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 作业从提交到结束经历的过程

用户在完成代码的编译后，通过 bsub 命令提交作业（Submitting），此时作业进入 LSF 作业调度系统，处于等待分配状态（Pending）。调度系统根据作业的优先级与当前可用计算资源进行综合判断，当条件满足时将作业分发到相应计算节点并将这些节点移出空闲队列（Dispatching）。计算节点完成作业的接收等准备工作后开始进行计算（Running）。作业执行完成后，节点被释放并再次进入空闲队列（Exiting）。

如果作业处于等待状态时，用户通过 bkill 将其杀死，此作业会直接退出 LSF，不再经过分发、执行、结束阶段；如果作业处于运行状态时，用户通过 bkill 将其杀死，则此作业不再继续运行并进入结束阶段。

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（Dispatched Time, Dt）。计算节点通常会在一小段准备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（End Time, Et）。假设作业申请的核数为 N，那么：

```
用户作业消耗的机时
= (Et - Dt) * 。N
```

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。此作业共申请 32 个 CPU 核。于是，作业消耗的机时为：

```
秒
8 * 核32 = 核秒256(*) = 核小时。0.07111(*)
```

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？

月度机时使用报告主要用于向用户展示当月用户所有任务所消耗的机时情况。用户作业可能开始于上一个月，可能开始于当月；用户的作业可能在当月便能结束，也可能需要到下个月才能结束。因此，在统计用户当月的机时使用量时，需要对作业的机时进行 “折合”。下面举例说明。

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？

当我们使用 bacct -l -u userName -q cpu -C 2014/3/1,2014/3/19 这样的命令查询某段时间内已结束的任务时，它会给出一个 “统计信息”。

```
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
```

值得注意的是，此 “统计信息” 并非我们机时统计时所真实使用到的信息。例如，Total CPU time consumed 部分是此段时间内完成的作业 “CPU 运行作业所花费的时间”（CPU_T）的简单叠加，并未包括作业被分发后在计算节点内的准备时间。同时，Total CPU time consumed 并未将每个作业的运行时间乘上相应的 CPU 核数。因此，此 “统计信息” 只能当作一个参考。

6 如何计算 GPU 部分的机时？

对于 GPU 用户提交的作业，其机时使用是按照 GPU 节点上 CPU 的机时使用量来计算的。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*2 个 GPU 设备），则此作业的机时使用量为：

```
小时
1 * 核32 = 核小时32(*)
```

机时统计细节与其他节点一致。