海量作业

作业组

为了便于管理海量的作业，可以对作业定义作业组。

作业组的名字是类似于 Linux的路径名的树状结构，如作业组 /test，而 /test/1 和 /test/2 都属于 /test

需要注意的是作业组是全局的，创建者是这个作业组的拥有者，作业组拥有者可以对这个作业组及其子组内的

所有作业进行控制（哪怕这个作业是其他人提交的，拥有者也能进行控制）。如果您不想作业被别人控制，请

确保提交到的作业组从 /开始拥有者都是自己。

创建作业组

```
bgadd /test□显□创建作业组
bsub -g /test□□用bsub提交作业时指定一个不存在的作业组，则作业组会被隐式创建
```

查看作业组

```
bjgroup /test□□后一列就是作业组的拥有者，请注意拥有者也是层次继承的
```

删除作业组

```
bgdel /test□集□已经配置自动清理空的隐式创建的作业组
```

有了作业组以后就可以对一组作业进行控制了

```
bjobs -g /test□查□指定作业组的作业
bkill -g /test 0□□止指定作业组的所有作业
```

作业序列

作业序列是一系列作业，这些作业执行相同的操作和资源要求，但是输入输出文件不同。这些作业共享同一个

作业 ID，并可以通过索引来区分每个子作业。

bsub 使用 `-J "arrayName[indexList, ...]"`

参数命名并创建一个作业序列，也可以理解为一个作业数组，中括号里面就是数组的下标，可以是一维数组也

可以是多维数组。 indexList = start[-end[:step]] 下标的起始、结束和步长均可指定。

作业调度系统提供了两个运行时变量 %I 和 %J， %I 为子作业的索引值， %J 为作业 ID

，一般用于输入输出文件名中；以及运行时环境变量 LSB_JOBINDEX

```
bsub -J "myArray[1-10]" myJob□提□一个有10个子作业的作业
bsub -J "myArray[1-10]" -i "input.%I" -o "output.%I" myJob□每□子作业定义不同的输入输出文件
bkill 123[1]□□掉 jobid 是123的第一个子作业
bkill 123□杀□ jobid 是123的整个作业
```

海量作业

提交大量的作业，将使得作业的管理将变得困难，虽然可以用上述作业组和作业序列进行批量管理，但是海量

作业仍然对调度系统会造成很大的压力，特别是分钟级或秒级的短作业在调度时会浪费大量的时间，因此可以

根据作业情况将多个作业合并为一个作业提交。对于不同的作业可以用两种方法来进行合并：

多作业串行执行

申请 1个 CPU

核，顺序执行每个串行作业，前一个运行完成后再运行下一个，可以将非常多的串行作业合并成一个。对于每

个串行作业运行时间都很短且运行时间不一定相同的适用这种方式提交。

4个串行作业串行执行合并为一个作业，提交到 x5650队列，脚本方式为：

```
$ cat job.lsf
#BSUB -q x5650
./a.out >& 1.out
./a.out >& 2.out
./a.out >& 3.out
./a.out >& 4.out

$ bsub < job.lsf
Job <3366369> is submitted to queue <x5650>.
```

多作业并行执行

申请 N个 CPU核，同时执行 N个串行作业， N不可大于单节点 CPU

核数，每个串行作业运行时间需要相同，最后需要 wait

命令等待所有作业运行完毕返回。对于每个串行作业运行时间较长且运行时间都完全相同的适用这种方式提交

。

12个串行作业并行执行合并为一个作业，提交到 x5650队列，脚本方式为：

```
$ cat job.lsf
#BSUB -q x5650
#BSUB -n 12
( ./a.out >& 1.out )&
( ./a.out >& 2.out )&
( ./a.out >& 3.out )&
( ./a.out >& 4.out )&
( ./a.out >& 5.out )&
( ./a.out >& 6.out )&
( ./a.out >& 7.out )&
( ./a.out >& 8.out )&
( ./a.out >& 9.out )&
( ./a.out >& 10.out )&
( ./a.out >& 11.out )&
( ./a.out >& 12.out )
wait

$ bsub < job.lsf
Job <3366370> is submitted to queue <x5650>.
```

Revision #8
Created 2021-05-09 22:10:41 CST by Yao Ge
Updated 2022-02-02 17:02:51 CST by Yao Ge