Introducció a l'AJO

Maig 2014

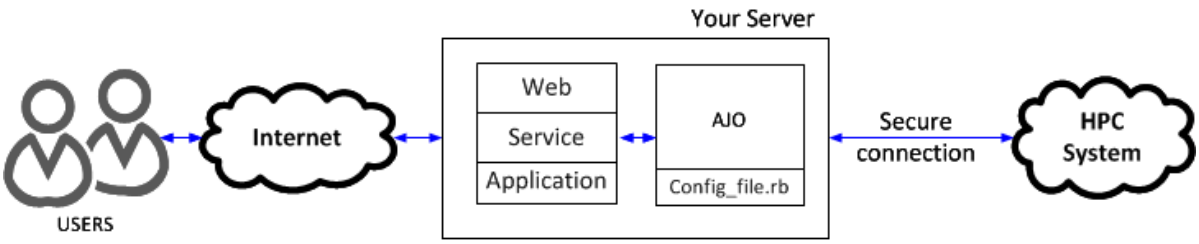
Continguts

1. Introducció 1
   1.1 Funcionalitats d'AJO 2
2. Instal·lació 2
3. Configurant AJO 3
   3.1 Opcions generals 3
   3.2 Grid Engine y opcions de codificació 4
   3.3 Paràmetres de carpeta y fitxer 5
4. Executant i testejant AJO 7
   4.1 Exemples 8

1. Introducció

AJO és un acrònim format per les inicials de Asynchronous Job Operator (operador de treballs asíncron). Aquesta eina ha estat dissenyada i desenvolupada amb l'objectiu de proporcionar una porta d'enllaç transparent entre una web, aplicació o servei i un sistema d'alt rendiment (HPC)

AJO trasllada les execucions a un sistema de cues compatible amb DRMAA (Distributed Resource Management Application API) com la família Grid Engine o TORQUE, permetent l'enviament, execució i recuperació de qualsevol tasca de forma simple i ràpida.



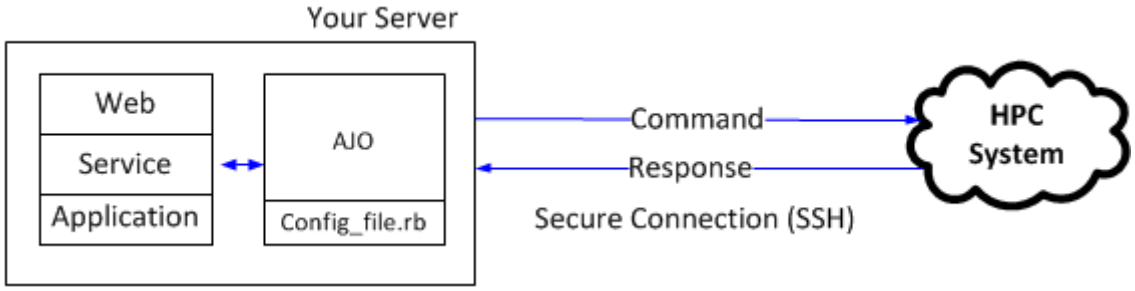
AJO BASIC OPERATION SCHEME

Tota la informació sobre el projecte AJO es troba disponible a http://rdlab.lsi.upc.edu/ajo. Per a més informació contacti amb rdlab@lsi.upc.edu.

1.1 Funcionalitats d'AJO

AJO proporciona tres funcionalitats principals que permeten controlar el procés d'execució completament.

- SUBMIT: Envia un treball al sistema HPC i retorna un TOKEN ID generat de manera segura.
- QUERY: Comprova l'estat d'un treball enviat prèviament.
- RETRIEVE: Obté el resultat d'un treball finalitzat.



A més a més, ofereix algunes funcionalitats extra que completen les anteriors. En el capítol "Executant i testejant AJO" es recullen els detalls per a totes les opcions.

- LIST: Mostra una llista d'identificadors de treballs enviats mitjançant AJO.
- CANCEL: Cancel·la un treball en execució enviat mitjançant AJO.
- ERASE: Esborra la informació al servidor d'un treball enviat mitjançant AJO.

2. Instal·lació

Per executar AJO es requereix un sistema GNU/Linux amb Ruby 1.9.1. o superior instal·lat i un client SSH. També és necessari un sistema de cues compatible en execució, com pot ser un de la família Grid Engine.

La versió més recent d'AJO es troba disponible al nostre repositori SVN emprant "public_ajo" com a usuari i contrasenya:

```
svn co http://svn-rdlab.lsi.upc.edu/subversion/ajo/public --username public_ajo
```

3. Configurant AJO

Atenció: Totes les tasques executades per AJO s'emmagatzemen per defecte en el *home* ($HOME/.executions) de l'usuari del sistema HPC.

3.1 Opcions generals

Al directori de descàrrega d'AJO es troben diversos arxius, incloent-hi un exemple de configuració anomenat *config.rb*. Aquest script conté totes les opcions necessàries per a córrer AJO correctament.

La primera part permet especificar opcions bàsiques com a quin servidor connectar-se o l'usuari a emprar, així com el directori on s'emmagatzemarà la informació al servidor o la ubicació del binari SSH.

```
# SSH options
SERVER = "server.name.domain"
USER = "username" # insert your Linux username between quotes
SSH_CMD = "/usr/bin/ssh #{USER}@#{SERVER}"
AJO_DIR = `#{SSH_CMD} 'echo $HOME'`.chomp("\n") + "/.executions"
```

D'altra banda, AJO envia comandes SSH directament al clúster, pel que és necessari fer servir SSH sense contrasenya mitjançant un parell de claus pública i privada. Es pot trobar més informació sobre l'ús d'SSH sense contrasenya a la xarxa: http://www.debian-administration.org/articles/152

Si AJO s'executarà per un servei del sistema amb el seu propi usuari (per exemple Apache o Tomcat) les claus han de ser al $HOME/.ssh de l'usuari que executa aquests serveis.

A més, si aquest usuari establirà connexió al servidor (*server*) amb més d'un usuari (*user1, user2*), es requereix d'un arxiu *config* al seu $HOME/.ssh. La clau pública es pot concatenar en un sol arxiu (id_rsa.pub), però les claus privades han de trobar-se en arxius diferents, de manera que l'arxiu *config* especificarà quina arxiu correspon a la clau de cada usuari.

A continuació és mostra un exemple de com quedaria l'arxiu, assumint que /var/www és el $HOME d'Apache (www-data):

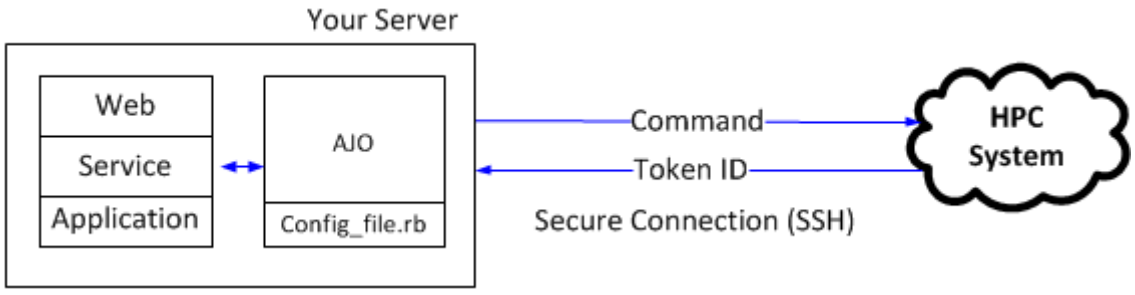
```
root@machine: cat /var/www/.ssh/config
      Host server
      HostName server.domain.com
      IdentityFile ~/.ssh/id_rsa.user1
      User    user1

      Host server
      HostName server.domain.com
      IdentityFile ~/.ssh/id_rsa.user2
      User    user2
```

3.2 Grid Engine y opcions de codificació

En aquesta secció es troben les principals opcions relatives a l'SGE com el *path* als binaris, l'arquitectura, etc. Es recomana no modificar aquesta secció de no ser que sigui estrictament necessari. A continuació es mostra un exemple:

```
# SGE options, normally you will not have to change this.
SGE_ROOT = `#{SSH_CMD} 'echo $SGE_ROOT'`.chomp "\n"
SGE_ARCH = `#{SSH_CMD} '#{SGE_ROOT}/util/arch'`.chomp "\n"
SGE_UTIL_PATH = SGE_ROOT + "/bin/" + SGE_ARCH
QSUB_CMD = SGE_UTIL_PATH + "/qsub"
QSTAT_CMD = SGE_UTIL_PATH + "/qstat"
QACCT_CMD = SGE_UTIL_PATH + "/qacct"
QDEL_CMD = SGE_UTIL_PATH + "/qdel"
```

A més, AJO fa servir un sistema de codificació per crear un *TOKEN ID* segur per a cada enviament de treball per assegurar la fiabilitat entre l'aplicació i el sistema HPC. Aquesta secció conté els paràmetres emprats per a codificar la informació.



Es recomana canviar el valor de la *sal* per quelcom aleatori, així com les constants USER i SERVER en les línies següents per algun valor més segur, com les cadenes "ajksdbh8os7dtfg8wky" o "I5787s4wa%^W/3w6u43", per a una millor codificació. És important fer servir cometes per delimitar les cadenes.

```
# Encryption options, replace USER and SERVER in second and third lines with
# something random for more security. But be careful - loosing CIPHER_KEY
# and CIPHER_IV will make decoding your job identifier impossible.
CIPHER_SALT = "ajo"
CIPHER_KEY = OpenSSL::PKCS5.pbkdf2_hmac_sha1(USER, CIPHER_SALT, 2000, 16)
CIPHER_IV = OpenSSL::PKCS5.pbkdf2_hmac_sha1(SERVER, CIPHER_SALT, 2000, 16)
```

3.3 Paràmetres de carpeta y fitxer

AJO empaqueta tots els fitxers locals i envia una copia al sistema HPC (a $HOME/.execucions per defecte), de manera que els arxius d'entrada i els directoris han de ser definits en l'arxiu de configuració. A més, cal definir també els arxius de sortida on recuperar les dades després de la execució.

A l'arxiu de configuració, després de la secció d'opcions generals, es troben els blocs de paràmetres de carpeta i fitxer. Es tracta de *hash* clau-valor en Ruby. Cal tenir en compte que la clau és un símbol (un tipus especial de variable que comença per dos punts (":")) i que el valor ha de ser una cadena de caràcters. Es possible afegir tants parells clau-valor com sigui necessari. Els parells amb valor buit seran ignorats.

El primer bloc, FOLDER_ARGS, permet especificar els paràmetres relatius a les carpetes que seran utilitzades per la comanda enviada al clúster. És possible utilitzar *paths* relatius i dreceres, com '~'. Les variables Bash de sistema també són acceptades. A continuació es mostra un exemple de l'ús del paràmetre:

```
FOLDER_ARGS = {
    :fld1 => "~/folder42",
    :fld2 => "",
    :fld3 => "",
}
```

El següent paràmetre, FILE_ARGS, permet seleccionar els arxius que seran utilitzats posteriorment en la comanda. El format és el mateix que en el bloc anterior:

```
FILE_ARGS = {
    :file1 => "~/file42",
    :file2 => "",
    :file3 => "",
}
```

El següent bloc mostra les variables de sortida (*output).* FOLDER_OUTPUT i FILE_OUTPUT tenen el mateix format i funció que els paràmetres descrits anteriorment, però aquests estaran disponibles posteriorment per a una fàcil recuperació, ja que serà possible descarregar-los ràpidament un cop la execució del treball hagi finalitzat. El seu aspecte és el següent:

```
FOLDER_OUTPUT = {
     :fld1 => "outputfolder42",
     :fld2 => "",
     :fld3 => "",
}

FILE_OUTPUT = {
     :file1 => "file1.txt",
}
```

El darrer paràmetre és RETRIEVE. Aquest permet especificar les variables de sortida que es desitja recuperar després d'una execució. El paràmetre es defineix de la següent manera:

```
RETRIEVE = {
     :folder1 => "file1.txt",
}
```

Finalment, trobem el bloc on s'especifica la comanda. En aquest s'hi poden escriure les comandes que es desitgen executar al clúster. El format dels paràmetres d'arxiu a emprar aquí és lleugerament complex:

- En primer lloc, les variables configurades en els blocs superiors són accedides en minúscules per al correcte processat dels *path* abans d'enviar el treball.
- En segon lloc, les comandes són cadenes de caràcters. Es recomana no fer servir cometes dobles (") dins de les comandes o bé escapar-les (mitjançant \").

Per usar els arguments prèviament configurats dins de de la comanda és necessari emprar interpolació (*string interpolation).* En Ruby es realitzat escrivint `#{variable}` (coixinet seguit del nom de la variable entre claus).
Per exemple, si es requereix mostrar el contingut de `'file1'` de `'FILE_ARGS'`, és necessari accedir al nom del paràmetre en minúscules, en aquest cas `'file_args'`, i seguidament accedir a l'índex específic en aquest paràmetre amb el nom entre claudàtors.

D'aquesta manera, la comanda quedaria:

```
"cat #{file_args[:file1]}".
```

Qualsevol variable prèviament configurada pot ser usada. A continuació es mostra un exemple de bloc de comanda:

```
def process_commands file_args, folder_args, file_output,
folder_output, input_dir, output_dir
[
     "uname -a > #{file_output[:file1]}",
     "echo 'hello' > #{file_output[:file1]}"
```

```
]
end
```

4. Executant i testejant AJO

AJO s'executa per línia de comandes. Aquestes són les opcions disponibles:

`--api`
: Demana a AJO que imprimeixi només informació pràctica. Útil si la sortida ha de ser analitzada.

`-c FILE or --config FILE`
: Per a cada acció, AJO necessita un arxiu de configuració, ja que utilitza les opcions SSH que conté per a la majoria d'operacions. Per defecte busca un arxiu de configuració de nom config.rb al directori on s'executa a no ser que s'invoqui AJO amb la opció `-c FILE`, on FILE és un arxiu de configuració diferent.

`-d DIR or --retrieve-directory DIR`
: Descarrega el resultat de la execució a DIR.

`--download-exec-folder ID`
: Recupera la carpeta on s'ha executat el treball al sistema HPC.

`-e ID or --erase`
: Esborra el directori associat al treball al servidor.

`-l or –list`
: Llista tots els identificadors disponibles per a consulta i recuperació de dades.

`--library FILE`
: Especifica la ubicació de la llibrería *libhpc*.

`--log-all`
: Força AJO a fer *log* de cada pas que realitza al executar una tasca.

  Per defecte, AJO crea un fitxer de registre (log) al directori on s'executa anomenat AJO.log i escriu en el només els missatges d'error. L'arxiu es sobreescriu a diari. En cas de que hi hagi algun problema amb AJO, es recomana executar la comanda amb el paràmetre --log-all per a obtenir més informació.

`-q ID or --query ID`
: Obté la informació sobre l'estat del treball amb identificador ID.

`--qstat ID`
: Obté la sortida de la crida qstat del treball amb identificador ID.

`--qstat-xml ID`
: Obté la sortida en format XML de la crida qstat del treball amb identificador ID.

`-r ID or --retrieve ID`
: Descarrega els resultats de la execució del treball especificat per ID.

`-s or --submit`
: Envia al clúster el treball especificat el l'arxiu de configuració.

`-v or –version`
: Mostra per pantalla la versió d'AJO i acaba.

`-x ID or --cancel ID`
: Cancel·la un treball en execució.

4.1 Exemples

Les operacions estàndard amb AJO inclouen l'enviament de treballs, la consulta de l'estat, la obtenció de resultats i l'esborrat dels arxius del treball al clúster.

Per enviar un treball és necessari executar la següent comanda:

`AJO -c config.rb -s`
: Aquesta comanda indicarà a AJO que l'arxiu a emprar és config.rb, ubicat al mateix directori des de el que executem, i que desitgem enviar al clúster el treball descrit per aquest. La comanda retornarà un identificador que s'haurà d'utilitzar posteriorment per altres operacions.

Per a consultar l'estat:

`AJO -c config.rb -q identifier_here`
: La comanda retornarà un missatge descrivint l'estat del treball.

Per a obtenir els resultats:

`AJO -c config.rb -r identifier_here`

També és possible especificar, mitjançant la opció `-d directory`, el directori on es desitja emmagatzemar les dades descarregades. A més, és possible demanar la descarrega de tota la sortida mitjançant el *flag* –a.

Per a esborrar del clúster los arxius del treball:

`AJO -c config.rb -e identifier_here`

Per llistar els identificadors del treballs disponibles per a recuperació:

`AJO -c config.rb –l`

Després de configurar AJO es pot, mitjançant aquesta comanda, comprovar que tot funciona correctament. Si no hi ha treballs enviats anteriorment mitjançant AJO, la sortida serà una línia en blanc. Si apareix un error, es recomana comprovar la sintaxi de l'arxiu de configuració (pot ser que manquin comes, punts i coma, etc.), actualitzar les dependències de Ruby, verificar la connexió de xarxa i verificar la connexió SSH.