محمود نادران طحان

۹۵/۰۴/۱۵

نسخه ۱

مرکز پردازش‌های سریع
دانشگاه شهید چمران اهواز

استفاده از برنامه مدیر کار برای اجرای برنامه‌ها

به طور معمول در خوشه‌های[1] محاسباتی، چند گره پردازشی[2] وجود دارند که برنامه‌های کاربران را اجرا می‌کنند. بنابراین، برای این که کاربر بتواند برنامه خود را بر روی خوشه اجرا کند، باید یک گره پردازشی بیکار[3] پیدا کند و سپس برنامه خود را بر روی آن گره اجرا کند. با توجه به این که این کار زمان‌بر و از حوصله کاربران خارج است، یک گره پردازشی به عنوان گره اصلی انتخاب می‌شود تا نقش رابط بین کاربران و خوشه را به عهده بگیرد. به همین خاطر، یک برنامه بر روی گره اصلی وجود دارد که وظیفه آن مدیریت سایر گره‌ها است تا در صورت رسیدن یک درخواست اجرا (برنامه کاربر)، یک گره بیکار پیدا کند و اجرای برنامه را به آن گره محول کند. به این برنامه‌ها اصطلاحاً برنامه «مدیر کار[4]» گفته می‌شود.

برنامه‌های مدیریتی با امکانات مختلفی موجود هستند که یکی از معروف‌ترین آن‌ها، مدیر کار TORQUE است. وظیفه اصلی آن پیدا کردن منبع (گره) خالی برای اجرای برنامه‌ها است و برای این کار از یک زمان‌بند[5] موسوم به MAUI استفاده می‌کند تا در صورت پر بودن همه گره‌های پردازشی، برنامه‌ها را در صف قرار دهد و زمان‌بندی کند. با توجه به این که برنامه زمان‌بند از دید کاربران پنهان است، بیش از این نیازی به معرفی آن نیست.

علاوه بر ساده‌سازی محیط استفاده از خوشه محاسباتی، برنامه مدیر کار یک نقطه قوت دیگر هم دارد و آن این است که در صورت به کار گیری آن، کاربر می‌تواند برنامه مورد نظر خود را به مدیر کار ارسال[6] کند و بعد از آن اتصال خود به خوشه را قطع کند. در این صورت نیازی نیست که همیشه کاربر به خوشه متصل بماند تا برنامه اجرا شود. مدیر کار (در اینجا TORQUE) به صورت خودکار در پس زمینه[7] برنامه کاربر را مدیریت می‌کند و نتایج خروجی تولید شده توسط برنامه کاربر را بر روی دیسک ذخیره می‌کند.

با توجه به این که کاربران برنامه‌های خود را باید به مدیر کار ارسال کنند، لازم است نحوه تعامل با آن را فرا بگیرند. به طور عادی، کاربران برای اجرای برنامه‌های خود یک دستور را در لینوکس تایپ می‌کنند. برای ارسال برنامه به مدیر کار یک اسکریپت[8] نوشته می‌شود که حاوی دو قسمت است. قسمت اول شامل تعدادی راهنما برای فرمان دادن به TORQUE است تا مشخصات برنامه در صف اجرا ثبت شوند. قسمت دوم حاوی دستور اجرای برنامه است که به طور عادی در خط فرمان لینوکس نوشته می‌شود. یک اسکریپت نمونه به شکل زیر است. توجه کنید که قسمت‌هایی که با رنگ قرمز نوشته شده‌اند، ثابت هستند و کاربران می‌توانند بدون تغییر آن‌ها را در اسکریپت‌های خود بنویسند. قسمت‌هایی که با رنگ آبی نوشته شده‌اند، بسته به نیاز کاربر می‌تواند تغییر کند.

[1] Cluster
[2] Processing node
[3] Idle
[4] Job manager
[5] Scheduler
[6] Submit
[7] background
[8] Script

```
#PBS -V
#PBS -q default
#PBS -j oe
#PBS -l nodes=1
#PBS -N job
#PBS -o /path/to/log/file
cd $PBS_O_WORKDIR
your command
```

در این اسکریپ، ابتدا تعدادی راهنما (خطوطی که با # شروع می‌شوند) وجود دارند که به وسیله آن‌ها به مدیر کار اعلام می‌شود، برنامه چه نیازها و مشخصاتی دارد. در این جا، به توضیح برخی از آن‌ها که بیشتر کاربرد دارند، می‌پردازیم.

- `#PBS -N job` نام برنامه را به مدیر کار اعلام می‌کند. این نام دلخواه است و زمانی به کار می‌آید که می‌خواهیم از وضعیت اجرای برنامه در لیست برنامه‌های موجود در صف مطلع شویم.
- `#PBS -o /path/to/log/file` به مدیر کار اعلام می‌کند که خروجی تولید شده بر روی صفحه نمایش را در یک فایل (که مسیر آن داده می‌شود) بنویسد. توجه کنید که این فایل حاوی اطلاعاتی است که به طور معمول بر روی صفحه نمایش نشان داده می‌شوند
- `your command` همان دستوری است که به طور معمول کاربر در محیط لینوکس وارد می‌کند تا برنامه خود را اجرا کند.

مثال شماره ۱) فرض کنید می‌خواهیم دستور `date` را به مدیر کار ارسال کنیم تا آن را بر روی یک گره اجرا کند و نتیجه (ساعت و تاریخ سیستم) را در یک فایل در مسیر `/home/mahmood/date.log` بنویسد. بنابراین اسکریپت مربوطه به صورت زیر خواهد بود.

```
#PBS -l nodes=1
#PBS -V
#PBS -q default
#PBS -N MyFirstExample
#PBS -o /home/mahmood/date.log
#PBS -j oe

cd $PBS_O_WORKDIR
date
```

برای ارسال اسکریپ، از دستور `qsub` استفاده می‌شود که باید نام اسکریپت را به این دستور بدهیم. اگر نام اسکریپت نوشته شده `first_example.tor` باشد، دستور را به شکل زیر در ترمینال لینوکس اجرا می‌کنیم.

```
[mahmood@cluster ~]$ qsub first_example.tor
277.cluster.scu.ac.ir
```

با توجه به این که زمان اجرای این دستور کم است، می‌توانیم با استفاده از دستور `ls` از ساخته شدن فایل `date.log` اطمینان حاصل کنیم. همچنین به کمک دستور `cat` می‌توانیم محتویات فایل را مشاهده کنیم.

```
[mahmood@cluster ~]$ ls date.log
date.log
[mahmood@cluster ~]$ cat date.log
Mon Jul  4 01:59:19 IRDT 2016
```

مثال شماره ۲) فرض کنید می‌خواهیم یک اسکریپت پوسته[۹] بنویسیم که چند دستور را به صورت پشت سر هم اجرا کند. خط اول، یک راهنما است که لینوکس از طریق آن متوجه می‌شود چند دستور موجود در این فایل را باید در خط فرمان لینوکس به صورت خودکار و پشت سر هم اجرا کند. خط دوم پیغام "hello1" را چاپ می‌کند. خط سوم، اجرای اسکریپت را به مدت ۶۰ ثانیه متوقف می‌کند و خط چهارم پیغام "hello2" را چاپ می‌کند. بنابراین یک فایل به نام `second_example.sh` می‌سازیم و دستورات زیر را در آن می‌نویسیم.

```
#!/bin/bash
echo "hello1"
sleep 60
echo "hello2"
```

حال، یک فایل به نام `second_example.tor` می‌سازیم و مانند مثال اول، راهنماهای PBS را در آن می‌نویسیم. فراموش نکنید که دستور مورد نظر در این مثال اسکریپت `second_example.sh` است که به صورت `./second_example.sh` فراخوانی می‌شود. همچنین مطلوب آن است که خروجی برنامه (پیغام‌های hello1 و hello2) در فایلی به نام `commands.log` نوشته شوند.

```
#PBS -l nodes=1
#PBS -V
#PBS -q default
#PBS -N MySecondExample
#PBS -o /home/mahmood/commands.log
#PBS -j oe

cd $PBS_O_WORKDIR
./second_example.sh
```

حال از دستور qsub استفاده می‌کنیم تا اسکریپ را اجرا کنیم. از آن جایی که در لینوکس، فایل‌های اسکریپت (که با `#!/bin/bash` شروع می‌شوند باید قابلیت اجرایی داشته باشند، ابتدا باید از دستور chmod با سوئیچ +x استفاده کنیم.

```
[mahmood@cluster ~]$ chmod +x second_example.sh
[mahmood@cluster ~]$ qsub second_example.tor
278.cluster.scu.ac.ir
```

با توجه به این که در این مثال یک وقفه ۶۰ ثانیه‌ای ایجاد کردیم، می‌توانیم روند اجرای آن را در صف برنامه‌ها مشاهده کنیم. برای این منظور بلافاصله بعد از qsub، دستور qstat را وارد می‌کنیم. در خروجی این دستور مشاهده می‌کنیم که کار شماره ۲۷۸ (همان شماره‌ای که در خروجی دستور qsub بود) نام MySecondExample دارد (همان عبارتی که جلوی `#PBS -N` نوشته شده است) و در وضعیت اجرا (R) قرار دارد. در ستون سمت چپ وضعیت سه عدد به شکل 00:00:00 نوشته شده است. این اعداد مدت زمان اجرای برنامه را نشان می‌دهند. مقدار زمان اجرا برنامه به مرور توسط مدیر کار Torque به روز می‌شود. بنا به دلایلی، این به روز رسانی لحظه‌ای صورت نمی‌گیرد. می‌توانید با اجرای چند بار دستور qstat این موضوع را دنبال کنید. مادامی که وضعیت کار در حالت R قرار دارد، یعنی برنامه در حال اجرا است.

```
[mahmood@cluster ~]$ qstat
Job id            Name             User    Time Use S Queue
------------ ------------ ---------- -------- - -----
279.cluster MySecondExample  mahmood          0 R default
```

[۹] Shell script

```
[mahmood@cluster ~]$ qstat
Job id            Name         User    Time Use S Queue
------------ ------------ ---------- -------- - -----
279.cluster MySecondExample  mahmood   00:00:00 R default
```

بعد از اجرای برنامه با وارد کردن دستور qstat متوجه می‌شویم که وضعیت برنامه در حالت C به معنای پایان یافته تغییر کرده است.

در این هنگام می‌توانیم با مشاهده محتویات فایل command.log از اجرای صحیح برنامه مطلع شویم.

```
[mahmood@cluster ~]$ qstat
Job id            Name         User    Time Use S Queue
------------ ------------ ---------- -------- - -----
279.cluster MySecondExample  mahmood   00:00:00 C default
[mahmood@cluster ~]$ ls commands.log
commands.log
[mahmood@cluster ~]$ cat commands.log
hello1
hello2
```

نکته: اگر بنا بر هر دلیل قصد داشته باشیم که یک برنامه در حال اجرا را از صف خارج کنیم، از دستور qdel استفاده خواهیم کرد. برای این کار، شماره برنامه در حال اجرا (ستون اول دستور qstat) را باید به این دستور بدهیم. این دستور عملاً باعث قطع اجرای دستور و کشتن[10] آن می‌شود و وضعیت برنامه به C تغییر می‌یابد. دقت کنید که در این حالت ممکن است خروجی برنامه معتبر نباشد. یک بار دیگر، مثال شماره ۲ را اجرا می‌کنیم و از این دستور برای حذف/کشتن برنامه در حال اجرا استفاده می‌کنیم. سپس با دستور qstat متوجه می‌شویم که برنامه از صف خارج شده است. با استفاده از دستور cat مشاهده خواهیم کرد که فقط hello1 در فایل نوشته شده است. این به معنی آن است که به دلیل قطع برنامه، پیغام hello2 در فایل چاپ نشده است.

```
[mahmood@cluster ~]$ qsub second_example.tor
280.cluster.scu.ac.ir
[mahmood@cluster ~]$ qstat
Job id            Name         User    Time Use S Queue
------------ ------------ ---------- -------- - -----
280.cluster MySecondExample  mahmood   00:00:00 R default
[mahmood@cluster ~]$ qdel 280
[mahmood@cluster ~]$ qstat
Job id            Name         User    Time Use S Queue
------------ ------------ ---------- -------- - -----
280.cluster MySecondExample  mahmood   00:00:00 C default
[mahmood@cluster ~]$ cat commands.log
hello1
```

[10] Kill