Index

Objectius

Aquesta pràctica té com a objectius introduir l'entorn de treball. A part de la logística per accedir al sistema, també es posarà en pràctica l'ús de sistemes de cues i l'execució sistemàtica de proves per a realitzar estudis paramètrics i/o de rendiment.

Presentació de la pràctica

Cal lliurar un document amb les respostes a les preguntes formulades, scripts/gràfics que es demanen i els comentaris que considereu oportuns.

Restriccions de l'entorn

Tot i que el desenvolupament de la pràctica és molt més interessant utilitzant dotzenes de computadors, s'assumeix que inicialment tindreu accés a un nombre força reduït de computadors amb diversos nuclis per node.

Material per a la realització de la pràctica

En els servidors de la UOC teniu el programari necessari per realitzar les execucions requerides. De totes maneres, és possible que tingueu que instal·lar software client per tal d'accedir als sistemes de la UOC.

Preguntes preliminars

Q1. Before searching for information (please provide a frank response): Do you know what is cyberinfrastructure and what does it mean?

Entenc que 'cyberinfrastructure' es la infraestructura de comunicacions, dispositius de xarxa i programari, que son necessaris per enviar dades entre diferents equips.

Q2. After exploring online: Please describe the concept of cyberinfrastructure and provide one example.

El concepte de cyberinfraestructura [1] , segons la universitat d'Indiana [2], consisteix en sistemes computacionals, dades i gestió de la informació, instruments avançats, entorns de visualització i persones, tot interconnectat per programari i xarxes avançades per millorar la productivitat escolar i disposar d'avanços en el coneixement impossible d'un altre forma. Un exemple es 'Open Science Grid' [3] que dona a investigadors l'accés a una xarxa distribuïda d'alta capacitat computacional per a investigacions.

Q3. How do you think you can interface with a data-centric cyberinfrastructure?

La forma de comunicar-se amb una 'data-centric cyberinfrastructure' podria ser a través d'equips encarregats de gestionar les consultes sobre les dades disponibles de la cyberinfraestructure, de tal manera que les dades no es moguin a través d'internet, encara que aquesta forma concentra el procés computacional en un mateix lloc.

Q4. How do you think cyberinfrastructure can be useful for science and engineering? Please provide an example.

Una cyberinfraestructura permet crear una plataforma on es concentri coneixement per tal que estigui disponible per investigadors, professors i alumnes de l'àmbit científic i d'enginyeria. Això millora l'accés a dades que poden facilitar estudis i recerca, que de forma individual seria impossible. Un exemple es XSEDE [5]que ofereix serveis de computació, dades i experiència a l'àmbit científic.

Enunciat de la pràctica

Utilitzarem un entorn tipus clúster de computació basat en GNU/Linux per a la realització de les pràctiques. A l'aula us proporcionem un manual d'ús bàsic del sistema de cues, si us plau utilitzeu-ho com a primera referència. També podeu trobar més informació a la web, per exemple al següent link podeu trobar més detalls del sistema de cues SGE (Sun Grid Engine) que hi ha instal·lat al clúster de la UOC:

http://gridscheduler.sourceforge.net/htmlman/htmlman1/qsub.html

L'accés a l'entorn de treball és mitjançant:

```
manelmdiaz@manelmdiaz-Desktop-Ubuntu:~$ ssh -p55000 capa20@eimtarqso.uoc.edu
Password:
Last login: Sun Sep 30 22:36:36 2018 from 94.248.90.19
Rocks 6.1 (Emerald Boa)
Profile built 14:11 01-Aug-2013
Kickstarted 16:38 01-Aug-2013
```

El nom_usuari es proporcionarà un cop sol·liciteu el vostre compte del sisema.

Descripció del sistema

Eimtarqso és el node principal d'un clúster de 10 nodes de còmput. Eimtarqso es l'únic node que heu d'accedir directament i a partir del qual podreu executar tasques a la resta dels nodes mitjançant el sistema de cues de SGE. És molt important que respectiu aquesta directiva ja que els recursos s'han de compartir amb la resta de companys. Els detalls del sistema els podem observar a través de la següent comanda (el resultat es veu a continuació):

```
[ivan@eimtarqso ~]$ qstat -f
queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@compute-0-0.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-1.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-2.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-3.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-4.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-5.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-6.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-7.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-8.local        BIP   0/0/4          0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-0-9.local        BIP   0/0/4          0.00     linux-x64
```

La sortida mostra 10 nodes de còmput (compute-0-0.local ... compute-0.9.local), cadascun dels nodes amb 4 cores totals (en la columna resv/used/tot) i una càrrega mitja total de 0.00.
Nota: tot i que el sistema us ho permeti, no us connecteu a cap node de còmput de

forma directa en cap cas.

Execució tradicional vs. en clúster:
Per exemplificar l'ús del sistema de cues (SGE) farem servir dues comandes de sistema que teniu per defecte al vostre path:
- hostname, proporciona el nom del servidor/node
- sleep, fa que el procés entri en espera durant el nombre de segons indicat
Típicament la forma de consultar el nom del servidor és el següent:

```
[ivan@eimtarqso ~]$ hostname
eimtarqso.uoc.edu
```

Podem executar aquesta comanda en altres nodes del clúster mitjançant un script de SGE. A continuació us proporcionem un exemple bàsic:

```
[ivan@eimtarqso ~]$ cat hostname_example.sge
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N hostname_jobname
#$ -o hostname.out
#$ -e hostname.err
hostname
```

La última línia de l'script fa la crida a la comanda que volem executar. La següent línia seria equivalent: echo `hostname` (noteu que les cometes son cap a l'esquerra...)

Per executar l'script haurem de fer servir qsub. Un exemple es mostra a continuació:
[ivan@eimtarqso ~]$ qsub hostname_example.sge
Your job 263534 ("hostname_jobname") has been submitted

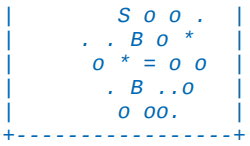
Primer accés al clúster

```
manelmdiaz@manelmdiaz-Desktop-Ubuntu:~$ ssh -p55000 capa20@eimtarqso.uoc.edu
Password:
Rocks 6.1 (Emerald Boa)
Profile built 14:11 01-Aug-2013

Kickstarted 16:38 01-Aug-2013

It appears that you have not set up your ssh key.
This process will make the files:
     /home/capa20/.ssh/id_rsa.pub
     /home/capa20/.ssh/id_rsa
     /home/capa20/.ssh/authorized_keys

Generating public/private rsa key pair.
Created directory &apos;/home/capa20/.ssh&apos;.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/capa20/.ssh/id_rsa.
Your public key has been saved in /home/capa20/.ssh/id_rsa.pub.
The key fingerprint is:
db:75:9a:f1:a6:a6:11:7a:88:7c:8f:b1:45:42:86:c3 capa20@eimtarqso.uoc.edu
The key&apos;s randomart image is:
+--[ RSA 2048]----+
|                 |
|     . .         |
|      E o        |
|       +         |
```



PREGUNTES 1,2 i 3

1. Quin és el resultat de la comanda hostname? On es pot trobar? Comproveu si s'han creat fitxers nous.

Resultat de la comanda hostname:

[capa20@eimtarqso ~]$ hostname
eimtarqso.uoc.edu

On es pot trobar: El nom del servidor es pot trobar en el símbol del sistema. → [capa20@**eimtarqso** ~]

Comproveu si s'han creat fitxers nous.
Un cop s'accedeix al clúster demana generar les claus ssh que genera els següents fitxers a la carpeta '.ssh':

/home/capa20/.ssh/id_rsa.pub
/home/capa20/.ssh/id_rsa
/home/capa20/.ssh/authorized_keys

També s'han creat les carpetes 'bio' i '.t_coffee'.

2. Executeu l'script d'exemple varies vegades (per exemple 3-4). Quin són els resultats obtinguts? És sempre el mateix? Per què?

He executat l'script diversos cops:

```
[capa20@eimtarqso ~]$ qsub hostname_example.sge
Your job 416945 ("hostname_jobname") has been submitted
[capa20@eimtarqso ~]$ qsub hostname_example.sge
Your job 416946 ("hostname_jobname") has been submitted
[capa20@eimtarqso ~]$ qsub hostname_example.sge
Your job 416947 ("hostname_jobname") has been submitted
[capa20@eimtarqso ~]$ qsub hostname_example.sge
Your job 416949 ("hostname_jobname") has been submitted
[capa20@eimtarqso ~]$ qsub hostname_example.sge
Your job 416950 ("hostname_jobname") has been submitted
```

El resultat del fitxer hostname.out mostra el nom del node que executa l'script. Cada cop que s'executa l'script s'afegeix una línia al final amb el nom del node que l'ha executat.

```
[capa20@eimtarqso ~]$ cat hostname.out
compute-0-9.local
compute-0-6.local
compute-0-2.local
compute-0-5.local
compute-0-4.local
```

El fitxer hostname.err està buit, entenc que es degut a que cap tasca ha retornat error.

```
[capa20@eimtarqso ~]$ cat hostname.err
[capa20@eimtarqso ~]$
```

3. Si executeu l'script d'exemple (el mateix fitxer) múltiples vegades a la vegada, teniu problemes per conservar el resultat de la sortida de cadascuna de les execucions? Com ho podeu fer per guardar el resultat de totes les execucions de forma independent?

He executat 7 vegades seguides l'script i al fitxer hostname.out només mostra dos resultats.

```
[capa20@eimtarqso ~]$ cat hostname.out
compute-0-4.local
compute-0-6.local
```

Per guardar de forma independent la sortida per a cada job utilitzarem variables especifica '$JOB_ID'. He creat el script 'hostname_example_p3.sge'

```
codi 'hostname_example_p3.sge'
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N hostname_jobname
#$ -o hostname_$JOB_ID.out
#$ -e hostname_$JOB_ID.err
hostname
```

Executo l'script dos cops de forma simultànea:

```
[capa20@eimtarqso ~]$ qsub hostname_example_p3.sge
Your job 417235 ("hostname_jobname") has been submitted
[capa20@eimtarqso ~]$ qsub hostname_example_p3.sge
Your job 417236 ("hostname_jobname") has been submitted
```

Resultat: es generen dos fitxers de sortida i dos d'errors

```
-rw-r--r-- 1 capa20 capa20    0 oct  1 20:19 hostname_417235.err
-rw-r--r-- 1 capa20 capa20   18 oct  1 20:19 hostname_417235.out
-rw-r--r-- 1 capa20 capa20    0 oct  1 20:19 hostname_417236.err
-rw-r--r-- 1 capa20 capa20   18 oct  1 20:19 hostname_417236.out
```

Un cop heu enviat un treball (job) a la cua mitjançant qsub, podeu cancel·lar-lo si és necessari (per exemple, si us heu adonat que hi ha un error o voleu fer canvis). Això es pot fer mitjançant la comanda SGE qdel.
Per a demostrar la seva utilització farem servir un altre script de prova que faci un sleep de varis segons i ens permeti veure que està a la cua, en execució i finalment cancel·lar-ho. L'script proposat és el següent:

```
[ivan@eimtarqso ~]$ cat sleep.sge
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N sleep_job
#$ -o sleep.out
sleep 100
```

Tasques (jobs) grans vs. múltiples tasques petites:

A vegades hem de fer estudis paramètrics o estadístics que impliquen l'execució d'un programa múltiples vegades. En aquest escenari tenim dues opcions bàsiques:

1. Utilitzar un script SGE que realitzi les diferents execucions com a part d'un únic treball
2. Enviar múltiples treballs per a cadascuna de les execucions.

En aquest curs esperem que utilitzeu la modalitat 2 ja que fa que l'accés al recursos sigui més justa entre tots els estudiants. Per exemple, si uns pocs estudiants realitzen execucions de gran durada en tots els recursos poden fer que la resta hagin d'esperar durant hores o fins i tot dies.

Avaluació de rendiment: estudi paramètric d'un codi d'exemple:

A continuació es presenten els passos necessaris per compilar i executar un programa per a la multiplicació de matrius (sense optimitzacions) que us proporcionem en aquesta pràctica:

- Compilar el programa mm.c amb:

  Sense optimitzacions: gcc –O0 mm.c -o mm

  Amb optimitzacions (compilador): gcc -O3 mm.c -o mm

- Executar el programa mitjançant el sistema de cues SGE. Noteu que el tamany de la matriu se li passa al programa mitjançant un paràmetre.

  IMPORTANT: no feu execucions directament a l'intèrpret de comandes ja que podeu saturar el node d'accés i les mesures que prengueu poden no ser vàlides per haver de compartir els recursos amb altres usuaris al mateix temps.

PREGUNTES 4 i 5

4. Com ho heu fet per copiar et fitxer mm.c des del vostre PC fins al servidor eimtarqso.uoc.edu ?

He utilitzat scp (Secure Copy) per copiar el fitxer 'mm.c' del meu equip al servidor eimtarqso.uoc.edu a la carpeta arrel del meu usuari.

```
manelmdiaz@manelmdiaz-Desktop-Ubuntu:~/Nextcloud/Manel/Estudios/UOC/Computacions
d&amp;apos;altes prestacions/PAC1$ scp -P 55000 mm.c capa20@eimtarqso.uoc.edu:
Password:
mm.c
```

Confirmo que el fitxer s'ha copiat correctament a la meva carpeta del servidor:

```
[capa20@eimtarqso ~]$ ls mm.c
mm.c
```

5. Proporcioneu el script SGE que heu fet servir per executar el programa.

Compilem sense optimitzacions i ens crearà el fitxer 'mm'.

```
[capa20@eimtarqso ~]$ gcc -O0 mm.c -o mm
-rwxrwxr-x 1 capa20 capa20 7976 oct  1 21:00 mm
```

Executem l'script 'mm.sge' i el resultat de la sortida es 'Done'.

```
[capa20@eimtarqso ~]$ qsub mm.sge
Your job 417247 ("mm_jobname") has been submitted
```

```
[capa20@eimtarqso ~]$  cat mm.out
Done.
```

Codi 'mm.sge':

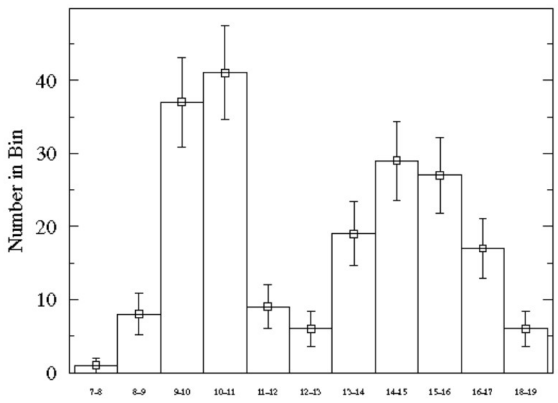
```
[capa20@eimtarqso ~]$ cat mm.sge
#!/bin/bash
```

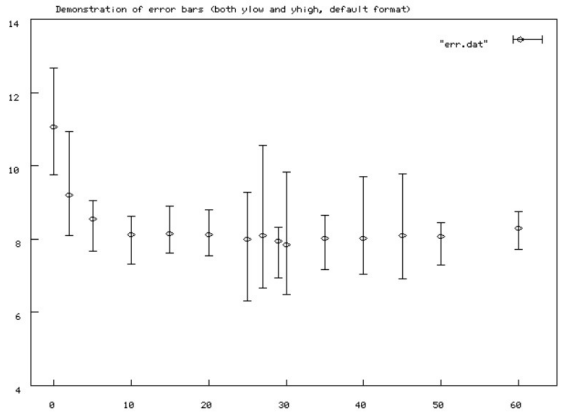
```
#$ -cwd
#$ -S /bin/bash
#$ -N mm_jobname
#$ -o mm.out
#$ -e mm.err
./mm 3
```

A continuació es demana fer un estudi paramètric de la multiplicació de matrius. El que es demana és que proporcioneu el temps d'execució del programa segons mida del problema (mida de les matrius, en aquest cas quadrades).

Quan es realitzen estudis de rendiment pot haver-hi certa variabilitat entre execucions degut a qüestions relacionades amb la contenció de memòria per la compartició de recursos amb altres processos del sistema, per efectes de cache, aleatorietat, etc.

Per tal de mitigar la variabilitat en les execucions, cal fer un estudi de caràcter estadístic on es realitzen varies execucions i es proporcionen mètriques estadístiques com ara la mitjana aritmètica, i els quartils. Podeu veure un parell d'exemple de gràfics utilitzat gnuplot (eina per fer gràfiques de codi obert amb "error bars", però podeu utilitzar la eina que més us agradi.





Com que haureu de realitzar vàries execucions (per exemple 4) per a cada configuració, és convenient que realitzeu aquestes execucions de forma sistemàtica i programada. Us proposem utilitzar scripts per realitzar aquesta tasca (per exemple shell scripts que cridin a qsub per a cadascuna de les configuracions).

Volem realitzar els següents estudis:

- Temps d'execució per a diferents mides de problema sense optimitzacions (-O0) i amb optimitzacions (-O3)
- Feu proves amb mides (paràmetre d'entrada): 100, 500, 1000 i 1500.

PREGUNTES 6, 7 i 8

6. Com ho heu fet per obtenir el temps d'execució de les diverses proves?

He utilitzat la funció clock_gettime() [4] de la llibreria time.h que retorna el temps real del rellotge de sistema. Al inici del programa he dessat el valor a la variable 'start_t' i al final del codi quan s'han fet els càlculs he desat el valor del moment a 'end_t'. Amb els valors del rellotge a l'inici i al final, he calculat la diferencia, tenint en compte l'estructura de les dades, per obtenir el temps en segons que ha trigat el programa a executar-se.

S'ha compilat el codi sense optimitzacions. He hagut d'afegir el paràmetre '-lrt' degut a un error 'undefined reference to `clock_gettime'':

```
[capa20@eimtarqso ~]$ gcc -O0 mm_time.c -o mm_time -lrt
```

He creat l'script següent per executar el programa per N = 100 i després l'he modificat per 250.

```
[capa20@eimtarqso ~]$ cat mm_time.sge
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N mm_time_jobname
#$ -o mm_time_$JOB_ID.out
#$ -e mm_time_$JOB_ID.err
./mm_time 100
```

He executat l'script amb N = 100 i N=250 i s'ha obtingut a la sortida següent:

```
[capa20@eimtarqso ~]$ qsub mm_time.sge
Your job 421943 ("mm_time_jobname") has been submitted
[capa20@eimtarqso ~]$ cat mm_time_421943.out
Mida: 100
Time: 0.015765 segons
Done.

[capa20@eimtarqso ~]$ qsub mm_time.sge
Your job 417311 ("mm_time_jobname") has been submitted
[capa20@eimtarqso ~]$ cat mm_time_417311.out
Mida: 250
Time: 0.271943 segons
Done.
```

He ressaltat en negreta el codi afegit.

Codi 'mm.c':

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main (int argc, char *argv[]) {
  struct timespec start_t, end_t;
  double elapsedSeconds;
  clock_gettime(CLOCK_MONOTONIC, &start_t);
  int     N; //size of columns and rows (matrices)
  int         tid, nthreads, i, j, k;
  double **a, **b, **c;
  double *a_block, *b_block, *c_block;
  double **res;
  double *res_block;

  if(argc<2){
    printf("Usage: mm matrix_size\n");
    exit(-1);
  }

  N = atoi(argv[1]);

  a = (double **) malloc(N*sizeof(double *)); /* matrix a to be multiplied */
  b = (double **) malloc(N*sizeof(double *)); /* matrix b to be multiplied */
  c = (double **) malloc(N*sizeof(double *)); /* result matrix c */

  a_block = (double *) malloc(N*N*sizeof(double)); /* Storage for matrices */
  b_block = (double *) malloc(N*N*sizeof(double));
  c_block = (double *) malloc(N*N*sizeof(double));

  /* Result matrix for the sequential algorithm */
  res = (double **) malloc(N*sizeof(double *));
  res_block = (double *) malloc(N*N*sizeof(double));

  for (i=0; i<N; i++)   /* Initialize pointers to a */
    a[i] = a_block+i*N;
```

```
  for (i=0; i<N; i++)   /* Initialize pointers to b */
    b[i] = b_block+i*N;

  for (i=0; i<N; i++)   /* Initialize pointers to c */
    c[i] = c_block+i*N;

  for (i=0; i<N; i++)   /* Initialize pointers to res */
    res[i] = res_block+i*N;

  for (i=0; i<N; i++) /* last matrix has been initialized */
    for (j=0; j<N; j++)
      a[i][j]= (i+j) * ((double) rand());
  for (i=0; i<N; i++)
    for (j=0; j<N; j++)
      b[i][j]= i * j * ((double) rand());
  for (i=0; i<N; i++)
    for (j=0; j<N; j++)
      c[i][j]= 0.0;

  for (i=0; i<N; i++) {
    for(j=0; j<N; j++) {
      for (k=0; k<N; k++) {
        c[i][j] += a[i][k] * b[k][j];
      }
    }
  }

  clock_gettime(CLOCK_MONOTONIC, &end_t);
     elapsedSeconds = (end_t.tv_sec - start_t.tv_sec) + (end_t.tv_nsec - start_t.tv_nsec) /
1000000000.0;

  printf("Mida: %d \n", N);
  printf("Time: %lf segons \n", elapsedSeconds);

  printf ("Done.\n");
  exit(0);
}
```

7. Proporcioneu els shell scripts (o programa, seqüència de comandes) fets servir per a realitzar les proves de forma sistemàtica.

Com el programa permet rebre per argument amb el valor N, es modificarà el script per tal que es pugui cridar amb una variable a través de la línia de comanda [6].

Script modificat per acceptar variable:

```
[capa20@eimtarqso ~]$ cat mm_time_p7.sge
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N mm_time_jobname
#$ -o mm_time_p7_$JOB_ID.out
#$ -e mm_time_p7_$JOB_ID.err
./mm_time $i
```

Executem la següent comanda per cridar el script per als diferents valors i:

```
[capa20@eimtarqso ~]$ for i in  {100 500 1000 1500}; do qsub mm_time_p7.sge $i; done
Your job 421938 ("mm_time_jobname") has been submitted
Your job 421939 ("mm_time_jobname") has been submitted
Your job 421940 ("mm_time_jobname") has been submitted
Your job 421941 ("mm_time_jobname") has been submitted
```

Resultat:

```
[capa20@eimtarqso ~]$ cat mm_time_p7*.out
Mida: 100
Time: 0.015749 segons
Done.
Mida: 500
```

```
Time: 2.172636 segons
Done.
Mida: 1000
Time: 19.175326 segons
Done.
Mida: 1500
Time: 69.721266 segons
Done.
```

8. Proporcioneu un gràfic de temps d'execució amb diferents mides, amb i sense optimitzacions, i compareu els resultats

Modifico el codi de 'mm_time.c' per tal que el resultat estigui un format que faciliti el seu posterior tractament (mida[integer] i temps[segons]), i torno a compilar sense optimitzacions també amb elles.

```
[capa20@eimtarqso ~]$ gcc -O0 mm_time.c -o mm_time -lrt

[capa20@eimtarqso ~]$ gcc -O3 mm_time.c -o mm_time_opt -lrt
```

Ara tenim els dos executables 'mm_time' sense optimitzacions i 'mm_time_opt' amb optimitzacions.

He creat un dos scripts (mm_time_p8.sge i mm_time_p8_opt.sge) on cadascú crida a l'executable sense optimització i amb optimització.

```
[capa20@eimtarqso ~]$ cat mm_time_p8.sge
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N mm_time_jobname
#$ -o mm_time_p8_$JOB_ID.out
#$ -e mm_time_p8_$JOB_ID.err
for i in  {100 500 1000 1500}; do
        ./mm_time $i
done

[capa20@eimtarqso ~]$ cat mm_time_p8_opt.sge
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N mm_time_jobname
#$ -o mm_time_p8_opt_$JOB_ID.out
#$ -e mm_time_p8_opt_$JOB_ID.err
for i in  {100 500 1000 1500}; do
        ./mm_time_opt $i
done
```

Envio a la cua 10 vegades el script 'mm_time_p8.sge'.

```
[capa20@eimtarqso ~]$ for i in {0..9}; do qsub mm_time_p8.sge $i; done
```

Comprovo que no hi ha cap treball pendent amb qstat -f i agrupo els resultat en un únic fitxer 'mm_time_p8_no_opt.out'.

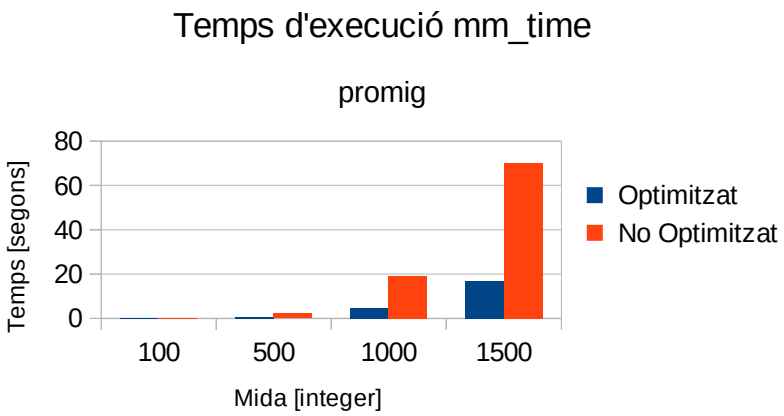
```
[capa20@eimtarqso ~]$ cat mm_time_p8*.out > mm_time_p8_no_opt.out
```

Faig el mateix però executant el script que crida a l'executable amb optimitzacions.

```
[capa20@eimtarqso ~]$ for i in {0..9}; do qsub mm_time_p8_opt.sge $i; done
[capa20@eimtarqso ~]$ cat mm_time_p8_opt*.out > mm_time_p8_opt.out
```

Els dos fitxers amb els resultats, mm_time_p8_no_opt.out i mm_time_p8_opt.out els he utilitzat en una fulla de càlcul per representar gràficament i comparar els resultats.

El temps d'execució es manté durant totes les proves. Es pot veure que a mesura que augmenti la mida, el temps augmenta ràpidament i la diferencia entre l'execució amb optimitzacions i sense elles es fa més evident. Si fem la mitja a través de totes les series, segons la gràfica següent, es veu clar.



PREGUNTES 9 i 10

Sistemes de cues i planificació.

9. Com ho fa el clúster per a poder accedir als fitxers del vostre $HOME des de qualsevol node de còmput del clúster? Quin sistema de fitxer penseu que fa servir el clúster de la UOC?

Segons informació del MIT [7], disposar un fitxer de xarxa com NFS permet compartir la carpeta /home per a tots els nodes del clúster.

Utilitzant comandes com 'df' i 'lsbkl' no mostra el sistema de fitxer del clúster. Llegint el fitxer /etc/fstab indica el sistema de fitxers ext4 o amb la comanda 'blkid' es mostra que utilitze 'ext4', que entenc que es el sistema local i no del clúster, degut a que entenc que hauria d'esser un de tipus compartit com NFS, GPFS o Lustre.

A continuació el resultat de les comandes executades:

```
[capa20@eimtarqso ~]$ lsblk -fm
NAME   FSTYPE LABEL UUID MOUNTPOINT       NAME     SIZE OWNER GROUP MODE
sda                                       sda    465,8G root  disk  brw-rw----
├─sda1                /                   ├─sda1  15,6G root  disk  brw-rw----
├─sda2                /var                ├─sda2   3,9G root  disk  brw-rw----
├─sda3                [SWAP]              ├─sda3  1000M root  disk  brw-rw----
├─sda4                                    ├─sda4     1K root  disk  brw-rw----
└─sda5                /state/partition1   └─sda5 445,3G root  disk  brw-rw----
sr0                                       sr0     1024M root  cdrom brw-rw----


[capa20@eimtarqso ~]$ cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Thu Aug  1 16:25:49 2013
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
```

```
#
UUID=151843ea-90fa-4779-ac0e-5a364f573f17 /                       ext4    defaults        1 1
UUID=26fc084f-aece-4f88-ba83-29953bc0dfb7 /state/partition1       ext4    defaults        1 2
UUID=f82f5ee2-db4c-424d-907d-95a833314e31 /var                    ext4    defaults        1 2
UUID=2887a2eb-487f-43e8-a51c-8e4180c7fc67 swap                    swap    defaults        0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0

# The ram-backed filesystem for ganglia RRD graph databases.
tmpfs /var/lib/ganglia/rrds tmpfs size=2044586000,gid=nobody,uid=nobody,defaults 1 0

[capa20@eimtarqso ~]$ blkid
/dev/sda1: UUID="151843ea-90fa-4779-ac0e-5a364f573f17" TYPE="ext4"
/dev/sda2: UUID="f82f5ee2-db4c-424d-907d-95a833314e31" TYPE="ext4"
/dev/sda3: UUID="2887a2eb-487f-43e8-a51c-8e4180c7fc67" TYPE="swap"
/dev/sda5: UUID="26fc084f-aece-4f88-ba83-29953bc0dfb7" TYPE="ext4"
```

10. Proporcioneu la planificació dels treballs indicats a la següent taula (assumint un sistema amb 8 CPUs) utilitzant les polítiques.

La utilització dels recursos es defineix com:
Utilització = recursos utilitzats (CPUs) / total dels recursos (CPUs) disponibles

Job #	Arrival Time	Runtime	#CPUs
1	1	4	5
2	2	6	2
3	2	2	6
4	2	2	1
5	5	3	6
6	5	3	7
7	6	3	2

A) FCFS

Política FCFS

	TIME																	
CPU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
CPU 1	J1	J1	J1	J1				J3	J3	J5	J5	J5	J6	J6	J6	J7	J7	J7
CPU 2	J1	J1	J1	J1				J3	J3	J5	J5	J5	J6	J6	J6	J7	J7	J7
CPU 3	J1	J1	J1	J1				J3	J3	J5	J5	J5	J6	J6	J6			
CPU 4	J1	J1	J1	J1				J3	J3	J5	J5	J5	J6	J6	J6			
CPU 5	J1	J1	J1	J1				J3	J3	J5	J5	J5	J6	J6	J6			
CPU 6		J2	J2	J2	J2	J2	J2	J3	J3	J5	J5	J5	J6	J6	J6			
CPU 7		J2	J2	J2	J2	J2	J2	J4	J4				J6	J6	J6			
CPU 8																		
Utilització	63%	88%	88%	88%	25%	25%	25%	88%	88%	75%	75%	75%	88%	88%	88%	25%	25%	25%

Utilització	63,19 %

B) EASY-backfilling (de tal forma que backfilling no permeti endarrerir un treball que estava davant en la cua)

Política EASY-backfilling

	TIME																	
CPU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
CPU 1	J1	J1	J1	J1	J3	J3	J5	J5	J5	J6	J6	J6	J7	J7	J7			
CPU 2	J1	J1	J1	J1	J3	J3	J5	J5	J5	J6	J6	J6	J7	J7	J7			
CPU 3	J1	J1	J1	J1	J3	J3	J5	J5	J5	J6	J6	J6						
CPU 4	J1	J1	J1	J1	J3	J3	J5	J5	J5	J6	J6	J6						
CPU 5	J1	J1	J1	J1	J3	J3	J5	J5	J5	J6	J6	J6						
CPU 6		J2	J2	J2	J2	J2	J2			J6	J6	J6						
CPU 7		J2	J2	J2	J2	J2	J2			J6	J6	J6						
CPU 8		J4	J4		J3	J3	J5	J5	J5									
Utilització	63%	100%	100%	88%	100%	100%	100%	75%	75%	88%	88%	88%	25%	25%	25%	0%	0%	0%

Utilització	75,83 %

La tasca nº 7 en cas que el seu temps d'execució sigui de 2 enlloc de 3, no endarreriria l'inici de la tasca 6 i podria haver començat en el temps nº8.

C) Utilització dels recursos (%)

La utilització de recursos es la següent:

Política FCFS → 63,19 %
EASY-backfilling → 75,83 %

Bibliografía

1: Wikipedia, Cyberinfrastructure, , https://en.wikipedia.org/wiki/Cyberinfrastructure
2: Craig A. Stewart, Stephen Simms, Beth Plale, What is Cyberinfrastructure? ,
3: , Open Science Grid, , http://opensciencegrid.org/
4: UNIX, The Single UNIX ® Specification, Version 2, 1997, http://pubs.opengroup.org/onlinepubs/7908799/xsh/clock_gettime.html
5: , , 2017, www.xsede.org
6: Chao Gao, Copy of Tutorial - Submitting a job using qsub - High Performance Computing at NYU - NYU Wikis, , https://wikis.nyu.edu/display/NYUHPC/Copy+of+Tutorial+-+Submitting+a+job+using+qsub
7: , Sun Grid Engine (SGE) QuickStart, , http://star.mit.edu/cluster/docs/0.92rc2/guides/sge.html