

Projeto Administrador de Cluster

Grupo: Victor Assis, Andre Correa, Rafael Lima

O foco do nosso grupo foi tentar deixar o cluster mais justo entre diferentes usuários, e foi pensado especialmente para o contexto de treinamento de redes neurais. Para isso, tentamos duas abordagens diferentes:

- Definir duas partições com diferentes comportamentos e níveis de prioridade (bem sucedido);
- Penalizar usuários com muita frequência de uso em detrimento dos usuários com pouca frequência de uso (mal sucedido).

Definindo as partições

O processo de treinamento de redes neurais é dividido em várias epochs. É possível ter uma idéia se a rede neural terá uma boa performance analisando as acurácias de validação calculadas ao fim de cada epoch. Por exemplo, é possível perceber caso o treinamento do modelo esteja overfittando, se está demorando mais do que o desejado para convergir ou ter uma noção se sequer vai convergir.

Caso duas pessoas submetam jobs de treinamento de redes neurais que ocupem completamente o cluster, a segunda que enviasse apenas teria um feedback quando o da primeira terminasse de rodar. Além disso, por o processo de treinamento poder demorar muitas horas, ela ainda precisaria esperar um tempo do próprio treinamento de seu modelo para ai sim decidir se interrompe o treinamento ou se permite ir até o fim. Se essas duas pessoas submetem o job no mesmo minuto, por uma diferença de segundos a segunda pessoa levaria muitas horas para ter informações sobre o treinamento.

Pensando nesse caso, criamos a partition Fair. Ela é capaz de suspender a execução de um job a cada 30 segundos e dar continuidade a execução de algum job que estava suspenso previamente. Especificamente no caso do nosso cluster, seria possível submeter até 4 vezes mais jobs ao mesmo tempo, sendo que a todo momento $\frac{1}{4}$ estaria rodando e $\frac{3}{4}$ estariam suspensos.

```
Activities Terminal mai 16 15:48 victor@sms-host:~
victor@sms-host:~
JOBID PARTITION NAME USER ST TIME NODES Nodelist(REASON)
177 fair job_taga victor R 0:50 2 compute[00-01]
178 fair job_taga victor S 0:30 2 compute[00-01]
179 fair job_taga victor S 0:30 2 compute[00-01]
180 fair job_taga victor S 0:30 2 compute[00-01]
[victor@sms-host ~]$ squeue
JOBID PARTITION NAME USER ST TIME NODES Nodelist(REASON)
177 fair job_taga victor R 1:00 2 compute[00-01]
178 fair job_taga victor S 0:30 2 compute[00-01]
179 fair job_taga victor S 0:30 2 compute[00-01]
180 fair job_taga victor S 0:30 2 compute[00-01]
[victor@sms-host ~]$ squeue
JOBID PARTITION NAME USER ST TIME NODES Nodelist(REASON)
177 fair job_taga victor R 1:01 2 compute[00-01]
178 fair job_taga victor S 0:30 2 compute[00-01]
179 fair job_taga victor S 0:30 2 compute[00-01]
180 fair job_taga victor S 0:30 2 compute[00-01]
[victor@sms-host ~]$ squeue
JOBID PARTITION NAME USER ST TIME NODES Nodelist(REASON)
178 fair job_taga victor R 0:31 2 compute[00-01]
177 fair job_taga victor S 1:01 2 compute[00-01]
179 fair job_taga victor S 0:30 2 compute[00-01]
180 fair job_taga victor S 0:30 2 compute[00-01]
[victor@sms-host ~]$ squeue
JOBID PARTITION NAME USER ST TIME NODES Nodelist(REASON)
178 fair job_taga victor R 0:32 2 compute[00-01]
177 fair job_taga victor S 1:01 2 compute[00-01]
179 fair job_taga victor S 0:30 2 compute[00-01]
180 fair job_taga victor S 0:30 2 compute[00-01]
[victor@sms-host ~]$ squeue
JOBID PARTITION NAME USER ST TIME NODES Nodelist(REASON)
178 fair job_taga victor R 0:38 2 compute[00-01]
177 fair job_taga victor S 1:01 2 compute[00-01]
179 fair job_taga victor S 0:30 2 compute[00-01]
180 fair job_taga victor S 0:30 2 compute[00-01]
[victor@sms-host ~]$
```

No print acima é possível observar 4 jobs do Victor rodando na partição Fair, cada job está ocupando completamente os recursos disponíveis. Na coluna ST (State) é possível observar que 1 job está R (Running) e outros 3 estão S (Suspended). Podemos acompanhar qual job está rodando através da coluna JOBID. Na quarta vez que o comando squeue foi rodado, observamos que o job 177 foi suspenso, dando lugar para o job 178.

O efeito colateral dessa estratégia é que o tempo total de execução de todos os jobs poderia ser até 4 vezes maior. Justamente pensando em contornar esse efeito colateral, criamos a partição Urgent. Jobs submetidos na partição Urgent possuem maior prioridade. Vamos supor que em uma fila de jobs temos 4 da partição Fair e 1 da Urgent, que foi submetido por último. Assim que houvesse recurso disponível, o job da partição Urgent furaria a fila e ocuparia o recurso, sem a possibilidade de ser suspenso mesmo que houvesse outro job Fair rodando.

```
Activities Terminal mai 16 15:44 victor@sms-host:~
victor@sms-host:~
Submitted batch job 183
[victor@sms-host ~]$ sbatch oi.slurm
Submitted batch job 184
[victor@sms-host ~]$ squeue
      JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
      181      urgent job_taga victor  PD       0:00      2 (Resources)
      182      urgent job_taga victor  PD       0:00      2 (Priority)
      183      urgent job_taga victor  PD       0:00      2 (Priority)
      184      urgent job_taga victor  PD       0:00      2 (Priority)
      177      fair job_taga victor  PD       0:00      2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p
riority partitions)
      178      fair job_taga victor  PD       0:00      2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p
riority partitions)
      179      fair job_taga victor  PD       0:00      2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p
riority partitions)
      180      fair job_taga victor  PD       0:00      2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p
riority partitions)
      167      fair job_rafa rafael  R       0:58      2 compute[00-01]
      168      fair job_rafa rafael  S       0:30      2 compute[00-01]
      169      fair job_rafa rafael  S       0:30      2 compute[00-01]
      170      fair job_rafa rafael  S       0:30      2 compute[00-01]
[victor@sms-host ~]$ squeue
      JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
      182      urgent job_taga victor  PD       0:00      2 (Resources)
      183      urgent job_taga victor  PD       0:00      2 (Priority)
      184      urgent job_taga victor  PD       0:00      2 (Priority)
      177      fair job_taga victor  PD       0:00      2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p
riority partitions)
      178      fair job_taga victor  PD       0:00      2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p
riority partitions)
      179      fair job_taga victor  PD       0:00      2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p
riority partitions)
      180      fair job_taga victor  PD       0:00      2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p
riority partitions)
      181      urgent job_taga victor  R       0:05      2 compute[00-01]
```

No print acima é possível observar 4 jobs do Rafael rodando na partição Fair, cada job está ocupando completamente os recursos disponíveis. Na coluna ST (State) é possível observar que 1 job está R (Running) e outros 3 estão S (Suspended). Na fila é possível observar alguns jobs do Victor, tanto da partição Fair quanto da partição Urgent. Decidimos colocar jobs de apenas 1 usuário na fila para que diferentes níveis de prioridade entre os usuários não afetasse esse experimento.

De outra máquina, o Rafael cancelou os seus jobs rodando. Ao rodar o squeue novamente, observamos que um job 181 da partição Urgent furou a fila, passando na frente dos jobs 177 a 180, da partição Fair. Também notamos que os outros jobs Urgent continuam pending, mostrando que a partição não permite suspensão de jobs enquanto rodam.

A ideia original seria um job Urgent ser capaz inclusive de suspender um job Fair para poder tomar conta de seu recurso. Pela documentação, o slurm.conf está configurado para que esse comportamento ocorra, porém infelizmente não conseguimos reproduzir esse comportamento. Provavelmente deixamos passar algum parametro de configuração, mas acreditamos que o comportamento gerado ainda assim possui valor e melhora a experiência de uso para os usuários.

Um comportamento que foi possível reproduzir é um job Urgent ser capaz de cancelar completamente um job Fair - não tiramos print, mas o Demay estava literalmente do lado quando aconteceu e inclusive foi ele que conseguiu interpretar o resultado, fiquei confuso - porém acreditamos que seria uma experiência ruim de uso ter um job cancelado por um job de outro usuário.

Para implementar a partição Fair, utilizamos as seguintes configurações:

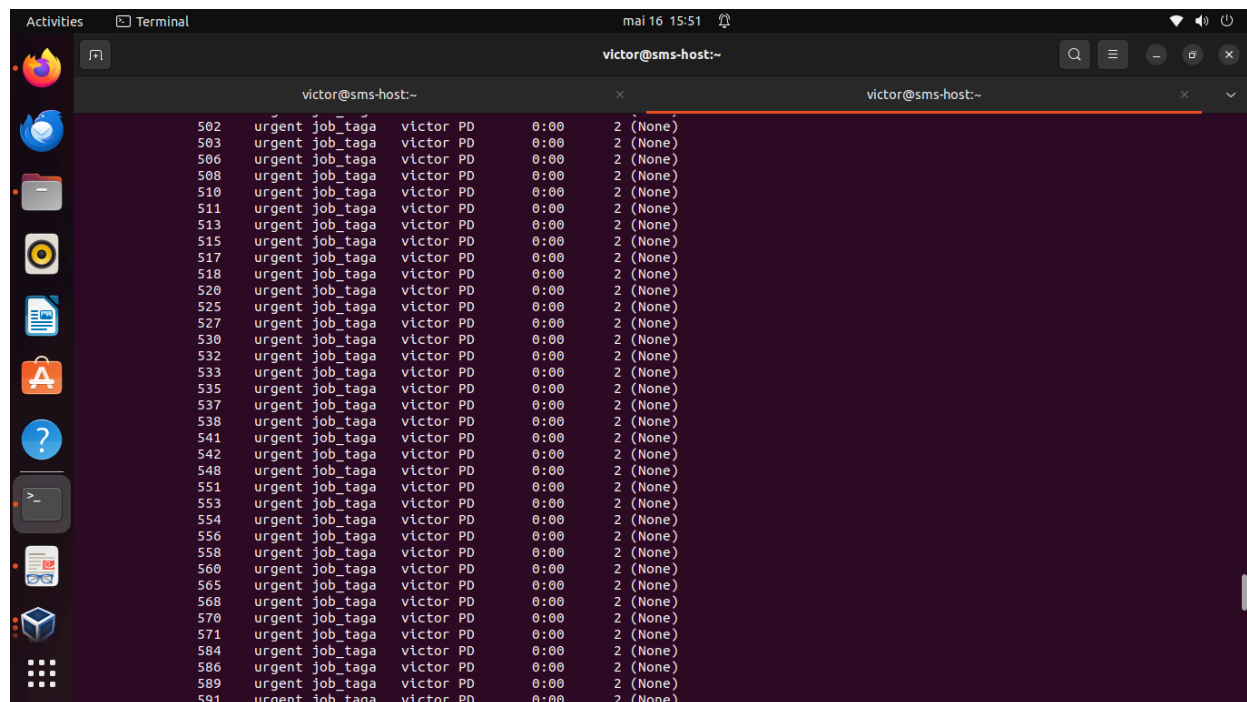
- PreemptMode=SUSPEND,GANG: permite suspender jobs, para cancelar usáramos CANCEL e para reagendar usamos RESCHEDULE
- PreemptType=preempt/partition_prio: define como critério de suspensão a prioridade da partição
- PartitionName=fair Nodes=compute0[0-1] Default=YES MaxTime=48:00:00 State=UP MaxMemPerNode=3000 Shared=YES Priority=10 PriorityTier=1

Para implementar a partição Urgent, utilizamos as seguintes configurações:

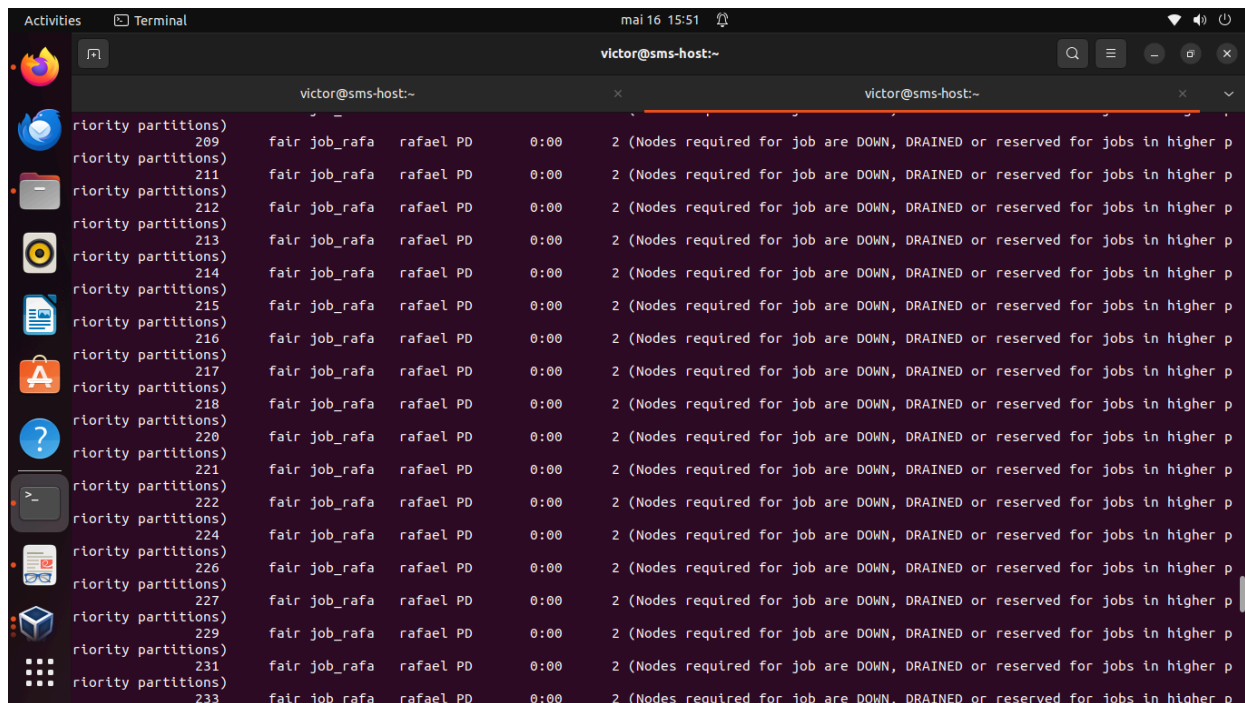
- PartitionName=urgent Nodes=compute0[0-1] Default=NO MaxTime=48:00:00 State=UP MaxMemPerNode=3000 Shared=YES Priority=10000 PriorityTier=3 OverSubscribe=FORCE:1

Penalizando usuários

É justo que usuários que usem pouco o cluster tenham maior prioridade em detrimento de usuários que utilizam muito o cluster. Para isso, utilizamos o tipo de prioridade multifatorial. Propositamente fizemos o job do André submeter quase nenhum job, ao mesmo tempo que o Victor e o Rafael submeteram 300 jobs pesados cada, esforço possível de ser observado nos prints abaixo.



```
victor@sms-host:~  
502 urgent job_taga victor PD 0:00 2 (None)  
503 urgent job_taga victor PD 0:00 2 (None)  
506 urgent job_taga victor PD 0:00 2 (None)  
508 urgent job_taga victor PD 0:00 2 (None)  
510 urgent job_taga victor PD 0:00 2 (None)  
511 urgent job_taga victor PD 0:00 2 (None)  
513 urgent job_taga victor PD 0:00 2 (None)  
515 urgent job_taga victor PD 0:00 2 (None)  
517 urgent job_taga victor PD 0:00 2 (None)  
518 urgent job_taga victor PD 0:00 2 (None)  
520 urgent job_taga victor PD 0:00 2 (None)  
525 urgent job_taga victor PD 0:00 2 (None)  
527 urgent job_taga victor PD 0:00 2 (None)  
530 urgent job_taga victor PD 0:00 2 (None)  
532 urgent job_taga victor PD 0:00 2 (None)  
533 urgent job_taga victor PD 0:00 2 (None)  
535 urgent job_taga victor PD 0:00 2 (None)  
537 urgent job_taga victor PD 0:00 2 (None)  
538 urgent job_taga victor PD 0:00 2 (None)  
541 urgent job_taga victor PD 0:00 2 (None)  
542 urgent job_taga victor PD 0:00 2 (None)  
548 urgent job_taga victor PD 0:00 2 (None)  
551 urgent job_taga victor PD 0:00 2 (None)  
553 urgent job_taga victor PD 0:00 2 (None)  
554 urgent job_taga victor PD 0:00 2 (None)  
556 urgent job_taga victor PD 0:00 2 (None)  
558 urgent job_taga victor PD 0:00 2 (None)  
560 urgent job_taga victor PD 0:00 2 (None)  
565 urgent job_taga victor PD 0:00 2 (None)  
568 urgent job_taga victor PD 0:00 2 (None)  
570 urgent job_taga victor PD 0:00 2 (None)  
571 urgent job_taga victor PD 0:00 2 (None)  
584 urgent job_taga victor PD 0:00 2 (None)  
586 urgent job_taga victor PD 0:00 2 (None)  
589 urgent job_taga victor PD 0:00 2 (None)  
591 urgent job_taga victor PD 0:00 2 (None)
```



```
victor@sms-host:~  
priority partitions)  
209 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
211 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
212 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
213 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
214 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
215 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
216 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
217 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
218 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
220 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
221 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
222 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
224 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
226 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
227 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
229 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
231 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p  
priority partitions)  
233 fair job_rafa rafael PD 0:00 2 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher p
```

A ideia desse experimento era reduzir ao máximo a prioridade dos jobs do Victor e do Rafael, enquanto a do André estaria intacta. Porém, quando fomos submeter um job do André, não foi possível observar o comportamento de furar fila. Novamente, não sabemos qual foi o erro cometido, mas como já havíamos demonstrado uma melhoria significativa na gestão do cluster a partir das partições e o prazo dos projetos estava muito apertado, decidimos simplesmente não investigar o que estava errado em nossa configuração. Registramos no relatório apenas para informar sobre a tentativa e a ideia que tínhamos em mente. No print abaixo é possível observar que os jobs do André não furaram a fila com jobs do Victor quando os jobs do Rafael foram cancelados.

```
[victor@sms-host ~]$ logout
[vagrant@sms-host ~]$ sudo su
[root@sms-host vagrant]# squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
706 urgent job_taga victor PD 0:00 2 (Priority)
705 urgent job_taga victor PD 0:00 2 (Resources)
707 urgent job_taga victor PD 0:00 2 (Priority)
708 urgent job_taga andre PD 0:00 2 (Priority)
709 urgent job_taga andre PD 0:00 2 (Priority)
710 urgent job_taga andre PD 0:00 2 (Priority)
711 urgent job_taga andre PD 0:00 2 (Priority)
701 fair job_rafa rafael R 4:15 2 compute[00-01]
703 fair job_rafa rafael S 3:30 2 compute[00-01]
704 fair job_rafa rafael S 3:31 2 compute[00-01]
702 fair job_rafa rafael S 3:30 2 compute[00-01]
[root@sms-host vagrant]# scancel -u rafael
[root@sms-host vagrant]# squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
706 urgent job_taga victor PD 0:00 2 (Resources)
707 urgent job_taga victor PD 0:00 2 (Priority)
708 urgent job_taga andre PD 0:00 2 (Priority)
709 urgent job_taga andre PD 0:00 2 (Priority)
710 urgent job_taga andre PD 0:00 2 (Priority)
711 urgent job_taga andre PD 0:00 2 (Priority)
705 urgent job_taga victor R 0:02 2 compute[00-01]
[root@sms-host vagrant]# squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
706 urgent job_taga victor PD 0:00 2 (Resources)
707 urgent job_taga victor PD 0:00 2 (Priority)
708 urgent job_taga andre PD 0:00 2 (Priority)
709 urgent job_taga andre PD 0:00 2 (Priority)
710 urgent job_taga andre PD 0:00 2 (Priority)
711 urgent job_taga andre PD 0:00 2 (Priority)
705 urgent job_taga victor R 0:11 2 compute[00-01]
[root@sms-host vagrant]# scancel 705
[root@sms-host vagrant]# squeue
```

Para tentar implementar o calculo da prioridade, tentamos utilizar as seguintes configurações:

- PriorityType=priority/multifactor: configura a definição de prioridade multifatorial
- PriorityUsageResetPeriod=6-0: define que a cada 6 dias a prioridade será resetada para não prejudicar
- PriorityWeightFairshare=10000: define uma alta prioridade para fazer uma distribuição justa de recursos para diferentes usuários
- PriorityWeightAge=1000: define uma media prioridade para a idade do job
- PriorityWeightPartition=10000: define uma alta prioridade para a partição do job

Códigos, Gráficos e Logs

Nós geramos os logs, estão no ZIP com todos os arquivos do projeto. Porém, como o nosso foco não foi melhorar o tempo de execução e sim deixar a gestão do cluster mais justa entre os usuários, não vimos sentido em utilizar os gráficos que seriam gerados pelos .py para a análise. Isso foi conversado tanto com o Prof. Michel quanto com o Demay, e ambos concordaram que não fazia sentido utilizar os gráficos. Só conseguiríamos testar a prioridade de usuário se algum tivesse poucos jobs enquanto os outros tivessem muitos, então os gráficos de quantidades de jobs por usuário também não fariam sentido. Por isso tiramos todos esses prints, são a evidência de melhoria que de fato faz sentido para

demonstrar o comportamento do cluster. Por fim, os códigos usados são apenas loops infinitos paralelizados, para que todo o recurso disponível fosse utilizado durante o experimento. Os arquivos também estarão inclusos no ZIP.