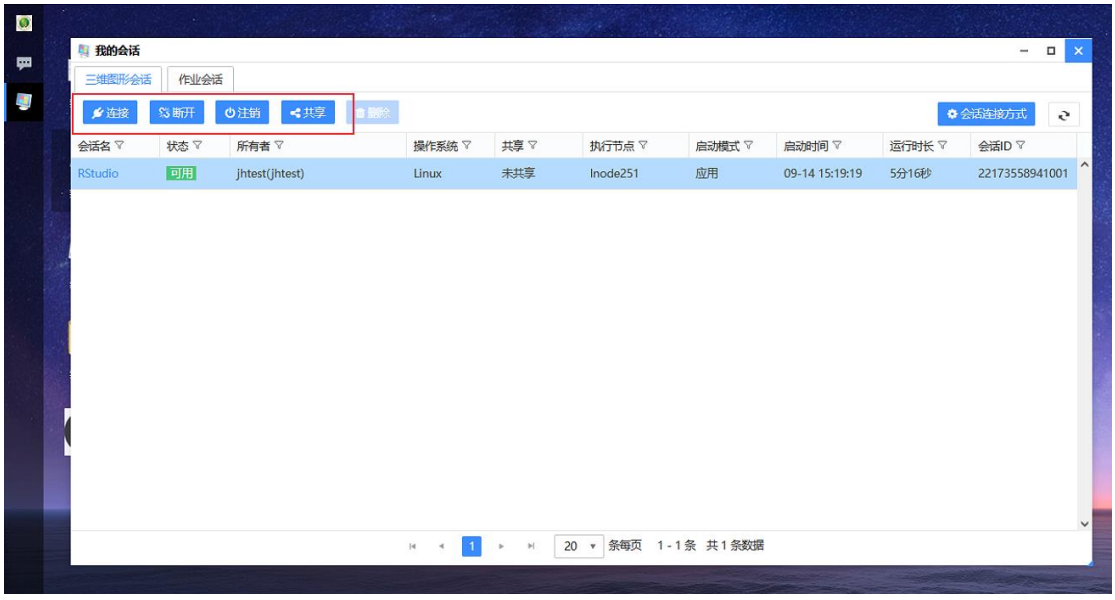


登录后即可使用 Rstudio 了，如下图所示：



会话管理

可在门户中“我的会话”中对已经打开的 Rstudio 会话管理，包含连接、断开、注销和共享操作，如下图：



第四章 注意事项

4.1 支持的浏览器版本

目前支持： IE8、firefox45、chrome50 以上。其他浏览器可以访问，但可能需要修改浏览器的安全配置项，建议使用 firefox。

4.2 上传下载打不开的原因

1）浏览器的版本

2）JH 客户端是否安装，版本是否正确。

4.3 提交作业选择 CPU 核数

提交作业时选择的核数，只会决定调度会分配多少核，并不能决定程序能使用多少核，具体应根据自己的程序实际能使用到多少核进行选择，串行程序则选择一个核心数，并行程序则根据实际需要进行选择。

第五章 集群私有镜像源(Conda 和 Pypi)

5.1 Conda 私有镜像源

Conda 私有镜像源地址：http://localhost/anaconda/

首次使用需拷贝 conda 私有镜像源配置文件至家目录（如果家目录已有.condarc 配置，建议将其重命名备份）：

```
cp /storage/public/share/conda-mirror/.condarc ~/
```

该配置文件配置了 conda 源、虚拟环境安装位置(~/conda_envs)和 conda 包存放位置（~/conda_pkgs），可按需调整配置。

以下以安装 bowtie2 为例介绍私有 conda 镜像源使用方法：

1、已经完成 conda 初始化工作（conda init）

2、加载 conda 基础环境变量： module load conda_base_py310

3、使用 conda 安装 bowtie2

```
conda create -n myTestBowtie2 -c bioconda bowtie2
```

4、激活环境并打印帮助信息

```
conda activate myTestBowtie2

bowtie2 -h
```

5.2 Pypi 私有镜像源

Pypi 私有镜像源地址： http://localhost/pypi/simple/

以安装 biopython 为例，进入一个 python 环境后，执行如下命令安装 biopython：

```
pip install -i http://localhost/pypi/simple --trusted-host localhost biopython
```

打印 biopython 的帮助信息：

```
python -c 'import Bio ; help(Bio)'
```