

BTU HPC Cluster

ANSYS FLUENT & SLURM ile İş Gönderme

1. HPC Cluster'a SSH ile giriş yapıyoruz:

```
tugrul@login: ~  
[ftugrul@yoga ~]$ ssh tugrul@10.15.62.251  
System information as of Mon Mar 29 05:37:52 +03 2021  
System load: 3.05 Processes: 109  
Usage of /home: unknown Users logged in: 2  
Memory usage: 7% IPv4 address for eth0: 10.128.110.103  
Swap usage: 0% IPv4 address for ext0: 10.15.62.251  
Temperature: 37.0 C IPv4 address for int0: 172.16.1.240  
  
BTU UNI  
High Performance Computing Cluster - Powered by CompecTA  
  
Name | MaxTimeLimit | Nodes  
-----  
short | 2 hours | 4 nodes  
mid | 1 day | 4 nodes  
long | 7 days | 4 nodes  
longer | 15 days | 4 nodes  
  
Job Scripts  
-----  
You can find example job scripts in: /cta/share/  
  
Application List etc.  
-----  
RUN : module avail  
RUN : btu-info, btu-queue, btu-summary (soon!)  
  
For General Help You Can:  
-----  
RUN : hpc-start-guide  
  
!!! IMPORTANT !!!  
-----  
Installation continues with the application!  
  
Last login: Mon Mar 29 05:36:53 2021 from 10.15.62.250  
tugrul@login:~$
```

2. Ev dizininiz altında yeni bir dizin oluşturun ve bu dizine geçin (bu örnekte FLUENT_TESTS adında bir dizin ile gösterilmiştir):

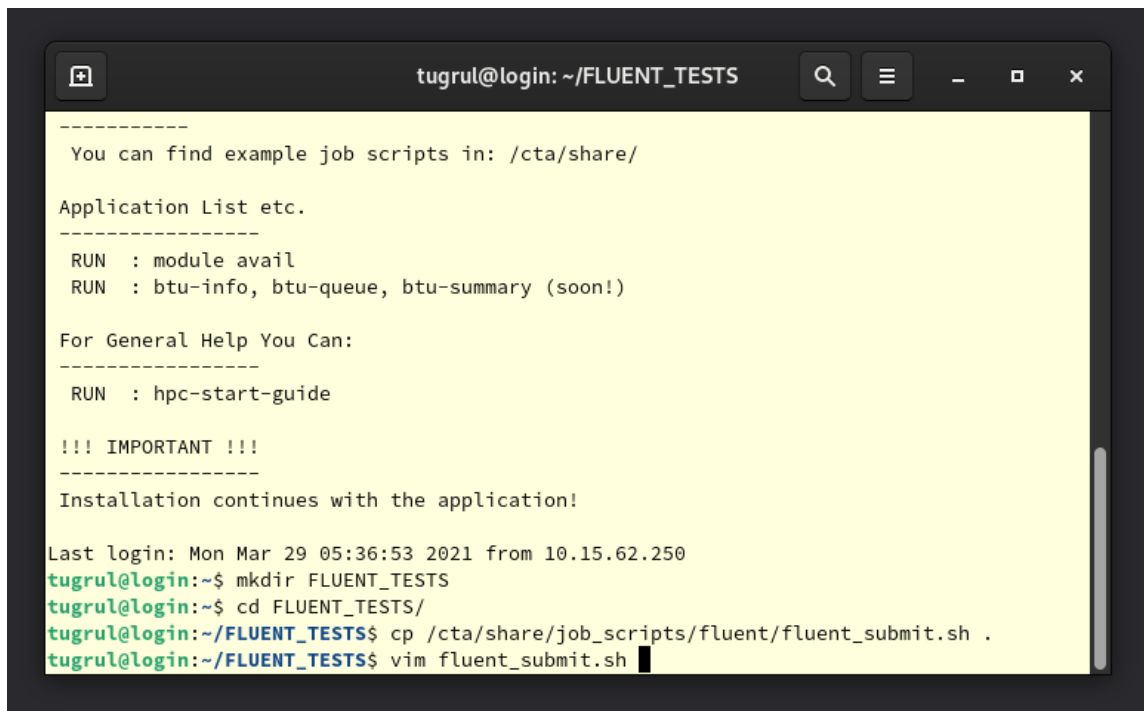
```
mkdir FLUENT_TESTS  
cd FLUENT_TESTS
```

3. Sonrasında HPC Cluster'a WinSCP ya da Filezilla gibi SFTP destekleyen bir uygulama ile bağlanarak case ve varsa data dosyalarınızı yine oluşturduğumuz bu dizinin içerisine kopyalayın. Biz bu örnek için cluster'daki örnek input'ları kullanacağız.
4. Aynı klasörün içine SLURM job script'i kopyalamamız gerekiyor. Bunun için biz /cta/share/job_scripts/fluent altındaki script'leri kullanıyoruz:

```
cp /cta/share/job_scripts/fluent/fluent_submit.sh .
```

5. Ardından job script dosyasını bir metin editörüyle açarak düzenliyoruz (bu örnekte "vim" metin editörü ile gösterilmiştir):

```
vim fluent_submit.sh
```



```
tugrul@login: ~/FLUENT_TESTS  
-----  
You can find example job scripts in: /cta/share/  
  
Application List etc.  
-----  
RUN : module avail  
RUN : btu-info, btu-queue, btu-summary (soon!)  
  
For General Help You Can:  
-----  
RUN : hpc-start-guide  
  
!!! IMPORTANT !!!  
-----  
Installation continues with the application!  
  
Last login: Mon Mar 29 05:36:53 2021 from 10.15.62.250  
tugrul@login:~$ mkdir FLUENT_TESTS  
tugrul@login:~$ cd FLUENT_TESTS/  
tugrul@login:~/FLUENT_TESTS$ cp /cta/share/job_scripts/fluent/fluent_submit.sh .  
tugrul@login:~/FLUENT_TESTS$ vim fluent_submit.sh
```

6. Açılan metin editöründe düzenlemek isteyeceğimiz satırlar şunlar:

```
#SBATCH --job-name=FLUENT_JOB
```

Bu satırda işin ismini belirtiyoruz. FLUENT_JOB yazan yere istediğiniz iş ismini yazın.

```
#SBATCH --nodes=2
```

Bu satırda işimizin kaç node (hesaplamayı yapacak makina, bilgisayar) olacağını belirtiyoruz. Bu örnekte biz 2 node seçtik

```
#SBATCH --ntasks-per-node 8
```

Burada işimizin bir node'da kaç CPU core'u kullanmasını istediğimizi belirtiyoruz.

```
#SBATCH --exclusive
```

Bu parametreyi değiştirmemiz gerek yok. Bu seçenek, makina başına 8 core istediğimiz ve geriye 2 adet de boş core kaldığı için, SLURM'a bu boş kalan 2 core'a başka bir iş göndermemesini ve sadece bizim kullanımımız için ayırmasını söylüyor.

```
#SBATCH --partition=long
```

Bu parametre işimizin hangi kuyruğa gideceğini belirtiyor. Şu an için Melkor HPC Cluster'da short, mid, long ve infinite isimli dört kuyruk tanımlı. Short 2 saat, mid 1 gün, long 7 gün ve infinite kuyruğu da sonsuz maksimum iş süresi belirtmeye yarıyor. İşiniz 7 günden uzun sürmeyecekse bu şekilde kalabilir, daha uzun sürecekse long yerine infinite yazın.

```
#SBATCH --time=4-0
```

Üstteki kuyruk seçtiğimiz parametreye göre burada bir süre belirtiyoruz. Buradaki 4-0 seçeneği 4 gün ve sıfır saat talep ediyor. GÜN-SAAT : DAKİKA şeklinde süre belirtilebilir. Örneğin 1 gün 12 saat için 1-12:00 girilebilir.

Eğer infinite kuyruğunu seçtiyseniz burada 4-0 yerine de infinite yazabilirsiniz.

```
#SBATCH --mail-type=ALL
```

Bu satır iş ile ilgili güncellemeleri size eposta ile gönderilmesini belirtiyor. Aynı şekilde bırakın.

```
# #SBATCH --mail-user=CHANGE_ME@btu.edu.tr
```

Bu satır iş ile ilgili güncellemelerin hangi eposta adresine gönderileceğini belirtiyor. En başındaki diyez ve boşluk olması bu opsiyonu geçersiz kılıyor. Dolayısı ile geçerli olması için şu şekilde düzenlenmesi gerek:

```
#SBATCH --mail-user=name.surname@btu.edu.tr
```

Yukarıda name.surname yazan kısma kendi eposta adresinizi girin.

JOURNAL DOSYASI SEÇENEKLERİ

Aşağıdaki satırlar işin çalışması için gereken Journal dosyası ile ilgili ayarları belirtiyor.

Fluent iş göndermeyi kolaylaştırmak için CompectA'nın yazdığı Journalist.py uygulaması ile journal dosyası üretiyoruz. Bununla ilgili ek bilgiyi bu dokümanın en altında bulabilirsiniz.

```
#EXTRA_ARGS='--no-init'
```

```
#EXTRA_ARGS='--no-data-read'
```

Buradaki EXTRA_ARGS parametresi Journalist uygulamasına verilecek ekstra seçenekleri belirtiyor. Bunun çalışması için başındaki diyezın kaldırılması lazım.

Örneğin sadece hybrid initialization yapmamasını istersek:

```
EXTRA_ARGS='--no-init'
```

Şeklinde düzenlememiz gerek.

Veya data okumamasını istersek:

```
EXTRA_ARGS='--no-data-read'
```

Olarak düzenlememiz gerek.

JOB SCRIPT'in DÜZENLENMİŞ HALİ:

```
tugrul@login: ~
#!/bin/bash

# Fluent Job Script
# Author: Ekrem Seren <ekrem@compecta.com>

### BEGIN SLURM OPTIONS ###

#SBATCH --job-name=FLUENT_JOB
#SBATCH --nodes=2
#SBATCH --ntasks-per-node 8
#SBATCH --exclusive
#SBATCH --partition=longer
#SBATCH --time=15-00:00:00

# Match comments on lines below if you would like to get email updates
#SBATCH --mail-type=ALL
# #SBATCH --mail-user=CHANGE_ME@btu.edu.tr

### END SLURM OPTIONS ###

# Extra args for Journalist - Journal Generator
# Do not do hybrid initialization
#EXTRA_ARGS='--no-init'

# Do not read data file:
EXTRA_ARGS='--no-data-read'

# keep this

1,1 Top
```

İşin ismini belirttik, node sayısı 3, node başına core sayısını 12 olarak belirledik (2 HPC Pack için), epostamızı girdik ve başındaki diyez ve boşluğu sildik ve no data read olan satırın başındaki diyezi kaldırdık. Bu iş için data dosyası okumayacağız ve hybrid initialization yapacağız.

Bu aşamadan sonra dosyayı kaydederek kapatıyoruz. (vim metin editörü için önce “:” sonra ise “wq” karakterleri girilir, ardından “enter”)

İŞİ SUBMIT ETME

Sonra bir SSH penceresi açıyoruz. Burada işi submit etmek için sbatch komutunu kullanacağız. Komut satırı mantıksal olarak şöyle olacak:

```
sbatch <job_script> <case_file> <iterasyon sayısı> <save_every_iteration>
```

Burada, sbatch'den sonra vereceğimiz ilk parametre job script'in ismi olmalı. Buradaki örnekte; fluent_submit.sh olacak.

İkinci parametre case dosyasının adı. Buradaki örnekte FFF-1.cas olacak.

Üçüncü parametre kaç iterasyon çözdürmek istediğimiz. Eğer üçüncü parametreyi vermezsek default olarak 1000 iterasyon çözer. Bu yüzden, eğer 1000 istiyorsak bunu belirtmek zorunda değiliz.

Dördüncü parametre, verilen iterasyon sayısı içerisinde kaç iterasyonda bir case ve data dosyası yazmak istediğimiz. Örneğin üçüncü parametrede 1000 iterasyon istediyseniz ve 200 iterasyonda bir case ve data yazmasını istersek, burada 200 belirtiyoruz. Bu parametre de zorunlu değil. Hiçbir şey belirtmezsek istenen tüm iterasyonlar bittikten sonra bir kez case ve data dosyası oluşturulacaktır.

İP UCU: Terminal penceresindeyken, bir komutu veya dosya adını otomatik olarak tamamlamak için ilk birkaç harfini yazdıktan sonra klavyeden TAB tuşuna basabiliriz. Eğer bastıktan sonra komutun veya dosya adını sonuna kadar tamamlamadıysa, birden fazla ihtimal var demektir. Peş peşe iki kere TAB tuşuna basarsak tüm ihtimalleri listeler.

Örnek işimizi şu şekilde gönderiyoruz:

```
sbatch fluent_submit.sh FFF-1.cas
```

```
tugrul@login: ~/FLUENT_TESTS
Application List etc.
-----
RUN : module avail
RUN : btu-info, btu-queue, btu-summary (soon!)

For General Help You Can:
-----
RUN : hpc-start-guide

!!! IMPORTANT !!!
-----
Installation continues with the application!

Last login: Mon Mar 29 05:36:53 2021 from 10.15.62.250
tugrul@login:~$ mkdir FLUENT_TESTS
tugrul@login:~$ cd FLUENT_TESTS/
tugrul@login:~/FLUENT_TESTS$ cp /cta/share/job_scripts/fluent/fluent_submit.sh .
tugrul@login:~/FLUENT_TESTS$ vim fluent_submit.sh
tugrul@login:~/FLUENT_TESTS$ sbatch fluent_submit.sh FFF-1.cas
Submitted batch job 3370
tugrul@login:~/FLUENT_TESTS$
```

Enter'a bastıktan sonra bize işin ID'sini belirten bir satır basacak. Buradaki örnekte job id 3370.

İşin kuyruktaki durumunu sorgulamak için **sqa** komutunu kullanıyoruz:

```
tugrul@login: ~/FLUENT_TESTS
-----
RUN : module avail
RUN : btu-info, btu-queue, btu-summary (soon!)

For General Help You Can:
-----
RUN : hpc-start-guide

!!! IMPORTANT !!!
-----
Installation continues with the application!

Last login: Mon Mar 29 05:36:53 2021 from 10.15.62.250
tugrul@login:~$ mkdir FLUENT_TESTS
tugrul@login:~$ cd FLUENT_TESTS/
tugrul@login:~/FLUENT_TESTS$ cp /cta/share/job_scripts/fluent/fluent_submit.sh .
tugrul@login:~/FLUENT_TESTS$ vim fluent_submit.sh
tugrul@login:~/FLUENT_TESTS$ sbatch fluent_submit.sh FFF-1.cas
Submitted batch job 3370
tugrul@login:~/FLUENT_TESTS$ sqa
      JOBID PARTITI NAME      USER  ACCOUNT      TIME  TIME_LIMIT  NODES  CPUS  MIN_MEM  STATE  NODELIST(REASON)
      3370  longer FLUENT_JOB  tugrul  users        2:04  15-00:00:00  2     40   6000M  RUNNING ne[01-02]
tugrul@login:~/FLUENT_TESTS$
```

Eğer sqa komutunu verdiğinizde işinizi listede göremezseniz, işiniz hemen sonlanmış demektir. Bu çoğu zaman job script içindeki bir yazım hatasından meydana gelir. Çıktı dosyasının en sonuna bakarak neden sonlandığı ile ilgili bir fikir sahibi olabilirsiniz. Desteğe ihtiyacınız olduğunda aşağıda bahsedilen çıktı dosyası ve job id ile birlikte support@compecta.com'a mail atabilirsiniz.

İşin bastığı çıktıyı görmek için (Solver'ın çıktılarını) tail komutunu kullanabiliriz. İşin çıktıları **slurm-<jobid>.out** isimindeki dosyaya yazılır. Buradaki örnekte dosya adı **slurm-621.out**

Aşağıdaki örnekte tail komutu ile sürekli olarak (-f) ve ilk okuma için 100 satır basacak şekilde komutu veriyoruz:

```
tugrul@login: ~/FLUENT_TESTS

-----
RUN  : module avail
RUN  : btu-info, btu-queue, btu-summary (soon!)

For General Help You Can:
-----
RUN  : hpc-start-guide

!!! IMPORTANT !!!
-----
Installation continues with the application!

Last login: Mon Mar 29 05:36:53 2021 from 10.15.62.250
tugrul@login:~$ mkdir FLUENT_TESTS
tugrul@login:~$ cd FLUENT_TESTS/
tugrul@login:~/FLUENT_TESTS$ cp /cta/share/job_scripts/fluent/fluent_submit.sh .
tugrul@login:~/FLUENT_TESTS$ vim fluent_submit.sh
tugrul@login:~/FLUENT_TESTS$ sbatch fluent_submit.sh FFF-1.cas
Submitted batch job 3370
tugrul@login:~/FLUENT_TESTS$ sqa
      JOBID PARTITI NAME      USER  ACCOUNT    TIME  TIME_LIMIT  NODES  CPUS  MIN_MEM  STATE  NODELIST(REASON)
      3370  longer FLUENT_JOB  tugrul  users      2:04  15-00:00:00  2     40   6000M  RUNNING ne[01-02]
tugrul@login:~/FLUENT_TESTS$ tail -f -n 100 slurm-3370.out
```

Komutu verdikten sonraki çıktı:

```
tugrul@login: ~/FLUENT_TESTS

 5  6.3614e-01  4.9427e-04  5.6639e-04  1.2549e-03  3.7838e-04  0:04:23  995
 6  5.4952e-01  3.7916e-04  4.6289e-04  1.3078e-03  3.6099e-04  0:04:14  994
 7  4.7543e-01  2.6468e-04  3.5395e-04  1.3047e-03  3.3680e-04  0:04:08  993
 8  4.4635e-01  2.3754e-04  3.1711e-04  1.2836e-03  3.0493e-04  0:04:02  992
 9  4.6309e-01  2.8110e-04  3.4077e-04  1.1603e-03  2.6926e-04  0:03:58  991
10  4.8793e-01  3.1814e-04  3.7265e-04  1.0150e-03  2.3391e-04  0:03:54  990
11  4.3958e-01  2.3581e-04  2.8997e-04  9.8906e-04  2.0312e-04  0:03:53  989
iter continuity x-velocity y-velocity z-velocity energy time/iter
12  4.3955e-01  2.3387e-04  2.8150e-04  9.4331e-04  1.7756e-04  0:03:50  988
13  4.2706e-01  2.0869e-04  2.5917e-04  9.3195e-04  1.5627e-04  0:03:48  987
14  4.1791e-01  2.0072e-04  2.5277e-04  9.1572e-04  1.3883e-04  0:03:47  986
15  4.2685e-01  2.2621e-04  2.7890e-04  9.5775e-04  1.2474e-04  0:03:45  985
16  4.0084e-01  2.0359e-04  2.5726e-04  9.2353e-04  1.1374e-04  0:03:45  984
17  4.0769e-01  2.4479e-04  2.8897e-04  1.0110e-03  1.0479e-04  0:03:45  983
18  3.7193e-01  1.9729e-04  2.4240e-04  9.7718e-04  9.6878e-05  0:03:44  982
19  3.5201e-01  1.8259e-04  2.2686e-04  9.4791e-04  8.9761e-05  0:03:43  981
20  3.3744e-01  1.7677e-04  2.1976e-04  9.1034e-04  8.3267e-05  0:03:42  980
21  3.3409e-01  1.8574e-04  2.2860e-04  8.3352e-04  7.7387e-05  0:03:42  979
22  3.0761e-01  1.6250e-04  2.0420e-04  8.0661e-04  7.1898e-05  0:03:41  978
iter continuity x-velocity y-velocity z-velocity energy time/iter
23  2.9321e-01  1.5326e-04  1.9176e-04  7.8414e-04  6.6922e-05  0:03:40  977
24  2.8270e-01  1.4871e-04  1.8512e-04  7.5935e-04  6.2106e-05  0:03:40  976
```


Bu komutu kesmek için CTRL+C tuşları kullanılabilir veya Terminal penceresi kapatılabilir.

İşin bulunduğu klasörde çıktı dosyalarını görmek için, pencereyi seçip F5 tuşu ile güncelleyin:

Bu aşamadan sonra işin bitmesini bekliyoruz. Eposta adresimizi job script'e girdiysek, iş bittiğinde SLURM eposta ile bilgi verecek.

Eposta başlığı şuna benzer olacak:

[BTU] Slurm Job_id=621 Name=FLUENT_JOB Began, Queued time 00:00:01

CompecTA Journalist.py

Fluent Journal File Generator

Melkor HPC Cluster'da Fluent işi göndermeyi kolaylaştırmak için Journalist.py isimli uygulamayı geliştirdik. Bir terminal penceresinden `journalist.py` yazılarak kullanılabilir.

Örneğin uygulamanın kullanım bilgisini görmek için:

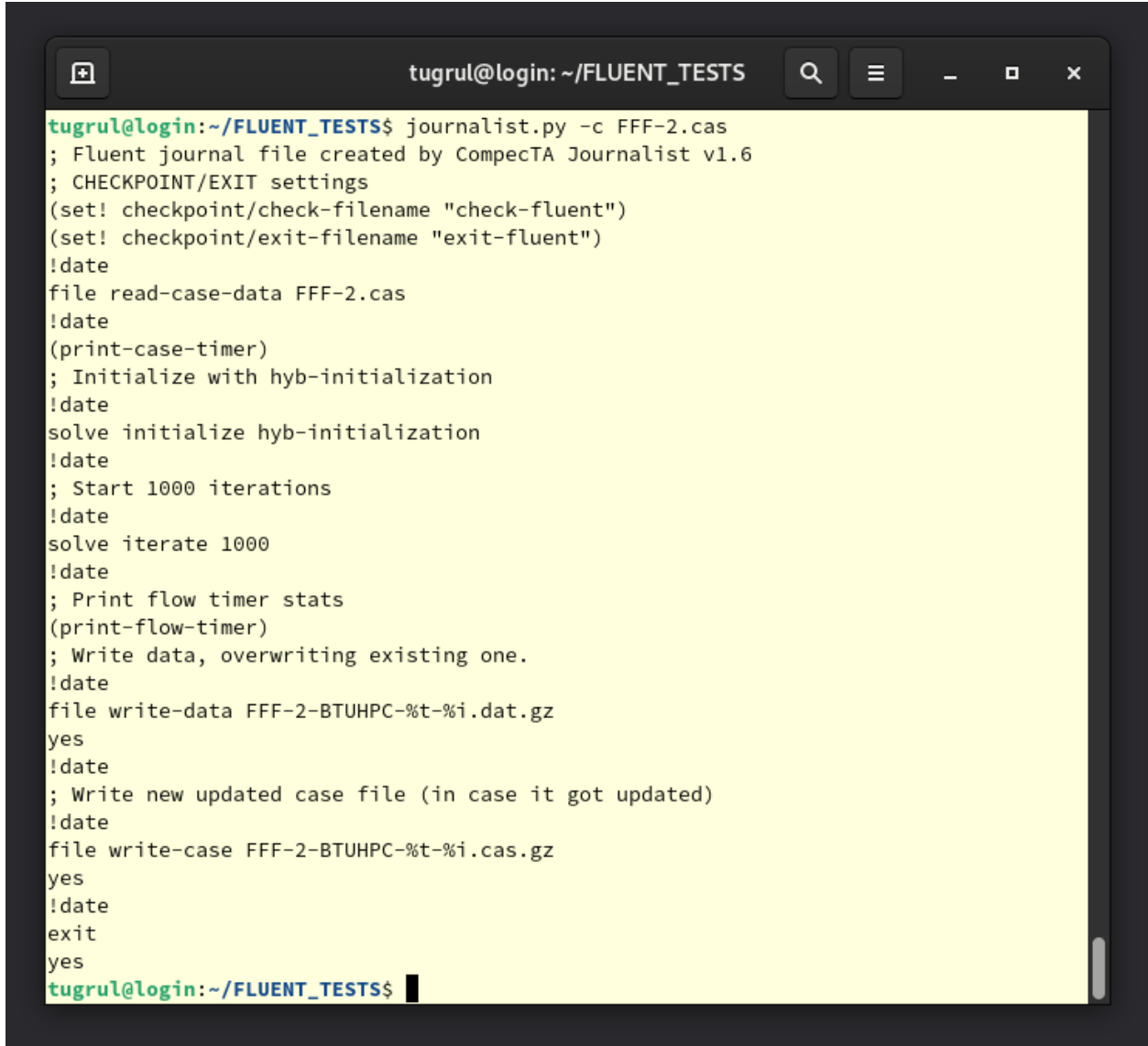
```
journalist.py --help
```

```
tugrul@login: ~  
  
tugrul@login:~$ journalist.py --help  
usage: CompectA Journalist [-h] -c CASE_FILE [-d DATA_FILE] [-nd] [-ni] [-if]  
      [-it ITERATIONS] [-trn TRANSIENT]  
      [-s SAVE_EVERY_ITERATIONS] [--version]  
  
Generate ANSYS Fluent Journal File.  
  
optional arguments:  
  -h, --help            show this help message and exit  
  -c CASE_FILE, --case-file CASE_FILE  
                        Fluent case file to read.  
  -d DATA_FILE, --data-file DATA_FILE  
                        Fluent data file to read. Default: <case-file>.dat.gz  
  -nd, --no-data-read  Do not read data file. Default: False (read data)  
  -ni, --no-init, --no-initialize  
                        Do not hyb-initialize. Default: False (do hyb-  
                        initialization)  
  -if, --init-flow     DO solve initialize initialize-flow instead of hyb-  
                        initialize  
  -it ITERATIONS, --iterations ITERATIONS  
                        Iteration count. Default: 1000  
  -trn TRANSIENT, --transient TRANSIENT  
                        Transient total time (s). Default: 1.0  
  -s SAVE_EVERY_ITERATIONS, --save-every-iterations SAVE_EVERY_ITERATIONS  
                        Write data file every X iterations. Default: Disabled  
  --version            show program's version number and exit  
  
CompectA HPC Solutions (c) 2020  
tugrul@login:~$
```

Verilen parametrelere göre journal dosyası oluşturup, çıktığı basar. Bu belgede yukarıda bahsettiğimiz job script içindeki `EXTRA_ARGS` parametresi bu uygulamaya ek parametre verir ve gerekli journal dosyasını işi çözmek için üretir.

Örneğin `deneme.cas.gz` için bir journal oluşturmak için:

```
journalist.py -c deneme.cas.gz
```



```
tugrul@login: ~/FLUENT_TESTS
tugrul@login:~/FLUENT_TESTS$ journalist.py -c FFF-2.cas
; Fluent journal file created by CompectA Journalist v1.6
; CHECKPOINT/EXIT settings
(set! checkpoint/check-filename "check-fluent")
(set! checkpoint/exit-filename "exit-fluent")
!date
file read-case-data FFF-2.cas
!date
(print-case-timer)
; Initialize with hyb-initialization
!date
solve initialize hyb-initialization
!date
; Start 1000 iterations
!date
solve iterate 1000
!date
; Print flow timer stats
(print-flow-timer)
; Write data, overwriting existing one.
!date
file write-data FFF-2-BTUHPC-%t-%i.dat.gz
yes
!date
; Write new updated case file (in case it got updated)
!date
file write-case FFF-2-BTUHPC-%t-%i.cas.gz
yes
!date
exit
yes
tugrul@login:~/FLUENT_TESTS$
```

Yukarıdaki örnekte uygulama journal dosyasını ekrana basıyor. Bunu bir dosyaya yazdırmak için büyükdür işaretini ">" kullanabiliriz. Örneğin:

```
tugrul@login: ~/FLUENT_TESTS
tugrul@login:~/FLUENT_TESTS$ journalist.py -c deneme.cas.gz -it 5000 > journal-test.jou
tugrul@login:~/FLUENT_TESTS$ ls
deneme.cas.gz  journal-test.jou
tugrul@login:~/FLUENT_TESTS$ cat journal-test.jou
; Fluent journal file created by Compecta Journalist v1.6
; CHECKPOINT/EXIT settings
(set! checkpoint/check-filename "check-fluent")
(set! checkpoint/exit-filename "exit-fluent")
!date
file read-case-data deneme.cas.gz
!date
(print-case-timer)
; Initialize with hyb-initialization
!date
solve initialize hyb-initialization
!date
; Start 5000 iterations
!date
solve iterate 5000
!date
; Print flow timer stats
(print-flow-timer)
; Write data, overwriting existing one.
!date
file write-data deneme-BTUHPC-%t-%i.dat.gz
yes
!date
; Write new updated case file (in case it got updated)
!date
file write-case deneme-BTUHPC-%t-%i.cas.gz
yes
!date
exit
yes
tugrul@login:~/FLUENT_TESTS$
```

Burada verdiğimiz komut:

```
journalist.py -c deneme.cas.gz -it 5000 > journal-test.jou
```

Bu komut **deneme.cas.gz** isimli case dosyası için 5000 iterasyondan oluşan bir journal dosyası oluştur ve bunu **journal-test.jou** isimli dosyaya yaz diyor.

Sonrasındaki **ls** komutu bulunduğumuz klasördeki dosyaları listeliyor. Ondan sonraki **cat journal-test.jou** komutu da **journal-test.jou** dosyasının içeriğini ekrana basıyor.