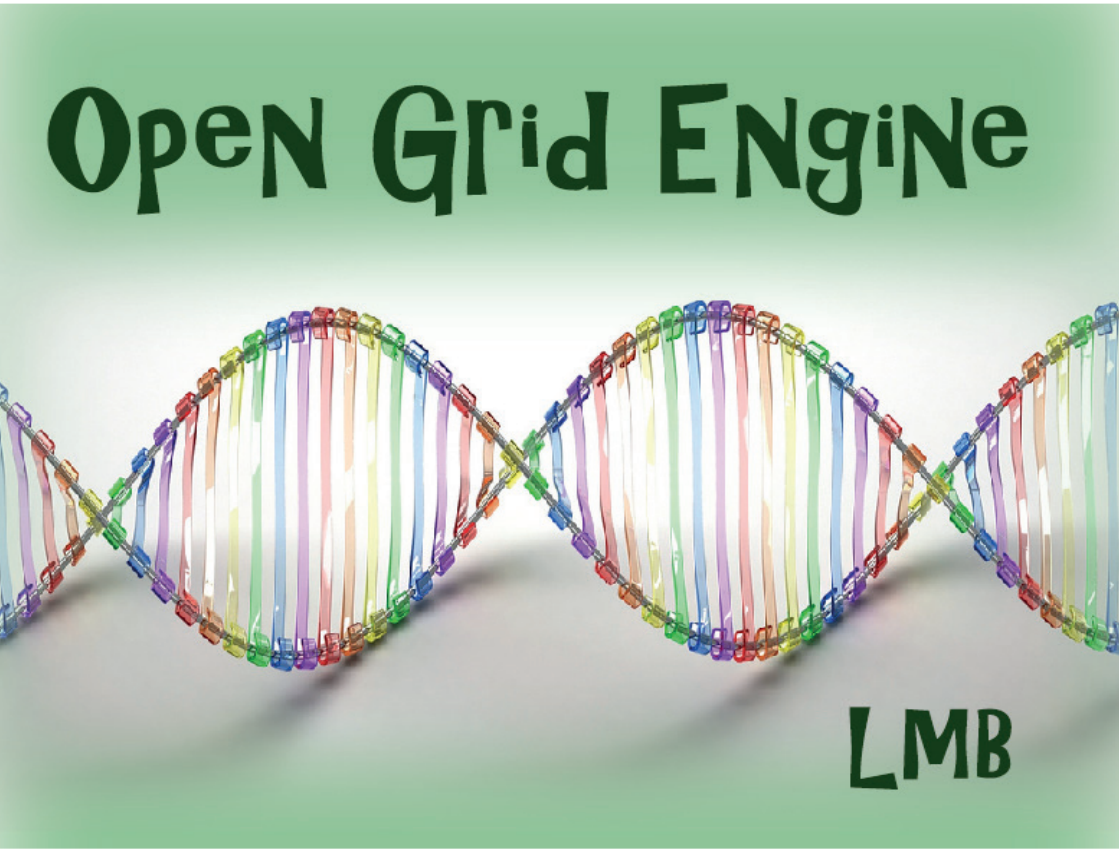
Documentos

ISSN 1677-9274
Dezembro, 2016
154

Instruções para uso do Open Grid Engine no Laboratório Multiusuário de Bioinformática



Embrapa

ISSN 1677-9274
Dezembro, 2016

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

Documentos 154

Instruções para uso do Open Grid Engine no Laboratório Multiusuário de Bioinformática

Leandro Carrijo Cintra

Embrapa Informática Agropecuária
Campinas, SP
2016

Embrapa Informática Agropecuária
Av. André Tosello, 209 - Barão Geraldo
Caixa Postal 6041 - 13083-886 - Campinas, SP
Fone: (19) 3211-5700
www.embrapa.br/informatica-agropecuaria
SAC: www.embrapa.br/fale-conosco/sac/

1ª edição
publicação digitalizada 2016

Dados Internacionais de Catalogação na Publicação (CIP)
Embrapa Informática Agropecuária

Cintra, Leandro Carrijo.

Instruções para uso do Open Grid Engine no Laboratório Multiusuário de Bioinformática / Leandro Carrijo Cintra.- Campinas : Embrapa Informática Agropecuária, 2016.

26 p. : il. ; cm. - (Documentos / Embrapa Informática Agropecuária, ISSN 1677-9274; 144).

1. Cluster. 2. Bioinformática. 3. Processamento de dados biológicos. I. Embrapa Informática Agropecuária. II. Título. III. Série.

CDD 570.285

Autor

Leandro Carrijo Cintra
Cientista da Computação, Doutor em Bioinformática Analista da Embrapa Informática Agropecuária, Campinas, SP.

Apresentação

Os avanços recentes na geração de dados nas áreas de biologia molecular e modelagem e simulação fizeram com que diversos projetos de pesquisa nessas áreas passassem a ser fortemente orientados à geração de grandes volumes de informações.

Assim, organizações, tais como a Embrapa, que mantém projetos de pesquisa nas áreas citadas fazem parte daquelas instituições que têm se deparado com os problemas de armazenar, processar e analisar grandes volumes de dados na atualidade. Este documento apresenta informações úteis para os novos usuários do Laboratório Multiusuário de Bioinformática da Embrapa, constituído pela empresa para o atendimento às demandas de seus projetos envolvendo o armazenamento e processamentos de dados biológicos.

Discute-se as principais características do sistema de *batch* Sun Grid Engine (SGE), (*Open Grid Engine*) que é utilizado para o gerenciamento do processamento no cluster. São apresentados os recursos desse sistema mais utilizados pelos usuários no decorrer das execuções de suas análises de dados. O documento deixa explícito as formas dos comandos que devem ser emitidos pelos usuários e esclarece os resultados que serão alcançados no que concerne ao gerenciamento das atividades de processamento dos dados nas análises. São fornecidos exemplos ao longo do texto, para que o usuário possa se orientar e utilizá-los como referência em suas próprias análises.

O presente trabalho tem sua relevância no treinamento dos recursos humanos que estão atuando no âmbito de projetos da Embrapa que exigem o processamento intenso e, por isso, demandam recursos computacionais especiais para a obtenção de seus resultados. No entanto, problemas similares ocorrem em outras organizações de ensino e pesquisa no País; e a discussão apresentada pode ser útil também para profissionais dessas, que queiram ampliar a sua capacidade de processamento utilizando clusters computacionais.

Silvia Maria Fonseca Silveira Massruhá
Chefe-geral
Embrapa Informática Agropecuária

Sumário

Instruções para uso do Open Grid Engine no Laboratório Multiusuário de Bioinformática

Leandro Carrijo Cintra

1. Introdução

Preliminarmente, deve-se fazer um esclarecimento importante, as informações que constam neste manual são úteis aos usuários que executam suas análises emitindo comandos diretamente no *shell* do Linux, não sendo necessárias em momento algum, para os usuários que trabalham com o Galaxy.

O Laboratório Multiusuário de Bioinformática (LMB) da Empresa Brasileira de Pesquisa Agropecuária (Embrapa) disponibiliza recursos computacionais para armazenamento e processamento de alto desempenho para serem utilizados no âmbito de projetos da Embrapa e de parceiros priorizando, mas não se limitando, àqueles que exigem a análise de dados biológicos. Essa demanda já está bastante evidenciada em várias áreas científicas e tende a se intensificar no futuro, como discutido em Hey et al. (2011).A arquitetura computacional adotada no laboratório segue o padrão relativo ao processamento de dados científicos, a saber, o uso de um *cluster* no qual os usuários podem se conectar remotamente via uma máquina

específica e submeter suas tarefas, também por intermédio único dessa máquina, a qual se encarregará de distribuir as atividades para as demais máquinas do *cluster*. No sistema em questão, que visa o processamento de dados biológicos, a plataforma adequada para essa tarefa é o Linux e uma ampla gama de análises são rotineiramente realizadas utilizando-se linhas de comando por intermédio de um *shell*. Dessa forma, é necessário que exista no *cluster* um sistema que se encarregue de priorizar as tarefas a serem realizadas, distribua estas tarefas entre as máquinas do *cluster* e acompanhe a execução de cada tarefa.

Sistemas com estas capacidades são tradicionalmente conhecidos como sistemas de filas (batch-queuing systems) e fazem parte de uma área mais ampla, conhecida como “Distributed Resources Management”(DRM), (SUN MICROSYSTEMS, 2005). No LMB implantou-se o sistema gestor de filas “Open Grid Engine”, no passado conhecido como Sun Grid Engine (SGE), e que por uma questão de tradição, será denominado por essa última forma ao longo deste documento. O SGE foi um projeto desenvolvido pela SUN e, posteriormente, tornou-se *open-source*. Com este documento, pretende-se facilitar a introdução nesse ambiente de análises para os atuais e futuros usuários do LMB.

Infraestruturas computacionais cuja principal finalidade seja fornecer processamento para seus usuários, enfrentam de forma mais, ou menos intensa, o problema de competição pelos recursos. As soluções adotadas pelos gestores sempre envolvem a priorização de projetos e atividades que serão atendidas pelo parque computacional. É justamente nesse ponto que um sistema gestor de filas torna-se imperativo, pois controlar a prioridade de execução dos *jobs* manualmente, só é viável em baixa escala.

Assim, toda a priorização de execução dos projetos do LMB é controlada automaticamente, e os usuários podem fazer a submissão de seus trabalhos (*jobs*) a qualquer instante, sem a necessidade de consultas a um operador humano para se determinar se é ou não possível usar a infraestrutura em um determinado período.

Outro grande motivo para se utilizar gestores de filas é que a arquitetura da solução computacional de alto desempenho torna-se transparente para os usuários finais. Desta forma, não há mais a necessidade de se saber em qual máquina deve-se executar um determinado *job*. O gestor de filas

gerencia isso, e muito mais. Assim, a inserção de novas máquinas à infraestrutura é transparente e estas passam a ser utilizadas assim que instaladas e configuradas, sem a necessidade de se levar ao conhecimento dos usuários que elas existem.

Os gestores de filas também possibilitam a produção de relatórios de uso dos recursos computacionais (CPU, memória, espaço utilizado em disco, etc...) mais elaborados, pois têm ferramentas para registrar as informações sobre os recursos consumidos e, posteriormente, emitir sumários sobre tais informações.

Um inconveniente com tais sistemas é que a forma de submissão dos trabalhos é diferente em relação ao que a maioria dos usuários está habituada (exceção feita àqueles que usam ou usaram outras infraestruturas de alto desempenho) a usar no seu dia a dia, exigindo assim um esforço inicial no aprendizado do novo sistema. Este documento traz algumas informações básicas sobre este novo modo de execução das análises.

1.1. Considerações sobre os programas que devem ser submetidos por meio do SGE

Normalmente, em grandes ambientes de processamento, os usuários têm acesso a uma única máquina da infraestrutura e fazem todas as suas submissões a partir dela. Neste momento no LMB, os usuários têm acesso a todas as máquinas e podem fazer uma submissão sem necessariamente utilizarem o gestor de filas. Procede-se dessa forma pois objetivava-se um ambiente diferente, mais transparente, no qual os usuários possam, se desejarem, verificar o que está ocorrendo nas máquinas e no que elas estão trabalhando. Dessa forma, surge a questão: quais processos devem ser submetidos por meio do gestor de filas? A resposta básica é: todos que envolvam intenso processamento de dados, pois apesar de poderem ser iniciados sem o uso do SGE, os processos que utilizam CPU intensamente serão interrompidos pelo sistema e só terão o seu trabalho totalmente executado se iniciados via o sistema de filas. A classificação de um processo como tendo uso intenso de Central Processing Unit (CPU) é algo subjetivo e exige o estabelecimentos de regras claras por parte do comite

gestor do LMB. Atualmente, em função de uma série de testes efetuados no ambiente, estabeleceu-se um limite de 30 minutos de uso de CPU para um processo ser considerado intensivo em CPU.

Por exemplo, não se deve passar a usar o comando *ls* com o auxílio do *qsub* (o comando que permite a submissão de *jobs* para o SGE), visto que nesse caso, precisamos de uma listagem dos arquivos que temos interesse imediatamente após se digitar o comando, e o *qsub* enfileiraria nosso trabalho para ser executado posteriormente. Outro ponto é que um ls quase não consome recursos da máquina que o executa e não faria sentido ter um programa com tais características aguardando na fila para gerar seus resultados. De forma geral, os comandos de sistema (tais como *ls, cd, mkdir, mv, top, vi*, etc...) deverão ser executados diretamente sem o auxílio do *qsub*.

Comandos tais como *cp, rsync, tar* podem ou não utilizar o *qsub*. Isto depende da quantidade de trabalho que estes comandos realizarão para o usuário. Por exemplo, se o usuário deseja copiar um *script* para realizar alterações neste e ainda ter uma fonte de backup, caso precise retornar ao conteúdo original; é evidente que esse procedimento de cópia não deverá ser efetuado com o *qsub*, pois é necessário ele ocorra imediatamente e a sobrecarga sobre as máquinas é desprezível. Por outro lado, se o usuário deseja utilizar o aplicativo tar para compactar um enorme volume de dados, poderia submeter esta tarefa por meio do gestor de filas; pois o usuário não precisa que isso entre em execução imediatamente e, apesar da sobrecarga na capacidade de processamento das máquinas ser mínima, existe para esse exemplo uma grande demanda para o sistema de IO.

Por outro lado, todos os comandos que envolvam processamento intenso de dados, tais como *scripts* desenvolvidos pelo usuário e todos os programas de bioinformática, deverão ser submetidos com o auxílio do *qsub*. Suas tarefas serão enfileiradas e executadas assim que houver recursos disponíveis. A priorização das tarefas é controlada diretamente pelo gestor de filas e o estabelecimento da política de prioridades é realizado pelos gestores do LMB.

Com relação àqueles casos em que o usuário ainda está avaliando quais as melhores estratégias para serem utilizadas em sua análise, por exemplo, testando parâmetros de ferramentas; então, não é interessante utilizar

o *qsub* nesses casos, dado que é necessário uma maior interatividade com o sistema para se chegar rapidamente a um conjunto de parâmetros adequados. No entanto, deve-se realizar todas as atividades de testes com conjuntos de dados pequenos. O motivo disso é o seguinte: o *cluster* está configurado para cancelar automaticamente trabalhos que ultrapassem o uso de 30 minutos de CPU que tenham sido executados diretamente, sem o uso do SGE. Nesse caso, o sistema considera que esta é uma tarefa intensiva em CPU e deveria ser executada via gestor de filas.

2. Utilizando o SGE

Com um sistema gestor de filas, e com o SGE não é diferente, não mais se entra com comandos e obtêm-se processos executando-os imediatamente. Ao contrário, submetem-se *jobs* (um conjunto de instruções que possibilitam ao SGE inicializar um processo corretamente) que serão enfileirados e disparados, ou seja, executados, quando houver disponibilidade nas máquinas. Isso ocorre de forma transparente para os usuários, e dessa forma, tudo que se precisa é conhecer os mecanismos básicos para se submeter os *jobs* e verificar seus status quando for necessário. Abaixo aparecem alguns dos principais comandos utilizados para se trabalhar com o SGE e, posteriormente, há um conjunto de exemplos que auxiliarão no entendimento global do sistema.

```
qsub – permite a submissão de um job
qstat – lista a fila de jobs
qalter – permite alterar propriedades de jobs que estão aguardando na fila
qdel – permite cancelar a execução de um determinado job
```

De todos esses, o *qsub* será o mais utilizado, por isso segue uma discussão mais detalhada sobre esse comando. Submeter um job com o *qsub* pode ser feito de duas formas diferentes. No primeiro caso, gera-se um *script* de configuração para o job e passa-se esse *script* como um parâmetro para o *qsub*. Esse método é indicado para a submissão de grandes *jobs*, que exijam configurações muito específicas. No geral, deseja-se executar um programa que atuará sob arquivos de entradas e produzirá

arquivos de saída. Nesse caso, é desejável uma forma mais rápida de se executar uma submissão, e por isso, este trabalho enfatiza esse segundo método, por se considerar que ele é mais útil.

Neste segundo método, basta utilizar-se o comando *qsub* <parâmetros> <enter>, então digita-se o comando a ser executado com todos os seus parâmetros, como se o estivéssemos executando a partir do *shell*. Ao final, utiliza-se crtl-D para terminar a edição e pronto, tem-se um job na fila. Uma forma equivalente e mais prática, principalmente quando se deseja submeter vários *jobs* ao mesmo tempo, é utilizar a linha de comando:

```
echo 'comando a ser submetido para a fila' | qsub <parâmetros> <enter>
```

O exemplo da seção 2.1 esclarece o uso do *qsub*, antes no entanto, considere uma questão muito importante relacionada com o diretório de trabalho dos *jobs* inicializados pelo SGE e os arquivos de saída e erro produzidos durante as execuções. Por padrão, todos os processos inicializados pelo SGE terão como diretório de trabalho o *home* do usuário que disparou os *jobs*. Se submetidos assim, todos os *jobs* deverão operar sobre arquivos com caminhos absolutos ou caminhos relativos a partir do diretório *home*, e ao final, eles produzirão dois arquivos no *home* do usuário, além, é lógico, dos arquivos de saída específicos do(s) programa(s) executados no job. Dos dois arquivos gerados pelo SGE, um terá a extensão .eID_JOB e conterá informações sobre erros ocorridos durante a execução do job; o outro arquivo terá a extensão .oID_JOB e conterá dados eventualmente direcionados para a saída padrão do(s) processo(s) executado pelo job. Em ambos os casos ID_JOB refere-se a um número de identificação único atribuído pelo SGE a cada job submetido ao sistema.

Caso os *jobs* sejam submetidos utilizando o *qsub* com o parâmetro -cwd, então os processos inicializados pelo SGE terão como diretório de trabalho o diretório do qual os *jobs* foram disparados pelo usuário. Se submetidos assim, todos os *jobs* deverão operar sobre arquivos com caminhos absolutos ou caminhos relativos ao diretório atual, e ao final, eles produzirão os dois arquivos mencionados anteriormente, não no *home* do usuário, mas no diretório do qual foi submetido o job.

Feito tal esclarecimento, pode-se apresentar alguns exemplos que ajudarão no entendimento do *qsub*.

2.1. Exemplo básico: Disparando o ls como um job pelo qsub

O comando *ls* é excelente para a ilustração do funcionamento do sistema gestor de filas, mas evidentemente, não se usa o *qsub* para rodar o ls no dia a dia, conforme a discussão na introdução da sessão 2 deste documento.

Considere um usuário chamado teste trabalhando no diretório /tmp. Dessa forma, para a submissão do ls com o diretório de trabalho baseado no *home* do usuário (/*home*/teste), pode-se fazer:

```
cd /tmp  #apenas para ficar claro que o job tem um diretório de
trabalho diferente do diretório de submissão
echo 'ls' | qsub
Your job 1824265 ("STDIN") has been submitted
```

Se executarmos o comando qstat, imediatamente após a submissão, poderemos ver o job na fila conforme ilustrado na Figura 1. Naquelas situações em que o *cluster* estiver com baixa carga de trabalho, então o job entrará em execução alguns segundos após a submissão. Naqueles momentos em que o *cluster* estiver sobrecarregado, o job submetido permanecerá na fila, sem entrar em execução, por um longo tempo, a depender das prioridades estabelecidas no sistema.

```
#Lista da fila de jobs do usuário teste
qstat
```

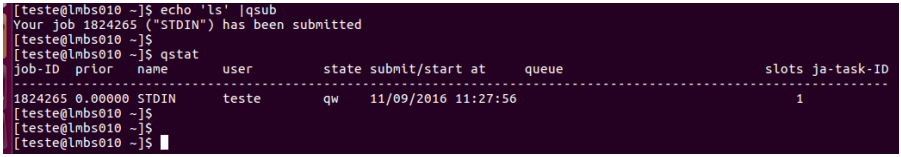


Figura 1. Ilustração da fila do usuário teste logo após a submissão do comando ls via *qsub*.

Evidentemente, no momento que o comando realmente for executado, ele deverá produzir seu resultado esperado que consiste da lista de arquivos do diretório de trabalho. No entanto, deve-se levar em consideração que para o processo que executará o *ls*, o diretório de trabalho será o /*home*/teste (o *home* do usuário teste) uma vez que se executou o comando *qsub* sem o parâmetro -cwd. Portanto, a saída padrão do *ls* será redirecionada para o arquivo /*home*/teste/STDIN.oJOB_ID (onde JOB_ID é o ID do job gerado durante a submissão da tarefa pelo *qsub*). Dessa forma, uma vez executado, deve-se encontrar no diretório /*home*/teste o arquivo /*home*/teste/STDIN.oJOB_ID com a lista de arquivos do *home* do usuário teste. Haverá também um arquivo nomeado /*home*/teste/STDIN.eJOB_ID que contém informações sobre possíveis erros ocorridos no processo, que deverá estar vazio para todos os casos em que a execução for bem-sucedida. A Figura 2 mostra o *home* do usuário teste após a execução do job e o conteúdo do arquivo /*home*/teste/STDIN.oJOB_ID

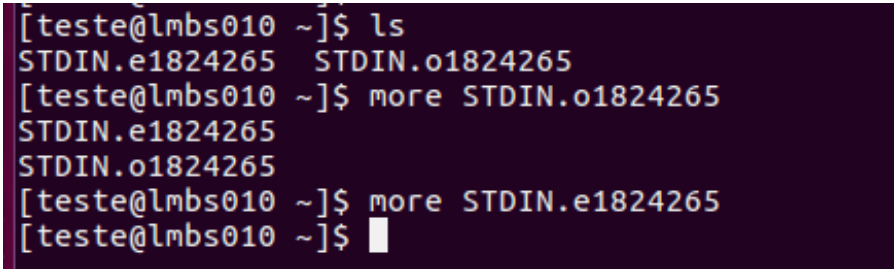


Figura 2. Ilustração do *home* do usuário teste logo após a execução do job com ID 1824265.

Nota-se que o arquivo STDIN.o1824265 contém a saída do comando *ls* e o arquivo de erro está vazio. Como o usuário teste foi criado apenas para os exemplos, sua *home* não tinha outros arquivos senão os dois criados pelo SGE

Agora, considere o parâmetro `-cwd` do *qsub*:

```
#o parametro -cwd no qsub garante que o job terá o diretório do qual
se realizou a submissão como o diretório de trabalho
cd /tmp
echo 'ls' | qsub  -cwd
```

Nesse caso, o diretório de trabalho para o processo ls que será criado futuramente pelo SGE será o diretório de submissão (o /tmp nesse caso). Assim, o comando ls produzirá uma listagem dos arquivos e diretórios do /tmp. Também, os dois arquivos criados pelo SGE durante a execução do job estarão no diretório /tmp ao final da execução.

Até o presente, nos exemplos mostrados, o ls não recebeu parâmetros. No entanto, isso poderia ser realizado simplesmente colocando os parâmetros desejados na string de caracteres que o aplicativo echo está transferindo para o *qsub*.

Também é aconselhável que a saída de um job seja depositada em um arquivo especialmente criado para isso e não no arquivo .oJOB_ID providenciado pelo SGE. No caso do ls pode-se fazer isso com um redirecionamento. No caso dos programas de bioinformática, a maioria tem um parâmetro indicando um arquivo de saída.

```
cd /tmp
echo ' ls -l > listagem.txt' | qsub -cwd
```

O comando acima produzirá o arquivo chamado /tmp/listagem.txt com a listagem longa de todos os arquivos e diretórios de /tmp. Isso ocorre porque utilizou-se o parâmetro `-cwd` para o *qsub* tendo-se o diretório /tmp como diretório de trabalho.

Não é interessante que se controlem os *jobs* do SGE usando diretamente o id destes. Isso é muito útil para o gestor de filas, mas para os humanos é mais natural trabalhar com nomes. Dessa forma, no momento da submissão pode-se atribuir um nome ao job. Tal nome aparecerá na fila e também será utilizado para a criação dos dois arquivos de saída do SGE. Para tanto, basta ser utilizado o parâmetro -N conforme exemplificado abaixo.

```
#Nomeando o job que irá executar o comando ls como meuLS. Esse nome
poderia ser outro qualquer.
echo 'ls -l > /tmp/listagem.txt' | qsub -cwd -N meuLS
```

Assim que a submissão terminar via o comando qstat, o job meuLS poderá ser visto na fila se ele ainda não tiver terminado sua execução. Ao final da execução o arquivo /tmp/listagem.txt terá uma listagem longa do diretório atual. Por outro lado, no diretório atual, os arquivos meuLS.eJOB_ID vazio e meuLS.oJOB_ID poderão ser encontrados com uma mensagem irrelevante.

Por fim, existem várias situações em que não se deseja que o SGE produza os arquivos de saída .oJOB_ID e .eJOB_ID. Para evitar isso, deve-se redirecionar o local onde estes dois arquivos são armazenados no sistema, e enviá-los para o dispositivo null, ou seja, descartá-los.

```
cd /tmp
echo 'ls -l > /home/teste/listagem.txt' | qsub -cwd -N meuls -o /dev/
null -e /dev/null
```

2.2. Exemplo para gerenciamento de memória

Este exemplo é sobre a submissão de um job que exija mais de 50G de memória para executar. Nesse caso é importante que essa necessidade seja informada ao SGE para que o sistema providencie uma máquina que tenha memória disponível suficiente para a execução do job. Do contrário, corre-se o risco de se ter o trabalho iniciado em uma máquina com a memória já sobrecarregada e, posteriormente, este deverá ser cancelado pelo sistema por falta de memória para continuar a execução. Na hipótese disso vir a ocorrer, o usuário obterá um erro e será obrigado e reiniciar o trabalho posteriormente. Considere a submissão de um job para a montagem genômica do genoma do Panda com reads de nova geração. Esse procedimento, em sua fase inicial, consome de 400G a 450G de memória RAM. Dessa forma, deve-se utilizar o parâmetro -l mf=450G no *qsub* para garantir que o job seja executado em uma máquina que tenha, no mínimo, 450G de memória livre. Veja o comando de submissão abaixo:

```
echo  SOAPdenovo31mer all -s panda.cfg -K 31 -p 48 -o montagemPanda.
out | qsub -cwd -l mf=450G
```

Evidentemente, todos os parâmetros do *qsub* discutidos anteriormente também se aplicam ao caso apresentado, e só foram omitidos para uma melhor clareza da linha de comando.

A política de uso do SGE adotada pelo LMB exige que toda submissão de *jobs* que consumirão grande volume de memória seja realizada utilizando-se o parâmetro `-l mf` (pode-se utilizar também -l mem_free) para que o sistema de gerenciamento de filas seja subsidiado com informações que possibilitem manter a estabilidade das máquinas.

2.3. Exemplo para envio de emails

No *qsub* existem os parâmetros `-m` e `-M` que controlam o envio de mensagens sobre o status dos *jobs* submetidos no gestor de filas. O parâmetro `-m` controla quando uma mensagem será enviada e o parâmetro `-M` controla para qual e-mail estas mensagens serão enviadas. O parâmetro `-m` pode ser modificado com os seguintes valores, incluindo composições com mais de uma opção:

‘b’ Um email é enviado no ínicio da execução do job

‘e’ Um email é enviado quando o job termina

‘a’ Um email é enviado quando um job é abortado e reinfileirado

‘s’ Um email é enviado quando um job é suspenso

‘n’ Nenhum email é enviado

Como exemplo, considere a submissão abaixo:

```
echo “ls” | qsub -cwd -m base -M leandro.cintra@embrapa.br
```

Nesse caso, o sistema enviará um email para todas as fases do job. Isso porém, pode levar a um resultado indesejado: caso uma máquina fique altamente sobrecarregada, o SGE pode temporariamente suspender um ou mais *jobs* rodando na máquina, fazendo com que um número excessivo de e-mails seja enviado para os casos em que o usuário utilize a opção s no parâmetro `-m`. Portanto, uma opção mais moderada seria enviar um e-mail no início e no fim da execução, com a configuração abaixo:

```
echo "ls" | qsub -cwd -m be -M leandro.cintra@embrapa.br
```

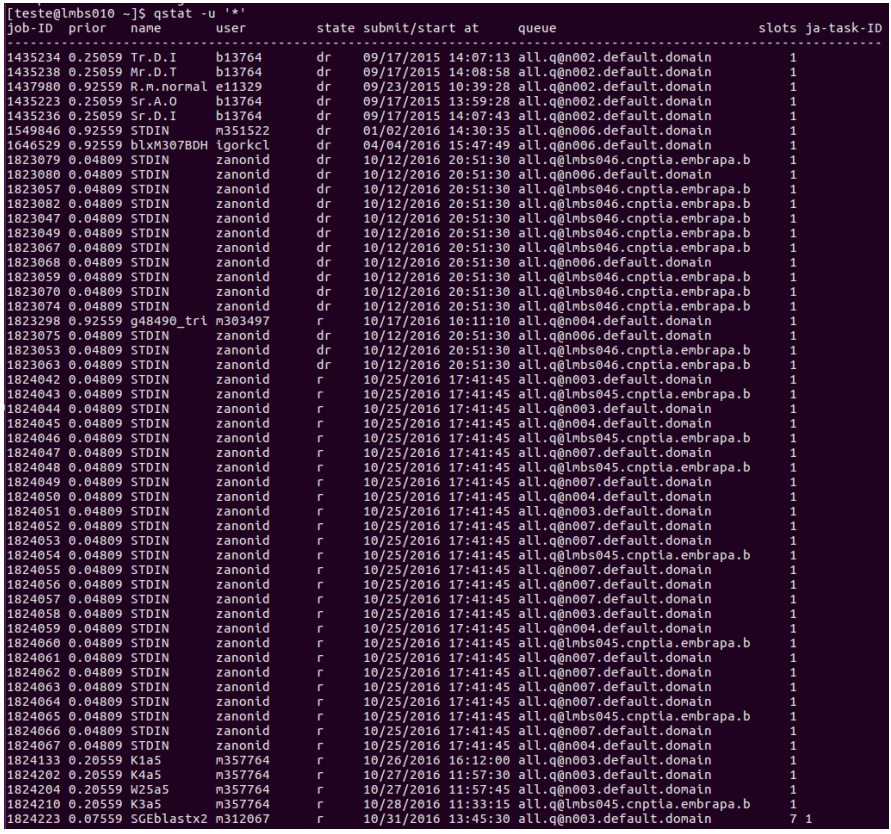
3. Verificando o status e controlando *jobs* submetidos

Para exibir uma lista dos seus *jobs* que estejam em execução ou aguardando para execução, o usuário deverá utilizar o comando qstat. Para exibir uma lista de todos os *jobs* de todos os usuários, que estejam executando ou aguardando para execução, o usuário deverá utilizar o comando `qstat -u '*'`. Atenção para as aspas simples envolvendo o asterisco. Elas são vitais para que o qstat produza o resultado desejado. A listagem produzida pelo qstat traz algumas informações importantes para a identificação da situação em que se encontra um job na fila. Para uma ilustração de tais informações, considere a Figura 3 que apresenta a listagem produzida para o comando:

```
qstat -u '*'
```

Por meio da referida figura pode-se identificar as colunas job-ID, prior, name, user, state, submit/start at, queue e slots. A coluna job-ID traz um número único que identifica cada job no sistema; prior informa a prioridade de um job e tem implicações apenas para o SGE decidir qual será o próximo job da fila a ter a sua execução iniciada; name apresenta o nome atribuído pelo usuário a um job; user informa o nome do usuário que submeteu o job; state apresenta o status atual do job; submit/start at informa a data de submissão ou de início de execução do job; queue informa a fila utilizada para aqueles *jobs* que já estejam em execução e por fim, slots

indica a quantidade de slots que um job está utilizando no sistema. Dentre todas estas informações uma coluna de particular interesse é a coluna de status. Seu conteúdo é baseado em alguns códigos pré-estabelecidos pelo SGE que indicam a situação atual de um job. Dentre os principais símbolos destacam-se o `qw` (queued) indicando que o job mantem-se em espera na fila aguardando sua vez de executar; o `r` (running) indicando que o job está em execução pelo sistema; o `d` (deleting) indicando que o job está em processo de exclusão em virtude de uma solicitação do usuário e o E (Error) indicando ter havido algum erro no momento que o SGE elegeu o job para execução.



```
[teste@lmbs010 ~]$ qstat -u '*'
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
1435234 0.25059 Tr.D.I     b13764       dr    09/17/2015 14:07:13 all.q@n002.default.domain          1
1435238 0.25059 Mr.D.T     b13764       dr    09/17/2015 14:08:58 all.q@n002.default.domain          1
1437980 0.92559 R.m.normal e11329       dr    09/23/2015 10:39:28 all.q@n002.default.domain          1
1435223 0.25059 Sr.A.O     b13764       dr    09/17/2015 13:59:28 all.q@n002.default.domain          1
1435236 0.25059 Sr.D.I     b13764       dr    09/17/2015 14:07:43 all.q@n002.default.domain          1
1549846 0.92559 STDIN      m351522      dr    01/02/2016 14:30:35 all.q@n006.default.domain          1
1646529 0.92559 blxM307BDH igorkcl      dr    04/04/2016 15:47:49 all.q@n006.default.domain          1
1823079 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823080 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@n006.default.domain          1
1823057 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823082 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823047 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823049 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823067 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823068 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@n006.default.domain          1
1823059 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823070 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823074 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823298 0.92559 g48490_tri m303497      r     10/17/2016 10:11:10 all.q@n004.default.domain          1
1823075 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@n006.default.domain          1
1823053 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1823063 0.04809 STDIN      zanonid      dr    10/12/2016 20:51:30 all.q@lmbs046.cnptia.embrapa.b     1
1824042 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n003.default.domain          1
1824043 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@lmbs045.cnptia.embrapa.b     1
1824044 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n003.default.domain          1
1824045 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n004.default.domain          1
1824046 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@lmbs045.cnptia.embrapa.b     1
1824047 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824048 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@lmbs045.cnptia.embrapa.b     1
1824049 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824050 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n004.default.domain          1
1824051 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n003.default.domain          1
1824052 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824053 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824054 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@lmbs045.cnptia.embrapa.b     1
1824055 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824056 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824057 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824058 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n003.default.domain          1
1824059 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n004.default.domain          1
1824060 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@lmbs045.cnptia.embrapa.b     1
1824061 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824062 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824063 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824064 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824065 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@lmbs045.cnptia.embrapa.b     1
1824066 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n007.default.domain          1
1824067 0.04809 STDIN      zanonid      r     10/25/2016 17:41:45 all.q@n004.default.domain          1
1824133 0.20559 K1a5       m357764      r     10/26/2016 16:12:00 all.q@n003.default.domain          1
1824202 0.20559 K4a5       m357764      r     10/27/2016 11:57:30 all.q@n003.default.domain          1
1824204 0.20559 W25a5      m357764      r     10/27/2016 11:57:45 all.q@n003.default.domain          1
1824210 0.20559 K3a5       m357764      r     10/28/2016 11:33:15 all.q@lmbs045.cnptia.embrapa.b     1
1824223 0.07559 SGEblastx2 m312067      r     10/31/2016 13:45:30 all.q@n003.default.domain          7 1
```

Figura 3. Listagem parcial da saída do comando qconf -u '*' mostrando os *jobs* de alguns usuários.

Pode-se identicar várias colunas de informação nesta listagem, sendo que neste trabalho enfatiza-se a coluna state que fornece o status do job na fila.

Para obter maiores informações sobre um job, tais como erros que possam ter ocorrido e que impediram que este executasse, utilize o comando qstat -j JOB_ID. O ID de um job poderá ser obtido a partir da listagem do comando qstat sem parâmetros. Nessa listagem, a primeira coluna é o *job-id* que traz um número único atribuído pelo SGE ao job, como discutido anteriormente.

Como colocado, o comando `qstat -j` JOB_ID pode fornecer informações sobre erros que tenham ocorrido com *jobs*. Essa afirmação está correta, no entanto, é necessário que se faça uma distinção entre erros nos *jobs* e erros nos programas a serem executados pelos *jobs*; pois o comportamento entre essas duas categorias no sistema é distinto.

Um erro no job ocorre antes do início da execução dos seus programas. Um exemplo seria o fato de o usuário que fez a submissão não existir em uma determinada máquina do *cluster*. Se o SGE selecionar essa máquina para executar o job, ele irá detectar que o usuário em questão não existe nela e irá marcar o job com um flag de erro, sendo que o mesmo ficará indefinidamente na fila, até que o administrador do *cluster* ou o usuário que fez a submissão intervenham. Para esses *jobs* que permanecem na fila com o flag de erro, o comando `qstat -j` JOB_ID é muito útil para fornecer informações sobre as causas do problema.

Porém, existem situações em que o erro ocorre após o início da execução dos programas, por exemplo, quando se informa para um programa um arquivo de entrada que não existe. Nesse caso, o programa irá terminar com um erro, mas do ponto de vista do SGE, ele fez o que tinha de fazer, ou seja, ele executou o trabalho. Logo, o job termina com sucesso e deixa a fila do sistema. Para se obter informações sobre *jobs* que já tenham terminado sua execução, deve-se utilizar o comando qacct, tal qual discutido na sessão 4.

O comando qdel JOB_ID permite cancelar um job que esteja em execução ou esteja esperando para entrar em execução.

O comando qalter JOB_ID é utilizado para alterar configurações de *jobs*

que estejam aguardando execução. Esta é uma característica relevante desse comando: ele atua apenas sobre *jobs* aguardando execução e não tem efeito sobre *jobs* já em execução.

4. Verificando informações de jobs já executados

O SGE mantém um registro com detalhes de todos os *jobs* que foram executados no sistema, no entanto, há um complicador para se acessar essas informações, pois para tanto é necessário se conhecer o JOB_ID que foi atribuído a um job no momento de sua criação. Essa, definitivamente, é uma informação que na maioria dos casos não se tem; a não ser que o usuário armazene o(s) JOB_ID(s) de alguma forma após a submissão, já prevendo essa busca a posteriori.

A seguir, tem-se comando que permite recuperar as informações de um job já executado.

```
qacct -j JOB_ID
```

Como geralmente os JOB_IDs não estão disponíveis, então, o que se faz rotineiramente é realizar busca considerando alguns parâmetros de restrição, sendo que os três principais parâmetros do qacct com essa finalidade são discutidos abaixo.

A primeira abordagem é o usuário restringir a busca apenas aos seus *jobs*. O exemplo a seguir demonstrada tal situação para o usuário m338700.

```
qacct -u m338700  -j
```

Nesse caso, todos os *jobs* executados pelo usuário m338700 desde que inicou seus trabalhos no *cluster* serão listados. É vital que o parâmetro `-j` seja utilizado, pois, do contrário, o comando irá apenas produzir um sumário para o uso de recursos computacionais por parte do usuário. É o parâmetro -j que garante a exibição de detalhes sobre os *jobs*.

No entanto, como a versão anterior do comando qacct lista todos os *jobs* do usuário, pode ainda, ser difícil localizar aquele de interesse, a depender da quantidade de trabalhos que o usuário já tiver terminado no *cluster*. Por isso, se um usuário tiver uma vaga noção da data de início da execução do job, ele ainda poderá fazer uma restrição que garanta a listagens dos *jobs* iniciados nos últimos dias:

```
#neste caso o qacct irá produzir uma listagem dos jobs iniciados nos últimos 7 dias
qacct -u m338700 -d 7
```

Ou, se o usuário desejar restringir a inicialização dos *jobs* a um certo período no tempo:

```
#neste caso o qacct irá localizar os jobs iniciados entre os instantes dados por -b e -e
qacct -u m338700 -b 1608010000.00 -e 1608312359.59 -j
```

A sintaxe acima garante que serão listados todos os *jobs* iniciados no mês de agosto de 2016. Os números passados como argumentos para os parâmetros `-b` e `-e` têm a seguinte conotação: YYMMDDHHMM.SS. Além disso, o parâmetro `-b` indica uma seleção de todos os *jobs* que iniciaram sua execução após o período informado, e o parâmetro `-e`, implica em seleção antes do período informado. Portanto, no exemplo acima está se estipulando a seleção dos *jobs* iniciados em agosto de 2016.

5. Conclusões

O sistema SGE é muito apropriado para um ambiente semelhante ao do LMB e cumpre uma função primordial neste. Evidentemente, o sistema também poderá ser utilizado em outros ambientes computacionais no âmbito da Embrapa, desde que atendidas algumas premissas básicas: como demonstrado ao longo do texto, o SGE opera em ambientes cuja interface com o usuário se baseia no uso de *shell* (uso de comandos) e não em

ambientes gráficos. Isso ocorre porque o sistema gerencia atividades que podem ser executadas em batch (não precisam iniciar imediatamente e não interajam com o usuário durante a execução). Sistemas com interface gráfica, na maioria dos casos, é interativo e exigem intervenção do usuário constantemente.

No caso específico do LMB, as análises em bioinformática executadas têm exatamente estas características. São sistemas que obtém algum(ns) arquivo(s) de dados como entrada e geram um ou mais arquivos de saída, sem exigir novas intervenções ao longo da execução. Dessa forma, atendem plenamente aos requisitos para serem executados em batch, fazendo com que o SGE seja um sistema de grande utilidade para o laboratório.

Além do mais, o SGE também cumpre um papel essencial quando organiza e prioriza a ordem de execução das tarefas. Como o LMB tem um número razoável de usuários, é importante que as tarefas sejam automaticamente escalonadas de acordo com as prioridades estabelecidas no regimento do laboratório e pelo responsável técnico.

De negativo com relação ao SGE tem-se a questão do treinamento dos usuários. No primeiro contato dos usuários com o ambiente, é necessário que ele se familiarize com a forma de operar a submissão de tarefas para o *cluster*. Isso, certamente, exige um certo esforço por parte do usuário, mas pode-se afirmar que esse é recompensado pelos diversos benefícios que o sistema traz. Para a equipe de suporte do LMB o sistema simplifica o gerenciamento do *cluster* e possibilita a obteção de informações de uso do ambiente. Para o usuário, o sistema facilita a submissão de diversas tarefas em paralelo no *cluster* e com isso, permite um ganho de tempo considerável na realização das análises.

Com todas as colocações feitas, torna-se claro que o sistema SGE foi uma solução adequada para as demandas de gerenciamento do processamento de dados no âmbito do Laboratório Multiusuário de Bioinformática da Embrapa.

6. Referências

HEY, T.; TANSLEY, S.; TOLLE, K. **O quarto paradigma**: descobertas científicas na era da eSience, São Paulo: Oficina de Textos, 2011. 263 p.

SUN MICROSYSTEMS. **N1 grid engine 6 user's guide**. Santa Clara, 2005. 180 p.

Embrapa

Informática Agropecuária

MINISTÉRIO DA
AGRICULTURA, PECUÁRIA
E ABASTECIMENTO

BRASIL
GOVERNO FEDERAL

CGPE 13548