Запуск задач на кластере Triton в системе SLURM

Условные обозначение в тексте:

!	Текст, на который следует обратить особое внимание

Моноширинным шрифтом набраны команды операционной системы или содержимое консоли. Например,

```
[user@triton ~]$ date

Fri Feb 28 12:13:14 YEKT 2014
```

Оглавление

Запуск программ осуществляется в системе **SLURM**. В результате запуска задача помещается в очередь заданий и ей присваивается **уникальный идентификатор** (его можно узнать командами `squeue`, `sacct`). По умолчанию задаче пользователя выделяется по 1 ГБ оперативной памяти на каждое вычислительное ядро (при необходимости размер исполняемой программы в байтах можно узнать с помощью команды `size`).

!	Необходимо использовать тот модуль установки переменных окружения, с которым программа была откомпилирована; Пока задача не просчиталась, нельзя ее перекомпилировать, удалять исполняемый файл и менять входные данные.

Модули

При работе на кластерах можно использовать различные компиляторы, библиотеки обмена сообщениями и пакеты прикладных программ (приложения), поэтому пользователь должен определить среду для решения своей задачи, выбрав нужное программное обеспечение. Выбор определяется модулем установки переменных окружения, требуемых для работы программы. Названия модулей содержат имена компиляторов, библиотек, пакетов, номера версий. Например, при загрузке модуля **openmpi-x86_64** программа пользователя будет откомпилирована системным компилятором **gcc** с библиотекой **openmpi**.

По умолчанию для пользователя загружается модуль **openmpi-x86_64**, который содержит следующие настройки: **OpenMPI GCC 4.4.7, openmpi-x86_64** (после запятой указано имя загруженного модуля в начале сеанса работы на кластере).

Список доступных на кластере модулей может отличаться и пополняться с введением нового программного обеспечения.

Работа с модулями обеспечивается командой **module**.

- **module avail -** выдача списка доступных модулей
- **module load <** имя модуля из списка **> -** загрузка модуля, выполняется из командной строки или из конфигурационного файла, настроит Вашу окружающую среду так, чтобы приложение могло использоваться;
- **module unload <** имя загруженного модуля **>** - выгрузка модуля отменит настройки переменных окружения, задаваемых данным модулем;
- **module switch <** имя загруженного модуля **> <** имя модуля **> -** замена загруженного модуля (первого) на указанный модуль (второй);
- **module list -** выдача списка загруженных для данной задачи модулей;
- **module show <** имя модуля из списка **> -** вывод полного имени файла с описанием команд изменения окружения, выполняемых при загрузке модуля;
- **module display <** имя модуля из списка **>-** вывод полного имени файла с описанием модуля как и в команде **module show;**
- **module whatis <** имя модуля из списка **>-** вывод на экран компилятора, библиотеки, приложения, устанавливаемых данным модулем;
- **module whatis -** вывод на экран списка модулей и соответствующих модулям настроек компилятора, библиотеки, приложения;
- **module clear -** выгрузка всех загруженных модулей на текущий момент.

Пример. Список модулей, доступных на кластере Triton на 01.03.2014, выданный с помощью команды

```
[user@triton ~]$ module avail

------------------- /usr/share/Modules/modulefiles -------------------
dot         module-cvs  module-info modules     null        use.own

--------------------------- /etc/modulefiles ---------------------------
mvapich2-x86_64 openmpi-x86_64
```

!	Настройка с помощью команды **module** имеет силу на текущий сеанс работы на кластере.

Для того, чтобы не настраивать заново в начале каждого сеанса работы среду для решения своей задачи, можно нужные настройки сохранить в $HOME/.bashrc пользователя, используя следующие команды:

- **module initadd <** имя модуля из списка доступных**>-** меняет $HOME/.bashrc , загружая указанный модуль для следующих сеансов работы;
- **module initlist -** выдача списка загруженных в $HOME/.bashrc модулей для следующих сеансов работы;
- **module initclear -** чистит $HOME/.bashrc, оставляя лишь модуль null, который не содержит никаких настроек.

Команды запуска задач

Система SLURM позволяет с помощью команд `srun` и `sbatch` работать соответственно в интерактивном и пакетном режимах.

!	Пакетный режим является основным в работе с кластером.

Команда **srun** для запуска интерактивной программы имеет вид:

```
[user@triton ~]$ srun -n <число процессов> -t <кол-во минут> <имя_программы или команда для выполнения> [параметры_программы...] [&]
```

!	При использовании **MPI** перед именем программы добавляется **mpiexec** `srun -n <число процессов> -t <кол-во минут> mpiexec <имя_программы или команда для выполнения> [параметры_программы...] [&]`

Команда **sbatch** для запуска программы в пакетном режиме имеет вид:

```
[user@triton ~]$ sbatch -n <число процессов> -t <кол-во минут> <имя_скрипта>
```

или

```
[user@triton ~]$ sbatch -n <число процессов> -t <кол-во минут> --wrap "<команда для выполнения либо команда запуска программы>"
```

!	При использовании **MPI** перед именем программы добавляется **mpiexec** `sbatch -n <число процессов> -t <кол-во минут> --wrap "mpiexec <имя_программы или команда для выполнения>"`

Команда `sbatch` использует следующий ключи:

-n — задаёт число процессов (tasks); если не задано, то по умолчанию n=1;

-t — заказывает время для решения задачи; при отсутствии t выделяется 30 минут (по умолчанию); 20 часов - максимальное время, выделяемое для счета задачи;

& — позволяет запустить интерактивную задачу в фоновом режиме, при котором пользователю доступна работа в командной строке и одновременно выдача результатов работы интерактивной задачи идет на экран.

-N позволяет задать число узлов (nodes) для задачи, если пользователю это важно:

--mem-per-cpu=<MB> — задаёт минимальную память в расчёте на одно ядро в мегабайтах; если не задано, то по умолчанию 1 ГБ;

--mem=<MB> — задаёт память на узле в мегабайтах.

!	Ключи **`--mem-per-cpu`** и **`--mem`** взаимно исключают друг друга.

-p debug (или **--partition=debug**) позволяет запускать задачи в специально выделенном **для отладки** программ разделе **debug** с максимальным временем счета **20 минут**

Пример. В результате интерактивного запуска

```
srun hostname
```

выдаётся имя узла, на котором запущен соответствующий процесс, например,

```
node16
```

При запуске в пакетном режиме команда запуска программы задаётся либо в **скрипте**, либо через **--wrap**, например,

```
[user@triton ~]$ sbatch mybat
```

или

```
[user@triton ~]$ sbatch -n 2 --wrap "srun hostname"
```

При этом скрипт `mybat` должен содержать следующие строки:

```
#!/bin/sh
#SBATCH -n 2
srun hostname &
wait
```

Скрипт запускается только на первом из выделенных узлов. Запуск нескольких процессов осуществляется командой srun. При этом все опции, указанные в командной строке или самом скрипте в строках **#SBATCH**, приписываются к каждой команде srun данного скрипта, если не переопределены в ней. Так, результирующий файл приведённого примера будет содержать 2 строки с именами узлов (возможно, одинаковых), на которых выполнятся 2 процесса задачи, сформированные командой srun. Если команды srun запускаются в фоновом режиме (символ & в конце строки), то они при наличии ресурсов могут выполняться одновременно.

По умолчанию стандартный **вывод пакетной задачи** будет направлен в файл **slurm-<id>.out**, а стандартный **поток ошибок** — в файл **slurm-<id>.err** текущего каталога. Выдаваемые результаты конкретной команды srun можно поместить вместо стандартного в указанный файл, добавив после команды символ перенаправления вывода

```
srun test > out_test &
```

Можно (чаще, при интерактивном запуске) параллельно просматривать результаты и сохранять их в файле, например:

```
srun --mem 40000 hostname | tee out_hostname
```

Описание всех опций и примеры команд можно посмотреть в **man**-руководстве, например, по команде

```
man srun
```

Примеры постановки задачи в очередь

```
user@triton:~$ sbatch -n 3 --wrap "mpiexec test2 3 5.1"
```

```
Submitted batch job 776
```
Результат: сформирована пакетная задача с запуском 3-х процессов test2 с 2 параметрами; Задаче присвоен уникальный идентификатор 776;

```
user@triton:~$ srun -N 2 sleep 30 &
[1] 22313
```
Результат: сформирована интерактивная задача в фоновом режиме; Номер фоновой задачи в текущем сеансе, 22313 - pid процесса srun на управляющей машине. Уникальный идентификатор можно узнать с помощью команд `squeue`, `sacct`.

Информационные команды

squeue — просмотр очереди (информации о задачах, находящихся в счете или в очереди на счет); возможно использование ключей, например:

```
user@triton:~$ squeue --user=`whoami`   — посмотреть только свои задачи;
user@triton:~$ squeue --states=RUNNING  — посмотреть считающиеся задачи;
user@triton:~$ squeue --long            — выдать более подробную информацию.
```

Пример

```
user@triton:~$ srun -N 2 sleep 30 &
[1] 22313
user@triton:~$ squeue
  JOBID PARTITION     NAME     USER    ST    TIME  NODES NODELIST(REASON)
   777    all        sleep     user    R     0:23    2      node[10,15]
```

JOBID — уникальный идентификатор задачи; никогда не используется повторно ;
PARTITION — название раздела, где считается задача;
NAME — имя задачи пользователя;
USER — логин пользователя;
ST — состояние задачи (R - выполняется, PD - в очереди);
TIME — текущее время счета;
NODES — количество узлов для задачи;
NODELIST(REASON) — список выделенных узлов.

sacct — просмотр задач текущего пользователя за сутки (с начала текущего дня);

Пример

```
user@triton:~$ sacct -a --starttime 2014-01-01

JobID      JobName    Partition   Account   AllocCPUS  State      ExitCode
---------- ---------- ---------- ---------- ---------- ---------- --------
522        sbatch     perm                  2          COMPLETED    0:0
522.batch  batch                            1          COMPLETED    0:0
777        sleep      debug                 2          CANCELLED+   0:0
780        sbatch     perm                  2          FAILED       0:0
780.batch  batch                            1          FAILED       127:0
783        sleep      perm                  2          RUNNING      0:0
```

JobID — уникальный идентификатор задачи, повторно не используется;
JobName — имя задачи пользователя;
Partition — название раздела, где считается задача;

State — состояние задачи: RUNNING — выполняется,
COMPLETED — закончилась,
FAILED — закончилась по ошибке,
CANCELLED+ — снята пользователем;
ExitCode — код возврата.

sinfo — просмотр информации об узлах (прежде всего, о состоянии узлов: доступны, заняты, свободны, ...).

Пример

```
user@triton:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up      20:00      1   idle node48
perm*        up   20:00:00     48   idle node[1-48]
```

PARTITION — название раздела, где считаются задачи,
***** - указывает на раздел по умолчанию;
AVAIL — состояние раздела узлов: **up** - есть доступ, **down** - нет доступа;
TIMELIMIT — максимальное время, выделяемое для счета задачи;
NODES — количество узлов;
STATE — состояние (в сокращённой форме):
idle - свободен,
alloc - используется процессом,
mix - частично занят, частично свободен,
down, drain, drng - заблокирован,
comp - все задания, связанные с этим узлом, находятся в процессе завершения;
***** - обозначает узлы, которые в настоящее время не отвечают (not responding);
NODELIST — список узлов.

scontrol — выдача детальной информации об узлах, разделах, задачах:

`user@triton:~$ scontrol show node node13` — об узле
`user@triton:~$ scontrol show partition` — о разделах;
`user@triton:~$ scontrol show job 174457` — о задаче.

Удаление задач из очереди

Для отмены выполнения задачи служит команда **`scancel`**:

`user@triton:~$ scancel <id1,id2,...,idn>` — снимает задачи с уникальными идентификаторами id1,id2,...,idn

`user@triton:~$ scancel -u user` — снимает все задачи пользователя user

`user@triton:~$ scancel --state=RUNNING 1,2,3` — снимает со счета уже стартовавшие задачи с идентификаторами 1,2 и 3

`CTRL+C` — снимает интерактивную задачу без фонового режима.

Пример

```
user@triton:~$ squeue
JOBID  PARTITION     NAME     USER     ST       TIME  NODES NODELIST(REASON)
977    perm          sleep    user     R        1:32    2      node[1-2]
```

Снять со счета интерактивную задачу, считающуюся в фоновом режиме.

```
user@triton:~$ scancel 977

srun: Force Terminated job 977
user@triton:~$ srun: Job step aborted: Waiting up to 2 seconds for job step to
finish.
slurmd[node1]: *** STEP 977.0 CANCELLED AT 2011-11-01T18:04:45 ***
srun: error: node1: task 0: Terminated
srun: error: node2: task 1: Terminated
Enter      — нажать
[1]+  Exit 15                 srun -p perm -N 2 sleep 1h


user@triton:~$ squeue
  JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
```

Список литературы

Инструкция основана на материалах сайта Параллельные вычисления в УрО РАН.

1. Модули | Параллельные вычисления в УрО РАН (URL: http://parallel.uran.ru/node/270).
2. Запуск задач на кластере в системе SLURM | Параллельные вычисления в УрО РАН (URL: http://parallel.uran.ru/node/313).