در راهنمای نحوه اتصال به خوشه چگونگی متصل شدن به ترمینال خوشه محاسباتی،از طریق پروتکلهای ssh و sftp آموزش داده شده است. در اینجا فرض بر این است که کاربر ثبت نام کرده و با نام کاربری و رمز عبور خویش به خط فرمان خوشه دسترسی دارد.
همانطور که در شکل نیز دیده می‌شود، کاربر reza به ترمینال خوشه متصل شده و در پوشه test دو فایل ذخیره کرده است. فایل m_code.f یک برنامه به زبان فرترن است (برای محاسبه عدد π) که با MPI موازی سازی شده.

```
[reza@saffron ~]$ cd test/
[reza@saffron test]$ ls
my_code.f  my_code.sh
[reza@saffron test]$
```

برای کامپایل کردن این برنامه می توان از کامپایلرهای مختلفی که روی خوشه نصب است استفاده کرد. برای مثال استفاده از کامپایلر gfortran به این صورت است:

```
[reza@saffron test]$ gfortran my_code.f
```

توجه داشته باشید که خوشه محاسباتی برای کامپایل برنامه ها و پکیجها، از گره اصلی (head node) استفاده می‌کند. اگر قصد دارید برنامه را بصورت موازی روی خوشه اجرا کنید، باید از کامپایلر mpif90 استفاده کنید. (برای سایر زبانها از mpif77، mpicc، mpic++ و ... استفاده کنید) اگر کامپایل برنامه بدون ایراد انجام شود، یک فایل اجرایی با نام a.out تولید خواهد شد، که قرار است توسط گره‌های محاسباتی اجرا شود.

```
[reza@saffron test]$ mpif90 my_code.f
[reza@saffron test]$ ls
a.out  my_code.f  my_code.sh
[reza@saffron test]$
```

راه‌های مختلفی برای اجرای فایل اجرایی a.out وجود دارد. ساده‌ترین راه استفاده از دستور زیر است:

```
[reza@saffron test]$ ./a.out
```

این دستور یک اجرای سریال بر روی گره اصلی را آغاز خواهد کرد! این روش اجرا برای همه کاربران کاملا **ممنوع** است.
برای اجرای موازی روی (مثلا ۱۶ هسته از) گره‌های محاسباتی، باید از دستور mpirun استفاده شود:

```
[reza@saffron test]$ mpirun -np 16 ./a.out
```

با این دستور، یک اجرای موازی روی گره‌های محاسباتی آغاز خواهد شد! البته بدون درنظر گرفتن این موضوع که ممکن است برنامه سایر کاربران قبلا روی همان گره‌ها در حال اجرا بوده باشد! به همین دلیل استفاده از این روش نیز برای همه کاربران **ممنوع** است.

استفاده همزمان چند کاربر از یک خوشه محاسباتی، نیازمند بکارگیری یک سیستم صف‌بندی مناسب است. خوشه سَفران از SGE برای صف‌بندی وظایف محاسباتی (job)استفاده می‌کند.

برای اجرای برنامه‌ها تحت نظارت SGE، باید ابتدا یک اسکریپت مناسب بنویسیم. کاربر reza چنین اسکریپتی را قبلا در پوشه test قرار داده است. برای مشاهده و تغییر محتوای اسکریپت m_code.sh، می‌توانیم از ویرایشگر متن nano که در ترمینال اجرا می‌شود، استفاده کنیم:

```
[reza@saffron test]$ nano my_code.sh
```

برای دانستن معنی سطرهایی که با علامت # آغاز شده‌اند راهنمای SGE را مطالعه کنید. سه سطر آخر به ترتیب اجرا خواهند شد. دستور date اول، زمان شروع اجرا و دستور date دوم، زمان اتمام اجرا را ثبت خواهند کرد (استفاده از آنها ضروری نیست). در بین این دو دستور، دستور اجرای سریال فایل a.out نوشته شده است،

```
  GNU nano 2.0.9                File: my_code.sh

#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -S /bin/bash
#
date
./a.out
date


^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cu
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U Un
```

دستور qsub برای ارسال اسکریپت (به صف اجرا) بکار میرود. با اجرای آن پیغامی دریافت خواهید کرد که حاوی یک شماره شناسایی است. این شماره را یادداشت کنید چون برای اطلاع از وضعیت اجرای برنامه خود به آن نیاز خواهید داشت.

```
[reza@saffron test]$ qsub my_code.sh
Your job 38 ("my_code.sh") has been submitted
[reza@saffron test]$
```

با اجرای دستور qstat می توانید از وضعیت فعلی صف اجرای برنامه‌ها مطلع شوید. مشاهده می‌کنید که job شماره ۳۸ در حال اجرا روی یک هسته از گره compute-1-6 می‌باشد.

```
[reza@saffron test]$ qstat
job-ID  prior   name       user  state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------
     38 0.55500 my_code.sh reza  r     04/22/2015 12:45:50 all.q@compute-1-6.local        1
[reza@saffron test]$
```

وقتی اجرای برنامه شما تمام شود اطلاعات آن دیگر توسط qstat نمایش داده نخواهد شد.

```
[reza@saffron test]$ qstat
[reza@saffron test]$
```

خروجی اسکریپت اجرا شده (نتایج مورد نظر یا اخطارهای احتمالی)، بصورت یک فایل جدید در همان پوشه کاری شما ذخیره خواهد شد.

```
[reza@saffron test]$ ls
a.out  my_code.f  my_code.sh  my_code.sh.o38
[reza@saffron test]$
```

با نگاه به فایل m_code.sh.o34، ملاحظه می‌کنیم که برنامه محاسباتی ما در مدت زمان ۲۸ ثانیه (بصورت سریال) اجرا شده است.

```
[reza@saffron test]$ cat my_code.sh
my_code.sh      my_code.sh.o38
[reza@saffron test]$ cat my_code.sh.o38
Wed Apr 22 13:15:47 AZST 2015
 PI calculated with 1000000000  points = 3.1416009435345842
Wed Apr 22 13:16:15 AZST 2015
[reza@saffron test]$
```

برای اجرای موازی همین برنامه باید دو کار انجام دهیم. اول اینکه در اسکریپت قبلی و در سطر مربوط به اجرای فایل a.out، دستور mpirun را اضافه کنیم.

```
#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -S /bin/bash
#
date
mpirun ./a.out
date
```

دوم اینکه از شکل کاملتر دستور qsub برای ارسال اسکریپت استفاده کنیم.

```
[reza@saffron test]$ qsub -pe orte 16 my_code.sh
Your job 39 ("my_code.sh") has been submitted
[reza@saffron test]$
```

اکنون اسکریپت برای پردازش موازی روی ۱۶ هسته ارسال شده است. مشاهده وضعیت صف نیز این موضوع را تائید می‌کند.

```
[reza@saffron test]$ qsub -pe orte 16 my_code.sh
Your job 39 ("my_code.sh") has been submitted
[reza@saffron test]$ qstat
job-ID  prior   name       user  state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------
     39 0.55500 my_code.sh reza  r     04/22/2015 12:49:50 all.q@compute-1-5.local        16
[reza@saffron test]$
```

این دفعه فایل خروجی مدت زمان اجرای برنامه را ۶ ثانیه نشان می‌دهد. فایل خروجی دیگری با نام m_code.sh.po39 نیز تولید شده است که ممکن است حاوی برخی خطاها و اخطارها باشد.

```
[reza@saffron test]$ ls
a.out  my_code.f  my_code.sh  my_code.sh.o38  my_code.sh.o39  my_code.sh.po39
[reza@saffron test]$ cat my_code.sh.o39
Wed Apr 22 13:19:50 AZST 2015
 PI calculated with            1000000000  points =    3.1415818318904005
Wed Apr 22 13:19:56 AZST 2015
[reza@saffron test]$ cat my_code.sh.po39
[reza@saffron test]$
```

اگر پس از ارسال یک job به صف، خواستید از اجرای آن جلوگیری کنید، باید از دستور qdel و شماره شناسایی مربوط به آن job استفاده کنید. مثلا برای توقف job شماره ۴۰ دستور زیر را باید اجرا کنیم:

```
[reza@saffron test]$ qsub -pe orte 4 my_code.sh
Your job 40 ("my_code.sh") has been submitted
[reza@saffron test]$ qdel 40
reza has registered the job 40 for deletion
[reza@saffron test]$
```

نحوه اجرای برنامه های مربوط به پکیجهای محاسباتی نیز کم و بیش به همین صورت است. برای مثال در اینجا کاربر reza قصد دارد یک شبیه‌سازی آزمایشی را با استفاده از نرم افزار LAMMPS، روی خوشه اجرا کند. کاربر قبلا نرم افزار را در دایرکتوری home خود نصب کرده و فایل اجرایی با نام lmp_openmpi را نیز با موفقیت تولید کرده است. کاربر همچنین اسکریپت شبیه‌سازی (in.lj) و اسکریپت صف (in – lj.sh) را آماده کرده و در پوشه test-lammps قرار داده است.

```
[reza@saffron test-lammps]$ ls
in.lj  in-lj.sh
[reza@saffron test-lammps]$ nano in-lj.sh
```

با نگاه به اسکریپت صف مشاهده می کنید که پس از دستور mpirun، ابتدا آدرس فایل اجرایی و سپس آدرس اسکریپت شبیه‌سازی (که حاوی داده های ورودی شبیه سازی است) آورده شده است.

```
  GNU nano 2.0.9        File: in-lj.sh

#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -S /bin/bash
#
mpirun $HOME/lammps-29Aug14/src/lmp_openmpi  <in.lj









                      [ Read 8 lines ]
^G Get He^O WriteO^R Read F^Y Prev P^K Cut Te^C Cur Pos
^X Exit  ^J Justif^W Where ^V Next P^U UnCut ^T To Spell
```

ارسال اسکریپت صف روی خوشه، درست شبیه مثال قبل است.

```
[reza@saffron test-lammps]$ qsub -pe orte 16 in-lj.sh
Your job 44 ("in-lj.sh") has been submitted
[reza@saffron test-lammps]$
```

پس از پایان اجرا، علاوه بر فایلهای خروجی SGE، فایلهای خروجی نرم افزار LAMMPS نیز تولید خواهند شد (مثل فایل log.lammps)

```
[reza@saffron test-lammps]$ ls
in.lj  in-lj.sh  in-lj.sh.o44  in-lj.sh.po44  log.lammps
[reza@saffron test-lammps]$
```

در اینجا محتوای فایل خروجی in – lj.sh.o44 را می‌توانید مشاهده کنید:

```
[reza@saffron test-lammps]$ cat in-lj.sh.o44
LAMMPS (29 Aug 2014)
Lattice spacing in x,y,z = 1.6796 1.6796 1.6796
Created orthogonal box = (0 0 0) to (167.96 167.96 167.96)
  2 by 2 by 4 MPI processor grid
Created 4000000 atoms
Setting up run ...
Memory usage per processor = 42.3553 Mbytes
Step Temp E_pair E_mol TotEng Press
       0         1.44   -6.7733681            0   -4.6133686   -5.0196696
     100   0.75932016   -5.7612878            0   -4.6223079    0.19054797
Loop time of 35.0282 on 16 procs for 100 steps with 4000000 atoms

Pair  time (%) = 17.8894 (51.0714)
Neigh time (%) = 2.15841 (6.16192)
Comm  time (%) = 13.7326 (39.2043)
Outpt time (%) = 0.0305565 (0.087234)
Other time (%) = 1.21726 (3.4751)

Nlocal:    250000 ave 250200 max 249824 min
Histogram: 2 3 1 1 1 1 2 3 1 1
Nghost:    75318.5 ave 75475 max 75140 min
Histogram: 2 3 1 0 2 1 0 1 1 5
Neighs:    9.38904e+06 ave 9.42544e+06 max 9.36203e+06 min
Histogram: 1 5 0 1 4 1 1 0 1 2

Total # of neighbors = 150224627
Ave neighs/atom = 37.5562
Neighbor list builds = 5
Dangerous builds = 0
[reza@saffron test-lammps]$
```