LSF 作业管理系统使用方法

上海交通大学高性能计算中心

http://hpc.sjtu.edu.cn

2013 年 12 月 5 日更新

目录

作业管理系统是高性能计算机的“指挥部”，它接收用户的作业请求，将作业分配到合适的节点上运行，最后将各节点的计算结果汇总给用户。作业管理系统能够提高计算资源的利用率、降低集群维护难度，因此高性能计算系统大都配备了作业管理系统。

IBM Platform LSF 是一个被广泛使用的作业管理系统，具有高吞吐、配置灵活的优点。上海交通大学 Pi 集群也使用了 LSF 作业管理系统。这份文档将指导您通过 LSF 提交和管理高性能计算作业。

遵循文档的操作规范和反馈方法，将帮助您顺利完成工作。也欢迎大家对文档内容提出建议，谢谢！

1 查看计算队列bqueues

作业队列是一系列可用的计算资源池，不同的队列在软硬件配置上有侧重，适合不同性质的作业。用户可以使用bqueues查看 Pi 集群可用的计算队列：

```
$ bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
cpu               40  Open:Active      -    -    -    -  4135     0  4135     0
fat               40  Open:Active      -    -    -    -    32     0    32     0
gpu               40  Open:Active      -    -    -    -   560     0   560     0
mic               40  Open:Active      -    -    -    -     0     0     0     0
```

Pi 集群可用的计算队列有四个，分别是cpu、fat、gpu和mic。各队列的硬件配置简要说明如下：

- cpu: 采用双路 8 核服务器，64GB 内存，共 332 台服务器，合计 5312 个 CPU 核心、约 21TB 内存。这个队列容量大，适合处理大型计算任务。
- fat: 采用双路 8 服务器，256GB 内存，共 20 台服务器，合计 320 个 CPU 核心、约 5TB 内存。这个队列适合进行大内存计算。
- gpu: 采用双路 8 核服务器，64GB 内存，每节点配备 2 块 NVIDIA K20M 加速卡，共 50 台服务器。合计 800 个 CPU 核心、约 3TB 内存。这个队列适合进行 CUDA 通用 GPU 计算。

- mic: 采用双路 8 核服务器，64GB 内存，每节点配备 2 块 Intel Xeon Phi 加速卡，共 5 台服务器。合计 80 核 CPU、约 300GB 内存。这个队列适合执行需要 MIC 加速的程序。

2 作业提交bsub

busb命令用于向 LSF 作业管理系统提交作业请求。bsub 可接收的参数很多，通过指定不同的运行参数，可以精细地设定作业运行需求。

```
$ bsub -h
```

下面分别介绍busb命令调用、提交作业的方法和额外的资源控制参数。

2.1 bsub 调用方法

在命令行中，用户可以通过如下三种方法使用 bsub 命令，三种方法各有优点。

1. 直接在命令行中输入完整参数;
2. 进入 bsub 环境交互提交;
3. 编写作业提交脚本供 bsub 处理;

2.1.1 直接输入完整参数

直接输入 bsub 完整参数，可以方便地提交单线程作业。下面这条命令提交了一个需要一个 CPU 核运行的单线程作业:

```
$ bsub -n 1 -q cpu -o job.out ./myprog "-j 3"
```

主要参数说明如下:

- -n指定所需的计算核心数。
- -q指定作业运行的队列，在 Pi 集群上可用的计算队列有 cpu、fat、gpu 和 mic。

- -o指定作业运行信息的输出文件。
- ./myprog是要提交运行的可执行文件，
- "-j 3"是传递给 myprog 的命令行参数。

当然，这种用法仅适用于简单的作业，更复杂的作业控制需要编写作业脚本。

2.2 交互式提交

键入 bsub 回车后，可进入 bsub 交互环境输入作业参数和作业程序。bsub 交互环境的主要有点是可以一次提交多个参数相同的作业。例如：

```
$ bsub
bsub> -n 1
bsub> -q cpu
bsub> -o job.out
bsub> PROG1
bsub> PROG2
bsub> PROG3
bsub> CTRL+d
```

等价于提交了 PROG1、PROG2 和 PROG3 三个作业程序：

```
$ bsub -n 1 -q cpu -o job.out PROG1
$ bsub -n 1 -q cpu -o job.out PROG2
$ bsub -n 1 -q cpu -o job.out PROG3
```

2.3 使用作业提交脚本

作业脚本是带有 “bsub 格式” 的纯文本文件。作业脚本易于编辑和复用，是提交复杂作业的最佳形式。下面是名为job.script的作业脚本的内容：

```
#=== Begin job.script ===#
#BSUB -n 1
#BSUB -q cpu
#BSUB -o job.out
```

```
./mytest "-j 3"
#==== End job.script ===#
```

其中以#BSUB开头的行表示 bsub 作业参数，其他#开头的行为注释行，其他行为脚本运行内容。作业脚本的使用方法很简单，只需要把脚本内容通过标准输入重定向给 busb：

```
$ bsub < job.script
```

以上脚本等价于如下命令：

```
$ bsub -n 1 -q cpu -o job.out ./mytest "-j 3"
```

bsub 默认会调用/bin/sh执行脚本内容，因此可以使用 Shell 编程脚本对作业参数进行处理和控制。下面这个作业脚本提交了一个需要 64 核心的 MPI 计算任务。这个脚本先设置了一些作业运行参数，调用其他脚本让运行相关的环境变量生效，然后使用一小段 Shell 脚本从$LSB_MCPU_HOSTS环境变量中创建 mpirun 所需的 nodelist，最后调用mpirun执行 MPI 并行作业。

```
#BSUB -J HELLO_MPI
#BUSB -q cpu
#BSUB -o job.out
#BSUB -e job.err
#BSUB -n 64

source /lustre/utility/intel/composer_xe_2014.3.163/bin/compilervars.sh intel64
source /lustre/utility/intel/mkl/bin/intel64/mklvars_intel64.sh
source /lustre/utility/intel/impi/4.1.1.036/bin64/mpivars.sh

MPIRUN=`which mpirun`
EXE="./mpihello"

cat /dev/null > nodelist

for host in `echo $LSB_MCPU_HOSTS |sed -e 's/ /:/g'|  sed 's/:n/\nn/g'`
do
echo $host >> nodelist
done
```

```
$MPIRUN -np $NP -machinefile nodelist $EXE
```

关于 MPI 程序和作业脚本的详细例子，请参考《并行程序示例》中的内容。

3 查看和终止作业

3.1 查看作业状态bjobs

检查已提交的作业的运行状态：

```
bjobs
```

以宽格式来显示作业运行状态：

```
bjobs -w
```

显示所有作业：

```
bjobs -a
```

显示正在运行的作业：

```
bjobs -r
```

显示等待运行 (pending) 的作业和等待的原因：

```
bjobs -p
```

显示已经挂起 (suspending) 的作业和挂起的原因：

```
bjobs -s
```

显示JOBID这个作业的所有信息：

```
bjobs -l JOBID
```

3.2 终止作业bkill

终止不需要的作业:

```
bkill
```

终止JOBID这个作业:

```
bkill JOBID
```

直接将作业JOBID从 LSF 中移除，而不等待该作业的进程在操作系统中终结:

```
bikill JOBID
```

3.3 监视作业输出bpeek

当作业正在运行时，显示它的标准输出，监视作业运行:

```
bpeek
```

查看JOBID的标准输出:

```
bpeek JOBID
```

3.4 作业历史信息bhist

显示作业的历史情况:

```
bhist
```

显示JOBID作业的历史情况:

```
bhist JOBID
```

4 错误排查

5 问题诊断

6 参考资料