

长安大学
高性能计算平台使用手册

目 录

第一章 高算平台介绍	4
第二章 命令行使用方式	4
2.1. 平台登录方式	4
2.1.1 Windows 用户访问方式	4
2.1.2 Linux 和 MAC 的登录方式	5
2.2. 数据上传方式	5
2.3. 景行资源管理与调度软件简介	5
2.4. 景行资源管理与调度命令行使用方式	9
2.4.1 加载指定工具	12
2.4.2 提交通用串行作业	13
2.4.3 提交 R 并行作业	14
2.4.4 提交 mpi 并行作业	14
2.4.5 提交 tensorflow 作业	14
2.4.6 提交 gpu 通用作业	14
2.4.7 提交 vasp 作业	15
2.4.8 提交 lammps 作业	15
2.4.9 提交 abaqus (6.14) 作业	15
2.4.10 提交 abaqus (2018) 作业	16
2.4.11 脚本提交模式	16
第三章 WEB 使用方式	17
3.1 文件传输工具安装介绍	17
3.2 远程图形显示工具安装介绍	18
3.3 登录	19
3.4 提交任务	20
3.4.1 ansys 任务提交	20
3.4.2 fluent 任务提交	21
3.4.3 CFX 仿真计算	21
3.5 作业管理使用	22
3.5.1 查看作业信息	27

3.5.2 挂起作业	30
3.5.3 继续作业	30
3.5.4 终止作业	30
3.6 会话管理	31
3.7 数据管理	34
3.7.1 用户家目录	35
3.7.2 作业数据区	35
3.8 注销	36
第四章 注意事项	36
4.1 支持的浏览器版本	36
4.2 上传下载打不开的原因	37
4.3 提交作业选择 CPU 核数	37

第一章 高算平台介绍

长安大学高性能计算平台由 2 个管理节点、1 个登录节点、22 个计算节点、3 个 GPU 节点、1 个胖节点组成，可用存储空间为 677 TB 并部署并行文件系统。各服务器的配置信息如下：

服务器	CPU	内存
管理节点	2 颗 Intel(R) Xeon(R) Gold 5215 CPU @ 2.50GHz 共 20 核	128G
计算节点	2 颗 Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz 共 40 核	192G
新计算节点	2 颗 Intel(R) Xeon(R) Gold 62458RCPU @ 2.70GHz 共 56 核	384G
GPU 节点	2 颗 Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz 共 40 核	192G
新 GPU 节点	2 颗 AMD EPYC 7402 CPU@2.80GHz 共 48 核	512G
胖节点	2 颗 Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz 共 80 核	1T

第二章 命令行使用方式

2.1. 平台登录方式

2.1.1 Windows 用户访问方式

可以通过第三方工具（xshell、putty 等）登录高算平台，xshell 的登录方式如下图所示：

```
Xshell for Xmanager Enterprise 5 (Build 0544)
Copyright (c) 2002-2015 NetSarang Computer, Inc. All rights reserved.

Type 'help' to learn how to use Xshell prompt.
[c:\~]ssh 10.33.0.102
```

输入用户名密码直接登录，地址为 10.33.0.102:



2.1.2 Linux 和 MAC 的登录方式

在 terminal 终端上输入：`ssh username@10.33.0.102` 输入密码登录 (username 为个人工号或者学号)。

注：登录节点为提供给大家登录、编译以及作业提交的节点，不允许在登录节点直接运行程序，严重者则注销使用账号。

2.2. 数据上传方式

可以通过 ftp 和第三方工具 winscp 或者 sftp 的方式上传。

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。

管理应用：对于由景行资源管理与调度软件管理的应用服务系统，用户可以远程运行本地工作站上没有的，甚至因为本地工作站的机器配置较低，无法完成安装运行的大型应用软件。例如，可以在配置较低的桌面机上流畅运行已加入集群的高端服务器上的 CAD、CAE 和 CFD 等大型应用工具。用户的作业实际是运行在集群后台的高端服务器上，其执行过程和结果完全透明地展现在用户的桌面上。

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，以及系统在一周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。

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 Fluent 和 cfx 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

```
[jhadmin@mu01 ooo]$ jjobs
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
292	jhadmin	RUN	normal	mu01	fat01	test	Nov 08

jjobs -a 显示最近完成的作业及正在运行或排队的作业

```
[jhadmin@mu01 ooo]$ jjobs -a
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
280	jhadmin	DONE	fluent	mu01	24*cu15	*fluent-test	Nov 08 09:38
281	jhadmin	DONE	fluent	mu01	40*cu15	*fluent-test	Nov 08 09:40
					32*cu04		

jjobs <作业号> 显示指定的作业状态

```
[****@HPC-M02 ~]$ jjobs 292
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
292	jhadmin	DONE	normal	mu01	fat01	test	Nov 08 13:53

jjobs -l <作业号> 显示指定作业的详细信息

```
[jhadmin@mu01 ooo]$ jjobs -l 292
```

Job <292>, Job Name <test>, User <jhadmin>, Project <default>, Status <DONE>,
Queue <normal>, Command <test>
Fri Nov 08 13:53:38: Submitted from host <mu01>, CWD </data/users/CHDHPC/jhadmi

```

n/ooo>, Output File <output.292>, Error File <error.2
92>;
Fri Nov 08 13:53:39: Started on <fat01>, Execution Home </data/users/CHD
HPC/jha
dmin>, Execution CWD </data/users/CHDHPC/jhadmin/ooo>,
Exe
cution user <jhadmin>;
Fri Nov 08 13:55:20: Done successfully. The CPU time used is 0.35 second
s.

```

SCHEDULING PARAMETERS:

	<i>r15s</i>	<i>r1m</i>	<i>r5m</i>	<i>r15m</i>	<i>ut</i>	<i>pg</i>	<i>io</i>	<i>ls</i>
<i>LoadSched</i>	-	-	-	-	-	-	-	-
<i>LoadStop</i>	-	-	-	-	-	-	-	-
	<i>it</i>	<i>tmp</i>	<i>swap</i>	<i>mem</i>	<i>TeslaV100-SXM2-32GB bytesread</i>			
<i>LoadSched</i>	-	-	-	-		-	-	
<i>LoadStop</i>	-	-	-	-		-	-	
	<i>bytesrecv</i>	<i>bytessent</i>	<i>byteswrite</i>	<i>cores</i>	<i>maxmem</i>	<i>maxswap</i>	<i>maxtmp</i>	
<i>LoadSched</i>	-	-	-	-	-	-	-	-
<i>LoadStop</i>	-	-	-	-	-	-	-	-
	<i>ndisk</i>	<i>rexpri</i>	<i>slots</i>					
<i>LoadSched</i>	-	-	-					
<i>LoadStop</i>	-	-	-					

作业终止

jctrl kill <作业号>

强制杀掉指定的作业

```
[jhadmin@mu01 ooo]$ jctrl kill 292  
Job <292> 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：

```
[jhadmin@mu03 ~]$ module avail  
  
-----  
----- /usr/share/Modules/modulefiles -----  
-----  
dot          module-git  module-info modules      null          use.own  
  
-----  
----- /etc/modulefiles -----  
-----  
  
cuda/cuda-10.0-x86_64          gfortran/gfortran-8.3-x86_64  
mpi/mvapich2-2.3.2-x86_64      mpi/openmpi-4.0.2-x86_64  
R/r-3.6.1-x86_64               gcc/gcc-8.3-x86_64  
mpi/mpich-3.2.1-x86_64         mpi/openmpi-1.10.7-x86_64  
python/python-3.8.0-x86_64     tensorflow/tensorflow-1.14.0-x86_64
```

使用 module load 可以加载模块，如 module load R/r-3.6.1-x86_64，就可以加载 R 工具：

```
[jhadmin@mu03 ~]$ module load R/r-3.6.1-x86_64
```

查看是否加载成功：

```
[jhadmin@mu03 ~]$ which R  
/apps/soft/R/r-3.6.1/bin/R
```

此时使用 `module list` 可以查看目前已经加载的工具：

```
[jhadmin@mu03 ~]$ module list
Currently Loaded Modulefiles:
  1) R/r-3.6.1-x86_64
```

使用 `module rm` 就可以取消加载工具，如 `module rm R/r-3.6.1-x86_64`，可以取消加载 R 工具：

```
[jhadmin@mu03 ~]$ module rm R/r-3.6.1-x86_64
```

如果想取消所有工具的加载，使用 `module purge` 命令。

2.4.2 提交通用串行作业

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：

```
[jhadmin@mu03 ~]$ jsub -q normal -n 1 -m cu01 -e error.%J -o output.%J -J
my_job ./test
Job <295> is submitted to queue <normal>
```

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 cu01 节点

<-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>：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。

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

```
[jhadmin@mu03 ~]$ jsub -n 80 -q rsnow -e error.%J -o output.%J test.r
```

2.4.4 提交 mpi 并行作业

【二进制程序】

cd 到测试文件所在的目录后，当前目录下存在 test

修改 ~/.bashrc，添加环境变量，若使用 openmpi, 添加：

```
module load mpi/openmpi-1.10.7-x86_64
```

若使用 mpich, 添加：

```
module load mpi/mpich-3.2.1-x86_64
```

若使用 mvapich2, 添加：

```
module load mpi/mvapich2-2.3.2-x86_64
```

提交命令：

```
[jhadmin@mu03 ~]$ jsub -n 80 -q para -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

```
[jhadmin@mu03 ~]$ jsub -q tensorflow -e error.%J -o output.%J test.py
```

提交到 gpu 队列

```
[jhadmin@mu03 ~]$ jsub -q tensorflow_gpu -e error.%J -o output.%J test.py
```

2.4.6 提交 gpu 通用作业

提交到 gpu 的队列中

```
[jhadmin@mu03 ~]$ jsub -q qgpu -e error.%J -o output.%J test
```

2.4.7 提交 vasp 作业

cd 到测试文件 (INCAR, KPOINTS, POSCAR, POTCAR) 所在的目录

```
[jhadmin@mu03 ~]$ jsub -q vasp -e error.%J -o output.%J ./vasp_std
```

2.4.8 提交 lammps 作业

cd 到计算文件 (.in) 所在的目录

```
[jhadmin@mu03 ~]$ jsub -n 32 -q lammps -e error.%J -o output.%J ./test.in
```

2.4.9 提交 abaqus (6.14) 作业

查看 abaqus 脚本提交方式:

```
[jhadmin@mu03 ~]$ abaqussubmit
```

Usage:

```
abaquussubmit np abaqusOpt
```

```
abaquussubmit np job=jobname [oldjob=oldjobname] [double] int
```

cd 到计算文件所在目录, 当前目录存在 Job-1.inp

```
[jhadmin@mu03 ~]$ abaqussubmit 8 job=Job-1 cpus=8 int
```

如果计算过程需要调用其他文件, 当前目录存在 Job-2.inp Job-1.odb 则:

```
[jhadmin@mu03 ~]$ abaqussubmit 8 job=Job-2 oldjob=Job-1 cpus=8 int
```

参数详解:

abaquussubmit: 执行程序

2 : 申请使用 CPU 数

Job : 进行计算的 inp 文件

oldjob : 需要使用的 odb 文件

cpus=2 : 计算文件本身支持的 cpu 数

int : 进行计算

暂时只支持用户最多使用 40 核进行并行计算。

2.4.10 提交 abaqus (2018) 作业

查看 abaqus 脚本提交方式:

```
[jhadmin@mu03 ~]$ abaqussubmit2018
```

Usage:

```
Abaquussubmit2018 np abaqusOpt
```

```
Abaquussubmit2018 np job=jobname [oldjob=oldjobname] [double] int
```

cd 到计算文件所在目录, 当前目录存在 Job-1. inp

```
[jhadmin@mu03 ~]$ abaqussubmit2018 8 job=Job-1 cpus=8 int
```

如果计算过程需要调用其他文件, 当前目录存在 Job-2. inp Job-1. odb 则:

```
[jhadmin@mu03 ~]$ abaqussubmit2018 8 job=Job-2 oldjob=Job-1 cpus=8 int
```

参数详解:

abaquussubmit2018 : 执行程序

2 : 申请使用 CPU 数

Job : 进行计算的 inp 文件

oldjob : 需要使用的 odb 文件

cpus=2 : 计算文件本身支持的 cpu 数

int : 进行计算

暂时只支持用户最多使用 40 核进行并行计算。

2.4.11 脚本提交模式

脚本提交支持#JSUB 来指定作业的提交选项, 使用该功能时, 必须在脚本的第一行指定 shell 类型, 如#!/bin/sh

shell 脚本格式如下:

```
#!/bin/sh
```

```
#JSUB -q para
```

```
#JSUB -n 100
```

```
#JSUB -e error.%J
```

```
#JSUB -o output.%J
```



```
#JSUB -J my_job  
./test
```

test：用来指定作业的执行命令。如 sleep 、 fluent 等。在 test 前设置作业提交的环境变量。

编写脚本时注意事项：

- (1) 只支持 Linux shell 脚本。
- (2) 提交命令支持#JSUB。脚本中文件内容最大长度为 4096 字节，否则作业提交失败。
- (3) 脚本中#JSUB 选项支持：-x、-P、-R、-q、-m、-n、-J、-i、-o、-e、-E、-Ep、-cwd。
- (4) 提交脚本中以#JSUB 开头的行，都是作业的命令选项信息。非#开头的行，都是作业的命令或者设置作业环境的 shell 命令。
- (5) 如果在脚本中指定了重复选项信息时，会打印提示信息，提示第二次及以后几次指定的选项不生效（默认第一次指定的选项生效）。

提交命令：

```
[jhadmin@mu01 ~]$ jsub < ./run.sh
```

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 <http://10.33.0.21:8080> 进入高算平台的登录页面，在页面上下载文件传输工具并安装。



The image shows a login page for the 'JH Innovation Software Application Portal'. At the top, there is a logo for '景行锐创 JH Innovation Software' and the text '应用门户'. Below the logo, there are two input fields: one for '用户名' (Username) and one for '密码' (Password). A blue button labeled '登录' (Login) is positioned below the password field. At the bottom, a message states: '为了保证您的使用，请在终端环境中下载安装以下应用：' (To ensure your use, please download and install the following applications in the terminal environment:). Below this message, there are two links: '远程图形显示工具' (Remote Graphics Display Tool) and '文件传输工具' (File Transfer Tool). The '文件传输工具' link is highlighted with a red box.

景行锐创 JH Innovation Software | 应用门户

用户名

密码

登录

为了保证您的使用，请在终端环境中下载安装以下应用：

[远程图形显示工具](#)、[文件传输工具](#)

3.2 远程图形显示工具安装介绍

为了实现任务的图形可视化，景行公司自己研发了一套实现软件图形传递的工具。



The image shows a login page for 'JH Innovation Software' (景行锐创). At the top, there is a logo and the text '应用门户' (Application Portal). Below this are two input fields: '用户名' (Username) and '密码' (Password). A blue '登录' (Login) button is positioned below the password field. At the bottom, a message states: '为了保证您的使用，请在终端环境中下载安装以下应用：' (To ensure your use, please download and install the following applications in the terminal environment:). Below this message, two links are provided: '远程图形显示工具' (Remote Graphics Display Tool) and '文件传输工具' (File Transfer Tool). The '远程图形显示工具' link is highlighted with a red box.

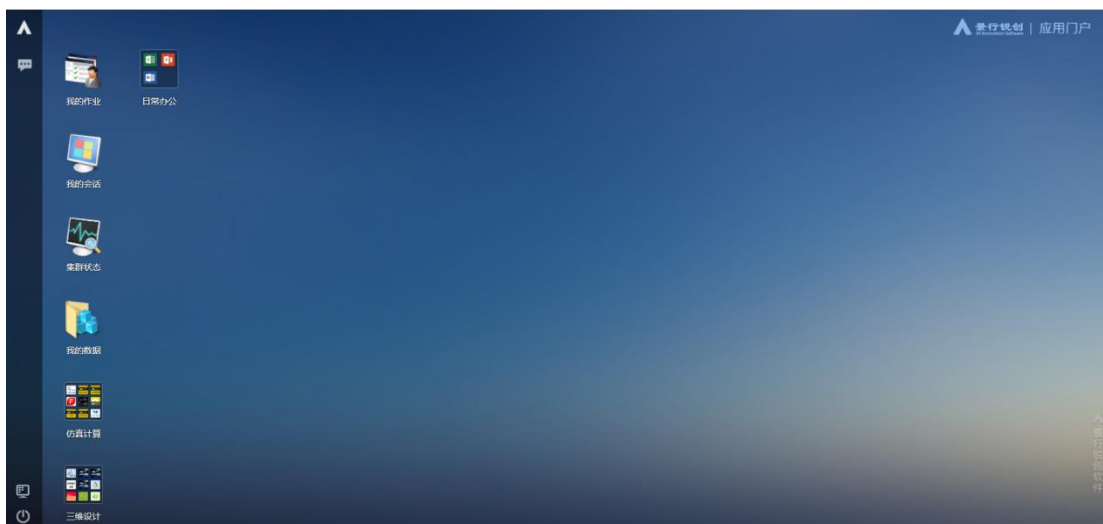
3.3 登录

访问 <http://10.33.0.21:8080> 进入高算平台的登录页面，如下：



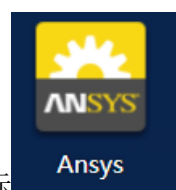
The image shows a login page for '长安大学高性能计算平台' (Chang'an University High-Performance Computing Platform). At the top, there is a logo and the text '长安大学' (Chang'an University) and '长安大学高性能计算平台'. Below this are two input fields: '用户名' (Username) and '密码' (Password). A blue '登录' (Login) button is positioned below the password field. At the bottom, a message states: '为了保证您的使用，请在终端环境中下载安装以下应用：' (To ensure your use, please download and install the following applications in the terminal environment:). Below this message, four links are provided: '远程图形显示工具' (Remote Graphics Display Tool), '文件传输工具' (File Transfer Tool), '平台使用说明' (Platform User Guide), and '培训PPT' (Training PPT).

输入用户名密码登录，登录进去的主页面如下：



3.4 提交任务

3.4.1 ansys 任务提交



在登录的主页面可以直接点击 snsys 的图标，点击之后会弹出 ansys 的提交页面，如下：

Ansys

—

×

目前集群中可用于计算【ansys】作业的CPU个数为640个

作业名：

计算文件*：

本地

远端

选中0个文件

其他文件：

本地

远端

选中0个文件

CPU数：

0

64

128

192

256

8

模块：

Mechanical

其他选项：

版本号：

18.1

取消

确定

作业名：用户可以自定义作业的名称。

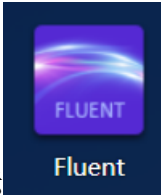
CPU 数：当前提交任务的 cpu 核数。

软件版本：ansys 的版本

设置完参数后点击“提交作业”，提交完之后页面会自动进入到作业的详细

信息页面

3.4.2 fluent 任务提交



在登录的主页面可以直接点击 fluent 的图标，点击之后会弹出 fluent 的提交页面，如下：

Fluent

目前集群中可用于计算【fluent】作业的CPU个数为640个

作业名：

CAS文件*：

本地

远端

选中0个文件

DAT文件：

本地

远端

选中0个文件

时间步长大小：

时间步数：

迭代步数*：

100

自动保存步数：

CPU数：

0

64

128

192

256

8

维数：

2d

计算结果名：

版本号：

18.1.0

图形界面支持：

关

取消

确定

将图中各选项填写完成（其中带*的是必填项），点击“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。如果提交作业时选择了“图形界面支持”，则可以在详细页面打开图形界面，图形界面上可以显示 Fluent 作业的运行过程。默认以二维应用程序图形展示作业的运行情况。

3.4.3 CFX 仿真计算

CFX 是一款通用的流体动力学仿真软件。其中 CFX 作业提交页面如图所示：

CFX

目前集群中可用于计算【cfx】作业的CPU个数为640个

作业名：

DEF文件*：

本地

远端

选中0个文件

RES文件：

本地

远端

选中0个文件

其他选项：

Interpolate?：

关

CPU数：

0

64

128

192

256

8

双精度：

关

版本号：

18.1

取消

确定

将图中各选项填写完成（其中带*的是必填项），点击“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。并且可以在作业信息页面打开图形界面，图形界面上可以显示 CFX 作业的运行过程。默认以二维应用程序图形展示作业的运行情况。

3.5 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：

作业号 ▾	状态 ▾	队列 ▾	执行节点 ▾	提交时间 ▾	执行时间 ▾	作业名 ▾
239	DONE	tensorflow_gpu	2*gpu01	11-04 15:22:10	11-04 15:22:10	test3.py
238	EXIT	tensorflow_gpu	2*gpu02	11-04 15:20:02	11-04 15:20:04	test3.py
237	DONE	tensorflow_gpu	2*gpu02	11-04 15:16:17	11-04 15:16:18	test3.py
230	DONE	qgpu	gpu02	11-04 11:59:50	11-04 11:59:50	./run.sh
229	DONE	rsnow	30*cu04	11-04 11:33:20	11-04 11:33:21	testjob1
228	DONE	rsnow	30*cu04	11-04 11:29:16	11-04 11:29:18	test_local_var.R
227	EXIT	rsnow	30*cu04	11-04 11:25:06	11-04 11:25:08	test_local_var.R
226	EXIT	rsnow	30*cu04	11-04 10:57:31	11-04 10:57:33	test_local_var.R
225	EXIT	rsnow	30*cu04	11-04 10:55:02	11-04 10:55:05	test_local_var.R
224	DONE	rsnow	30*cu04	11-04 10:46:35	11-04 10:46:37	testjob1
223	DONE	rsnow	30*cu04	11-04 10:45:12	11-04 10:45:12	testjob1
222	DONE	rsnow	30*cu04	11-04 10:44:05	11-04 10:44:06	testjob1
221	EXIT	rsnow	30*cu04	11-04 10:35:08	11-04 10:35:08	testjob1

1 2 3 4 20 条每页 共 68 条数据

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、UNKNOWN、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：

我的作业						
✕ 终止 挂起 继续 作业置顶 作业置底 删除数据 重新提交						
作业号	状态	队列	执行节点	提交时间	执行时间	作业名
239	等于	tensorflow_gpu	2*gpu01	11-04 15:22:10	11-04 15:22:10	test3.py
238		tensorflow_gpu	2*gpu02	11-04 15:20:02	11-04 15:20:04	test3.py
237	筛选 清空	tensorflow_gpu	2*gpu02	11-04 15:16:17	11-04 15:16:18	test3.py
230	DONE	qgpu	gpu02	11-04 11:59:50	11-04 11:59:50	./run.sh
229	DONE	rsnow	30*cu04	11-04 11:33:20	11-04 11:33:21	testjob1
228	DONE	rsnow	30*cu04	11-04 11:29:16	11-04 11:29:18	test_local_var.R
227	EXIT	rsnow	30*cu04	11-04 11:25:06	11-04 11:25:08	test_local_var.R
226	EXIT	rsnow	30*cu04	11-04 10:57:31	11-04 10:57:33	test_local_var.R
225	EXIT	rsnow	30*cu04	11-04 10:55:02	11-04 10:55:05	test_local_var.R
224	DONE	rsnow	30*cu04	11-04 10:46:35	11-04 10:46:37	testjob1
223	DONE	rsnow	30*cu04	11-04 10:45:12	11-04 10:45:12	testjob1
222	DONE	rsnow	30*cu04	11-04 10:44:05	11-04 10:44:06	testjob1
221	EXIT	rsnow	30*cu04	11-04 10:35:08	11-04 10:35:08	testiob1
1 2 3 4 20 条每页 共 68 条数据						

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：

我的作业						
✕ 终止 挂起 继续 作业置顶 作业置底 删除数据 重新提交						
作业号	状态	队列	执行节点	提交时间	执行时间	作业名
239	等于	tensorflow_gpu	2*gpu01	11-04 15:22:10	11-04 15:22:10	test3.py
238		tensorflow_gpu	2*gpu02	11-04 15:20:02	11-04 15:20:04	test3.py
237	筛选 清空	tensorflow_gpu	2*gpu02	11-04 15:16:17	11-04 15:16:18	test3.py
230	DONE	qgpu	gpu02	11-04 11:59:50	11-04 11:59:50	./run.sh
229	DONE	rsnow	30*cu04	11-04 11:33:20	11-04 11:33:21	testjob1
228	DONE	rsnow	30*cu04	11-04 11:29:16	11-04 11:29:18	test_local_var.R
227	EXIT	rsnow	30*cu04	11-04 11:25:06	11-04 11:25:08	test_local_var.R
226	EXIT	rsnow	30*cu04	11-04 10:57:31	11-04 10:57:33	test_local_var.R
225	EXIT	rsnow	30*cu04	11-04 10:55:02	11-04 10:55:05	test_local_var.R
224	DONE	rsnow	30*cu04	11-04 10:46:35	11-04 10:46:37	testjob1
223	DONE	rsnow	30*cu04	11-04 10:45:12	11-04 10:45:12	testjob1
222	DONE	rsnow	30*cu04	11-04 10:44:05	11-04 10:44:06	testjob1
221	EXIT	rsnow	30*cu04	11-04 10:35:08	11-04 10:35:08	testiob1
1 2 3 4 20 条每页 共 68 条数据						

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：

我的作业

[✕ 终止](#)
[⏸ 挂起](#)
[▶ 继续](#)
[📄 作业置顶](#)
[⬇ 作业置底](#)
[🗑 删除数据](#)
[↶ 重新提交](#)

作业号 ▾	状态 ▾	队列 ▾	执行节点 ▾	提交时间 ▾	执行时间 ▾	作业名 ▾
239	DONE	tens		11-04 15:22:10	11-04 15:22:10	test3.py
238	EXIT	tens		11-04 15:20:02	11-04 15:20:04	test3.py
237	DONE	tensorflow_gpu	2*gpu02	11-04 15:16:17	11-04 15:16:18	test3.py
230	DONE	qgpu	gpu02	11-04 11:59:50	11-04 11:59:50	./run.sh
229	DONE	rsnow	30*cu04	11-04 11:33:20	11-04 11:33:21	testjob1
228	DONE	rsnow	30*cu04	11-04 11:29:16	11-04 11:29:18	test_local_var.R
227	EXIT	rsnow	30*cu04	11-04 11:25:06	11-04 11:25:08	test_local_var.R
226	EXIT	rsnow	30*cu04	11-04 10:57:31	11-04 10:57:33	test_local_var.R
225	EXIT	rsnow	30*cu04	11-04 10:55:02	11-04 10:55:05	test_local_var.R
224	DONE	rsnow	30*cu04	11-04 10:46:35	11-04 10:46:37	testjob1
223	DONE	rsnow	30*cu04	11-04 10:45:12	11-04 10:45:12	testjob1
222	DONE	rsnow	30*cu04	11-04 10:44:05	11-04 10:44:06	testjob1
221	EXIT	rsnow	30*cu04	11-04 10:35:08	11-04 10:35:08	testiob1

1 2 3 4 20 条每页 共 68 条数据

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：

我的作业

[✕ 终止](#)
[⏸ 挂起](#)
[▶ 继续](#)
[📄 作业置顶](#)
[⬇ 作业置底](#)
[🗑 删除数据](#)
[↶ 重新提交](#)

作业号 ▾	状态 ▾	队列 ▾	执行节点 ▾	提交时间 ▾	执行时间 ▾	作业名 ▾
239	DONE	tensorflow_gpu	2*gpu01		11-04 15:22:10	test3.py
238	EXIT	tensorflow_gpu	2*gpu02		11-04 15:20:04	test3.py
237	DONE	tensorflow_gpu	2*gpu02	11-04 15:16:17	11-04 15:16:18	test3.py
230	DONE	qgpu	gpu02	11-04 11:59:50	11-04 11:59:50	./run.sh
229	DONE	rsnow	30*cu04	11-04 11:33:20	11-04 11:33:21	testjob1
228	DONE	rsnow	30*cu04	11-04 11:29:16	11-04 11:29:18	test_local_var.R
227	EXIT	rsnow	30*cu04	11-04 11:25:06	11-04 11:25:08	test_local_var.R
226	EXIT	rsnow	30*cu04	11-04 10:57:31	11-04 10:57:33	test_local_var.R
225	EXIT	rsnow	30*cu04	11-04 10:55:02	11-04 10:55:05	test_local_var.R
224	DONE	rsnow	30*cu04	11-04 10:46:35	11-04 10:46:37	testjob1
223	DONE	rsnow	30*cu04	11-04 10:45:12	11-04 10:45:12	testjob1
222	DONE	rsnow	30*cu04	11-04 10:44:05	11-04 10:44:06	testjob1
221	EXIT	rsnow	30*cu04	11-04 10:35:08	11-04 10:35:08	testiob1

1 2 3 4 20 条每页 共 68 条数据

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：

我的作业

[✖ 终止](#)
[⏸ 挂起](#)
[▶ 继续](#)
[⏴ 作业置顶](#)
[⏵ 作业置底](#)
[🗑 删除数据](#)
[↺ 重新提交](#)

作业号	状态	队列	执行节点	提交时间	执行时间	作业名
239	DONE	tensorflow_gpu	2*gpu01	11-04 15:22:10	11-04 15:22:10	test3.py
238	EXIT	tensorflow_gpu	2*gpu02	11-04 15:20:04	11-04 15:20:04	test3.py
237	DONE	tensorflow_gpu	2*gpu02	11-04 15:16:17	11-04 15:16:18	test3.py
230	DONE	qgpu	gpu02	11-04 11:59:50	11-04 11:59:50	./run.sh
229	DONE	rsnow	30*cu04	11-04 11:33:20	11-04 11:33:21	testjob1
228	DONE	rsnow	30*cu04	11-04 11:29:16	11-04 11:29:18	test_local_var.R
227	EXIT	rsnow	30*cu04	11-04 11:25:06	11-04 11:25:08	test_local_var.R
226	EXIT	rsnow	30*cu04	11-04 10:57:31	11-04 10:57:33	test_local_var.R
225	EXIT	rsnow	30*cu04	11-04 10:55:02	11-04 10:55:05	test_local_var.R
224	DONE	rsnow	30*cu04	11-04 10:46:35	11-04 10:46:37	testjob1
223	DONE	rsnow	30*cu04	11-04 10:45:12	11-04 10:45:12	testjob1
222	DONE	rsnow	30*cu04	11-04 10:44:05	11-04 10:44:06	testjob1
221	EXIT	rsnow	30*cu04	11-04 10:35:08	11-04 10:35:08	testjob1

1 2 3 4 20 条每页 共 68 条数据

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：

我的作业

[✖ 终止](#)
[⏸ 挂起](#)
[▶ 继续](#)
[⏴ 作业置顶](#)
[⏵ 作业置底](#)
[🗑 删除数据](#)
[↺ 重新提交](#)

作业号	状态	队列	执行节点	提交时间	执行时间	作业名
239	DONE	tensorflow_gpu	2*gpu01	11-04 15:22:10	11-04 15:22:10	test3.py
238	EXIT	tensorflow_gpu	2*gpu02	11-04 15:20:04	11-04 15:20:04	test3.py
237	DONE	tensorflow_gpu	2*gpu02	11-04 15:16:17	11-04 15:16:18	test3.py
230	DONE	qgpu	gpu02	11-04 11:59:50	11-04 11:59:50	./run.sh
229	DONE	rsnow	30*cu04	11-04 11:33:20	11-04 11:33:21	testjob1
228	DONE	rsnow	30*cu04	11-04 11:29:16	11-04 11:29:18	test_local_var.R
227	EXIT	rsnow	30*cu04	11-04 11:25:06	11-04 11:25:08	test_local_var.R
226	EXIT	rsnow	30*cu04	11-04 10:57:31	11-04 10:57:33	test_local_var.R
225	EXIT	rsnow	30*cu04	11-04 10:55:02	11-04 10:55:05	test_local_var.R
224	DONE	rsnow	30*cu04	11-04 10:46:35	11-04 10:46:37	testjob1
223	DONE	rsnow	30*cu04	11-04 10:45:12	11-04 10:45:12	testjob1
222	DONE	rsnow	30*cu04	11-04 10:44:05	11-04 10:44:06	testjob1
221	EXIT	rsnow	30*cu04	11-04 10:35:08	11-04 10:35:08	testjob1

1 2 3 4 20 条每页 共 68 条数据

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：

我的作业

× 终止 挂起 继续 作业置顶 作业置顶 删除数据 重新提交

作业号	状态	队列	作业数据	作业详情	作业历史
239	DONE	tensorflow_gp	test3.py	用户: jhadmin 作业号: 239 队列: tensorflow_gpu 状态: DONE 命令: test3.py 提交节点: mu03 数据删除时间: 执行节点: 2*gpu01 槽数: 2 节点名称: /data/槽数/CPU% jhadmin/tensorflow gpu01 0 / 40 0% 提交目录: /data/users/CHDHPC/jhadmin/tensorflow	
238	EXIT	tensorflow_gp			
237	DONE	tensorflow_gp			
230	DONE	qgpu			
229	DONE	rsnow			
228	DONE	rsnow			
227	EXIT	rsnow			
226	EXIT	rsnow			
225	EXIT	rsnow			
224	DONE	rsnow			
223	DONE	rsnow			
222	DONE	rsnow			
221	EXIT	rsnow			

资源需求与使用

资源需求: span[ptile=2] CPU执行时间: 0.58 s 提交时间: 2019-11-04 15:22:10
交换区使用量: 内存使用量: 24 MB 执行时间: 2019-11-04 15:22:10
结束时间: 2019-11-04 15:22:33

其它

1 2 3 4 20 条每页 共 68 条数据

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：

我的作业

× 终止 挂起 继续 作业置顶 作业置顶 删除数据 重新提交

作业号	状态	队列	执行节点	提交	作业数据	作业详情	作业历史
249	RUN	testq	cu04	11-0	用户: wangli 作业号: 249 队列: testq 状态: RUN 命令: ./hmain 提交节点: mu01 数据删除时间: 执行节点: cu04 槽数: 1 提交目录: /data/users/CHDHPC/wangli/fangan1 执行目录: /data/users/CHDHPC/wangli/fangan1		
235	DONE	memorylimit	cu15	11-0			
234	EXIT	memorylimit	cu04	11-0			
233	EXIT	memorylimit	cu15	11-0			
232	EXIT	memorylimit	cu04	11-0			
231	EXIT	memorylimit	cu04	11-0			
218	DONE	testq	cu04	11-0			
217	DONE	testq	cu04	11-0			
216	DONE	testq	cu04	11-0			
215	DONE	testq	cu04	11-0			
214	DONE	normal	cu04	11-0			
213	DONE	normal	cu04	11-0			
212	DONE	testq	cu04	11-0			
210	DONE	normal	cu01	11-0			

资源需求与使用

资源需求: CPU执行时间: 1.38 s 提交时间: 2019-11-04 17:35:20
交换区使用量: 内存使用量: 34 MB 执行时间: 2019-11-04 17:35:21
结束时间:

其它

作业动态输出

1 2 3 20 条每页 共 55 条数据

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：

我的作业			
<div> ✖ 终止 ⏏ 挂起 ▶ 继续 ⏴ 作业置顶 ⏵ 作业置底 🗑 删除数据 ↶ 重新提交 </div>			
作业号 ▾	状态 ▾	队列 ▾	作业数据 作业详情 作业历史
239	DONE	tensorflow_gp	Job <239>, User <jhadmin>, Project <default>, Command <test3.py> Mon Nov 04 15:22:10: Submitted from host <mu03>, to Queue <tensorflow_gpu>, CWD </data/users/CHDHPC/jhadmin/tensorflow>, Output File <out.tt>, Error File <err.tt>, 2 Processors Requested, Each unique Host requests<2> GPUs <TeslaV100-SXM2-32GB>, Requested Resource ;
238	EXIT	tensorflow_gp	Mon Nov 04 15:22:10: Dispatched to <2*gpu01>; Mon Nov 04 15:22:10: Running with execution Pid <110011>, Execution Home </data/users/CHDHPC/jhadmin>, Execution CWD </data/users/CHDHPC/jhadmin/tensorflow>, Execution User <jhadmin>;
237	DONE	tensorflow_gp	Mon Nov 04 15:22:33: Done successfully. The CPU time used is 0.58 seconds;
230	DONE	qgpu	Summary of time in seconds spent in various states by Mon Nov 04 15:22:33
229	DONE	rsnow	PEND PSUSP RUN USUSP SSUSP UNKNW TOTAL
228	DONE	rsnow	0 0 23 0 0 0 23
227	EXIT	rsnow	
226	EXIT	rsnow	
225	EXIT	rsnow	
224	DONE	rsnow	
223	DONE	rsnow	
222	DONE	rsnow	
221	EXIT	rsnow	
<div> ⏴ ⏵ 1 2 3 4 ▶ ⏴ 20 条每页 共 68 条数据 </div>			

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

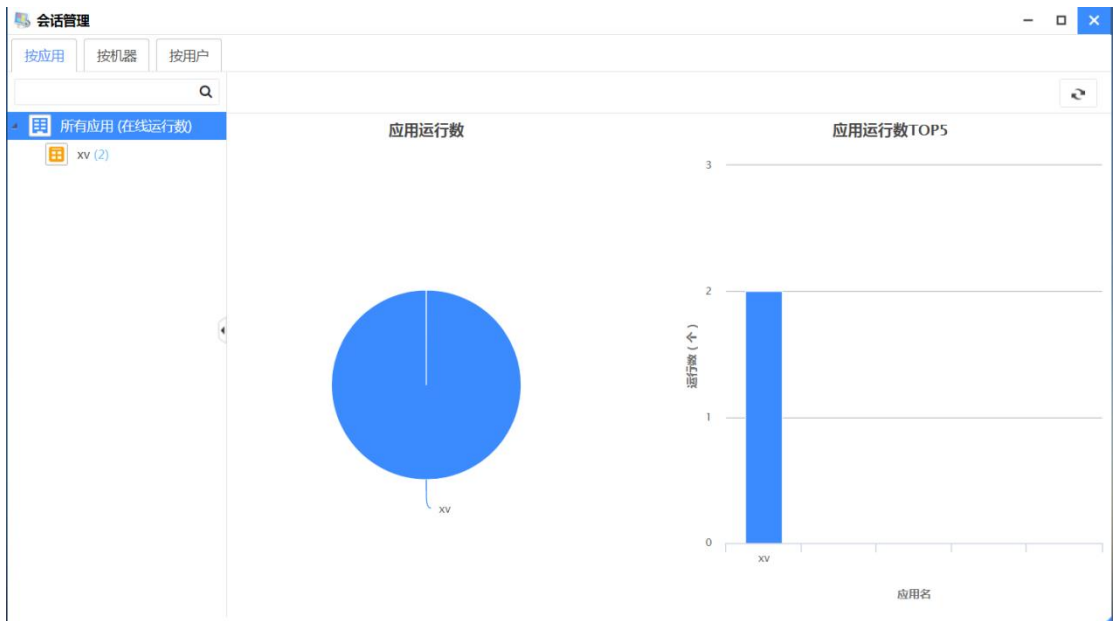
仅能对 PSUSP、USUSP 状态的作业进行唤醒操作，使挂起的作业可以继续运行。可以同时选择一个或多个作业进行继续操作，也可以在作业信息页面对作业进行继续操作。其中 PSUSP 状态的作业继续后状态变成 PEND，USUSP 状态的作业继续后状态变成 RUN。

3.5.4 终止作业

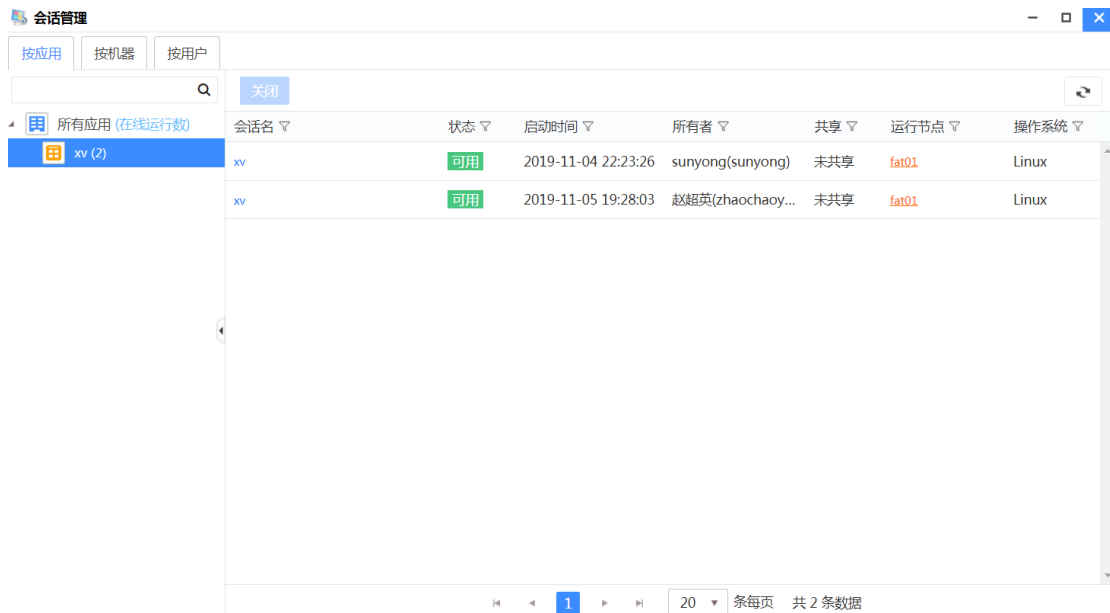
仅能对 PEND、RUN、PSUSP、USUSP 状态的作业进行终止操作。可以同时选择一个或多个作业进行终止操作，也可以在作业信息页面对作业进行终止操作。其中，对作业进行终止后作业状态变成 EXIT。

3.6 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：

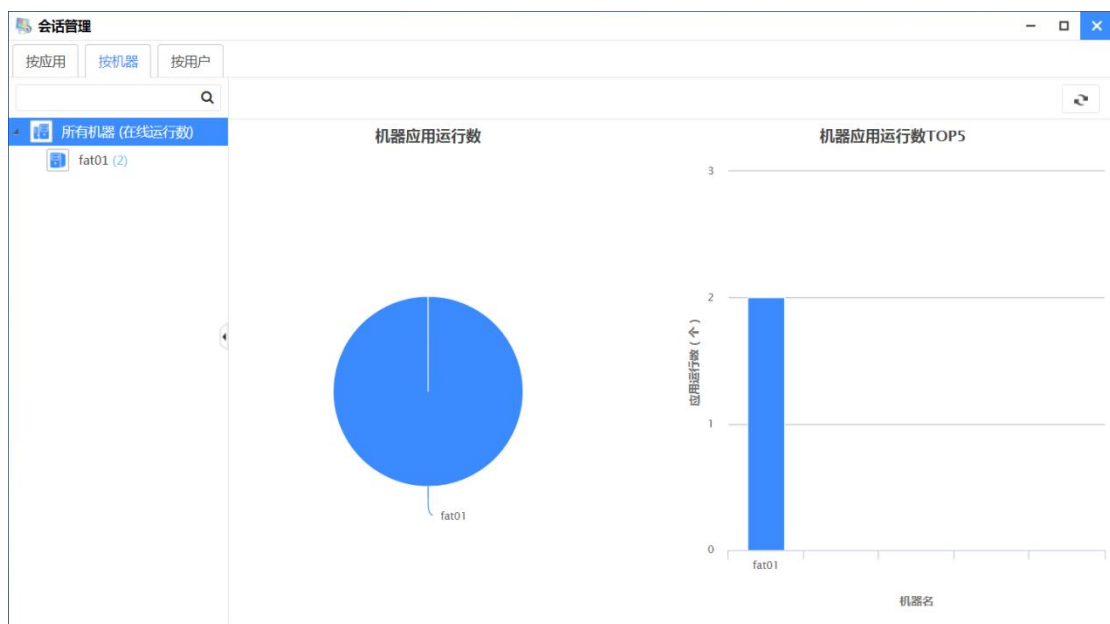


管理员的会话列表分为按应用、按机器和按用户。其中“按应用”的会话管理将所有用户的会话按应用进行分类。点击“按应用”，首页会出现饼状图和柱状图，饼状图统计了所有应用运行数和比例，柱状图统计了应用运行数 TOP5。在左边以 tree 状列举了所有启动的会话类别，并在每个类别后的“（）”中注有该种应用运行的数量。点击每种种类，页面右边会切换成该种应用的会话列表，提供了关闭的功能给管理员。管理员可以强制关闭所有用户的会话应用。

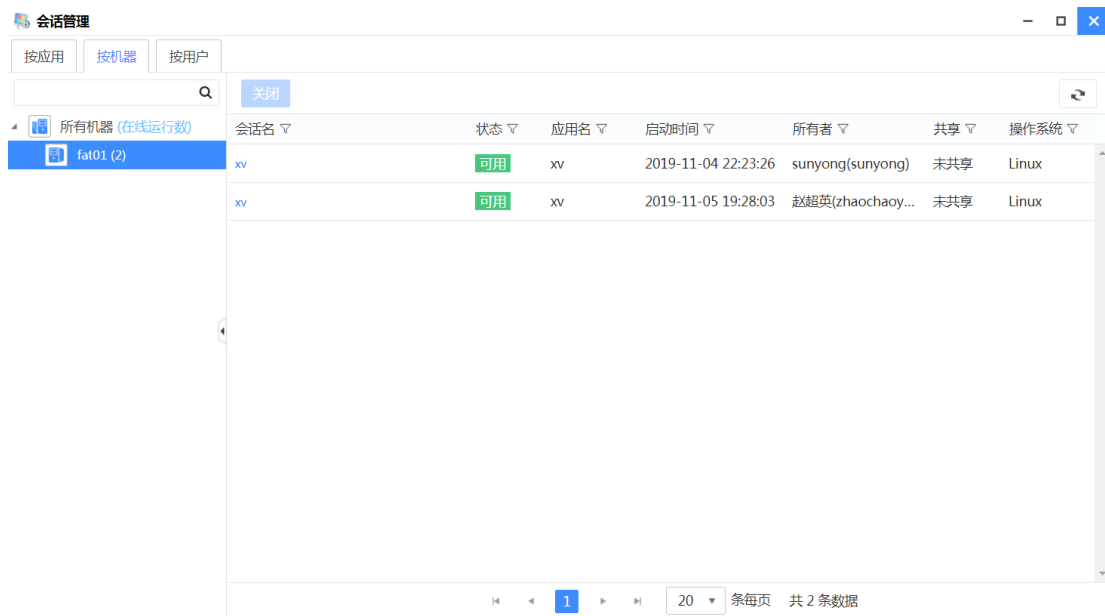


会话名	状态	启动时间	所有者	共享	运行节点	操作系统
xv	可用	2019-11-04 22:23:26	sunyong(sunyong)	未共享	fat01	Linux
xv	可用	2019-11-05 19:28:03	赵超英(zhaochaoy...)	未共享	fat01	Linux

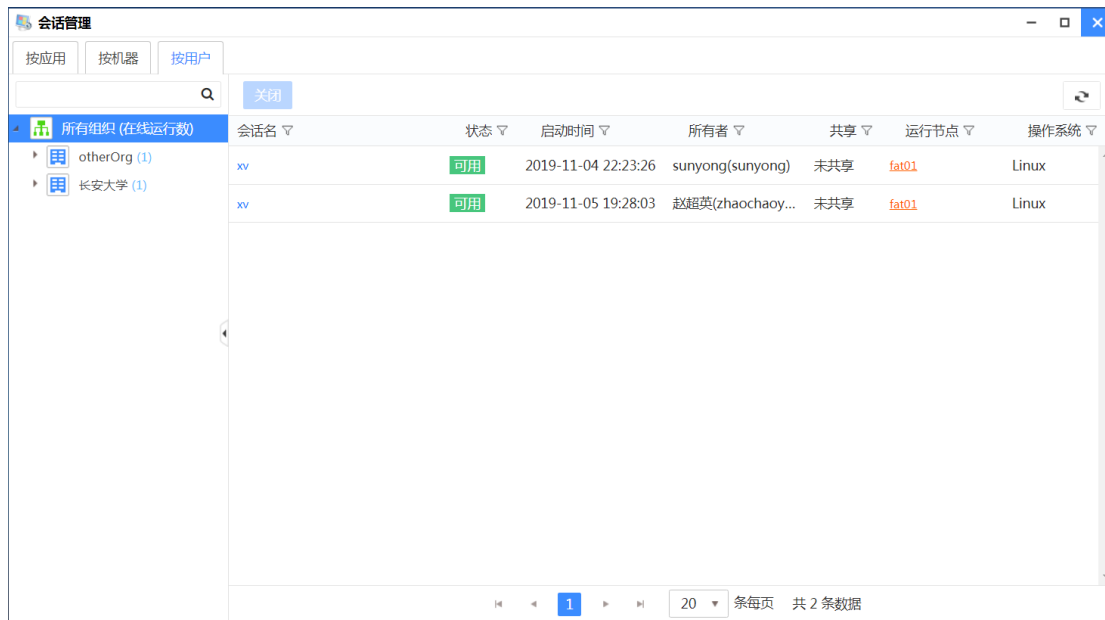
其中按机器的会话管理将所有用户的会话按运行机器进行分类。点击“按机器”，首页会出现饼状图和柱状图，饼状图统计了所有机器上的应用运行数，柱状图统计了机器应用运行数 TOP5。在左边以 tree 状列举了所有会话的运行节点，并在每个类别后的“（）”中注有运行的数量。点击每个机器名称，页面右边会切换成该机器目前运行的会话列表，提供了关闭的功能给管理员。管理员可以强制关闭所有用户的会话应用。



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。



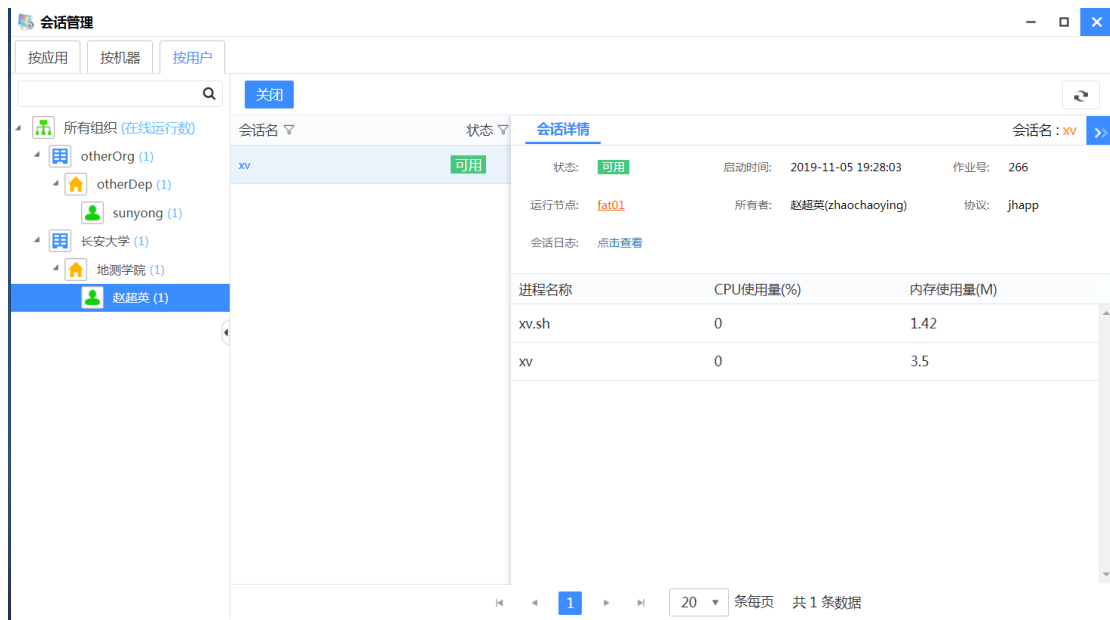
其中按用户的会话管理将所有用户的会话按组织机构进行了分类。点击“按用户”，直接用列表显示所有组织用户的会话列表。在左边以 tree 状列举了所有组织单位，并在每个类别后的“（）”中注有会话运行的数量。点击组织机构的名称，页面右边会切换成该每个组织机构的会话列表，提供了关闭和刷新的功能给管理员。管理员可以强制关闭所有用户的会话应用。



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：



3.7 数据管理

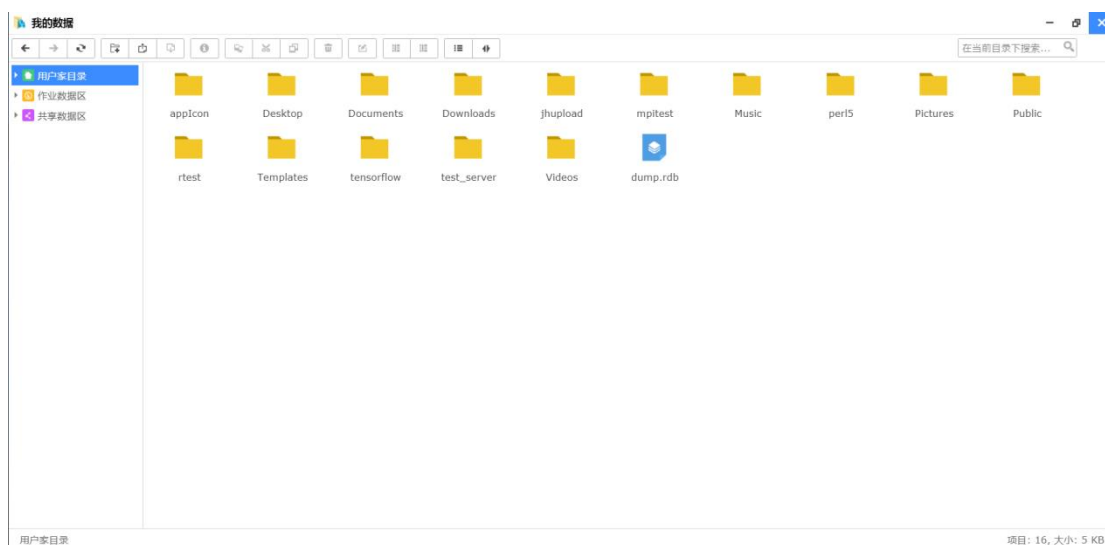
云端数据管理主要是对数据列表、云端家目录、云端工作区中的数据进行管理。对数据的操作主要有查看数据详细信息、删除数据、对数据列表进行排序。对文件的主要操作有新建文件夹、新建文本文件、上传文件、打开文件、下载文件、选择文件、查看文件信息、剪切文件、复制文件、粘贴文件、删除文件、重命名文件、对文件进行前后处理、对文件进行排序。

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：

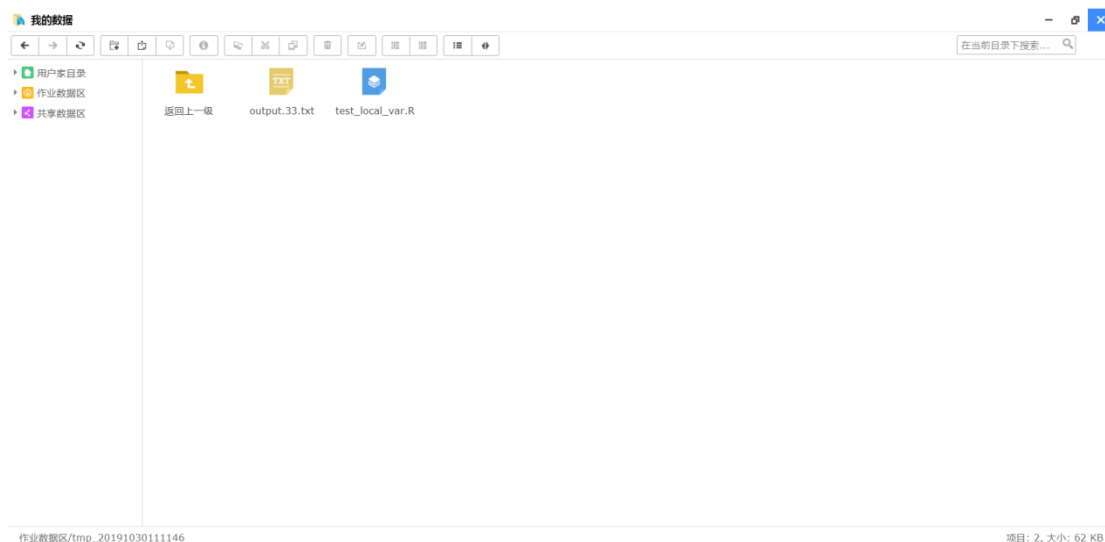


数据列表页面

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。

3.8 注销



点击左下角的退出按钮可以退出当前登录用户。

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) jhfileclient 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，串行程序则选择一核，并行程序则根据实际需要进行选择。