
# GUIA RÀPIDA D'ÚS DEL SISTEMA CONDOR

UAB
Universitat Autònoma
de Barcelona

Aquest document pretén donar una introducció breu per utilitzar el sistema Condor i executar-ne treballs. Cal indicar que els exemples inclosos en aquesta guia mostren algunes opcions amb valors que s'ajusten a la configuració del sistema Condor tal i com està configurat a les aules de la UAB. Per a una descripció més completa i detallada el lector pot utilitzar altres documents produïts pel grup Condor de la Universitat de Wiscondin-Madison i que és poden trobar a http://www.cs.wisc.edu/condor. Allà s'hi pot trobar el manual complet i alguns capítols de llibre de caràcter general i introductori.

Condor és un sistema software que crea una entorn de Càlcul d'Alta Productivitat (*High-Throughput Computing*) aprofitant la potència de càlcul d'un conjunt de computadors interconnectats mitjançant una xarxa de comunicació. Condor, de la mateixa forma que altres sistema de cues, disposa d'un mecanisme de cues de treballs, de polítiques de planificació, mecanismes de monitorització i gestió de recursos. Els usuaris envien els seus treballs a Condor, Condor els posa en una cua i decideix quan i on s'executaran segons les necessitats dels treballs i la disponibilitat de recursos, monitoritza el progrés dels treballs i informa a l'usuari de la seva finalització. El funcionament del sistema es basa en una sèrie de conceptes clau:

- En el sistema es diferencien tres tipus de màquina: el gestor central (o Central Manager), les màquines clients i les màquines treballadores. És possible que una màquina sigui client i treballadora alhora. El gestor central, en canvi, acostuma a ser una màquina dedicada només a aquesta tasca.
- Els usuaris envien els seus treballs des de màquines clients, en les que hi tindran els seus programes i els fitxers de dades necessaris. La cua de tots els treballs enviats es manté sempre en aquesta màquina de forma permanent fins a la seva finalització.
- El gestor central s'encarrega de buscar un màquina treballadora adient per a cada treball enviat pels usuaris. El paper del gestor central és fonamentalment de mitjancer entre màquines clients i màquines treballadores. A part d'això, s'encarrega també de mantenir algunes estadístiques generals sobre l'ús del sistema per tal de garantir l'ús equitatiu de recursos entre els diferents usuaris.
- Els treballs i les màquines es descriuen mitjançant un formalisme similar als anuncis classificats dels diaris (*ClassAds*).
- Els treballs que es poden executar sobre un sistema Condor poden ser de diferents tipus (anomenats *universes*). Els universos més importants són: vanilla, standard, java, MPI i PVM.
- *Checkpoint* i migració de treball. Amb determinats tipus de treballs, Condor és capaç d'interrompre la seva execució i continuar-la posteriorment en una altra màquina com si no s'hagués produït cap interrupció temporal.

Condor funciona sobre diferents plataformes de tipus Unix i Windows. Els exemples inclosos en aquesta guia pertanyen a entorns Linux, però en la majoria de casos, són fàcilment transportables a entorns Windows.

# A.- Execució bàsica de treballs

Els passos fonamentals per executar treballs amb Condor són:

**1.- Preparar el treball perquè pugui ser executat de forma no interactiva.**

Els treballs han d'estar preparats de forma que puguin executar-se de forma *batch*, és a dir, les dades d'entrada i els resultats de sortida han d'estar en fitxers perquè quan s'estiguin executant no podran fer operacions d'entrada i sortida interactives. Si l'aplicació utilitza els dispositius d'entrada i sortida estàndard (STDIN, STDOUT i STDERR), aquests també es redirigiran a fitxers.

**2.- Triar un univers Condor per a l'execució del treball**

Condor executa cada treball sota un determinat entorn (o univers). Aquest univers determina la forma com s'executarà el treball i les possibles accions que s'hi podran fer. Els principals universos són: *vanilla* i *standard* (per a aplicacions seqüencials), MPI i PVM (per a aplicacions paral·leles), java (per aplicacions escrites en Java). Per a aplicacions seqüencials, que són les que es contemplen en aquest document, l'univers *vanilla* és el més simple i immediat perquè utilitza el programa de l'usuari sense necessitat de cap modificació. L'univers *standard* ofereix majors prestacions però requereix que l'aplicació sigui enllaçada (*linkada*) amb una llibreria de Condor.

**3.- Preparar un fitxer de descripció del treball**

Qualsevol treball que es vulgui executar en el sistema Condor ha de ser descrit mitjançant un fitxer de text (ASCII pur) on s'indiquen les seves característiques i els seus requeriments.

**4.- Enviar el treball**

Els treballs s'envien a Condor enviant el fitxer de descripció mitjançant la comanda ***condor_submit***.

Exemple 1

Aquest és un fitxer de descripció molt simple per executar un programa que es diu *hello* i que tenim en un subdirectori condor del compte de l'usuari USER_A.

Fitxer de descripció *hello.sub* (el nom pot ser qualsevol).

```
# Inici del fitxer (aquesta línea és un comentari)
Universe   = vanilla
Executable = /home/USER_A/condor/hello
Output     = hello.out
Error      = hello.err
Log        = hello.log
InitialDir = /home/USER_A/condor
Should_transfer_files = YES
When_to_transfer_output = ON_EXIT
Queue
#final del fitxer
```

El resultat de l'execució deixarà tres fitxers: hello.out, hello.err i hello.log dins del subdirectori *condor* (aquest subdirectori de treball ho indica el camp *IntialDir*; l'ús de subdirectoris és una bona pràctica per tenir ordenats els diferents treballs i els seus resultats). En el fitxers hello.out i hello.err hi haurà tot el que el programa hello hagi escrit a l'STDOUT i l'STDERR, respectivament. El fitxer *hello.log* l'omple Condor amb informació relativa a tot el que li va passant al programa *hello* al llarg del temps. L'ordre *Queue* és la que indica que el programa descrit anteriorment sigui enviat a Condor. Abans hi ha dues línies que li indiquen a Condor que sempre ha de moure els fitxers entre la màquina client i la treballadora. Aquestes opcions són innecessàries si es disposa d'un sistema d'arxius compartits, però aquesta no és la situació en què es troben els ordinadors de les aules de la UAB.

Per enviar el treball cal fer :

```
% condor_submit hello.sub
Submitting job(s).
1 job(s) submitted to cluster 1.
```

El treball ha rebut l'identificador 1. Posteriorment, amb l'ordre ***condor_q*** podem conèixer l'estat de tots els nostres treballs.

```
% condor_q
-- Submitter: perdita.cs.wisc.edu : <128.105.165.34:1027> :
 ID      OWNER            SUBMITTED    RUN_TIME ST PRI SIZE CMD
  1.0    USER_A           6/16 06:52   0+00:00:00 I  0   0.0  hello
1 jobs; 1 idle, 0 running, 0 held
```

Cada treball apareix en un línia separada on, entre d'altres informacions, s'indica el seu identificador, l'instant en que es va enviar i el seu estat. Els estats més significatius són *I=idle* (en espera), *R=running* (en execució). És important recordar que, normalment, els treballs no es posen en execució immediatament. Cal esperar que Condor refresqui la seva informació, busqui una màquina adient, mogui tots els fitxers necessaris i finalment arrenqui la nostra aplicació en la màquina remota.

Si després de passats uns quants minuts el treball continua sense posar-se en estat de *running*, això pot indicar alguna mena d'error. Al final del document es comenta una mica més les possibles causes d'error.

Si el treball s'executa correctament en el fitxer de *log* podrem veure una cosa com aquesta:

```
000 (8135.000.000) 05/25 19:10:03 Job submitted from host:
<128.105.146.14:1816>
...
001 (8135.000.000) 05/25 19:12:17 Job executing on host:
<128.105.165.131:1026>
...
005 (8135.000.000) 05/25 19:13:06 Job terminated.
      (1) Normal termination (return value 0)
            Usr 0 00:00:37, Sys 0 00:00:00  -  Run Remote Usage
            Usr 0 00:00:00, Sys 0 00:00:05  -  Run Local Usage
            Usr 0 00:00:37, Sys 0 00:00:00  -  Total Remote Usage
            Usr 0 00:00:00, Sys 0 00:00:05  -  Total Local Usage
      9624  -  Run Bytes Sent By Job
      7146159  -  Run Bytes Received By Job
      9624  -  Total Bytes Sent By Job
      7146159  -  Total Bytes Received By Job
...
```

## 5.- Eliminar un treball

Per eliminar un treball cal utilitzar l'ordre ***condor_rm*** indicant-li l'identificador del treball concret que volem eliminar.

```
% condor_rm 1
Cluster 1 has been market for removal.
```

## 6.- Veure l'estat de totes les màquines del sistema

Per saber en quin estat es troben totes les màquines que poden executar treballs Condor es pot usar la comanda *condor_status*.

```
% condor_status

Name          OpSys       Arch   State      Activity   LoadAv Mem   ActvtyTime

aula01.siee.u LINUX       INTEL  Unclaimed  Idle       0.020   502  0+01:30:04
aula02.siee.u LINUX       INTEL  Unclaimed  Idle       0.000   502  0+01:10:04
aula03.siee.u LINUX       INTEL  Unclaimed  Idle       0.000   502  0+02:15:04
aula04.siee.u LINUX       INTEL  Claimed    Busy       0.040   502  0+00:00:01

                     Machines Owner Claimed Unclaimed Matched Preempting

         INTEL/LINUX        4     0       1         3       0          0

               Total        4     0       1         3       0          0
```

La informació més rellevant mostrada amb aquesta comanda és:

*Name*: són els noms de les màquines.
*OpSys*: el seu sistema operatiu.
*Arch*: l'arquitectura de la CPU.
*State*: estat de la màquina (Unclaimed or Claimed).
*Activity*: activitat que està fent aquell node des del punt de vista de Condor (Idle, desocupada; Owner, algú l'està usant; i Busy, executant un treball Condor).

A les últimes línies hi ha un resum de l'estat de totes les màquines del sistema.

## B.- Execució de diferents tipus de treballs

En aquest punt s'inclouen diferents exemples de fitxers de descripció que il·lustren algunes de les situacions més habituals.

### 1.- Execució de treballs que utilitzen arguments d'entrada i fitxers de dades

El fitxer de descripció pot ser d'aquest estil:

```
# Inici del fitxer (aquesta línea és un comentari)
Universe   = vanilla
Executable = /usr1/USER_A/condor/hello
Output     = hello.out
Error      = hello.err
Log        = hello.log
InitialDir = /usr1/USER_A/condor
Arguments  = -1 -n
Transfer_input _files = fitxer1 fitxer2
Should_transfer_files = YES
When_to_transfer_output = ON_EXIT
Queue
#final del fitxer
```

En aquest exemple s'han afegit camps que indiquen els arguments que cal passar-li al programa en el moment de la seva execució (camp *Arguments*) i els fitxers de dades d'entrada que cal enviar amb el treball (camp *Transfer_input_files*). El camp *Should_transfer_files* amb valor YES indica que Condor sempre mogui els fitxers (això és necessari per a les aules de la UAB però seria innecessari si es disposa d'un sistema d'arxius compartit). Si el treball genera fitxers de sortida, aquests seran enviats de tornada quan el treball acabi (indicat pel camp *When_to_tranfer_output*).

### 2.- Execució de treballs que fan càlculs paramètrics

En aquest exemple es vol executar 10 instàncies d'un mateix programa *hello*. El resultat de cada execució es guardarà en un subdirectori diferent. Els subdirectoris han d'estar prèviament creats i en aquest exemple tenen nom *dir_0*, *dir_1*, ...*dir_10*. L'ordre *Queue 10* és la que indica quantes instàncies s'enviaran. La macro *$(Process)* és la que serveix per "personalitzar" cada instància. Condor fa que *$(Process)* prengui automàticament 10 valors diferents: de 0 a 9. En l'exemple, aquest valor s'utilitza afegit al nom dels subdirectoris i dels fitxers de sortida. Així es diferenciaran els noms del directoris i es crearan fitxers de sortida amb noms diferents (els fitxers *hello.err* i *hello.log* tindran el mateix nom dins de cada subdirectori). El valor de *$(Process)* també se li passa com un argument al programa. Lògicament, si cada execució hagués de fer servir fitxers d'entrada diferents, aquests es podrien posar en els subdirectoris pertinents o usar noms diferents aprofitant la macro *$(Process)*.

```
# Inici del fitxer (aquesta línea és un comentari)
Universe   = vanilla
Executable = hello
Output     = hello.out.$(Process)
Error      = hello.err
Log        = hello.log
Arguments  = -1 –n -$(Process)
InitialDir = dir_$(Process)
Queue 10    #final del fitxer
```

## 4.- Altres arguments útils per a descriure treballs

En el fitxer de descripció del treball es poden especificar força opcions per controlar l'execució dels treballs. La llista completa es pot consultar a la part del manual dedicada a la comanda *condor_submit*. En aquest apartat es comenten uns pocs arguments que poden ser útils en les primeres fases d'aprenentatge del sistema.

A) Especificar Requeriments especials mitjançant el camp *Requirements*

Aquest argument serveix per indicar alguna necessitat particular del nostre treball. Per exemple, posar això al fitxer de descripció

```
Requirements = (machine == condor_2.siee.uab.es)
```

indica que volem executar el treball només a la màquina condor_2.siee.uab.es

Per defecte, Condor acostuma a insertar una quants requeriments per defecte. Es poden veure si fem un *condor_submit –v*. Per l'exemple 1 de la pàgina 5, els requeriments introduïts per Condor serien:

```
Requirements = (OpSys == "LINUX") && (Arch == "INTEL") && (Disk
>=  DiskUsage)  &&  ((Memory  *  1024)  >=  ImageSize)  &&
(HasFileTransfer)
```

Bàsicament, s'indica quin ha de ser el Sistema Operatiu de la màquina que executi el treball, el tipus de CPU, l'espai mínim de disc disponible, la mida de la seva memòria principal i que el dimoni Condor d'aquella màquina tingui capacitat de transferir fitxers remots.

B) Especificar preferències mitjançant el camp *Rank*

Aquest argument serveix per indicar-li a Condor algun criteri perquè intenti executar el nostre treball en alguna màquina que tingui alguna característica concreta. Per exemple, posar això al fitxer de descripció

```
Rank = KFLOPS
```

faria que Condor ordenés les màquines en ordre de preferència d'acord al seu valor en KFLOPS i, per tant, que el nostre treball s'executés en la màquina de major potència de càlcul disponible en aquell moment.

C) Rebre notificacions amb els camps *Notification* i *Notify_user*.

Aquests camps permeten controlar la recepció de missatges que Condor envia per informar del progrés dels treball.

```
Notify_user  = joan.ningu@uab.es
Notification = Error
```

Això faria que Condor enviés un missatge si el treball acabés erròniament (per defecte, Condor envia un missatge quan el treball acaba; això es pot desactivar posant la *notification* a *never*).

## 5.- Execució de treballs amb capacitat d'interrupció/rearrencada

L'univers *vanilla* és el més bàsic perquè permet executar qualsevol tipus de treball. Per a treballs seqüencials Condor disposa d'un altre univers que proporciona funcionalitats superiors: és l'univers *standard*.

Un dels avantatges d'aquest univers són la possibilitat de que l'execució d'un treball sigui suspesa en una màquina i que pugui continuar en una màquina diferent en un instant de temps posterior com si no hagués hagut interrupció. Això es coneix com a *checkpoint*. L'altra avantatge és que el treball no necessita que els seus fitxers d'entrada i sortida es moguin. El treball els utilitza fent un accés remot i, per tant, el treball depèn menys de les limitacions d'espai de la màquina remota on s'executa el treball.

Val a dir que l'opció de *checkpoint* té algunes limitacions i que no totes les aplicacions se'n poden beneficiar. Per exemple, les aplicacions lo poden usar crides *fork()* o *exec()*, no poden tenir fitxers ni comunicacions TCP/IP oberts, etc (al manual hi ha la llista completa de limitacions).

Per utilitzar aquest univers el que cal fer primer és enllaçar la nostra aplicació amb les llibreries de Condor encarregades de fer el *checkpoint* i l'accés remot a fitxers. La comanda per fer això és la ***condor_compile***.

```
condor_compile gcc -O -o my_program my_program.c        (per a programes C)
condor_compile  pgf77  -O  -o  my_program  my_program.f  (per  a  programes
Fortran)
condor_compile ld -o my_program my_program.o       (per a programes objecte)
condor_compile make –f MyMakefile          (per a programes compilats amb make)
```

Un cop fet això, cal crear un fitxer de descripció on canvien alguns dels atributs. L'univers ha de ser *standard* i, en aquest cas no cal especificar que es moguin els fitxers perquè es llegiran i escriuran directament a la màquina des de la que s'envia el treball. Posteriorment, el treball s'envia de la mateixa forma que els treballs que s'han vist anteriorment.

Exemple: per enviar una aplicació *hello.c* en aquest univers, primer l'enllacem:

```
$ condor_compile gcc -O -o hello.std hello.c

LINKING  FOR  CONDOR  :  /usr/bin/ld  -L/usr2/condor/lib  -Bstatic  -m
elf_i386     -dynamic-linker     /lib/ld-linux.so.2     -o     hello.std
/usr2/condor/lib/condor_rt0.o             /usr/lib/gcc-lib/i386-redhat-
linux/2.96/../../../crti.o                /usr/lib/gcc-lib/i386-redhat-
linux/2.96/crtbegin.o    -L/usr2/condor/lib    -L/usr/lib/gcc-lib/i386-
redhat-linux/2.96    -L/usr/lib/gcc-lib/i386-redhat-linux/2.96/../../..
/tmp/ccJwYS1A.o                     /usr2/condor/lib/libcondorzsyscall.a
/usr2/condor/lib/libz.a -lgcc -lc -lnss_files -lnss_dns -lresolv -lc -
lnss_files -lnss_dns -lresolv -lc -lgcc /usr/lib/gcc-lib/i386-redhat-
linux/2.96/crtend.o                       /usr/lib/gcc-lib/i386-redhat-
linux/2.96/../../../crtn.o       /usr2/condor/lib/libcondorc++support.a
/usr2/condor/lib/libcondorzsyscall.a(condor_file_agent.o): In function
`CondorFileAgent::open(char       const       *,       int,       int)':
/scratch/cbuild/build/src/condor_ckpt/condor_file_agent.C:77:  the  use
of     `tmpnam'     is     dangerous,     better     use     `mkstemp'
gcc: file path prefix `/usr2/condor/lib/' never used
```

I el fitxer de descripció seria de la forma:

```
# Inici del fitxer (aquesta línea és un comentari)
Universe   = standard
Executable = hello.std
Output     = hello.out.
Error      = hello.err
Log        = hello.log
Queue
#final del fitxer
```

## C.- Alguns problemes típics. O per què no s'executen les aplicacions?

El mecanisme que utilitza Condor per descriure treballs i màquines és molt potent i expressiu però alhora pot provocar que un treball no s'executi per culpa d'algun camp especificat en el fitxer de descripció.

Cal tenir en compte també que, en general, el sistema no respon immediatament quan se li envia un treball. Els diferents dimonis de Condor estan configurats a la UAB perquè reaccionin i actualitzin la seva informació al cap d'uns quants minuts.

De totes formes, si un treball no es posa en execució passats uns pocs minuts (3 ó 4) i hem comprovat que hi ha màquines disponibles, és possible que hi hagi alguna causa que impedeixi l'execució del treball. Tot i que no hi ha cap regla fixa per tractar aquests problemes, aquí es comenten algunes accions que poden indicar la causa del problemes.

En alguns casos, els treballs sembla que sempre estan en estat d'espera però el que realment passa és que el sistema els intenta executar i fallen en el moment inicial. Condor intenta repetidament posar el procés en marxa, cada vegada falla, però des del punt de vista de l'usuari la comanda condor_q només sembla mostrar que el treball encara espera trobar una màquina. Aquest tipus d'error es pot detectar sobre tot en el fitxer de *log* del treball si se n'ha definit un per al treball.

El següents exemples mostren un parell de casos de treballs que fallaven. Deduir la causa de l'error a partir de la informació del *log* de vegades no és immediata (Exemple 1), però si més no permet apropar-nos-hi.

Exemple 1

```
000   (087.000.000)   10/22   11:19:36   Job   submitted   from   host:
<158.109.66.171:41679>
...
001   (087.000.000)   10/22   11:24:45   Job   executing   on   host:
<158.109.66.171:46716>
...
007 (087.000.000) 10/22 11:24:45 Shadow exception!
        Error from starter on condor-1.siee.uab.es: Failed to execute
'/home/mas
enar/examples/shelljob condor_exec.exe': Permission denied
        0  -  Run Bytes Sent By Job
        0  -  Run Bytes Received By Job
...
```

En aquest cas, l'error es produïa perquè s'estava enviant un treball que consistia en un fitxer de comandes (*shellscript*) en el que a la primera línia hi faltava això #/bin/csh.

Exemple 2

```
000   (088.000.000)   10/22   11:19:36   Job   submitted   from   host:
<158.109.66.171:41679>
...
001   (088.000.000)   10/22   11:24:45   Job   executing   on   host:
<158.109.66.171:46716>
...
007 (018.000.000) 04/04 22:02:09 Shadow exception!
        Error from starter on condor-1.siee.uab.es:
        Failed to open standard input file
        '/home/masenar/exemples/simple/analisis.in':
   No such file or directory (errno 2)
```

Aquest segon error es produïa en un treball en el què accidentalment s'havia eliminat un dels fitxers d'entrada (analisis.in) abans de que el treball es posés a executar.

La comanda *condor_q –ana* també ens pot donar una mica més d'informació del que està passant.

joann@condor-1:~/examples$ condor_q -ana -submitter masenar

```
--  Submitter:  joann@condor-1.siee.uab.es  :  <158.109.66.171:41679>  :
condor-1.siee.uab.es
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
---
087.000:  Run analysis summary.  Of 224 machines,
    222 are rejected by your job's requirements
      2 reject your job because of their own requirements
      0 match, but are serving users with a better priority in the
pool
      0 match, match, but reject the job for unknown reasons
      0 match, but will not currently preempt their existing job
      0 are available to run your job

1 jobs; 1 idle, 0 running, 0 held
```

Si el resultat d'aquesta comanda ens diu que el nostre treball és rebutjat per totes les màquines (les dues primeres categories) i amb la comanda *condor_status* podem comprovar que hi ha màquines disponibles (*idle*) les característiques de les quals s'adiuen amb els requeriments del nostre treball nostre aleshores podem sospitar que alguna cosa va malament (errors en el fitxer de descripció o en la fase inicial d'execució del treball).

*Per a qualsevol comentari sobre errades o omissions en aquest document, envieu un missatge a* *pr.oliba@uab.es*