π 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

目录

1 作业在 LSF 系统中的生命周期 2

2 一个并行作业的 CPU 机时是如何计算的？ 2

3 能否给出一个计算作业机时的例子？ 2

4 π 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 6

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 = (Et - Dt) * 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 π 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 <xxxxx>, 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 coreseconds = 0.07111 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 π 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 月份的机时使用时，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 月份的机时使用时，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π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π 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
- “Manual page: bacct” http://www-01.ibm.com/support/knowledgecenter/SSETD4_9.1.2/lsf_command_ref/bacct.1.dita?lang=en