

1. 用户上机操作

在超算中心集群系统上，用户提交作业的基本步骤如下所示：

- 准备模型数据文件和作业脚本文件。
- 用户账号创建和登录
- 传输文件到集群系统登录节点上
- 提交作业
- 作业管理
- 作业结果查看，将结果文件下载到本地硬盘

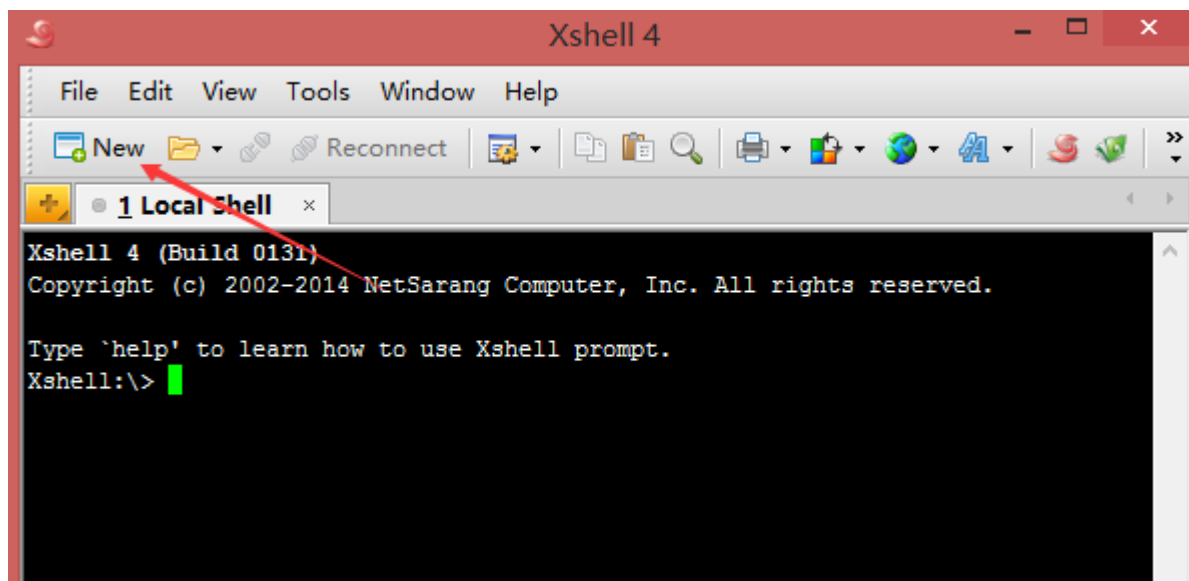
其中，准备模型数据文件和作业脚本文件，用户可以在本地完成。用户账号的创建需要联系集群系统管理员。本文主要介绍其余几个步骤的进行方法。

1.1 用户登录

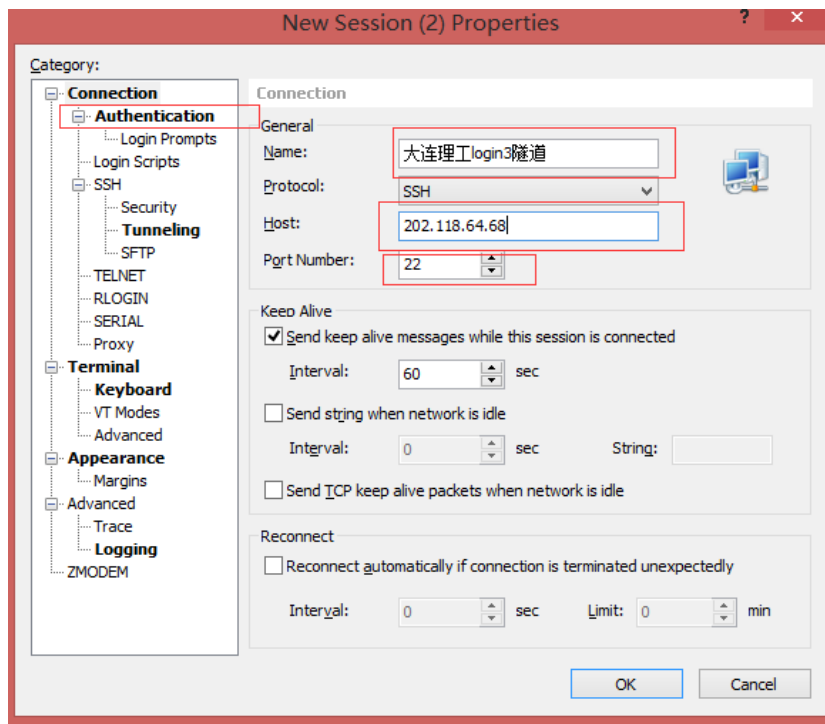
用户可以通过代理的方式登陆 GridViewweb 界面。

用户首先需要通过 ssh 用隧道连接点上登陆节点。客户端软件可以使用 putty, secureCRT, Xshell 等多种客户端连接软件，这里已 xshell 为例说明隧道连接方法：

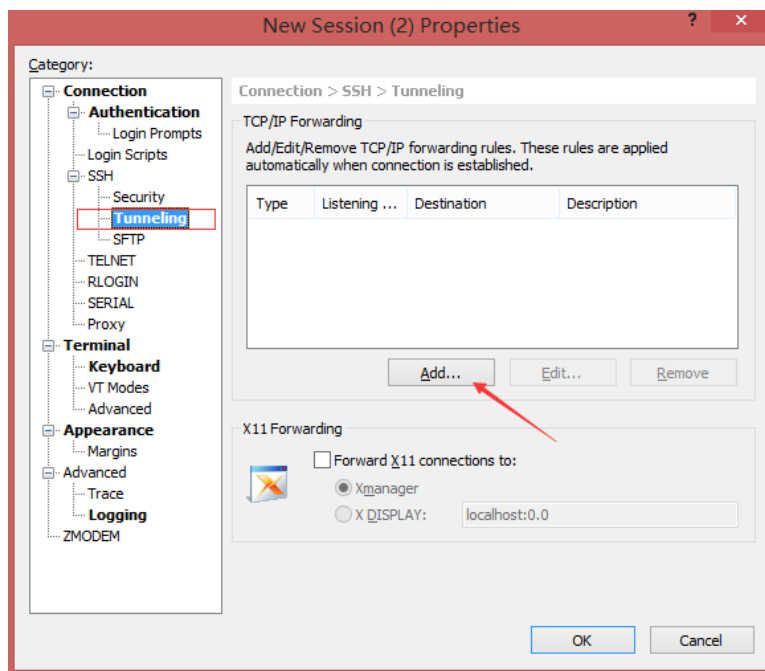
首先打开 xshell，建立新的连接：



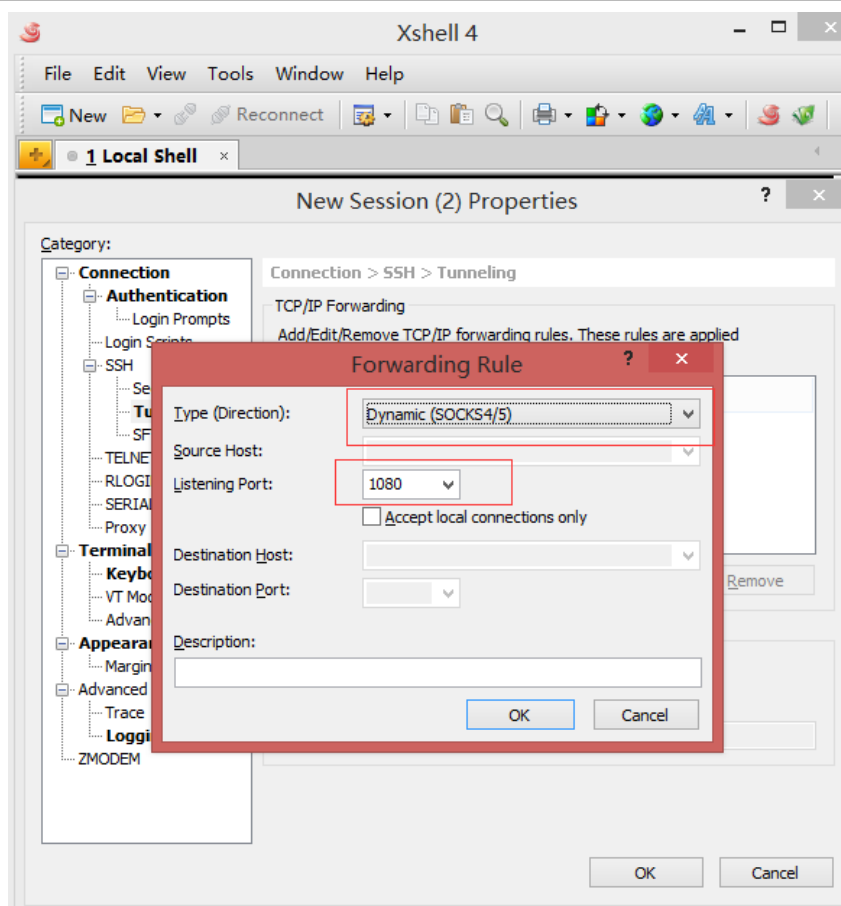
输入连接名称和带连接节点的 ip 和端口号



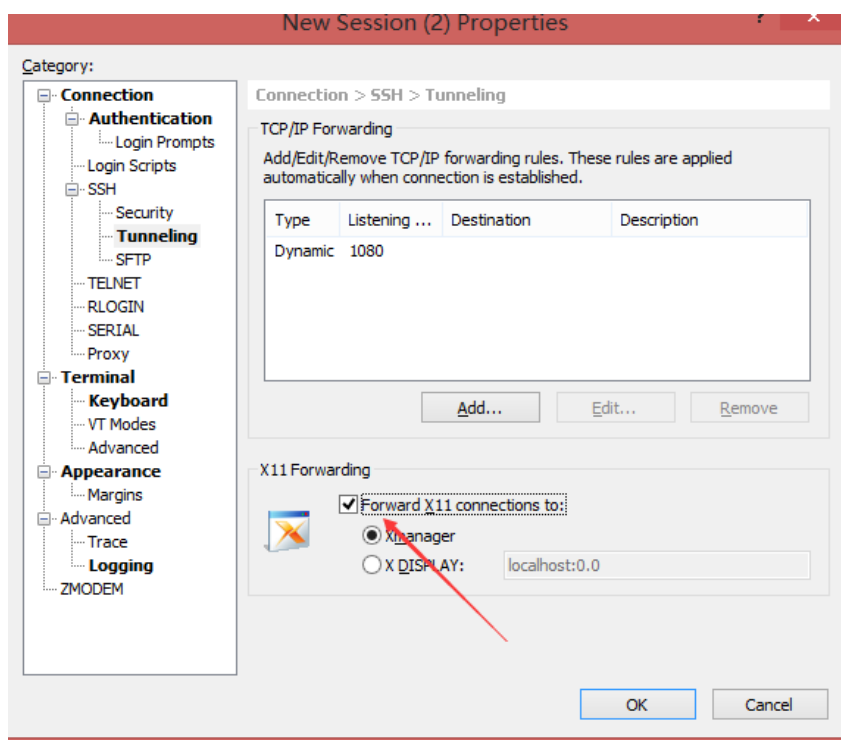
然后点击隧道，如下图所示：



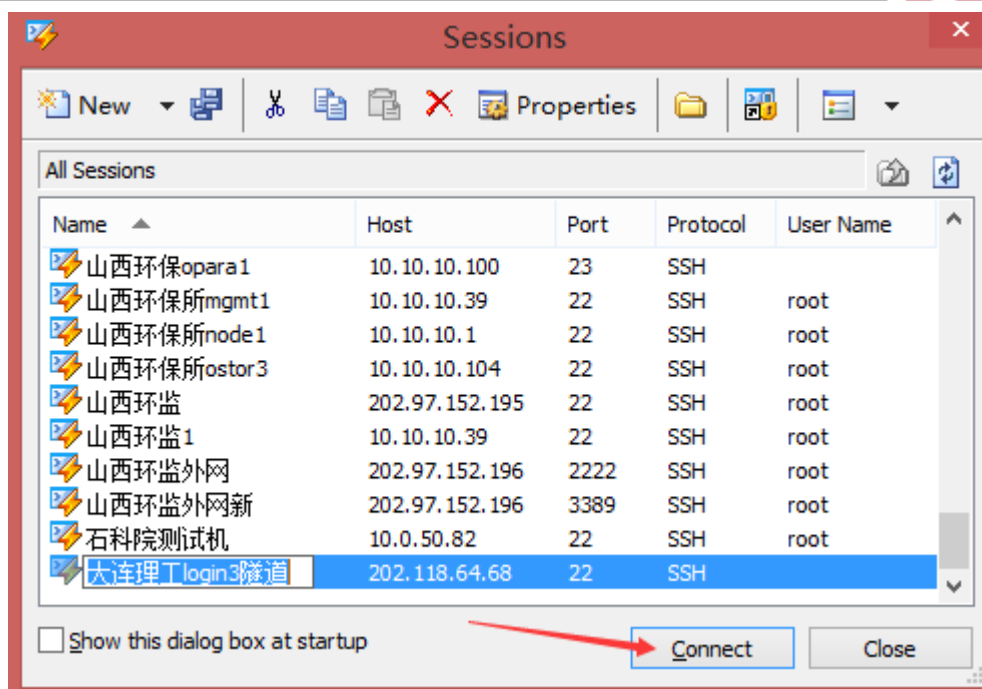
點選 add 按钮后，选好重定向种类和端口号：



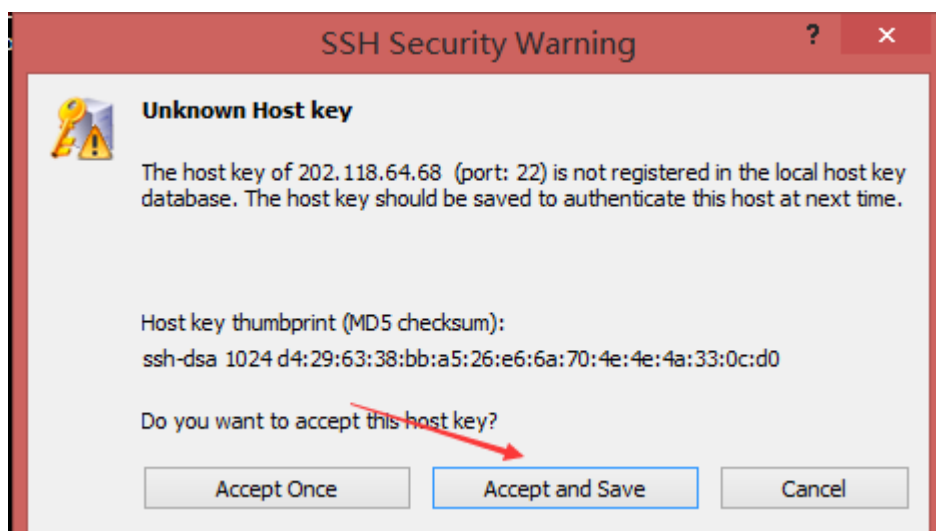
确认后勾选 X11 forwarding



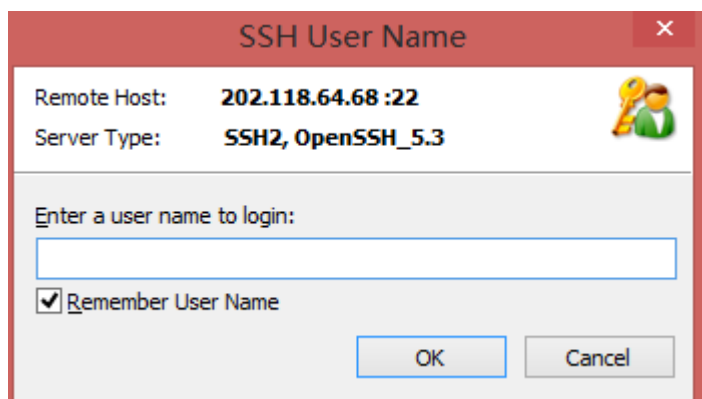
再点击 ok。然后用新建的隧道连接登陆节点。

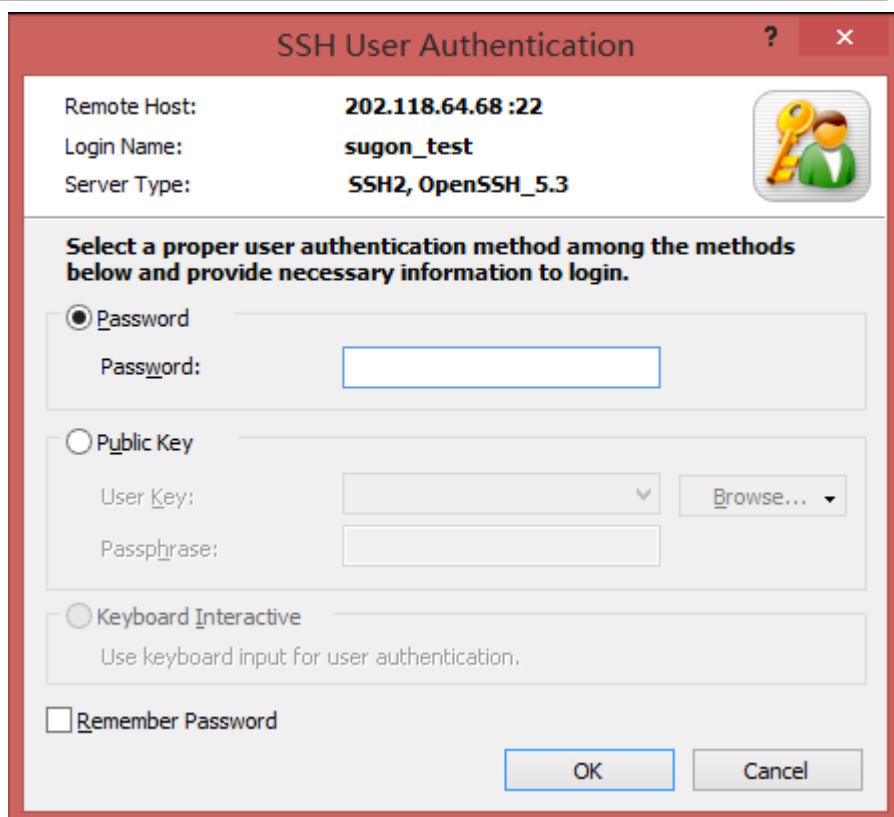


确认 hostkey



输入用户名和密码:





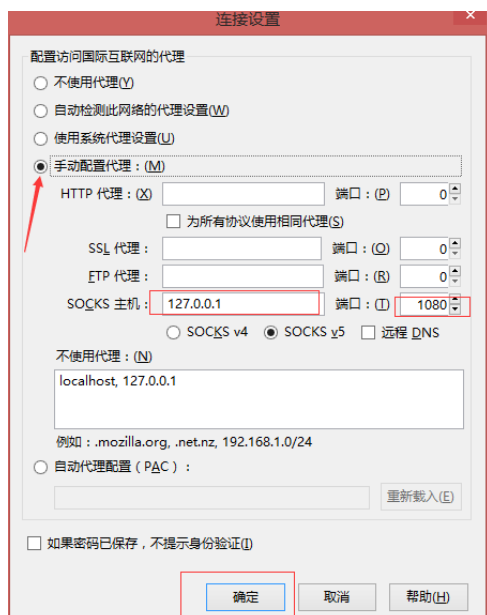
接下来，在浏览器里设置代理，举 firefox 为例



接下里，选择高级->网络->连接->设置：



设置手动配置代理，和响应的 IP 和端口号：



点击确定，配置完成。接下来用户就可在浏览器里输入 GridView 的 IP 和端口号进行登录了。

用户可以使用注册的用户名和密码来登陆 GridView Web 界面，或者用 ssh 的方式采用客户端软件来登录登录节点：



```
[sugon_pre@node39 sugon_pre]$ pwd
/public/home/sugon_pre
```

用户通过客户端软件 ssh 登录到节点后，会自动登录到自身的家目录：/public/home/用户名；

在该路径下系统已经自动生成三个文件夹：software, sourcecode, 和 app。建议用户将使用的软件装到 software 文件夹，将源代码相关文件放到 sourcecode 目录下，将用户的算例等文件放到 app 文件夹下。其中 software 目录下内置 profile.d 文件夹，其中包含集群系统的所有环境变量文件。

1.2 文件传输

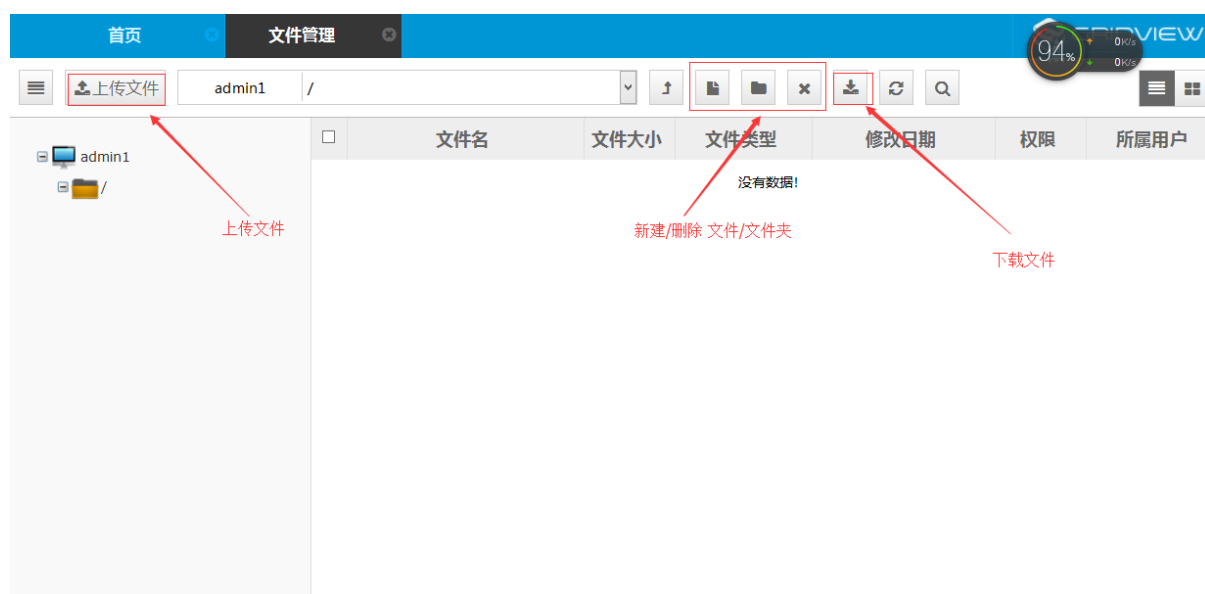
用户上传模型文件或者下载结果文件都需要进行文件传输。用户有两种方式可以同超算中心的集群系统进行文件传输。一种是通过 FTP 服务的方式（暂时不开通），一种是利用 GridView Web 界面基于 SSH 协议进行。Winscp

1.2.1 基于 GridView Web 界面进行文件传输

用户用用户名和密码登录到 GridView 的 web 界面后，可以选择如下操作，进入用户的文件管理界面：



在该界面下可以进行上传文件/下载文件/以及新建文件夹，删除文件夹等，如下图所示：

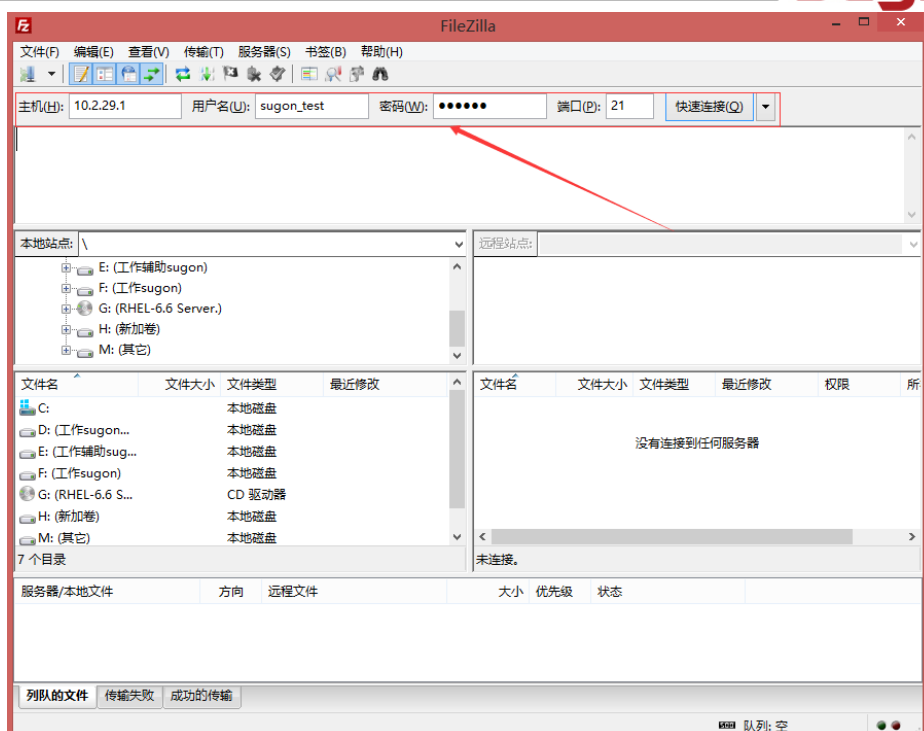


1.2.2 FTP 服务传输文件

用户可以通过 FTP 服务进行文件的传输。登录时，需要使用主机名和主机密码。注意 FTP 服务是 24 小时连续服务的。

在 windows 系统下，推荐用户使用免费的 ftp 软件 Filezilla，下载的官方网址为：
<http://sourceforge.net/projects/filezilla/>

安装 Filezilla 软件后，打开软件界面，可以按照下图进行配置，完成后单击快速链接，或者回车键既可以登陆 ftp 站点。在文件菜单中，可以选择“站点管理器”保存 ftp 地址，用户名和密码。下次登录时只需要从“站点管理器”中选择相应的 FTP 地址即可以直接登陆。



用户也可以选择使用其他 FTP 软件如 cuteftp, flashfxp 等进行登录。当需要下载或上传的文件数量较多时, 建议用 tar 命令将文件进行打包后下载或上传, 加快文件传输速度, 减少文件传输的出错概率。

注意, 因为 windows 操作系统和 linux 操作系统对于个别字符的处理方式不同, 上传的文本文件在 linux 系统下, 需要用 dos2unix 命令先处理一下, 然后再继续使用, 否则可能导致作业提交出错。

1.3 提交作业

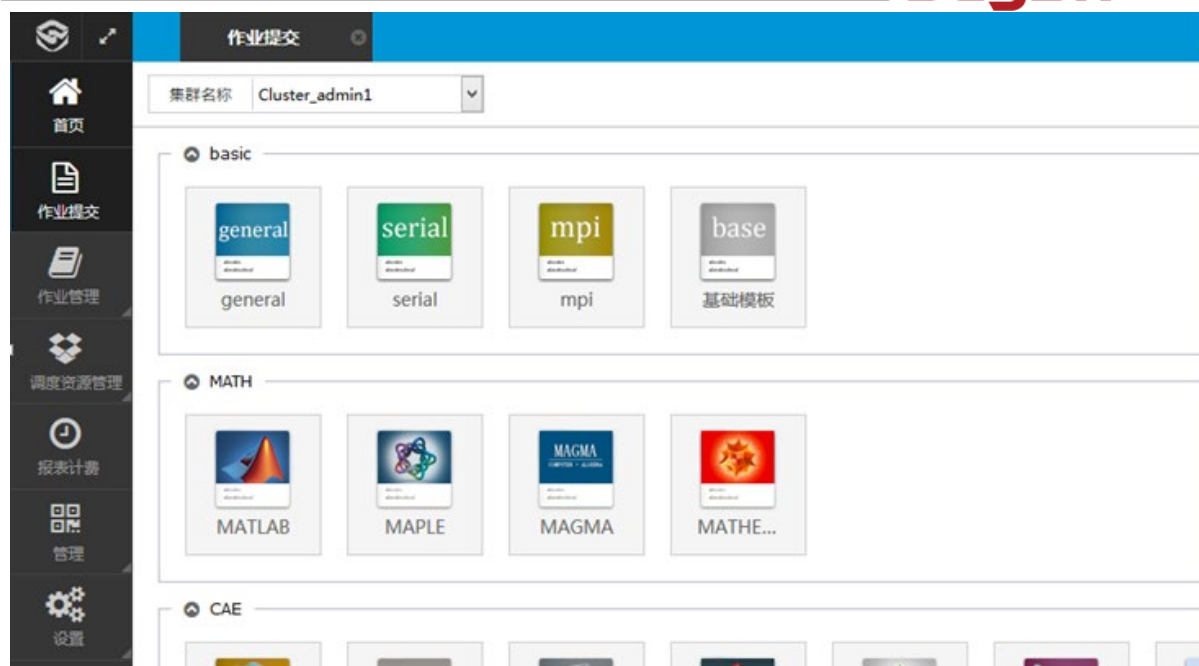
在超算中心的集群系统上提交作业, 用户有两种方式可供选择。一种是通过 GridView Web 界面提交, 一种是通过命令行的方式, 用 pbs 命令提交。

用户严禁使用任何前台或者后台方式直接运行程序, 所有计算任务必须通过 GridView web 界面提交或命令行用 PBS 提交。否则, 后果自负。

具体提交作业方法如下所示。

1.3.1 用 GridView web 界面提交作业

用户通过 GridView web 界面提交作业, 系统缺省可用 portal 有四种, general/serial/mpi/基础模板, 如下图所示:



其中：

- general 为通用模板，既可提交并行作业，也可以提交串行作业。
- Serial 为串行作业使用模板，可以提交串行和支持 openmp 的作业，串行作业仅能一个核心内运行，openmp 仅能在一个单独的节点上运行。
- Mpi 为并行作业使用模板。可以提交并行作业，作业可以在多个节点上并行执行。
- 基础模板同样可以提交并行作业和串行作业。

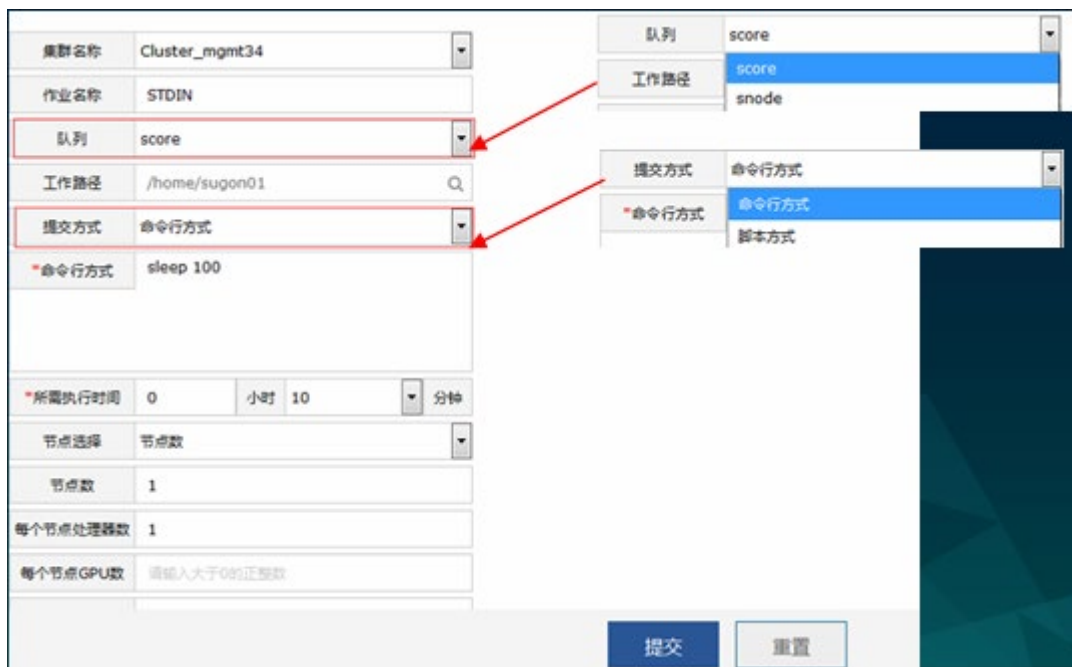
4.3.1.1 基础模板作业提交

用户选择基础模板后，出现如下界面：

集群名称	Cluster_mgmt34	▼
作业名称	STDIN	
队列	score	▼
工作路径	/home/sugon01	🔍
提交方式	命令行方式	▼
*命令行方式	sleep 100	
*所需执行时间	0	小时 10 分钟
节点选择	节点数	▼
节点数	1	
每个节点处理器数	1	
每个节点GPU数	请输入大于0的正整数	
<div style="float: right;"> <div style="margin-right: 10px;">提交</div> <div>重置</div> </div>		

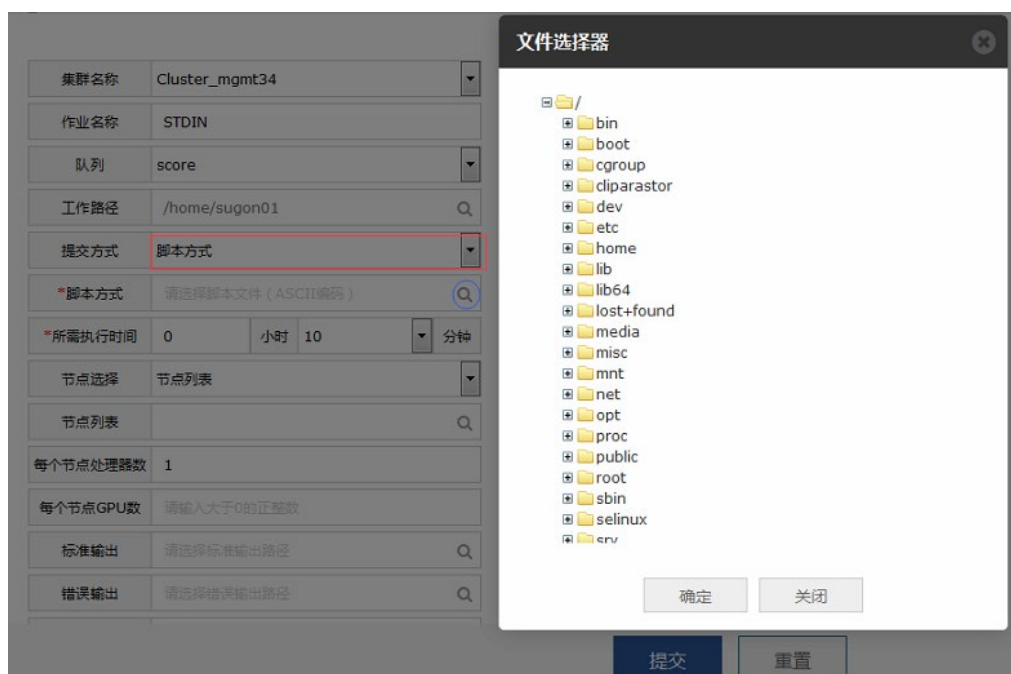
在该界面，标记为*的为必须填写的栏目。

基础模板作业提交时，用户可以选择默认的集群(目前，大工超算中心集群系统为单集群)，填写作业名称，或使用缺省名。用户可以根据自身作业的需要，选择不同的作业队列，关于队列的详细说明参见 3.2 节计算队列分布。工作路径默认为用户的家目录，用户可以根据需要选择。作业提交方式有命令行方式和脚本方式两种。用户仅能选择其中的一种。



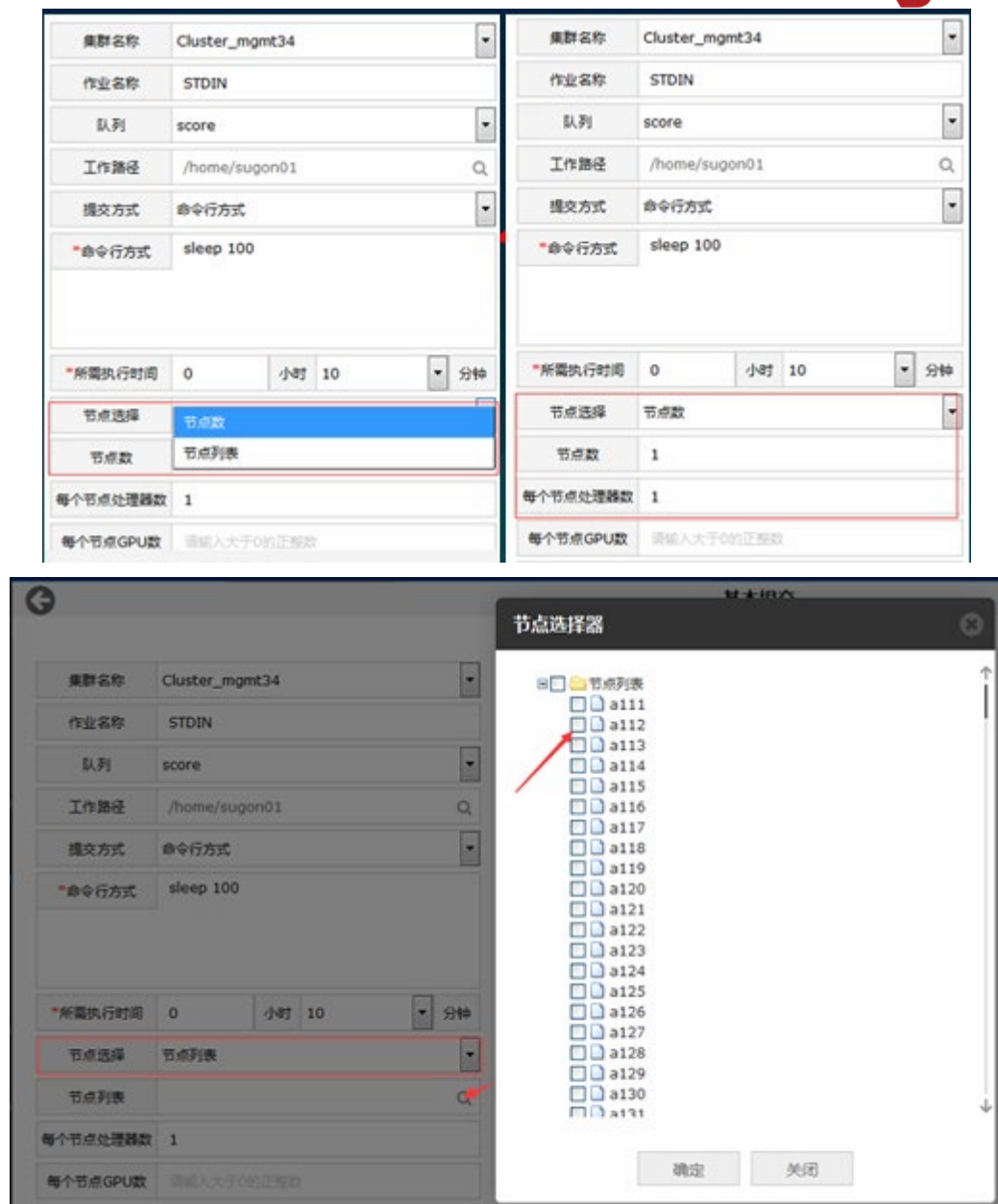
The screenshot shows a web-based job submission form. The 'Queue' (队列) dropdown is set to 'score'. The 'Work Path' (工作路径) is '/home/sugon01'. The 'Submission Method' (提交方式) dropdown is set to 'Command Line' (命令行方式), and the 'Command Line Method' (命令行方式) dropdown is also set to 'Command Line' (命令行方式). The command field contains 'sleep 100'. The 'Required Execution Time' (所需执行时间) is set to 0 hours and 10 minutes. The 'Node Selection' (节点选择) dropdown is set to 'Node Count' (节点数), and the 'Node Count' (节点数) is 1. The 'Processors per Node' (每个节点处理器数) is 1. The 'GPUs per Node' (每个节点GPU数) field is empty with a placeholder '请输入大于0的正整数'. At the bottom, there are 'Submit' (提交) and 'Reset' (重置) buttons.

如果选择用脚本方式提交作业，用户可以选择已经写好的脚本文件，如下图所示：



The screenshot shows the same job submission form, but the 'Submission Method' (提交方式) dropdown is now set to 'Script' (脚本方式). The 'Script Method' (脚本方式) dropdown is set to 'Select Script File (ASCII encoding)' (请选择脚本文件 (ASCII编码)). A file selector dialog titled '文件选择器' (File Selector) is open, showing a directory tree with various system directories like bin, boot, cgroup, etc. The 'Submit' (提交) and 'Reset' (重置) buttons are visible at the bottom.

用户根据自己作业情况输入作业所需的执行时间，然后可以选择作业运行的节点数或节点列表。如下图所示：



集群名称 Cluster_mgmt34

作业名称 STDIN

队列 score

工作路径 /home/sugon01

提交方式 命令行方式

*命令行方式 sleep 100

*所需执行时间 0 小时 10 分钟

节点选择 节点数

节点数 节点列表

每个节点处理器数 1

每个节点GPU数 请输入大于0的正整数

节点选择 节点数

节点数 1

每个节点处理器数 1

每个节点GPU数 请输入大于0的正整数

节点选择器

节点列表

a111

a112

a113

a114

a115

a116

a117

a118

a119

a120

a121

a122

a123

a124

a125

a126

a127

a128

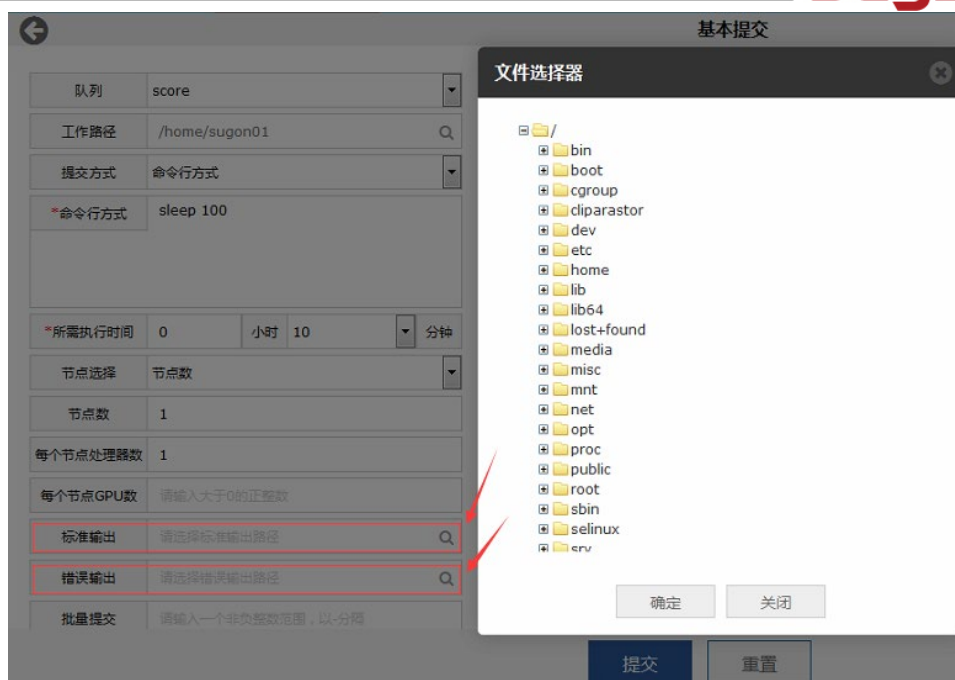
a129

a130

a131

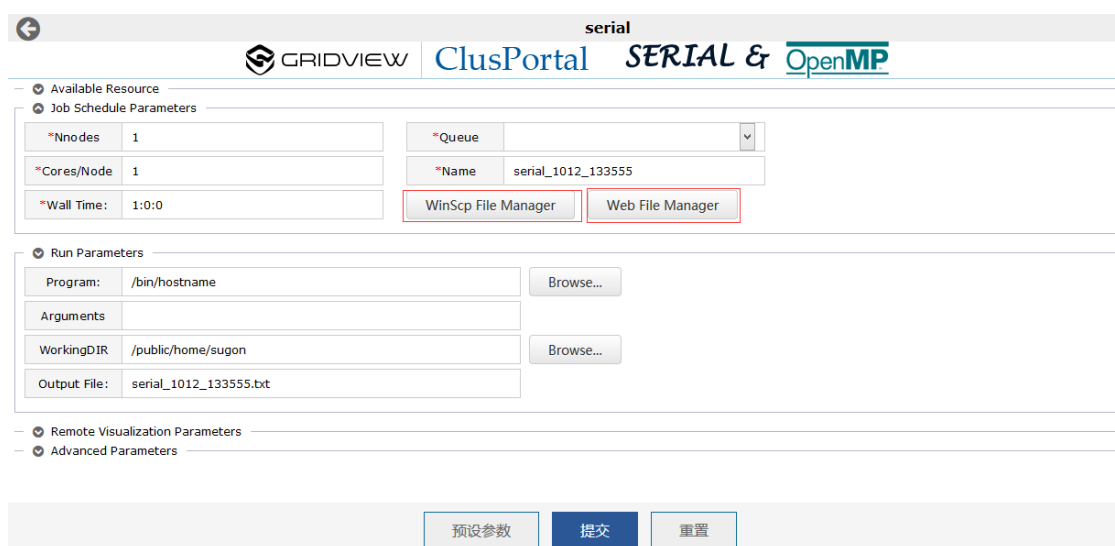
确定 关闭

节点数或节点列表选择完毕后，用户可以指定标准输出文件和错误输出文件名字和路径，也可以用缺省设置。



4.3.1.2 Serial portal 模板作业提交

Serial portal 模板作业提交界面如下，具体提交方式同基础模板方式类似：



CPU Time: 如果部署了 ClusQuota 集群资源配额计费系统，将显示您目前可用的机时配额，计量单位为“CPU*Hours”。例如，系统显示目前可用配额为 120 CPU*Hours，表明最多可以用 12 个 CPU 并行运算 10 小时。本次计算任务结束之后，将按照“CPU 并行数*实际运行时间”扣除相应的机时配额。队列状态的“Charge Rate”一栏表示它们在 ClusQuota 系统中的计费比率，例如，优先权高的工作队列其计费比率也相应要高一些。

Nnodes: 本次计算任务需要使用多少个节点。本 portal 中只能选 1

Cores/Node: 本次计算任务每个节点需要使用多少 CPU 核。

Wall Time: 本次计算任务预计将运行多长时间。根据系统的调度策略，WallTime 较短的任务将有机会优先运行；不过须注意，一旦 WallTime 时间到了而程序尚未运行结束，本次任务将被强行终止。因此请合理预估 WallTime 的长短。此外，如果部署了 ClusQuota 集群资源配额计费系统，本次任务申请的机时资源不允许超过您目前可用的机时配额。

Queue: 本次计算任务将使用的工作队列。

Name: 本次计算任务的名称。

WinScp File Manager: 启动 WinSCP 程序上传/下载计算任务的输入输出文件。

Web File Manager: 启动文件管理

Program: 选择本次计算任务的可执行程序或命令。

Arguments: 如果应用程序运行时需要提供自定义的参数, 请在此输入。

Working DIR: 本次计算任务的工作目录。

Output File: 计算过程中的标准输出和标准错误输出信息, 将被重定向保存为文件。

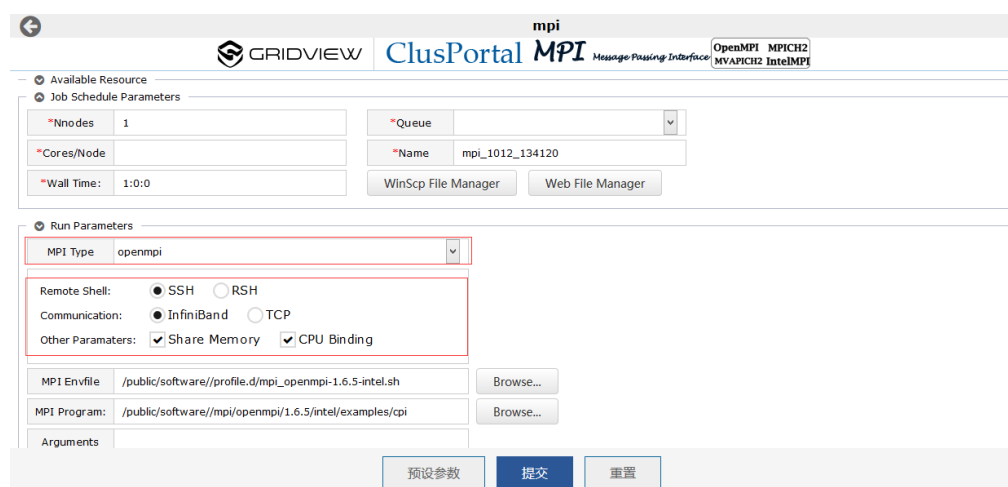
PBS Options: 如果需要手动添加 PBS 作业的高级参数, 可在此处设置。这类参数的行首必须包含“#PBS”关键字, 将被加到 PBS 脚本文件的开始处。该选项默认无需设置。

Pre Commands: 如果运行 mpirun 命令之前需要做前处理操作, 可在此处设置相关命令参数, 命令行格式必须遵循 bash 脚本规范。该选项默认无需设置。

Post Commands: 如果在 mpirun 命令运行结束之后需要做后处理操作, 可在此处设置相关命令参数, 命令行格式必须遵循 bash 脚本规范。该选项默认无需设置。

4.3.1.3 MPI portal 模板作业提交

Mpi portal 模板作业提交界面如下, 具体提交方式同基础模板方式类似:



多数参数说明请见 Serial Portal 模板作业提交, 下列参数为 MPI portal 模板独有的, 说明如下:

MPI Type: 选择 MPI 并行环境, 如 Open MPI 或 Intel MPI。

Remote Shell: 多节点并行任务, MPI 初始化并行环境时, 节点之间的访问模式。建议采用默认的 SSH 模式。

Communication: 节点连接方式, InfiniBand 和 TCP 两种方式。建议采用默认的 InfiniBand 方式。

Other Paramasters: 多节点并行任务, 节点之间数据交换采用何种网络。如果勾选“Share Memory”选项, 表示同一节点内的 MPI 进程采用共享内存方式进行数据交换; 如果勾选“CPU Binding”选项, 表示将 MPI 进程与固定的 CPU 核心绑定, 防止进程漂移。开启这两个选项通常可以提高 MPI 程序的运行速度。

MPI Envfile: 文件环境变量配置文件。用户可浏览集群, 选择配置文件。

MPI Program: 选择本次计算任务的可执行程序。本 Portal 安装后提供基本的 MPI 版本 CPI 可进行测试, 用户也可浏览选择集群中的其他 MPI 程序。

Arguments: 如果 MPI 应用程序运行时需要提供自定义的参数, 请在此输入。

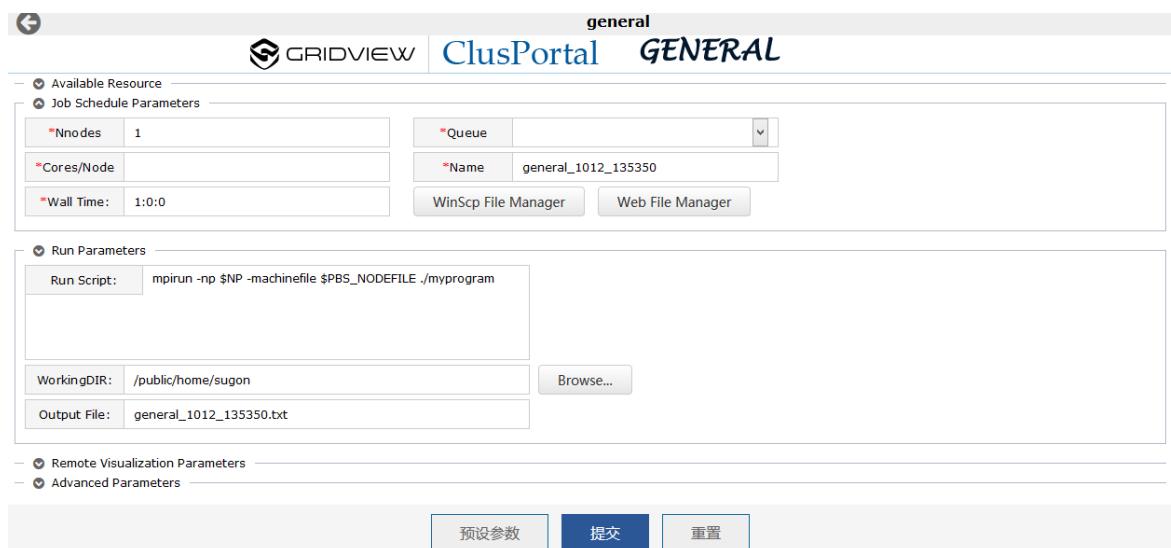
MPI Options: 如果需要手动添加 MPI 并行时的高级参数, 可在此处设置, 这些参数将被传递成为 mpirun 命令行参数的一部分。该选项默认无需设置。

Pre Commands: 如果运行 mpirun 命令之前需要做前处理操作, 可在此处设置相关命令参数, 命令行格式必须遵循 bash 脚本规范。该选项默认无需设置。

Post Commands: 如果在 mpirun 命令运行结束之后需要做后处理操作, 可在此处设置相关命令参数, 命令行格式必须遵循 bash 脚本规范。该选项默认无需设置。

4.3.1.4 General portal 模板作业提交

General portal 模板作业提交界面如下，具体提交方式同基础模板方式类似：



参数说明同基础模板/Serial portal 模板/MPI portal 模板。

1.3.2 用命令行方式提交作业

用户可以通过 ssh 客户端软件直接登录到集群系统的登录节点上，用命令行的方式提交作业。请注意，用户可以在登录节点查看文件，编辑文件，查看作业，查看资源使用情况等。但用户不允许在登陆节点上运行计算程序。

在登录节点上提交作业只允许通过 pbs 作业调度系统提交作业。Pbs 提交作业是通过 qsub 命令来执行，qsub 命令通过脚本文件提交作业到作业管理系统，具体格式如下：

qsub <PBS 作业脚本>

PBS 作业脚本本质上是一个 shell 脚本，注释行以“#”开头，pbs 运行参数以“#PBS”开头。

PBS 作业脚本里可以直接调用 shell 命令和系统命令。脚本里运行参数说明如下表所示：

运行参数	说明
-a <作业开始运行的时间>	向 PBS 系统指定作业运行的开始时间。 作业运行时间格式为： [[[[CC]YY]MM]DD]hhmm[.SS]
-A <用户名>	使用不同的用户来提交作业，缺省使用当前用户名
-o <标准输出文件的路径> -e <标准错误输出的路径>	该参数指定标准错误输出的位置，缺省的情况下，PBS 系统把标准输出和标准错误输出放在用户 qsub 命令提交作业的目录下。 标准错误输出：<作业名>.o<作业号> 标准错误输出：<作业名>.e<作业号>

	路径使用如下格式标准： [<节点名>:] <路径名>
-N <作业名>	指定提交的作业名
-q <目标队列>	指定作业提交的目标队列，其中目标队列可以是目标队列、目标节点名或者是目标节点上的队列。如果目标队列是一个路由队列，那么服务器可能把作业路由到新的队列中。如果该参数没有指定，命令 qsub 会把作业脚本提交到缺省的队列中。
-l <申请资源列表>	<p>该参数指定作业脚本申请的 PBS 系统资源列表。</p> <p>申请资源列表使用如下格式：</p> <p style="text-align: center;"><资源名>=[<数量>] [, 资源名=[<数量>] , ...]</p> <p>例如作业希望申请在双路节点上申请 5 个 CPU 资源的情况，则可以在脚本中如下：</p> <pre>#PBS -l nodes=2:ppn=2+1:ppn=1</pre>
登陆 SHELL 继承来的变量	包括\$HOME, \$LANG, \$LOGNAME, \$PATH, \$MAIL, \$SHELL 和 \$TZ。
\$PBS_O_HOST	qsub 提交的节点名称
\$PBS_O_QUEUE	qsub 提交的作业的最初队列名称
\$PBS_O_WORKDIR	qsub 提交的作业的绝对路径
\$PBS_JOBID	作业被 PBS 系统指定的作业号
\$PBS_JOBNAME	用户指定的作业名，可以在作业提交的时候用 qsub -N <作业名> 指定，或者在 PBS 脚本中加入 #PBS -N <作业名>。
\$PBS_NODEFILE	<p>PBS 系统指定的作业运行的节点名。该变量在并行机和机群中使用。当在 PBS 脚本中用 #PBS -l nodes=2:ppn=2 指定程序运行的节点数时，可以使用 \$PBS_NODEFILE 在脚本中引用 PBS 系统指定的作业运行的节点名。比如：</p> <pre>#PBS -l nodes=2:ppn=2</pre> <pre>mpirun -np 4 -machinefile \$PBS_NODEFILE <程序名></pre>
\$PBS_QUEUE	PBS 脚本在执行时的队列名

在集群计算系统上提交的作业通常分为如下几类：

- 普通的串行程序，仅使用一个计算核心即可。
- 同一节点内运行的 OpenMP 或基于 threads 的共享内存程序，仅使用一个节点内的多个核心。
- 利用消息传递方式的跨节点的 MPI 并行程序。

➤ 复杂的 OpenMP+MPI 混合并行程序。

4.3.2.1 普通的串行程序作业脚本示例：

脚本	脚本说明
#PBS -N test	指定作业名：test
#PBS -l nodes=1:ppn=1	指定作业资源：1 个节点，1 个核心
#PBS -l walltime=12:00:00	指定作业运行时间 12 天
#PBS -q blades	指定作业运行队列 blades
cd \$HOME/test/	进入工作目录 \$HOME/test
./a.out > \$HOME/result/a.result	执行命令，并将结果重定向到 ./a.out > \$HOME/result/a.result 文件中。

上面为一个普通的串行作业脚本示例，用户可以通过 qsub 命令，加上该脚本的文件名，就可提交作业。脚本中给指定了作业名称，作业所需资源，作业的运行时间，作业运行所用队列，以及作业执行的目录。用户把作业的可执行文件和目录更换到用户的自己的内容就可成功提交用户自己的作业。

4.3.2.2 共享内存并行作业

脚本	脚本说明
#PBS -N test_OpenMP	指定作业名：test_OpenMP
#PBS -l nodes=1:ppn=10	指定作业资源：1 个节点，10 个核心
#PBS -l walltime=12:00:00	指定作业运行时间 12 天
#PBS -q blades	指定作业运行队列 blades
export OMP_NUM_THREADS=10	设置 OpenMP 线程数环境变量
cd \$HOME/test_OpenMP/	进入工作目录 \$HOME/test_OpenMP
./a.out > \$HOME/result/a.result.OpenMP	执行命令，并将结果重定向到 ./a.out > \$HOME/result/a.result.OpenMP 文件中。

该类作业包括 OpenMP 并行方式的作业，也包括不使用 OpenMP 而是通过 POSIX 等系统底层所编写的多线程程序。

用户请注意，这里的用户申请的节点数，核心数，需要同 OMP_NUM_THREADS 一致，且该数值不应该超出队列中单节点的物理核心数。同时，还要注意用户可执行程序的输入文件如需要设定 OpenMP 的核心数，也要同上面的参数设置一致。

4.3.2.3 MPI 并行作业

脚本	脚本说明
#PBS -N test_MPI	指定作业名：test_MPI
#PBS -l nodes=2:ppn=20	指定作业资源：2 个节点，40 个核心

#PBS -l walltime=12:00:00	指定作业运行时间 12 天
#PBS -q blades	指定作业运行队列 blades
echo This jobs is \$PBS_JOBID@\$PBS_QUEUE	打印作业名字和作业所属队列信息
cd \$PBS_O_WORKDIR	进入工作目录 \$PBS_O_WORKDIR
mpirun -np 40 -machinefile \$PBS_NODEFILE ./vasp	运行 Vasp 可执行文件，指定 40 个线程，以及所用的节点名

该类作业为 MPI 并行方式的作业，请用户注意用户设定的核心数数值不应该超出队列中单节点的物理核心数。

4.3.2.4 OpenMP+MPI 混合并行作业

脚本	脚本说明
#PBS -N test_OMP_MPI	指定作业名：test_OMP_MPI
#PBS -l nodes=2:ppn=20	指定作业资源：2 个节点，40 个核心
#PBS -l walltime=12:00:00	指定作业运行时间 12 天
#PBS -q blades	指定作业运行队列 blades
echo This jobs is \$PBS_JOBID@\$PBS_QUEUE	打印作业名字和作业所属队列信息
cd \$PBS_O_WORKDIR	进入工作目录 \$PBS_O_WORKDIR
mpirun -np 40 -machinefile \$PBS_NODEFILE -mca btl self,openib,sm ./test_OMP_MPI	运行 test_OMP_MPI 可执行文件，指定 40 个线程，以及所用的节点名，并指定共享内存，和用 InfiniBand 网络计算

该类作业为 OpenMP+MPI 混合并行方式的作业，请用户注意用户设定的核心数数值不应该超出队列中单节点的物理核心数。

4.3.2.5 其它类的作业提交

超算中心的集群系统除了刀片服务器外，还包括胖节点，mic 节点和 gpu 节点。

对于胖节点，mic 节点和 gpu 节点上的作业提交，需要用户选择合适的队列提交作业(作业队列必须包含用户打算使用的节点)。其用户自身的计算程序需要支持自身的资源需求，也就是 gpu 队列上提交的作业，计算程序需要支持 gpu，胖节点队列上提交的作业，计算程序需要支持胖节点；mic 队列上提交的作业，计算程序需要支持 mic 节点。

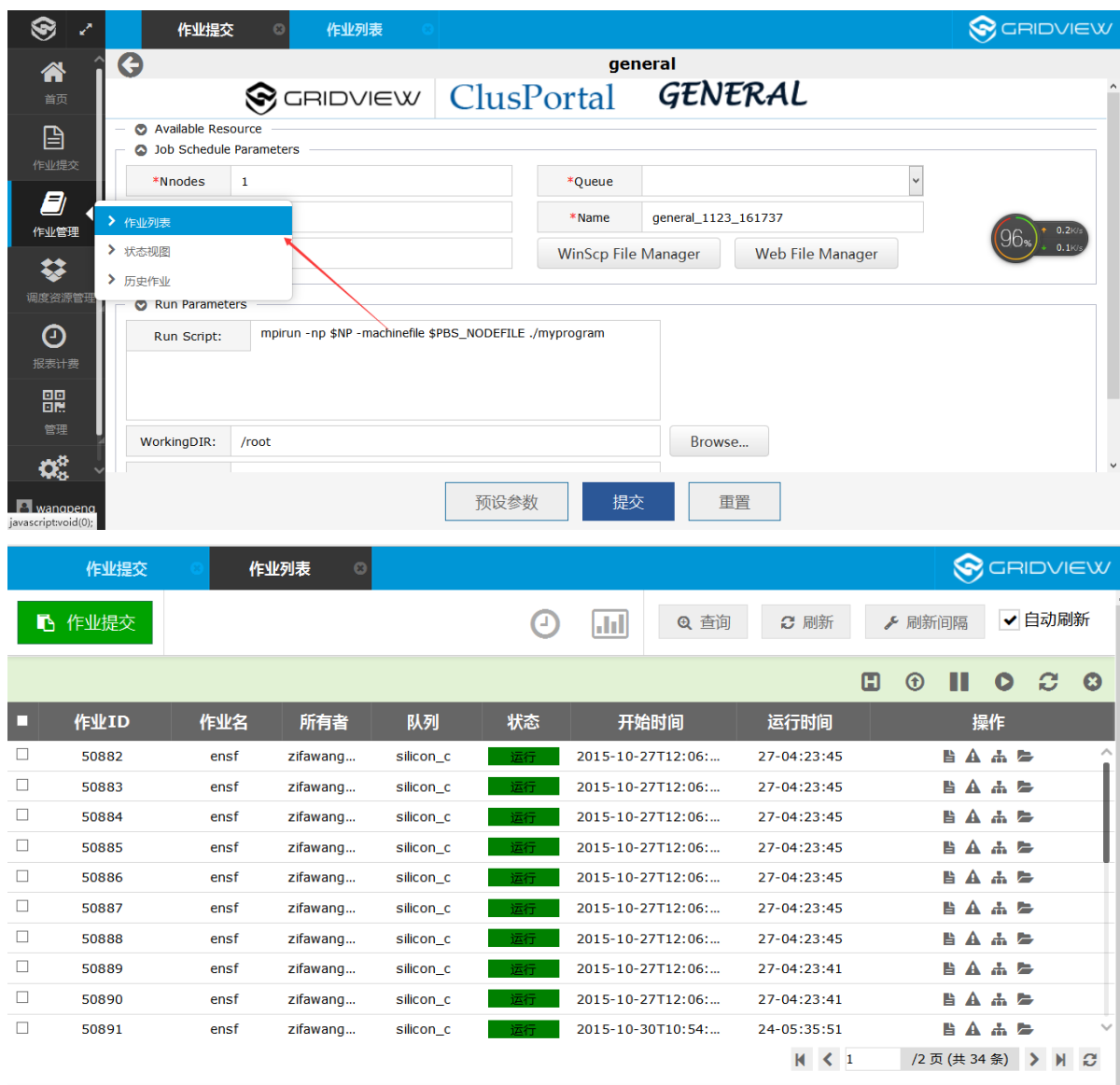
1.4 作业管理

在超算中心的集群系统上进行作业管理，用户有两种方式可供选择。一种是通过 GridView Web 界面，一种是通过命令行的方式，用 pbs 命令管理查询。

1.4.1 用 GridView Web 界面管理作业

1.4.1.1 用户作业状态查询

提交过作业后，可以点击作业列表菜单查询作业状态，如下图所示：



The screenshot shows the GridView Web interface. The top navigation bar has '作业提交' (Job Submission) and '作业列表' (Job List). The '作业列表' tab is active, displaying a table of jobs. A red arrow points to the '作业列表' menu item in the left sidebar.

作业ID	作业名	所有者	队列	状态	开始时间	运行时间	操作
50882	ensf	zifawang...	silicon_c	运行	2015-10-27T12:06:...	27-04:23:45	[Icons]
50883	ensf	zifawang...	silicon_c	运行	2015-10-27T12:06:...	27-04:23:45	[Icons]
50884	ensf	zifawang...	silicon_c	运行	2015-10-27T12:06:...	27-04:23:45	[Icons]
50885	ensf	zifawang...	silicon_c	运行	2015-10-27T12:06:...	27-04:23:45	[Icons]
50886	ensf	zifawang...	silicon_c	运行	2015-10-27T12:06:...	27-04:23:45	[Icons]
50887	ensf	zifawang...	silicon_c	运行	2015-10-27T12:06:...	27-04:23:45	[Icons]
50888	ensf	zifawang...	silicon_c	运行	2015-10-27T12:06:...	27-04:23:45	[Icons]
50889	ensf	zifawang...	silicon_c	运行	2015-10-27T12:06:...	27-04:23:41	[Icons]
50890	ensf	zifawang...	silicon_c	运行	2015-10-27T12:06:...	27-04:23:41	[Icons]
50891	ensf	zifawang...	silicon_c	运行	2015-10-30T10:54:...	24-05:35:51	[Icons]

Page 1 / 2 (共 34 条)

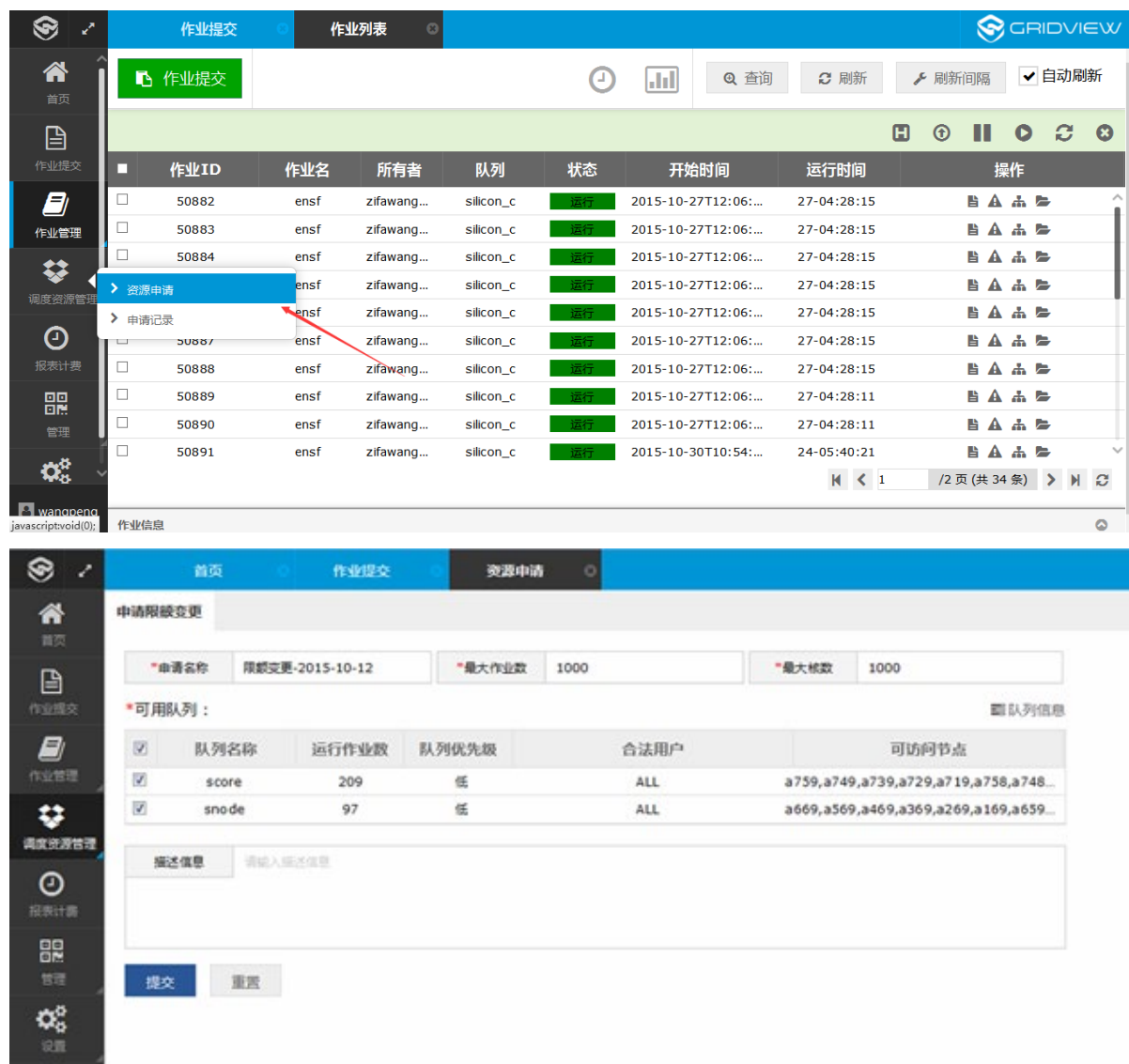
1.4.1.2 用户作业管理

用户可以对列表中队列进行操作。注意用户可以看到其他用户作业的运行状态，但用户并不能对其他用户的作业进行操作，仅能对于自己的作业进行操作。操作方式是点选左上角的快捷按钮，可以删除作业，保留作业，挂起作业，恢复作业，释放作业，以及重新运行作业等。如下图所示：



1.4.1.3 用户可用资源申请

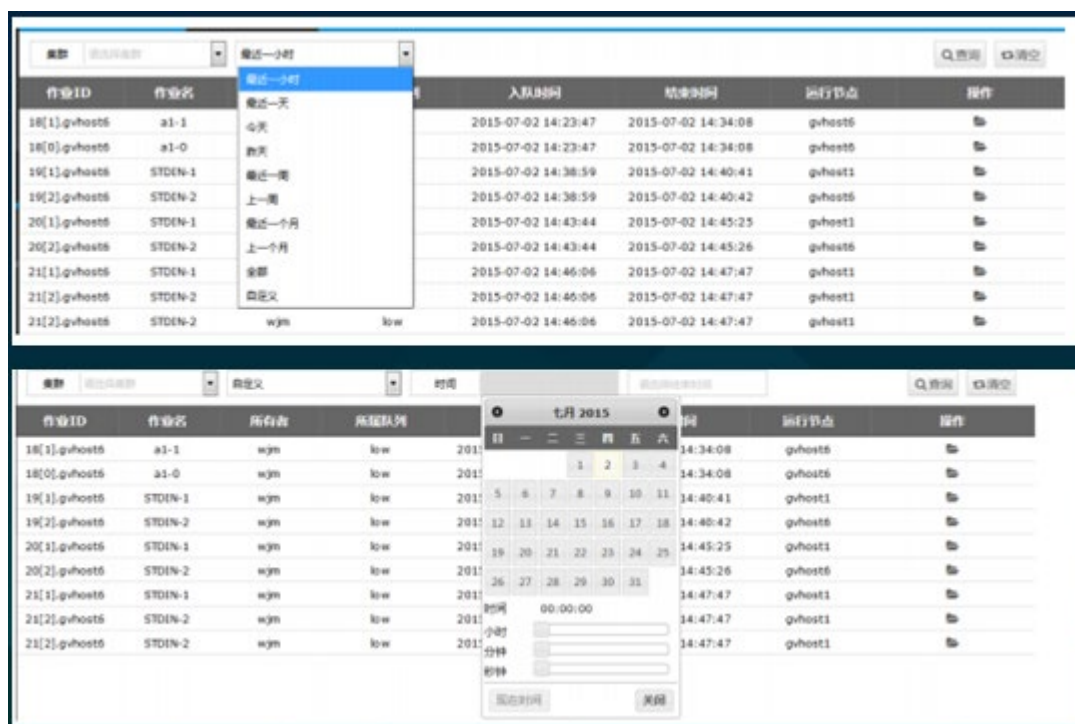
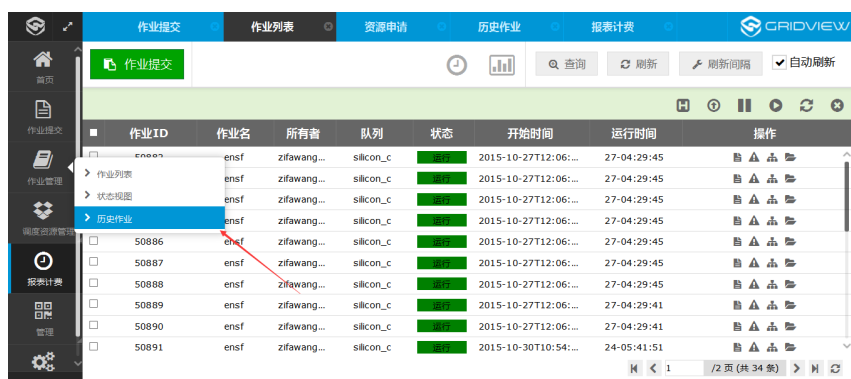
用户还可以对自身可用资源进行申请(最大作业数, 最大核心数), 需要点击左侧的资源申请菜单项:



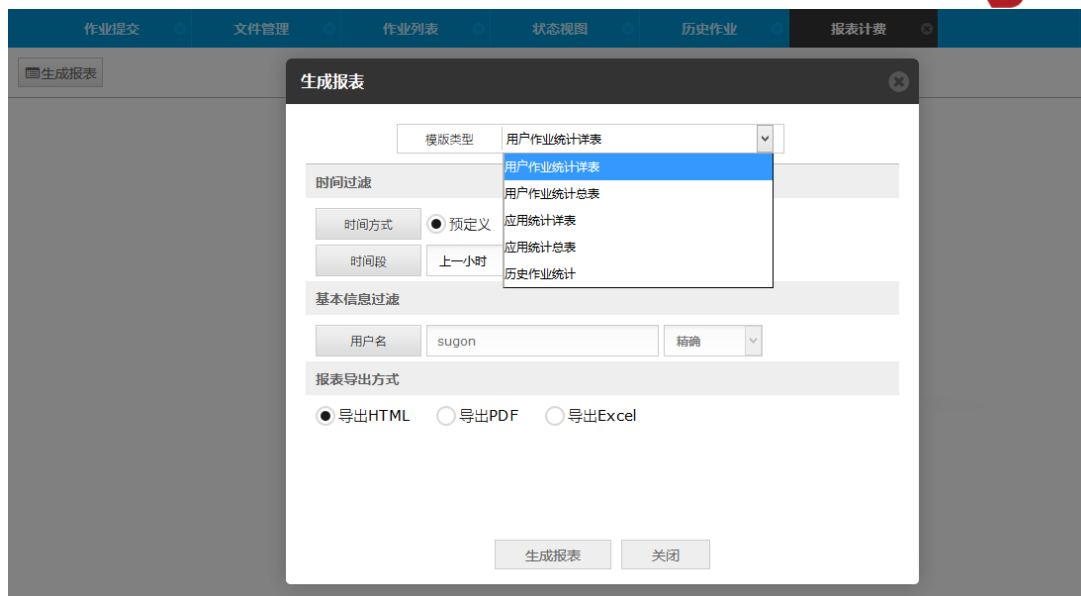
提交申请后, 等待系统管理员审批通过, 即可使用申请的新的资源量。

1.4.1.4 用户历史作业查询和报表生成

用户可以通过点击左侧的历史作业菜单，来查询自己的历史作业运行情况，如下图所示：



用户还可对历史作业按条件查询，并生成报表。



用户作业统计报表详表

时间范围: 2015-07-02 14:00:00--2015-07-02 15:00:00

生成时间: 2015-07-02 15:10:19

操作人员: wjm

用户: wjm

日期	作业总数	核数	CPU使用时间(小时)	WallTime(小时)	内存使用量(MB)	Efficiency(%)
2015-07-02	9	9	0.0000	5.3917	24.91	0.00
合计	9	9	0.0000	5.3917	24.91	0.00

用户作业统计报表总表

时间范围: 2015-07-02 14:00:00--2015-07-02 15:00:00

生成时间: 2015-07-02 15:10:46

操作人员: wjm

用户名	队列名	CPU使用时间		WallTime		内存使用量		Efficiency
		数值(小时)	百分比(%)	数值(小时)	百分比(%)	数值(MB)	百分比(%)	百分比(%)
wjm	low	0.0000	0.00	5.3917	100.00	24.91	100.00	0.00
	总计(按队列)	0.0000		5.3917		24.91		0.00
总计(按用户)		0.0000		5.3917		24.91		0.00

1.4.2 用命令行方式管理作业

1.4.1.1 用户作业状态查询

提交过作业后，用户可以查看作业状态：

```
qstat
```

```
[root@login1 bin]# qstat
```

Job ID	Name	User	Time Use	S	Queue
1002.mgmt34	TT	sugon	00:00:00	C	score
1003.mgmt34	TT	sugon		R	score

```
[root@login1 bin]# qstat -a
```

```
mgmt34:
```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	S	Elap Time
1002.mgmt34	sugon	score	TT		0	1	24	--	01:00:00	C
1003.mgmt34	sugon	score	TT	18442	1	1	--	01:00:00	C	--

Job Id 作业标识符 PBS 自动指定

Username 用户名

Queue 队列名

Jobname 作业名 由作业提交人规定的名称

SessID Session 标识符 仅当作业运行时才有值

NDS 使用的节点数

TSK 使用的 cpu 数或 task 数

Req'd Memory 作业所需的内存数

Req'd Time 作业所需的 cpu 时间或 wall time

S 作业状态

Elap Time 作业已经运行的时间

查看节点状态:

```
pestat
```

```
[root@login1 bin]# pestat
```

node	state	load	phymem	ncpus	allmem	resi	usrs	tasks	jobidlist
a110	free	0.07	129010	24	129010	11229	7/2	0	
a111	free	0.00	129010	24	129010	11228	6/1	0	
a112	free	0.03	129010	24	129010	11224	6/1	0	
a113	free	0.06	129011	24	129011	11228	6/1	0	
a114	free	0.03	129010	24	129010	11230	6/1	0	
a115	free	0.00	129010	24	129010	11231	6/1	0	
a116	free	0.11	129011	24	129011	11226	6/1	0	
a117	free	0.02	129011	24	129011	11229	6/1	0	
a118	free	0.15	129010	24	129010	11226	6/1	0	
a119	free	0.02	129011	24	129011	11225	6/1	0	
a120	free	0.03	129010	24	129010	11226	6/1	0	
a121	free	0.02	129010	24	129010	11226	6/1	0	
a122	free	0.00	129010	24	129010	11227	6/1	0	
a123	free	0.02	129010	24	129010	11240	6/1	0	
a124	free	0.00	129010	24	129010	11227	6/1	0	
a125	free	0.00	129010	24	129010	11226	6/1	0	
a126	free	0.02	129010	24	129010	11228	6/1	0	
a127	free	0.01	129010	24	129010	11232	6/1	0	
a128	free	0.00	129010	24	129010	11224	6/1	0	
a129	free	0.08	129010	24	129010	11233	6/1	0	

node 节点名

state 节点状态

Load 节点负载

Phymem 物理内 MB

ncpus cpu/核心数

Allmem	分配内存数
Resi	寄存器数
usrs	用户数
Tasks	作业数
Joblist	作业列表

1.4.1.2 用户作业管理

用户可以对列表中队列进行操作。注意用户可以看到其他用户作业的运行状态，但用户并不能对其他用户的作业进行操作，仅能对于自己的作业进行操作。操作方式是点选左上角的快捷按钮，可以删除作业，保留作业，挂起作业，恢复作业，释放作业，以及重新运行作业等。如下图所示：

删除作业

```
qdel -p jobid
```

交换作业顺序

```
qorder 111.node1 112.node1
```

挂起作业：

```
qhold 111.node1
```

取消作业挂起

```
qrls 111.node1
```

更改作业运行队列：

```
qmove high 111.node1
```

1.5 源码编译

有些用户需要对自行安装源码编译的软件，或编译自编的代码，需要进行源码编译。目前在超算中心集群系统上部署的编译器和编译命令如下表所示：环境变量的选取。

1.5.1 c/fortran 编译器汇总

超算中心集群系统 C/fortran 编译器汇总

编译器	安装目录	版本	相关命令
GNU C	/usr/bin	4.4.7	gcc myprog.c
GNU C++	/usr/bin	4.4.7	g++ myprog.cpp
GNU Fortran	/usr/bin	4.4.7	gfortran myprog.f
PGI C	/public/software/pgi_2015	15.10	pgcc myprog.c
PGI C++	/public/software/pgi_2015	15.10	pgCC myprog.cpp
PGI Fortran77	/public/software/pgi_2015	15.10	pgf77 myprog.f
PGI Fortran90	/public/software/pgi_2015	15.10	pgf90 myprog.f90 pgf90 myprog.cuf
PGI Fortran95	/public/software/pgi_2015	15.10	Pgf95 myprog.f95 pgf95 myprog.f95

Intel C/C++	/public/software/compiler/intel/composer_xe_2013_sp1.0.080/	14.0.0	icc/icpc myprog.c
	/public/software/compiler/intel/composer_xe_2015.2.164/	15.0.2	icc/icpc myprog.c
	/public/software/intel_xe_2015u5	15.0.4	icc/icpc myprog.c
Intel Fortran	/public/software/compiler/intel/composer_xe_2013_sp1.0.080/	14.0.0	ifort myprog.f
	/public/software/compiler/intel/composer_xe_2015.2.164/	15.0.2	ifort myprog.f
	/public/software/intel_xe_2015u5	15.0.4	ifort myprog.f
Nvidia CUDA Toolkit	/public/software/cuda-6.0	6.0	nvcc myprog.cu
	/public/software/cuda-6.5	6.5	nvcc myprog.cu
	/public/software/cuda-7.5	7.5	nvcc myprog.cu

1.5.2 mpi 编译器汇总

超算中心集群系统 MPI 编译器汇总

MPI 实现方式	安装目录	版本	MPI 编译器	
			c 编译器	Fortran 编译器
mvapich2	/public/software/mpi/mvapich2/2.1/intel	2.1	mpicc mpicxx	mpif77 mpif90
	/public/software/mpi/mvapich2/2.1/gnu	2.1	mpicc mpicxx	mpif77 mpif90
mpich	/public/software/mpi/mpich/3.1.4/intel	3.1.4	mpicc mpicxx	mpif77 mpif90
	/public/software/mpi/mpich/3.1.4/gnu	3.1.4	mpicc mpicxx	mpif77 mpif90
openmpi	/public/software/mpi/openmpi/1.6.5/intel	1.6.5	mpicc mpicxx	mpif77 mpif90
	/public/software/mpi/openmpi/1.6.5/gnu	1.6.5	mpicc mpicxx	mpif77 mpif90
	/public/software/mpi/openmpi/1.8.5/intel	1.8.5	mpicc mpicxx	mpif77 mpif90
	/public/software/mpi/openmpi/1.8.5/gnu	1.8.5	mpicc mpicxx	mpif77 mpif90
IntelMPI	/public/software/mpi/intelmpi/5.0.2.044/	5.0.2	mpiicc mpiicpc	mpiifort
			mpicc mpigcc mpicxx mpigxx	mpif77 mpif90 mpifc

1.5.3 环境变量的选取

超算中心集群系统 C/fortran 编译器环境变量的选取

编译器	版本	环境变量选取命令
GNU	4.4.7	系统缺省值
PGI	15.10	source software/profile.d/pgi_2015-env.sh
Intel	14.0.0	source software/profile.d/compiler_intel-composer_xe_2013_sp1.0.080.sh
	15.0.2	source software/profile.d/compiler_intel-composer_xe_2015.2.164.sh
	16.0.1	source software/profile.d/compiler_intel-composer_xe_2016.u1.sh
Nvidia CUDA	6.0	source software/profile.d/cuda-6.0.sh
	6.5	source software/profile.d/cuda-6.5.sh
	7.5	source software/profile.d/cuda-7.5.sh

超算中心集群系统 mpi 编译器环境变量的选取

MPI 实现 方式	版本	环境变量选取命令
mvapich2	Intel 2.1	source software/profile.d/mpi_mvapich2-2.1-intel.sh
	Gnu 2.1	source software/profile.d/mpi_mvapich2-2.1-gnul.sh
mpich	Intel 3.1.4	source software/profile.d/mpi_mpich-3.1.4-intel.sh
	Gnu 3.1.4	source software/profile.d/mpi_mpich-3.1.4-gnu.sh
openmpi	Intel 1.6.5	source software/profile.d/mpi_openmpi-1.6.5-intel.sh
	Gnu 1.6.5	source software/profile.d/mpi_openmpi-1.6.5-gnu.sh
	Intel 1.6.5	source software/profile.d/mpi_openmpi-1.8.5-intel.sh
	Gnu 1.6.5	source software/profile.d/mpi_openmpi-1.8.5-gnu.sh
IntelMPI	5.0.2	source software/profile.d/mpi_intelmpi-5.0.2.044.sh

超算中心集群系统数学库环境变量的选取

数学库	版本	环境变量选取命令
fftw	float 2.1.5	source software/profile.d/mathlib_fftw-2.1.5-float.sh
	double 2.15	source software/profile.d/mathlib_fftw-2.1.5-double.sh
	float 3.3.3	source software/profile.d/mathlib_fftw-3.3.3float.sh
	float 3.3.3	source software/profile.d/mathlib_fftw-3.3.3-float.sh
lapack	Intel 3.4.2	source software/profile.d/mathlib_lapack-3.4.2-intel.sh
	Gnu 3.4.2	source software/profile.d/mathlib_lapack-3.4.2-gpu.sh
acml	Gfortran 5.3.1	source software/profile.d/mathlib_acml-5.3.1-gfortran.sh
	ifort 5.3.1	source software/profile.d/mathlib_acml-5.3.1-ifort.sh

附录 1 qsub 命令说明

qsub

submit pbs job

Synopsis

```
qsub[ -a date_time ] [ -A account_string ] [ -b secs ] [ -c checkpoint_options ]
[ -C directive_prefix ] [ -d path ] [ -D path ] [ -e path ] [ -f ] [ -F ] [ -h ]
[ -I ] [ -j join ] [ -k keep ] [ -l resource_list ]
[ -m mail_options ] [ -M user_list ] [ -n ] [ -N name ] [ -o path ]
[ -p priority ] [ -P user[:group] ] [ -q destination ] [ -r c ] [ -S path_list ]
[ -t array_request ] [ -u user_list ]
[ -v variable_list ] [ -V ] [ -W additional_attributes ] [ -X ] [ -z ] [script]
```

Description

To create a job is to submit an executable script to a batch server. The batch server will be the default server unless the -q option is specified. The command parses a script prior to the actual script execution; it does not execute a script itself. All script-writing rules remain in effect, including the "#!" at the head of the file.

See discussion of PBS_DEFAULT under Environment Variables below. Typically, the script is a shell script

which will be executed by a command shell such as sh or csh.

Options on the qsub command allow the specification of attributes which affect the behavior of the job.

The qsub command will pass certain environment variables in the Variable_List attribute of the job.

These

variables will be available to the job. The value for the following variables will be taken from the environment of the qsub command: HOME, LANG, LOGNAME, PATH, MAIL, SHELL, and TZ. These values will be assigned to a new name which is the current name prefixed with the string "PBS_O_". For example, the job will have access to an environment variable named PBS_O_HOME which have the value of the variable HOME in the qsub command environment.

In addition to the above, the following environment variables will be available to the batch job.

PBS_O_HOST the name of the host upon which the qsub command is running.

PBS_SERVER

the hostname of the pbs_server which qsub submits the job to.

PBS_O_QUEUE

the name of the original queue to which the job was submitted.

PBS_O_WORKDIR

the absolute path of the current working directory of the qsub command.

PBS_ARRAYID

each member of a job array is assigned a unique identifier (see -t).

PBS_ENVIRONMENT

set to PBS_BATCH to indicate the job is a batch job, or to PBS_INTERACTIVE to indicate the job is a PBS interactive job, see -I option.

PBS_JOBID

the job identifier assigned to the job by the batch system. It can be used in the stdout and stderr paths. TORQUE replaces \$PBS_JOBID with the job's jobid (for example, #PBS -o /tmp/\$PBS_JOBID.output).

PBS_JOBNAME

the job name supplied by the user.

PBS_NODEFILE

the name of the file contain the list of nodes assigned to the job (for parallel and cluster systems).

PBS_QUEUE

the name of the queue from which the job is executed.

Options

-a*date_time*

Declares the time after which the job is eligible for execution.

The date_time argument is in the form:

[[[[CC]YY]MM]DD]hhmm[.SS]

Where CC is the first two digits of the year (the century), YY is the second two digits of the year, MM is the two digits for the month, DD is the day of the month, hh is the hour, mm is the minute, and the

optional SS is the seconds.

If the month, MM, is not specified, it will default to the current month if the specified day DD, is in the future. Otherwise, the month will be set to next month. Likewise, if the day, DD, is not specified, it will default to today if the time hhmm is in the future. Otherwise, the day will be set to tomorrow. For example, if you submit a job at 11:15am with a time of -a 1110, the job will be eligible to run at

11:10am tomorrow.

-A*account_string*

Defines the account string associated with the job. The account_string is an undefined string of characters and is interpreted by the server which executes the job. See section 2.7.1 of the PBS ERS.

-b*seconds*

Defines the maximum number of seconds qsub will block attempting to contact pbs_server. If pbs_server is down, or for a variety of communication failures, qsub will continually retry connecting to pbs_server for job submission. This value overrides the CLIENTRETRY parameter in torque.cfg. This is

a non-portable TORQUE extension. Portability-minded users can use the PBS_CLIENTRETRY environmental variable. A negative value is interpreted as infinity. The default is 0.

-c*checkpoint_options*

Defines the options that will apply to the job. If the job executes upon a host which does not support checkpoint, these options will be ignored.

Valid checkpoint options are:

none

No checkpointing is to be performed.

enabled

Specify that checkpointing is allowed but must be explicitly invoked by either the `qhold` or `qchkpt` commands.

shutdown

Specify that checkpointing is to be done on a job at `pbs_mom` shutdown.

periodic

Specify that periodic checkpointing is enabled. The default interval is 10 minutes and can be changed by the `$checkpoint_interval` option in the MOM config file or by specifying an interval when the job is submitted

interval=minutes

Checkpointing is to be performed at an interval of minutes, which is the integer number of minutes of wall time used by the job. This value must be greater than zero.

depth=number

Specify a number (depth) of checkpoint images to be kept in the checkpoint directory.

dir=path

Specify a checkpoint directory (default is `/var/spool/torque/checkpoint`).

-C*directive_prefix*

Defines the prefix that declares a directive to the `qsub` command within the script file. See the paragraph on script directives in the Extended Description section.

If the `-C` option is presented with a `directive_prefix` argument that is the null string, `qsub` will not scan

the script file for directives.

-d*path*

Defines the working directory path to be used for the job. If the `-d` option is not specified, the default

working directory is the home directory. This option sets the environment variable `PBS_O_INITDIR`.

-D*path*

Defines the root directory to be used for the job. This option sets the environment variable `PBS_O_ROOTDIR`.

-e*path*

Defines the path to be used for the standard error stream of the batch job. The path argument is of the form:

[hostname:]path_name

Where `hostname` is the name of a host to which the file will be returned and `path_name` is the path name on that host in the syntax recognized by POSIX. The argument will be interpreted as follows:

path_name

Where `path_name` is not an absolute path name, then the `qsub` command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the `hostname` component.

hostname:path_name

Where `path_name` is not an absolute path name, then the `qsub` command will not expand the path name relative to the current working directory of the command. On delivery of the standard error, the path name will be expanded relative to the users home directory on the `hostname` system.

path_name

Where `path_name` specifies an absolute path name, then the `qsub` will supply the name of the host on which it is executing for the `hostname`.

hostname:path_name

Where `path_name` specifies an absolute path name, the path will be used as specified.

If the `-e` option is not specified, the default file name for the standard error stream will be used. The

default name has the following form:

job_name.esquence_number

where `job_name` is the name of the job, see `-N` option, and `sequence_number` is the job number assigned when the job is submitted.

-f

Job is made fault tolerant. Jobs running on multiple nodes are periodically polled by mother superior. If one of the nodes fails to report, the job is canceled by mother superior and a failure is reported. If a job is fault tolerant, it will not be canceled based on failed polling (no matter how many nodes

fail to report). This may be desirable if transient network failures are causing large jobs not to complete,

where ignoring one failed polling attempt can be corrected at the next polling attempt.

If TORQUE is compiled with PBS_NO_POSIX_VIOLATION (there is no config option for this), you have to use `-W fault_tolerant=true` to mark the job as fault tolerant.

-F

Specifies the arguments that will be passed to the job script when the script is launched. The accepted syntax is:

```
qsub -F "myarg1 myarg2 myarg3=myarg3value" myscript2.sh
```

Quotation marks are required. qsub will fail with an error message if the argument following `-F` is not a quoted value. The pbs_mom server will pass the quoted value as arguments to the job script when it launches the script.

-h

Specifies that a user hold be applied to the job at submission time.

-I

Declares that the job is to be run "interactively". The job will be queued and scheduled as any PBS batch job, but when executed, the standard input, output, and error streams of the job are connected through qsub to the terminal session in which qsub is running. Interactive jobs are forced to not rerunnable. See the "Extended Description" paragraph for additional information of interactive jobs.

-j join

Declares if the standard error stream of the job will be merged with the standard output stream of the job.

An option argument value of `oe` directs that the two streams will be merged, intermixed, as standard output. An option argument value of `eo` directs that the two streams will be merged, intermixed, as standard error.

If the join argument is `n` or the option is not specified, the two streams will be two separate files.

-k keep

Defines which (if either) of standard output or standard error will be retained on the execution host. If

set for a stream, this option overrides the path name for that stream. If not set, neither stream is retained on the execution host.

The argument is either the single letter "e" or "o", or the letters "e" and "o" combined in either order.

Or the argument is the letter "n".

e

The standard error stream is to retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by:

```
job_name.esquence
```

where `job_name` is the name specified for the job, and `sequence` is the sequence number component of the job identifier.

o

The standard output stream is to retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by:

```
job_name.osequence
```

where `job_name` is the name specified for the job, and `sequence` is the sequence number component of the job identifier.

eo

Both the standard output and standard error streams will be retained.

oe

Both the standard output and standard error streams will be retained.

n

Neither stream is retained.

-l resource_list

Defines the resources that are required by the job and establishes a limit to the amount of resource that can be consumed. If not set for a generally available resource, such as CPU time, the limit is infinite. The `resource_list` argument is of the form:

resource_name[=[value]][,resource_name[=[value]],...]

In this situation, you should request the more inclusive resource first. For example, a request for procs should come before a gres request.

In TORQUE 3.0.2 or later, qsub supports the mapping of -l gpus=X to -l gres=gpus:X. This allows users who are using NUMA systems to make requests such as -l ncpus=20,gpus=5 indicating they are not concerned with the GPUs in relation to the NUMA nodes they request, they only want a total of 20 cores and 5 GPUs.

-mmail_options

Defines the set of conditions under which the execution server will send a mail message about the job. The mail_options argument is a string which consists of either the single character "n", or one or more

of the characters "a", "b", and "e".

If the character "n" is specified, no normal mail is sent. Mail for job cancels and other events outside of

normal job processing are still sent.

For the letters "a", "b", and "e":

a

mail is sent when the job is aborted by the batch system.

b

mail is sent when the job begins execution.

e

mail is sent when the job terminates.

If the -m option is not specified, mail will be sent if the job is aborted.

-Muser_list

Declares the list of users to whom mail is sent by the execution server when it sends mail about the job.

The user_list argument is of the form:

user[@host][,user[@host],...]

If unset, the list defaults to the submitting user at the qsub host, i.e. the job owner.

-nnode-exclusive

Allows a user to specify an exclusive-node access/allocation request for the job. This affects only [cpuset](#)s and compatible schedulers.

-Nname

Declares a name for the job. The name specified may be up to and including 15 characters in length. It

must consist of printable, non white space characters with the first character alphabetic.

If the -N option is not specified, the job name will be the base name of the job script file specified on the command line. If no script file name was specified and the script was read from the standard input, then the job name will be set to STDIN.

-opath

Defines the path to be used for the standard output stream of the batch job. The path argument is of the form:

[hostname:]path_name

where hostname is the name of a host to which the file will be returned and path_name is the path name on that host in the syntax recognized by POSIX. The argument will be interpreted as follows:
path_name

Where path_name is not an absolute path name, then the qsub command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the hostname component.

hostname:path_name

Where path_name is not an absolute path name, then the qsub command will not expand the path name relative to the current working directory of the command. On delivery of the standard output, the path name will be expanded relative to the users home directory on the hostname system.

path_name

Where path_name specifies an absolute path name, then the qsub will supply the name of the host on which it is executing for the hostname.

hostname:path_name

Where path_name specifies an absolute path name, the path will be used as specified.

If the `-o` option is not specified, the default file name for the standard output stream will be used. The

default name has the following form:

`job_name.osequence_number`

where `job_name` is the name of the job, see `-N` option, and `sequence_number` is the job number assigned when the job is submitted.

-ppriority

Defines the priority of the job. The priority argument must be a integer between -1024 and +1023 inclusive. The default is no priority which is equivalent to a priority of zero.

-Puser[:group]

Allows a root user to submit a job as another user. TORQUE treats proxy jobs as though the jobs were submitted by the supplied username. This feature is available in TORQUE 2.4.7 and later, however, TORQUE 2.4.7 does not have the ability to supply the `[:group]` option. The `[:group]` option is available in TORQUE 2.4.8 and later.

-qdestination

Defines the destination of the job. The destination names a queue, a server, or a queue at a server. The `qsub` command will submit the script to the server defined by the destination argument. If the destination is a routing queue, the job may be routed by the server to a new destination.

If the `-q` option is not specified, the `qsub` command will submit the script to the default server. See `PBS_DEFAULT` under the Environment Variables section on this man page and the PBS ERS section 2.7.4, "Default Server".

If the `-q` option is specified, it is in one of the following three forms:

`queue`

`@server`

`queue@server`

If the destination argument names a queue and does not name a server, the job will be submitted to the named queue at the default server.

If the destination argument names a server and does not name a queue, the job will be submitted to the default queue at the named server.

If the destination argument names both a queue and a server, the job will be submitted to the named queue at the named server.

-ry/n

Declares whether the job is rerunnable. See the `qrerun` command. The option argument is a single character, either `y` or `n`.

If the argument is `"y"`, the job is rerunnable. If the argument is `"n"`, the job is not rerunnable. The default value is `y`, rerunnable.

-Spath_list

Declares the shell that interprets the job script.

The option argument `path_list` is in the form:

`path[@host][,path[@host],...]`

Only one path may be specified for any host named. Only one path may be specified without the corresponding host name. The path selected will be the one with the host name that matched the name of the execution host. If no matching host is found, then the path specified without a host will be selected, if present.

If the `-S` option is not specified, the option argument is the null string, or no entry from the `path_list` is selected, the execution will use the users login shell on the execution host.

-tarray_request

Specifies the task ids of a job array. Single task arrays are allowed.

The `array_request` argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimited list. Examples: `-t 1-100` or `-t 1,10,50-100`.

An optional **slot limit** can be specified to limit the amount of jobs that can run concurrently in the job

array. The default value is unlimited. The slot limit must be the last thing specified in the `array_request` and is delimited from the array by a percent sign (%).

This sets the slot limit to 5. Only 5 jobs from this array can run at the same time.

You can use [qalter](#) to modify slot limits on an array. The server parameter `max_slot_limit` can be used to set a global slot limit policy.

-uuser_list

Defines the user name under which the job is to run on the execution system.

The `user_list` argument is of the form:

```
user[@host][,user[@host],...]
```

Only one user name may be given per specified host. Only one of the user specifications may be supplied without the corresponding host specification. That user name will be used for execution on any host not named in the argument list. If unset, the user list defaults to the user who is running `qsub`.

-v*variable_list*

Expands the list of environment variables that are exported to the job.

In addition to the variables described in the "Description" section above, `variable_list` names environment variables from the `qsub` command environment which are made available to the job when it executes. The `variable_list` is a comma separated list of strings of the form `variable` or `variable=value`. These variables and their values are passed to the job.

-V

Declares that all environment variables in the `qsub` commands environment are to be exported to the batch job.

-W*additional_attributes*

The `-W` option allows for the specification of additional job attributes. The general syntax of the `-W` is in the form:

```
-W attr_name=attr_value[,attr_name=attr_value...]
```

Note if white space occurs anywhere within the option argument string or the equal sign, "=", occurs within an `attribute_value` string, then the string must be enclosed with either single or double quote marks.

PBS currently supports the following attributes within the `-W` option.

`depend=dependency_list`

Defines the dependency between this and other jobs. The `dependency_list` is in the form:

```
type[:argument[:argument...]][,type:argument...]
```

The argument is either a numeric count or a PBS job id according to type. If argument is a count, it must be greater than 0. If it is a job id and not fully specified in the form `seq_number.server.name`, it will be expanded according to the default server rules which apply to job IDs on most commands. If argument is null (the preceding colon need not be specified), the dependency of the corresponding type is cleared (unset). These are the valid dependencies:

`synccount:count`

This job is the first in a set of jobs to be executed at the same time. Count is the number of additional jobs in the set.

`syncwith:jobid`

This job is an additional member of a set of jobs to be executed at the same time.

In the above and following dependency types, `jobid` is the job identifier of the first job in the set.

`after:jobid[:jobid...]`

This job may be scheduled for execution at any point after jobs `jobid` have started execution.

`afterok:jobid[:jobid...]`

This job may be scheduled for execution only after jobs `jobid` have terminated with no errors. See the `qsub` warning under "Extended Description".

`afternotok:jobid[:jobid...]`

This job may be scheduled for execution only after jobs `jobid` have terminated with errors. See the `qsub` warning under "Extended Description".

`afterany:jobid[:jobid...]`

This job may be scheduled for execution after jobs `jobid` have terminated, with or without errors.

`on:count`

This job may be scheduled for execution after count dependencies on other jobs have been satisfied. This form is used in conjunction with one of the before forms, see below.

`before:jobid[:jobid...]`

When this job has begun execution, then jobs `jobid...` may begin.

`beforeok:jobid[:jobid...]`

If this job terminates execution without errors, then jobs jobid... may begin. See the csh warning under "Extended Description".

beforenotok:jobid[:jobid...]

If this job terminates execution with errors, then jobs jobid... may begin. See the csh warning under "Extended Description".

beforeany:jobid[:jobid...]

When this job terminates execution, jobs jobid... may begin.

If any of the before forms are used, the jobs referenced by jobid must have been submitted with a dependency type of on.

If any of the before forms are used, the jobs referenced by jobid must have the same owner as the job being submitted. Otherwise, the dependency is ignored.

Array dependencies make a job depend on an array or part of an array. If no count is given, then the entire array is assumed. Array dependency examples are [here](#).

afterstartarray:arrayid[count]

After this many jobs have started from arrayid, this job may start.

afterokarray:arrayid[count]

This job may be scheduled for execution only after jobs in arrayid have terminated with no errors.

afternotokarray:arrayid[count]

This job may be scheduled for execution only after jobs in arrayid have terminated with errors.

afteranyarray:arrayid[count]

This job may be scheduled for execution after jobs in arrayid have terminated, with or without errors.

beforestartarray:arrayid[count]

Before this many jobs have started from arrayid, this job may start.

beforeokarray:arrayid[count]

If this job terminates execution without errors, then jobs in arrayid may begin.

beforenotokarray:arrayid[count]

If this job terminates execution with errors, then jobs in arrayid may begin.

beforeanyarray:arrayid[count]

When this job terminates execution, jobs in arrayid may begin.

If any of the before forms are used, the jobs referenced by arrayid must have been submitted with a dependency type of on. If any of the before forms are used, the jobs referenced by arrayid must have the same owner as the job being submitted. Otherwise, the dependency is ignored.

Error processing of the existence, state, or condition of the job on which the newly submitted job is a deferred service, i.e. the check is performed after the job is queued. If an error is detected, the new job will be deleted by the server. Mail will be sent to the job submitter stating the error.

Dependency examples:

```
qsub -W depend=afterok:123.big.iron.com /tmp/script
qsub -W depend=before:234.hunk1.com:235.hunk1.com
/tmp/script
qsub script.sh -W depend=afterokarray:427[]
```

This assumes every job in array 427 has to finish successfully for the dependency to be satisfied.

```
qsub script.sh -W depend=afterokarray:427[][5]
```

This means that 5 of the jobs in array 427 have to successfully finish in order for the dependency to be satisfied.

group_list=g_list

Defines the group name under which the job is to run on the execution system. The g_list argument is of the form:

```
group[@host][,group[@host],...]
```

Only one group name may be given per specified host. Only one of the group specifications may be supplied without the corresponding host specification. That group name will be used for execution on any host not named in the argument list. If not set, the group_list defaults to the primary group of the user under which the job will be run.

interactive=true

If the interactive attribute is specified, the job is an interactive job. The `-I` option is a alternative method of specifying this attribute.

`stagein=file_list`

`stageout=file_list`

Specifies which files are staged (copied) in before job start or staged out after the job completes execution. On completion of the job, all staged-in and staged-out files are removed from the execution system. The `file_list` is in the form:

`local_file@hostname:remote_file[,...]`

regardless of the direction of the copy. The name `local_file` is the name of the file on the system where the job executed. It may be an absolute path or relative to the home directory of the user. The name `remote_file` is the destination name on the host specified by `hostname`. The name may be absolute or relative to the users home directory on the destination host. The use of wildcards in the file name is not recommended. The file names map to a remote copy program (`rcp`) call on the execution system in the follow manner:

For stagein: `rcp hostname:remote_file local_file`

For stageout: `rcp local_file hostname:remote_file`

Data staging examples:

`-W stagein=/tmp/input.txt@headnode:/home/user/input.txt`

`-W stageout=/tmp/output.txt@headnode:/home/user/output.txt`

If TORQUE has been compiled with `wordexp` support, then variables can be used in the specified paths. Currently only `$PBS_JOBID`, `$HOME`, and `$TMPDIR` are supported for stagein.

`umask=XXX`

Sets `umask` used to create stdout and stderr spool files in `pbs_mom` spool directory.

Values starting with 0 are treated as octal values, otherwise the value is treated as a decimal `umask` value.

`-X`

Enables X11 forwarding. The `DISPLAY` environment variable must be set.

`-z`

Directs that the `qsub` command is not to write the job identifier assigned to the job to the commands standard output.

Operands

The `qsub` command accepts a script operand that is the path to the script of the job. If the path is relative, it will be expanded relative to the working directory of the `qsub` command.

If the script operand is not provided or the operand is the single character `"-"`, the `qsub` command reads the script from standard input. When the script is being read from Standard Input, `qsub` will copy the file to a temporary file. This temporary file is passed to the library interface routine `pbs_submit`. The temporary file is removed by `qsub` after `pbs_submit` returns or upon the receipt of a signal which would cause `qsub` to terminate.

Standard Input

The `qsub` command reads the script for the job from standard input if the script operand is missing or is the single character `"-"`.

Input Files

The script file is read by the `qsub` command. `Qsub` acts upon any directives found in the script. When the job is created, a copy of the script file is made and that copy cannot be modified.

Standard Output

Unless the `-z` option is set, the job identifier assigned to the job will be written to standard output if the job is successfully created.

Standard Error

The `qsub` command will write a diagnostic message to standard error for each error occurrence.

Environment Variables

The values of some or all of the variables in the `qsub` commands environment are exported with the job, see the `-v` and `-V` options.

The environment variable `PBS_DEFAULT` defines the name of the default server. Typically, it corresponds to the system name of the host on which the server is running. If `PBS_DEFAULT` is not set, the default is defined by an administrator established file.

The environment variable `PBS_DPREFIX` determines the prefix string which identifies directives in the script.

The environment variable `PBS_CLIENTRETRY` defines the maximum number of seconds `qsub` will block. See the `-b` option above. Despite the name, currently `qsub` is the only client that supports this option.

TORQUE. cfg

The `torque.cfg` file, located in `PBS_SERVER_HOME` (`/var/spool/torque` by default) controls the behavior of the `qsub` command. This file contains a list of parameters and values separated by whitespace. `QSUBSLEEP` takes an integer operand which specifies time to sleep when running `qsub` command. Used to prevent users from overwhelming the scheduler.

`SUBMITFILTER` specifies the path to the submit filter used to pre-process job submission. The default path is `libexecdir/qsub_filter`, which falls back to `/usr/local/sbin/torque_submitfilter` for backwards compatibility. This `torque.cfg` parameter overrides this default.

```
SERVERHOST
QSUBHOST
QSUBSENDUID
XAUTHPATH
CLIENTRETRY
VALIDATEGROUP
DEFAULTTCKPT
VALIDATEPATH
RERUNNABLEBYDEFAULT
```

For example:

```
QSUBSLEEP 2
RERUNNABLEBYDEFAULT false
```

Extended Description

Script Processing:

A job script may consist of PBS directives, comments and executable statements. A PBS directive provides a way of specifying job attributes in addition to the command line options. For example:

```
:
#PBS -N Job_name
#PBS -l walltime=10:30,mem=320kb
#PBS -m be
#
step1 arg1 arg2
step2 arg3 arg4
```

The `qsub` command scans the lines of the script file for directives. An initial line in the script that begins with the characters `"#!"` or the character `":"` will be ignored and scanning will start with the next line. Scanning will continue until the first executable line, that is a line that is not blank, not a directive line, nor a line whose first non white space character is `"#"`. If directives occur on subsequent lines, they will be ignored.

A line in the script file will be processed as a directive to `qsub` if and only if the string of characters starting with the first non white space character on the line and of the same length as the directive prefix matches the directive prefix.

The remainder of the directive line consists of the options to `qsub` in the same syntax as they appear on the command line. The option character is to be preceded with the `"-"` character.

If an option is present in both a directive and on the command line, that option and its argument, if any, will be ignored in the directive. The command line takes precedence.

If an option is present in a directive and not on the command line, that option and its argument, if any, will be processed as if it had occurred on the command line.

The directive prefix string will be determined in order of preference from:

The value of the `-C` option argument if the option is specified on the command line.

The value of the environment variable `PBS_DPREFIX` if it is defined.

The four character string `#PBS`.

If the `-C` option is found in a directive in the script file, it will be ignored.

User Authorization:

When the user submits a job from a system other than the one on which the PBS Server is running, the name under which the job is to be executed is selected according to the rules listed under the `-u` option. The user submitting the job must be authorized to run the job under the execution user name. This authorization is provided if:

The host on which qsub is run is trusted by the execution host (see `/etc/hosts.equiv`)

The execution user has an `.rhosts` file naming the submitting user on the submitting host.

C-Shell `.logout` File:

The following warning applies for users of the c-shell, `csch`. If the job is executed under the `csch` and a `.logout`

file exists in the home directory in which the job executes, the exit status of the job is that of the `.logout`

script, not the job script. This may impact any inter-job dependencies. To preserve the job exit status, either remove the `.logout` file or place the following line as the first line in the `.logout` file

```
set EXITVAL = $status
```

and the following line as the last executable line in `.logout`

```
exit $EXITVAL
```

Interactive Jobs:

If the `-I` option is specified on the command line or in a script directive, or if the "interactive" job attribute declared true via the `-W` option, `-W interactive=true`, either on the command line or in a script directive, the job is an interactive job. The script will be processed for directives, but will not be included with the job.

When the job begins execution, all input to the job is from the terminal session in which qsub is running.

When an interactive job is submitted, the qsub command will not terminate when the job is submitted.

Qsub

will remain running until the job terminates, is aborted, or the user interrupts qsub with an SIGINT (the

control-C key). If qsub is interrupted prior to job start, it will query if the user wishes to exit.

If the user response "yes", qsub exits and the job is aborted.

One the interactive job has started execution, input to and output from the job pass through qsub.

Keyboard

generated interrupts are passed to the job. Lines entered that begin with the tilde (`~`) character and contain special sequences are escaped by qsub. The recognized escape sequences are:

`~.`

Qsub terminates execution. The batch job is also terminated.

`~susp`

Suspend the qsub program if running under the C shell. "susp" is the suspend character, usually `CNTL-Z`.

`~asusp`

Suspend the input half of qsub (terminal to job), but allow output to continue to be displayed.

Only works under the C shell. "asusp" is the auxiliary suspend character, usually `CNTL-Y`.

Exit Status

Upon successful processing, the qsub exit status will be a value of zero.

If the qsub command fails, the command exits with a value greater than zero