

# $\pi$ 集群机时统计常见问题集

上海交通大学高性能计算中心

<http://hpc.sjtu.edu.cn>

2014年5月27日更新

1	作业在LSF系统中的生命周期	1
2	一个并行作业的CPU机时是如何计算的?	1
3	能否给出一个计算作业机时的例子?	2
4	$\pi$ 用户机时月报是如何计算的?	3
5	如何读懂bacct输出的机时信息?	4
6	如何计算GPU部分的机时?	5
7	参考资料	5

## 1 作业在LSF系统中的生命周期

用户通过**bsub**命令提交作业(Submitting)后,作业进入LSF作业调度系统,处于等待状态(Pending)。调度系统根据作业的优先级与当前可用资源,将作业分发(Dispatching)到相应计算节点。然后作业开始在节点上运行(Running),直至作业结束退出(Exiting)。

用户可以通过**bkill**终止处于等待状态或者运行状态的作业。用户终止处于等待状态的作业,不消耗机时;用户终止正在运行的作业,仍需要扣除已经消耗的机时。

## 2 一个并行作业的CPU机时是如何计算的?

当作业完成分发进入计算节点后,计算节点已被用户作业所占用,因此机时计算时间自此开始(Dt, Dispatched Time)。计算节点通常会在一小段准备(约几秒钟)时间后正式开始进行计算。当CPU完成计算后,作业结束,机时计算时间自此终止(Et, End Time)。若作业申请的核数为N,若这个作业消耗的机时T,可以这样计算:

$$T = (E_t - D_t) * N$$

显然，如果作业处于等待状态时被杀死，则其消耗的机时为0；如果作业在运行时被杀死，则作业的结束时间为作业被杀死的时间。

### 3 能否给出一个计算作业机时的例子？

在 $\pi$ 集群中，作业执行的各主要时间点可由LSF提供的**bacct**命令与**bjobs**命令查询出来。其中，**bacct**命令适合查询已经结束的任务，**bjobs**命令适合查询正在运行的任务。

以下是一个作业的查询结果，我们将以此为例说明任务的各时间节点。

```
Job <12345>, Job Name <HELLO_MPI>, User <xxxxxx>, Project <default>, Status
    <DONE>, Queue <cpu>, Command <#BUSB -q cpu;#BSUB -J HELLO
    _MPI;#BSUB -L /bin/bash;#BSUB -o %J.out;#BSUB -e %J.err;#B
    SUB -n 32;#BSUB -R "span[ptile=16]"; MODULEPATH=/lustre/ut
    ility/modulefiles:$MODULEPATH;module purge;module load ope
    nmpi/gcc/1.6.5; mpirun ./test_mpi>
Wed Apr  2 11:50:03: Submitted from host <mu07>, CWD <$HOME/mpi_test/my_test>,
    Output File <%J.out>, Error File <%J.err>;
Wed Apr  2 11:50:05: Dispatched to 32 Hosts/Processors <16*node313> <16*node118
    >;
Wed Apr  2 11:50:13: Completed <done>.

Accounting information about this job:
      CPU_T      WAIT      TURNAROUND  STATUS      HOG_FACTOR      MEM      SWAP
        6.35         2          10     done         0.6345       1M      32M
```

根据上述输出，我们不难发现作业的提交时间为Wed Apr 2 11:50:03，被分发的时间为Wed Apr 2 11:50:05，完成的时间为Wed Apr 2 11:50:13。这个作业共申请32个CPU核，从作业分发到程序结束共8秒。因此，作业消耗的机时为：

$$8 * 32 = 256 \text{ coreseconds} = 0.07111 \text{ corehours}$$

根据上述输出，作业从被提交到被分发，共经历了2秒的等待时间 (WAIT)。另外，作业被分发之后计算节点用于运算的时间 (CPU\_T) 为6.35秒。

## 4 $\pi$ 用户机时月报是如何计算的？

月度机时使用报告向用户展示当月所有任务的机时消耗。当月任务包括：在上个月或更早时间提交、并且一直运行到现在的作业，简称“长作业”；或者是本月提交运行的作业。对长作业的计时，说明如下。

某作业于2014年2月26日12点0分0秒被分发进入计算节点 (Dispatched Time)，到2014年3月5日12点0分0秒运行结束 (End Time)。那么，在统计2014年3月份的机时使用，此作业被纳入当月机时计算的时间段为2014年3月1日0点0分0秒至2014年3月5日12点0分0秒。

同理，某作业于2014年3月22日12点0分0秒被分发进入计算节点 (Dispatched Time)，到2014年4月6日12点0分0秒运行结束 (End Time)。那么，在统计2014年3月份的机时使用，此作业被纳入当月机时计算的时间段为2014年3月22日12点0分0秒至2014年3月31日23点59分59秒。

现在举例说明月度机时使用统计时怎样使用bacct和bjobs来正确计算用户的机时使用量。假设今天为2014年4月5日，现在我们来统计2014年3月用户userName的CPU列队的机时使用情况，我们需要使用以下两条命令。

```
$ bacct -l -u userName -q cpu -C 2014/3/1,2014/4/7
$ bjobs -l -u userName
```

bacct命令只能统计已结束作业的情况，而bjobs只能统计正在运行的作业的情况。于是 bacct -l -u userName -q cpu -C 2014/3/1,2014/4/7 则是计算用户userName于2014/3/1至2014/4/7日在CPU队列上执行完成的作业的详细情况。

细心的读者可能已经发现了，为什么时间段是2014/3/1至2014/4/7，而不是2014/3/1至2014/3/31？缘由如下：由于bacct命令只能统计已结束作业的情况，如果使用命令 bacct -l -u userName -q cpu -C 2014/3/1,2014/3/31 则无法统计出在2014/4/1至今天 (2014/4/5) 结束的作业，如果有作业开始于2014/3/10结束于2014/4/2，那么就会发生漏

算。让bacct的命令查询截止时间稍稍超出查询当天的时间，就能够保证不会漏算某些已经结束的作业。对于开始于2014/3/20而至今仍在运算的这类作业，则需要利用bjobs进行查询。

至于所有通过bacct和bjobs查询出来的作业，在进行月度机时统计时均只会计入当月“折合”的部分，并不会重复计算，这一点用户可以根据月度报告提供的信息进行验算。

另外，需要再次明确的一点是，作业的机时统计开始于作业被分发后 (Dispatched Time) 结束于作业结束后 (End Time)，包括作业进入计算节点后的准备时间以及CPU运行作业的所花费的时间。

## 5 如何读懂bacct输出的机时信息？

bacct命令返回的机时汇总信息如下：

```
$ bacct -l -u userName -q cpu -C 2014/3/1,2014/3/19
SUMMARY:      ( time unit: second )
Total number of done jobs:      1      Total number of exited jobs:      2
Total CPU time consumed:      2.3      Average CPU time consumed:      0.8
Maximum CPU time of a job:      1.8      Minimum CPU time of a job:      0.2
Total wait time in queues:      7.0
Average wait time in queue:      2.3
Maximum wait time in queue:      3.0      Minimum wait time in queue:      2.0
Average turnaround time:      6 (seconds/job)
Maximum turnaround time:      9      Minimum turnaround time:      3
Average hog factor of a job: 0.10 ( cpu time / turnaround time )
Maximum hog factor of a job: 0.20      Minimum hog factor of a job: 0.03
Total throughput:      0.01 (jobs/hour) during 350.37 hours
Beginning time:      Mar 5 09:38      Ending time:      Mar 19 23:59
```

需要注意的是：我们不使用bacct计算总机时。我们只是用bacct和bjobs提供的单个作业的起止时间、计算核心数等信息，逐个累加得到某段时间内消耗的总机时。bacct提供的机时汇总结果并不具有参考价值。bacct得到的机时汇总结果，往往与我们提供给您的月报账单有差异，造成差异的原因包括但不限于：

1. bacct计算的机时结果，没有乘以作业使用的核数；
2. bacct只统计已完成作业消耗的及时，未统计仍在运行的作业所消耗的机时；

3. bacct通常还会受限于用户指定的参数，只统计某个时间段内提交且完成的作业所消耗的机时，而在此时间段之前提交且持续运行的长作业，往往会被漏掉；

如果您需要准确了解某一段时间内您的机时使用情况，欢迎您随时致信 $\pi$ 管理员。您可以使用机时月报中提供的作业列表，逐个审核作业的机时消耗。如果您对月报账单中某个作业的机时计算结果有疑问，欢迎致信 $\pi$ 管理员(邮件中请说明计时有误的作业ID)。我们计划在下半年，发布供用户自助使用的机时统计程序，敬请期待。

## 6 如何计算GPU部分的机时？

运行在GPU队列上的作业，机时使用是按照CPU耗时折算而来的。GPU节点上配置了2颗CPU(共16核)，外加2块K20M加速卡。GPU机时的计费单位是“卡小时”(cardhours)，卡小时与用于计算CPU机时费的单位“核小时”(corehours)换算关系如下：

```
1 cardhour = 8 corehours
```

例如，某作业于2014年3月22日12点0分0秒被分发进入GPU节点(Dispatched Time)，到2014年3月22日13点0分0秒运行结束(End Time)，共申请了32个CPU核-2个GPU节点的资源，即使用了4块加速卡。则此作业的机时使用量为：

```
1 * 32 = 32 corehours = 4 cardhours
```

机时统计细节与其他节点一致。

## 7 参考资料

- “LSF Command Reference” [http://www.bsc.es/support/LSF/9.1.2/print/lsf\\_command\\_ref.pdf](http://www.bsc.es/support/LSF/9.1.2/print/lsf_command_ref.pdf)
- “Manual page: bacct” [http://www-01.ibm.com/support/knowledgecenter/SSETD4\\_9.1.2/lsf\\_command\\_ref/bacct.1.dita?lang=en](http://www-01.ibm.com/support/knowledgecenter/SSETD4_9.1.2/lsf_command_ref/bacct.1.dita?lang=en)