????

集群使用作业调度系统管理所有计算作业，该系统接受用户的作业请求，并将作业合理的分配到合适的节点上运行，因此所有用户均应通过作业调度系统提交计算作业，不可直接在任何节点上直接运行。用户使用 `bsub` 命令向作业调度系统提交作业， `bsub` 选项非常繁多，可对作业进行非常细致的控制，这里简要介绍常用选项和方法。

bsub ????

?????

`bsub [options] command [arguments]`

- `[options]` 为 bsub 的选项，可以设定队列、CPU核数等
- `command` 为计算程序，如果是 MPI 并行程序需要使用 mpirun 启动
- `[arguments]` 为计算程序的参数

> 例：提交一个作业到 e5v3ib 队列，需要 24核的 MPI 并行程序
>
> ```
> $ bsub -q e5v3ib -n 24 "module load oneapi/2024.0/mpi && mpirun ./app"
> Job <3206000> is submitted to queue <e5v3ib>
> ```

????

```
bsub < jobfile
```

jobfile 为作业的 shell 脚本文件，文件名任意且不需要运行权限，脚本内容如下：

```
#BSUB [options]
command [arguments]
```

脚本中以 `#BSUB` 开头的行后跟 `bsub` 的选项，其它行为作业运行脚本

> 例：提交一个作业到 e5v3ib 队列，需要 48核，需要大内存节点，作业名为 MgSiO3 ，标准输出文件为 `out`，标准错误输出文件为 `err`， Intel MPI 的并行作业 0
>
> ```
> $ cat job.lsf
> #BSUB -q e5v3ib
> #BSUB -n 48
> #BSUB -J MgSiO3
> #BSUB -o out
> #BSUB -e err
> module load ips/2018u4
> mpirun ./app
>
> $ bsub < job.lsf
> Job <3207099> is submitted to queue <e5v3ib>.
> ```
>
> 等价如下命令行方式
>
> ```
> $ bsub -q e5v3ib -n 48 -J MgSiO3 -o out -e err "module load
> ips/2018u4;mpirun ./app"
> Job <3207099> is submitted to queue <e5v3ib>.
> ```

bsub ????

- `-J job_name`：作业名称

????

- `-n min_tasks[,max_tasks]`：作业需要 CPU核数；例：需要四核 `-n 4`；需要 4~8核均可 `-n 4,8`
- `-m`：作业运行的节点或节点组，多个节点写在双引号内并用空格分隔，节点组对应的具体节点可用 `bmgroup` 命令查看，此选项很复杂。在 hostname/hostgroup 前后可用这些符号：后加 `!` 指定头结点、后加 `+[num]` 指定节点使用顺序。例：指定在 c04n01 和 c04n02 运行 `-m "c04n01 c04n02"`；指定可在 f01n01~n03，但是最希望在 f01n01、次希望在 f01n02 `-m "f01n01+2`

`f01n02+1 f01n03"`；

- `-R "res_req"`

  ：资源请求串，此选项非常复杂；例：有的队列某些节点内存较大，需要大内存节点可以指定 -R largemem

- `-R "select[hname!=host_name]"`：排除 host_name节点，如果要排除多个节点中间用 &&连接，如 `-R "select[hname!=x001 && hname!=x002]"`排除 x001和 x002节点
- `-x`：作业需要独占节点，无论申请多少核，作业均独占所运行的节点
- `-W [hour:]minute`：作业运行最长时间，超过这个时间则被 kill

CPU??

- `-R affinity[core:cpubind=core:membind=localprefer:distribute=pack]`：作业调度系统将进行 CPU亲和性绑定，注意可能会和程序本身（如 MPI）的绑定冲突，使用前请测试！

?????

- `-r`：如果计算节点或系统故障则自动重新运行作业
- `-Q "exit_code [exit_code ...]"`：根据作业退出码自动重新提交作业。使用空格分隔多个退出码，all指所有退出码，加 `~`排除一个或多个退出码。

????

- `-I`

  ：交互式作业，可在作业运行期间和程序进行交互（如输入参数等），可在调试期间使用，正常计算

  请勿使用

- `-K`：等待作业执行完才返回
- `-i input_file`：标准输入文件
- `-o output_file`：标准输出文件
- `-e error_file`：标准错误输出文件
- 以上三个选项的文件名中可以包含 `%J`用于表示 JOBID。如果没有用 `-o`或 `-oo`指定标准输出文件，那么系统会自动设定为 `output_%J`；如不想要输出文件请设置 `-o /dev/null`

更多选项见 官方文档

GPU ??

提交作业时使用 `-gpu` 选项申请所需的 GPU 资源，计算进程只可见作业调度系统分配的 GPU。CPU 核自动按照申请节点 GPU 的比例分配，如一节点 8个 GPU 和 40个 CPU 核，申请 2个 GPU 则分配 10个 CPU 核。

`-gpu` 的各个选项用 :分隔，默认值为 `num=1:mode=shared:mps=no:j_exclusive=yes`，常用选项如下

- `num=number`：每台主机需要 GPU的数量
- `mode=shared | exclusive_process`： GPU运行模式，`shared` 对应 Nvidia/AMD DEFAULT compute mode、`exclusive_process` 对应 Nvidia EXCLUSIVE_PROCESS
- `mps=yes | no`：开启或关闭 Nvidia Multi-Process Service (MPS)。关闭 MPS，多进程通过时间分片的方式共享 GPU；开启 MPS，多进程共享一个 CUDA Context 并发执行，增加了 GPU利用率
- `aff=yes | no`：是否强制进行严格的 GPU-CPU 亲和性绑定，还需要配合 `-R affinity[core:cpubind=core:membind=localprefer:distribute=pack]` 才能一同完成 GPU-CPU 亲和性绑定

????

一个计算任务可能分成几步，而每一步对资源的需求不同，因此需要分开提交，但这些作业之间又具有依赖关系，bsub 可使用选项 -w 'dependency_expression' 指定依赖关系。如果计算任务分成几步，但是每步对资源需求一样，那么请写在一个作业任务中依次执行。

- `-w 'done(job_ID | "job_name")'`：需要 job_ID 或 job_name 作业完成且状态为 DONE，即退出码为 0
- `-w 'ended(job_ID | "job_name")`'：需要 job_ID 或 job_name 作业完成或退出，状态为 EXIT 或 DONE

- 支持逻辑表达式 && (AND)、|| (OR)、! (NOT)
- 孤儿作业（即依赖条件不可能满足的）1分钟后会被自动终止

更多详细信息见官方文档

MPI/OpenMP ????

OpenMP (Open Multi-Processing) 是一种共享内存方式的单进程多线程并行编程技术；MPI (Message Passing Interface) 是一种多进程基于信息传递的并行编程技术。OpenMP 的特点是单节点、进程内、多线程、基于共享内存的并行运算；MPI 的特点是单或多节点、进程间、非共享内存、基于消息传递的并行运算。

混合并行编程模型构建的应用程序可以同时使用 OpenMP 和 MPI，节点内 NUMA内进程内使用 OpenMP 共享内存并行可降低内存需求，跨节点跨 NUMA跨进程使用 MPI 消息传递可大规模并行。需要注意的是，并不是一个节点一个 MPI进程是最优的，这往往会导致跨 NUMA 的内存访问，因此需要通过测试确定最佳配比。

mpirun一般会根据环境变量 LSB_MCPU_HOSTS启动相应的 MPI 进程，因此可以通过下列方法改变此环境变量中每个节点的 CPU核数，以匹配 MPI/OpenMP混合作业的 MPI 进程分布：

1. `#BSUB -n` 指定的仍然是总 CPU核数
2. 提交作业脚本中需要在计算命令前首先运行

   ```
   source /fs00/software/lsf/misc/ompthreads.sh [N]
   ```

3. 每个 MPI 进程的 OpenMP 线程数量可以用环境变量 `OMP_NUM_THREADS` 指定或上述命令行参数指定，同时指定时命令行参数优先，需要保证每个节点的 CPU 核数可以被线程数整除！

??????

?????

- LSB_JOBID: 作业ID
- LSB_QUEUE: 队列名称
- LSB_JOBNAME: 作业名称
- LSB_DJOB_NUMPROC: 分配的CPU总核数
- LSB_DJOB_HOSTFILE: 分配的节点列表文件，每行一个
- LSB_HOSTS: 分配的节点列表，每个CPU核一个节点名的纯节点列表
- LSB_MCPU_HOSTS: 分配的节点和核数列表，每个节点名和CPU核数的列表

```
LSB_DJOB_NUMPROC=6
LSB_HOSTS="node1 node1 node1 node2 node2 node2"
LSB_MCPU_HOSTS="node1 3 node2 3"
```

```
$ cat $LSB_DJOB_HOSTFILE
node1
node1
node1
node2
node2
node2
```

> LSB_HOSTS 和 LSB_MCPU_HOSTS 以不同的格式包含相同的信息，LSB_MCPU_HOSTS 比 LSB_HOSTS 更短更精简，如果 LSB_HOSTS 超过 4096 字节，则仅有 LSB_MCPU_HOSTS。

??????

????

提交一个串行作业到e52660队列，命令行方式和脚本方式分别为：

```
$ bsub -q e52660 ./app
Job <3279929> is submitted to queue <e52660>.
$ cat job.lsf
#BSUB -q e52660
./app
```

```
$ bsub < job.lsf
Job <3279930> is submitted to queue <e52660>.
```

MPI ????

MPI程序需要使用 `mpirun` 启动

提交一个需要 48核的 Intel MPI 并行作业到 e5v3ib, 命令行方式为:

```
$ bsub -q e5v3ib -n 48 "module load ips/2018u4;mpirun ./app"
Job <3280120> is submitted to queue <e5v3ib>.
```

提交一个需要 48核的 Open MPI 并行作业到 e5v3ib, 脚本方式为:

```
$ cat job.lsf
#BSUB -q e5v3ib
#BSUB -n 48
module load iccifort/15.0.3 imkl/11.2.3 openmpi/1.10.0-iccifort-15.0.3
mpirun ./app

$ bsub < job.lsf
Job <3280122> is submitted to queue <e5v3ib>.
```

OpenMP ????

OpenMP 不能跨节点, 因此 `-n` 不能指定超过一个节点的 **CPU**核数

提交一个需要 64核的 OpenMP 并行作业到 e7v4ib, 使用程序参数 -nt 指定线程数量, 命令行方式为:

```
$ bsub -q e7v4ib -n 64 "./app-nt \$LSB_DJOB_NUMPROC"
Job <3348175> is submitted to queue <e7v4ib>.
```

提交一个需要 64核的 OpenMP 并行作业到 e7v4ib, 使用环境变量 `OMP_NUM_THREADS`

指定线程数量, 脚本方式为:

```
$ cat job.lsf
#BSUB -q e7v4ib
#BSUB -n 64
OMP_NUM_THREADS="$LSB_DJOB_NUMPROC"
./app

$ bsub < job.lsf
Job <3348182> is submitted to queue <e7v4ib>.
```

MPI/OpenMP ????

每个 MPI进程跑 6个 OpenMP线程

通过环境变量 `OMP_NUM_THREADS` 指定

```
#BSUB -q 6140ib
#BSUB -n 72
export OMP_NUM_THREADS=6
source /fs00/software/lsf/misc/ompthreads.sh
module load ips/2018u4
mpirun ./run
```

通过命令行参数指定（有些计算程序需要通过命令行参数指定线程数量）

```
#BSUB -q 6140ib
#BSUB -n 72
source /fs00/software/lsf/misc/ompthreads.sh 6
module load ips/2018u4
mpirun ./openmx -nt 6
```

GPU ??

提交一个需要 1个 GPU 的作业到 e5v4p100ib 队列

```
bsub -q e5v4p100ib -gpu num=1 ./gpu_app
```

提交一个需要 4个 GPU 的作业到 62v100ib 队列, 进行 GPU-CPU 绑定

```
bsub -q 62v100ib -gpu "num=4:aff=yes" ./gpu_app
```

Revision #41
Created 9 May 2021 16:38:05 by Yao Ge
Updated 23 June 2024 16:06:54 by Yao Ge