Ali
Distribution de Tâches sur un Réseau

Guide Administrateur

Thomas Bucher

Août 2004

Introduction

Comme le titre l'indique, Ali est un système permettant de distribuer des tâches au sein d'un réseau. À la base, ce système a été créé pour soulager la charge du serveur Aladin du CDS (Centre de Données Astronomiques de Strasbourg). Au fur et à mesure que le système évoluait, il s'est avéré qu'il pouvait être réexploitable dans bien d'autres situations, moyennant une bonne compréhension de son fonctionnement, et quelques possibles modifications.

Ce document a pour but l'explication du fonctionnement d'Ali, de la façon de le configurer de manière fonctionnelle et optimale, et d'expliquer certaines parties de code, afin de pouvoir les modifier selon ses besoins. Ainsi, ce document est axé sur l'administration d'Ali. Pour son utilisation ou pour plus d'informations sur certains points (tels que la description des tâches), veuillez vous référer au Guide Utilisateur.

1 Structure du système

Ali se compose de 4 entités distinctes, chacune possédant un petit nom personnel :

- le serveur (Sali) qui répond aux requêtes des clients et renvoie les résultats,
- le moniteur (Mali) qui garde à jour une table de l'état des noeuds exécuteurs,
- l'exécuteur (Wali) qui exécute les tâches ordonnées par Sali,
- le client (Cali) qui soumet des requêtes au serveur.

Il va de soi que Wali et Cali possèdent plusieurs instances ; une ferme ne serait pas une ferme si il n'y avait qu'une seule machine (Wali) et il est normal que plusieurs utilisateurs (donc plusieurs Cali) puissent soumettre des tâches simultanément. De plus, le serveur peut aussi exister en plusieurs exemplaires ; ainsi, il est possible de réserver un serveur pour un certain type de tâches ou de clients, ou alors d'installer un serveur sur chaque machine cliente afin de minimiser les transferts réseau, etc.

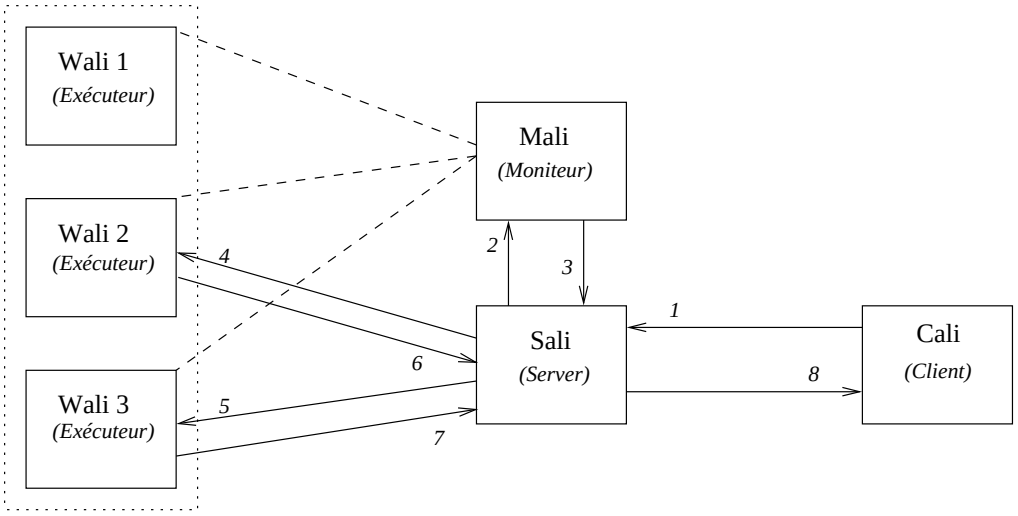

FIG. 1 – Schéma général de l'environnement Ali

La figure 1 représente la configuration générale de l'environnement Ali, ainsi que les interactions basiques :

- 1 : le client envoie sa requête (contenant des tâches) au serveur.
- 2 : le serveur demande au moniteur quels sont les Wali appropriés aux tâches.
- 3 : le moniteur renvoie l'adresse des Wali appropriés.

– 4, 5 : le serveur ordonne aux Wali d'exécuter les tâches.
– 6, 7 : les Wali renvoient les résultats à Sali.
– 8 : si toutes les tâches sont finies, Sali renvoie les résultats à Cali.

Le fonctionnement détaillé de chacune des entités est présenté dans les sections qui suivent.

2 Le serveur (Sali)

2.1 Fonctionnement

De manière générale, le fonctionnement du serveur est simple (voir la Figure 2). C'est un serveur multi-processus (par opposition à un serveur mutli-threadé), c'est à dire qu'il écoute sur un port donné, et pour toute connexion, il crée un processus fils qui va communiquer avec le client. Pour la suite, on appellera aussi ce processus fils *serveur* (alors que c'est plutôt une instance du serveur dédiée à un client).

Pour faire simple, je vais simplement commenter, dans l'ordre, les différentes étapes de l'algorithme du serveur décrit sur la Figure 2.

2.1.1 Initialisation

Cette opération n'est faite qu'une seule fois au lancement du serveur (père), et ne peut ˆêtre réitérée.

Le serveur initialise tout d'abord sa configuration, selon les valeurs fixées dans le fichier de configuration et celles données dans la ligne de commande. Les options de la ligne de commande sont domninantes sur celles du fichier de configuration.

Ensuite, le serveur lit le fichier *$HOME/.ali/sali_auth_clients* pour créer sa liste de clients autorisés. C'est cette liste qui va permettre de refuser ou pas les clients lors de leur connexion.

2.1.2 Attente de connexion - fork

Suite à l'initialisation, le serveur se met en écoute sur le port configuré. Ainsi, pour toute connexion d'un client sur ce port, le serveur se duplique en un processus fils qui va servir le client. Une fois la duplication effectuée, le père incrémente son *compteur de fils* et se remet en attente d'une autre connexion.

La suite de l'algorithme ne concerne que les processus fils. Dès que le fils est créé, l'identité du client est vérifiée. Si celui-ci n'est pas autorisé, le serveur lui envoie un message d'erreur et le déconnecte.

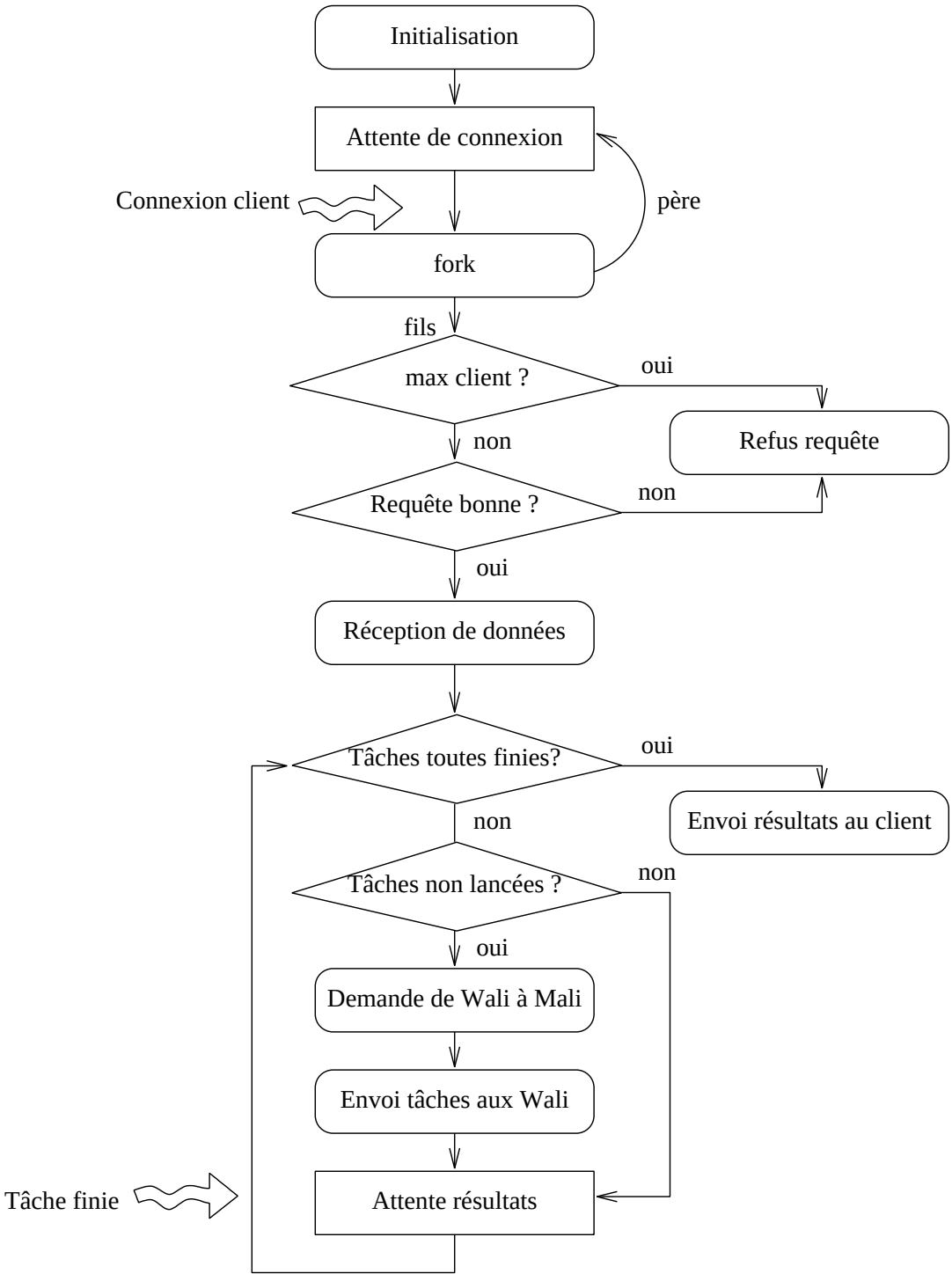

FIG. 2 – Fonctionnement général du serveur

2.1.3 Max client

Avant toute chose, le serveur vérifie si le nombre maximum de clients n'est pas atteint. En effet, pour chaque client connecté, il y a une occurrence du serveur dupliquée. Si un grand nombre de clients se connectent en même temps, la machine serveur pourrait être surchargée. Cette limite est configurable via l'option `maxclient=entier`. Si cette limite est atteinte, le serveur renvoie un message d'erreur au client, et le déconnecte.

Note pour développeur : on peux imaginer de faire évoluer le serveur en ajoutant d'autres tests que le nombre de clients maximum, comme par exemple vérifier la charge totale du cluster. Ainsi, on pourrait proposer au client de différer sa requête, selon certains paramètres de priorité, etc.

2.1.4 Requête bonne

Une fois les vérifications de sécurité faites, le client envoie sa requête. Le format de la requête est décrit dans le Guide Utilisateur (Section 2). Après réception de celle-ci, le serveur l'analyse, et remplit une liste de structures, chacune représentant une tâche à effectuer.

2.1.5 Réception de données

Une fois la requête validée, le serveur crée un répertoire unique et se place dedans. Ensuite, la réception des données peut commencer. En fait, il s'agit simplement de la réception des fichiers d'entrée primaires indiqués dans la requête. Ces fichiers sont stockés sur le disque du serveur, en attendant leur utilisation.

2.1.6 Tâches toutes finies

Tant qu'une tâche n'est pas finie, le serveur attend ou tente de relancer d'autres tâches. En fait, une fois les premières tâches lancées, le serveur se met en écoute d'un port résultats qu'il communique à chaque Wali lors de l'envoi des ordres. C'est sur ce port que les Wali vont se connecter pour renvoyer les résultats. Ce port est propre à chaque instance du serveur (pour chaque job, il y a un port différent).

2.1.7 Tâches non lancées

Si certaines tâches sont non lancées et peuvent l'être (i.e ne sont pas dépendantes d'autres tâches non finies), alors le serveur les lance.

2.1.8 Demande de Wali à Mali

Pour chaque tâche pouvant être lancée, le serveur demande l'exécuteur le plus approprié (pour la tâche) au moniteur. Le moniteur lui renvoie alors l'adresse du Wali approprié, ou bien une valeur nulle si aucun n'est utilisable. Si aucun Wali n'est approprié pour cette tâche, alors son compteur d'essais est incrémenté... si elle a dépassé son quota d'essais, alors le serveur renvoie une erreur à l'utilisateur.

2.1.9 Envoi des tâches aux Wali

Les tâches sont respectivement envoyées à leurs Wali appropriés ; les données envoyées sont : le port de résultat du serveur (celui sur lequel Wali se connecte pour envoyer les résultats), l'identificateur de la tâche, la commande à exécuter, les fichiers en entrée (IFILES), les noms des fichiers en sortie (OFILES) et les noms des fichiers liés (LFILES). En retour, le serveur obtient l'identificateur de l'ordre passé à Wali (en fait, le PID du fils gérant cet ordre du coté de Wali). Cet identificateur permettra à Sali d'ordonner à un Wali de tuer une tâche (en cas de timeout, par exemple).

2.1.10 Attente de résultats

Comme décrit plus haut, le serveur se met en écoute d'un port résultat, en attente de connexion des Wali. Lorsque qu'un Wali se connecte, soit la tâche s'est bien passée, alors Sali réceptionne les OFILES et les stocke sur le disque ; soit la tâche a échoué et Sali incrémente le compteur d'essais de la tâche.

De plus, lors de l'attente de résultats, Sali lance une horloge qui permet de vérifier régulièrement si une tâche lancée n'a pas dépassé son temps limite (*timeout*, fixé par l'utilisateur ou par défaut). La périodicité de cette horloge est réglable via l'option `updatetimer=nb_sec`. Si une tâche dépassé son temps, un ordre est envoyé au Wali en question pour la tuer, et elle est alors relancée sur un autre noeud.

Un autre détail important de cette attente de résultat est que Sali sert aussi de passerelle aux données provenant des fichiers liés. Si un Wali se connecte et envoie un message ayant pour code *MSGCODE_LINK*, alors ce message est directement renvoyé à l'utilisateur servi.

2.1.11 Envoi des résultats au client

Une fois toutes les tâches finies, ou lorsqu'une erreur critique est arrivée, les résultats sont envoyés au client. Dans le premier cas, un code de succès

et tous les FOFILES sont renvoyés au client. Dans le second cas, un code d'échec et un message d'erreur sont renvoyés.

2.2 Options de configuration

Voici les différents paramètres de configuration du serveur. Ils peuvent être mis dans le fichier de configuration, ou directement dans la ligne de commande lançant le serveur. Les options de la ligne de commande sont dominantes sur celles du fichier de configuration.

`sali_port=entier` : port principal sur lequel Sali attend les connexions des clients.
Valeur par défaut : 6060

`mali_name=chaine` : nom réseau du moniteur chargé de donner les Wali appropriés (ex. : *paladin* ou *paladin.u-strasbg.fr*).
Valeur par défaut : machine locale

`mali_port=entier` : port de Mali.
Valeur par défaut : 6061

`wali_port=entier` : port sur lequel Sali contacte les Wali.
Valeur par défaut : 6062

`maxclient=entier` : nombre de clients pouvant se connecter simultanément au serveur.
Valeur par défaut : 10

`updatetimer=entier` : nombre de secondes entre chaque vérification du timeout de chaque tâche lancée.
Valeur par défaut : 20

`maxruntablesize=entier` : nombre de tâches pouvant être lancées simultanément.
Valeur par défaut : 10

`maxtasktries=entier` : nombre de tentatives supplémentaires affecté par défaut au tâches.
Valeur par défaut : 0

`requests_log=fichier` : fichier où les requêtes sont archivées. Seules la date, l'IP du client et la balise %CMD sont archivées dans ce fichier.
Valeur par défaut : *$HOME/ali/sali_req_log*

`auth_clients=fichier` : liste des clients autorisés. Le fichier contient simplement une liste des noms réseau (ex : paladin.u-strasbg.fr) des clients autorisés.
Valeur par défaut : *$HOME/ali/sali_auth_clients*

`user=uid` : uid du processus. Lorsque l'uid est renseigné, le processus prend celui-ci. Cette option permet de restreindre les droits du processus lorsqu'il est lancé au démarrage par root.
Valeur par défaut : -

`daemon=booléen` : mode d'exécution du serveur ; lorqu'il est en mode démon, le serveur reste actif même lorsque le terminal se ferme, par contre les commandes interactives sont désactivées.
Valeur par défaut : true

3 Le moniteur (Mali)

3.1 Fonctionnement

Le moniteur est un simple démon qui met à jour les informations qu'il possède sur les exécuteurs connus. Ces informations lui permettent de déterminer, pour une tâche donnée, quel exécuteur est le plus approprié pour celle-ci ; c'est le service qu'il rend au serveur. Le fonctionnement général du moniteur est décrit sur la Figure 3 et détaillé dans les sous-rubriques suivantes.

3.1.1 Initialisation

Tout comme le serveur, le moniteur commence d'abord par déterminer sa configuration, via le fichier de configuration ou les options de la ligne de commande.

Ensuite, il initialise sa liste d'exécuteurs connus, à partir du fichier *$HOME/.ali/mali_workers*. Ce fichier contient le nom de chaque machine hébergeant un exécuteur.

Enfin, le moniteur se divise en deux : un thread (appelé *Updater*) qui va mettre à jour régulièrement les informations, et le processus maître qui va rendre le service à Sali.

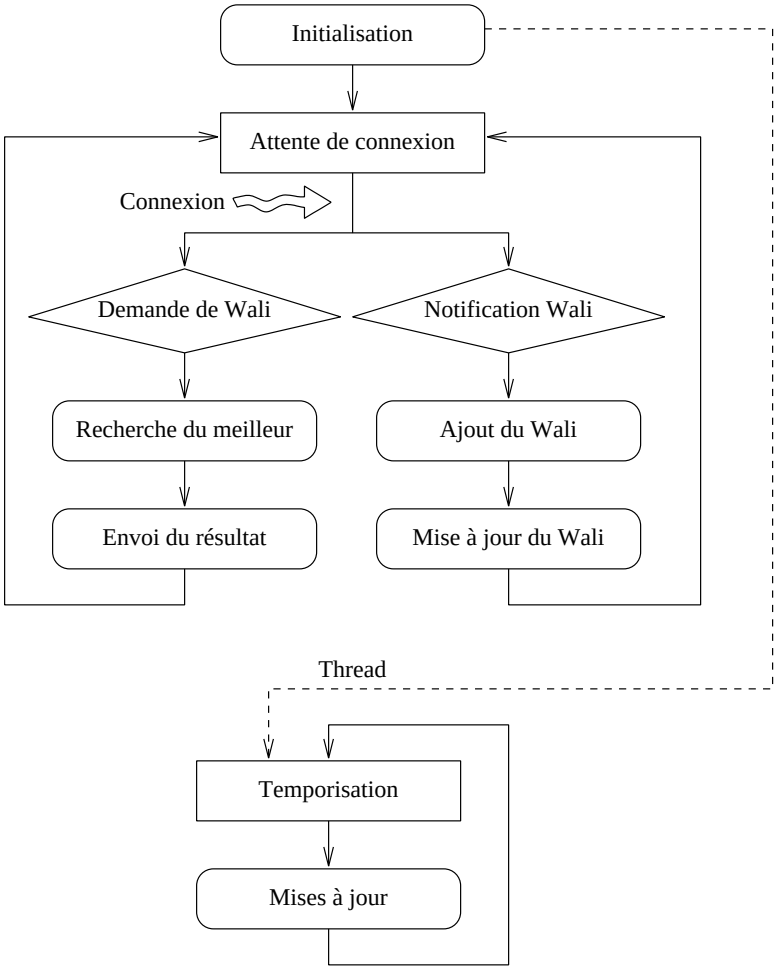

FIG. 3 – Fonctionnement général du moniteur

3.1.2 Attente de connexion

Contrairement au serveur, le moniteur ne se duplique pas pour répondre aux requêtes du serveur. En effet, le traitement de ces demandes s'avère relativement court, c'est pourquoi il n'est pas nécessaire de compliquer inutilement la structure du programme.

Outre le service rendu au serveur, le moniteur propose aussi de renvoyer les informations sur les exécuteurs au format XML (simplifié) pour, par exemple, afficher l'état de la ferme sur une page Web. Actuellement, une version allégée de ce moniteur Web est disponible sur :
http ://paladin.u-strasbg.fr/cgi-bin/workers.pl.

3.1.3 Demande de Wali

Ici, un serveur se connecte pour accéder au service ; il demande au moniteur de lui renvoyer un exécuteur approprié pour une tâche donnée.

En fait, dans l'implémentation actuelle, le serveur est déjà connecté ; il ne se connecte qu'un fois à son initialisation, et garde ainsi une socket ouverte. Ainsi, pour recevoir l'adresse d'un exécuteur, il envoie au moniteur simplement une requête contenant la liste des items requis par la tâche et le coût de celle-ci.

3.1.4 Recherche du meilleur

Tout d'abord, le moniteur élimine tous les exécuteurs ne possédant par les items requis par la tâche en question. Ensuite, il cherche simplement celui dont la charge CPU (*cpuscore*) est la plus faible. Pour l'instant, la charge réseau (*netscore*) n'est pas encore prise en compte dans la décision.

Afin de ne pas reprendre deux fois de suite le même exécuteur, la charge de celui-ci est (virtuellement) augmentée, proportionnellement au coût de la tâche en question.

3.1.5 Envoi du résultat

L'adresse de l'exécuteur choisi est simplement renvoyée au serveur. Cette adresse a une valeur nulle si aucun exécuteur n'a pu être choisi.

3.1.6 Notification d'un Wali

La liste des exécuteurs monitorés n'est pas figée. En effet, il se peut qu'une ou plusieurs machines soient insérées dans l'environnement Ali, alors que celui-ci est en fonctionnement. Au lieu de devoir arrêter le moniteur, modifier son fichier d'exécuteurs et le relancer, la gestion des nouveaux exécuteurs est faite automatiquement.

En effet, dès qu'un nouvel exécuteur se notifie, il est directement ajouté dans la liste des exécuteurs monitorés ainsi que dans le fichier des Wali connus.

D'autre part, certains exécuteurs peuvent être inactifs pendant un certain moment pour une quelconque raison. Étant donné que les exécuteurs inactifs ne sont pas mis à jour à la même fréquence que les actifs, il est intéressant qu'ils se notifient pour être tout de suite pris en compte.

3.1.7 Mise à jour du Wali

Un exécuteur qui se notifie au moniteur ne prend pas l'initiative d'envoyer directement ses informations.

C'est pourquoi le moniteur les lui demande dans un deuxième temps, afin de mettre à jour sa table le plus rapidement possible.

3.1.8 Temporisation

Cet état est celui de l'*Updater* : il dort la plupart du temps. Il se réveille toutes les n secondes pour effectuer les mises à jour des exécuteurs.

3.1.9 Mises à jour

Pour chaque exécuteur présent dans la liste, le moniteur (ici, le thread *Updater*) va faire une requête d'information. Si l'exécuteur ne répond pas, ou qu'il y a un problème de communication quelconque, on le considère comme inactif.

Les exécuteurs inactifs ne sont pas mis à jour à la même fréquence que les actifs. Ceci évite d'encombrer inutilement le réseau avec des paquets destinés à une mort certaine. En fait, les exécuteurs inactifs sautent k mises à jours normales, k étant configurable via l'option `uptoskip=entier`. Au bout de k mises à jour, les exécuteurs inactifs sont de nouveau testés, et ils ont alors l'occasion de repasser en mode actif.

Les exécuteurs actifs renvoient simplement les informations les concernant (charge CPU, nombre de tâches exécutées, etc...). La liste des items qu'ils proposent étant statique, elle n'est transférée qu'une fois ; lors de leur première mise à jour ou juste après une inactivité.

La charge réseau est testée en envoyant un certain nombre d'octets, et en mesurant le temps mis entre son départ et son retour. La taille du paquet est configurable.

3.2 Options de configuration

Voici les quelques options de configuration proposées par le moniteur. De la même manière que le serveur, elles peuvent être présentes dans le fichier de configuration ou sur la ligne de commande (ces dernières étant prédominantes).

`max_workers=entier` : nombre maximum d'exécuteurs monitorés par Mali.
Valeur par défaut : 16

`refresh_rate=entier` : nombre de secondes entre chaque mise à jour. Une petite valeur permet d'avoir des informations plus actuelles sur les exécuteurs, mais il y a un risque de saturation du réseau si beaucoup d'exécuteurs sont monitorés et la taille du paquet écho est grande.
Valeur par défaut : 1

`echo_size=entier` : nombre de Ko composant l'écho qui sert à tester la charge du réseau vers chaque exécuteur.
Valeur par défaut : 1

`workers_file=fichier` : fichier contenant les exécuteurs connus.
Valeur par défaut : *$HOME/.ali/mali_workers*

`uptoskip=entier` : nombre de mises à jour que les exécuteurs inactifs "sautent".
Valeur par défaut : 10

`mali_port=entier` : port sur lequel moniteur écoute.
Valeur par défaut : 6061

`wali_port=entier` : port sur lequel le moniteur contacte les exécuteurs.
Valeur par défaut : 6062

`debug_level=entier` : niveau de verbosité de debuggage.
Valeur par défaut : 0

`item_file=fichier` : le fichier d'items contient la structure connnue du cluster, en termes d'items. Ce fichier ressence tous les chaque item disponible sur le réseau afin de déterminer les décompositions possibles.
Valeur par défaut : *$HOME/ali/conf/mali_struc_items*

`user=uid` : uid du processus. Lorsque l'uid est renseigné, le processus prend celui-ci. Cette option permet de restreindre les droits du processus lorsqu'il est lancé au démarrage par root.
Valeur par défaut : -

`daemon=booléen` : mode d'exécution du moniteur ; lorqu'il est en mode démon, le serveur reste actif même lorsque le terminal se ferme, par contre les commandes interactives sont désactivées.

Valeur par défaut : true

4 L'exécuteur (Wali)

4.1 Fonctionnement

Globalement, un exécuteur n'est qu'un démon qui exécute une commande qu'on lui envoie. Ceci sous certaines conditions, et avec un minimum d'encapsulation. Comme précédemment, le fonctionnement général est décrit par un schéma (Figure 4), et sera détaillé dans les sous-sections suivantes.

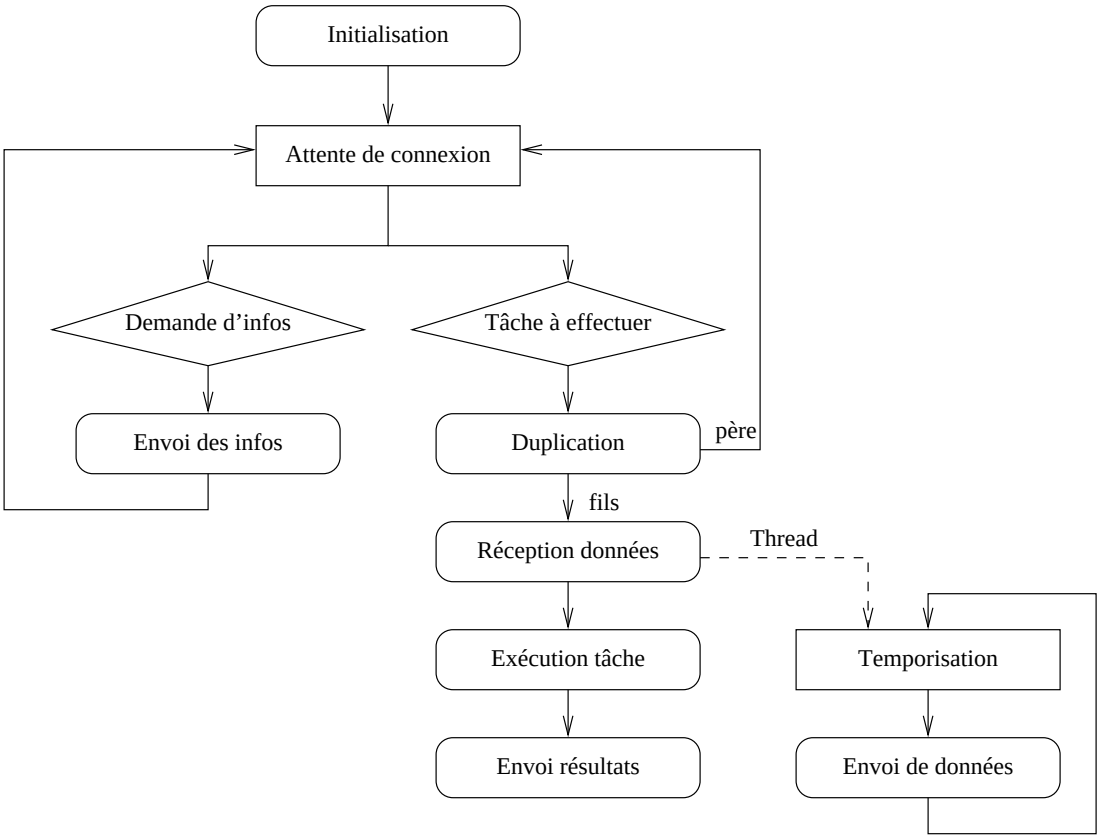

Fig. 4 – Fonctionnement général d'un exécuteur

4.1.1 Initialisation

Tout comme le serveur et le moniteur, l'exécuteur se configure via le fichier de configuration et les paramètres passés sur la ligne de commande.

Ensuite, à l'instar du serveur, l'exécuteur établit une liste de clients (ici, Sali) autorisés. Cette restriction permet d'éviter l'intrusion de serveurs non fiables dans l'environnement Ali.

4.1.2 Attente de connexion

Deux types de connexion peuvent arriver sur le port d'écoute de l'exécuteur : une demande d'information provenant du moniteur, et un ordre d'exécution de tâche provenant du serveur.

4.1.3 Demande d'information

En fait, le moniteur ne se connecte qu'une seule fois à l'exécuteur, et garde une socket ouverte entre l'exécuteur et lui tant qu'il est actif.

Ainsi, il n'envoie qu'un simple message, demandant les informations d'état de l'exécuteur.

L'exécuteur ne se duplique pas pour répondre aux demandes du moniteur.

4.1.4 Envoi des informations

L'exécuteur tient à jour une variable qui correspond à la charge courante de la machine. Cette charge est calculée par un thread, qui mesure régulièrement le temps mis pour effectuer une boucle un certain nombre de fois ; cette boucle étant non prioritaire par rapport aux autres programmes, elle ne surcharge pas inutilement la machine.

4.1.5 Tâche à effectuer

Ici, un serveur se connecte pour ordonner à l'exécuteur d'effectuer une tâche. Ce serveur est accepté seulement s'il se trouve dans la liste des clients autorisés de l'exécuteur.

4.1.6 Duplication

Afin de d'effectuer la tâche demandée, l'exécuteur se duplique. Le père reprend son écoute du port, et le fils est chargé de gérer la tâche. De plus, le fils se place dans un répertoire temporaire unique, afin de ne pas interférer avec les autres tâches en cours.

4.1.7 Réception des données

L'exécuteur (fils) commence alors à recevoir les différentes informations sur la tâche, dont la plus importante : la commande à lancer. Il reçoit aussi les fichiers d'entrée.

Dès que la réception est finie, et si la tâche nécessite des fichiers liés, alors l'exécuteur lance un thread (le *Linker*) qui sera chargé d'envoyer les nouvelles données des fichiers liés au serveur, au fur et à mesure de leur évolution.

4.1.8 Exécution de la tâche

L'exécuteur se duplique encore une fois, et lance la commande donnée dans un Bourne-Shell (sh). Cette duplication est nécéssaire, car tout le code du processus est remplacé par celui de la commande (voir les sources plus plus de detail).

L'application est terminée avec succès si sa fin est normale (c'est à dire non interrompue) et sa valeur de retour égale à *EXIT_SUCCESS*.

4.1.9 Envoi des résultats

Soit la tâche s'est terminée avec succès, et dans ce cas le code *JOB_SUCCESS* ainsi que les fichiers de sortie sont renvoyés, soit elle a échoué et seul le code *JOB_FAILURE* est renvoyé.

4.1.10 Temporisation

Une temporisation cadence le thread *Linker*. En effet, si celui-ci devait renvoyer des données dès qu'un fichier est mis à jour (écriture d'un caractère, par exemple), alors il risquerait de surcharger inutilement le réseau.

C'est pourquoi, le *Linker* ne vérifie que toutes les secondes si un fichier liés a été mis à jour, et si c'est le cas envoie les nouvelles données au serveur.

Le *Linker* se termine lorsque l'exécuteur positionne un drapeau. Alors, le *Linker* est assuré que tous fichiers liés sont complets, vérifie si de nouvelles données sont présentes, si oui les envoie, et enfin termine son exécution.

4.1.11 Envoi de données

L'envoi des nouvelles données d'un fichier lié se fait grâce à un message contenant deux choses : l'identifiant du fichier (son nom), et les données. Ainsi, à réception de ce message, le serveur n'a qu'à le retourner à son client. C'est le client qui est chargé de mettre à jour son fichier lié à partir des données du message.

4.2 Options de configuration

Voici les options permettant de configurer l'exécuteur :

`wali_port=entier` : port d'écoute de l'exécuteur.
Valeur par défaut : 6062

`mali_name=chaine` : nom réseau du moniteur auquel l'exécuteur se notifie.
Valeur par défaut : *paladin.u-strasbg.fr*

`mali_port=entier` : port d'écoute du moniteur auquel l'exécuteur se notifie.
Valeur par défaut : 6061

`item_file=fichier` : nom du fichier des items proposés par l'exécuteur.
Valeur par défaut : *$HOME/.ali/wali_avail_items*

`auth_file=fichier` : nom du fichier des clients autorisés.
Valeur par défaut : *$HOME/.ali/wali_auth_clients*

`nb_cpu=entier` : nombre de processeurs de la machine où tourne l'exécuteur. Cette valeur est utilisée par le moniteur pour son heuristique d'augmentation de charge.
Valeur par défaut : 1

`debug_level=entier` : niveau de verbosité de debuggage.
Valeur par défaut : 0

`user=uid` : uid du processus. Lorsque l'uid est renseigné, le processus prend celui-ci. Cette option permet de restreindre les droits du processus lorsqu'il est lancé au démarrage par root.
Valeur par défaut : -

`daemon=booléen` : mode d'exécution du worker ; lorqu'il est en mode démon, le serveur reste actif même lorsque le terminal se ferme, par contre les commandes interactives sont désactivées.
Valeur par défaut : true

5 Installation

L'installation de l'environnement Ali est des plus simples. Il suffit de se munir de la dernière version du paquetage Ali, par exemple : *ali-1.00.tar.gz*. Tout d'abord, il faut le préparer :

```
% gunzip ali-1.00.tar.gz
% tar -xvf ali-1.00.tar
% cd ali-1.00
```

Ensuite, il vous suffit alors d'effectuer les quatre opérations suivantes (dans la racine du paquetage) :

```
% ./configure
% make
% su
% make install
```

Le script *configure* permet de pré-configurer les sources afin que la compilation soit la plus compatible avec la machine. De plus, il initialise les fichiers de configurations de manière adaptée. Le nombre de processeurs, par exemple, est déterminé que les machines sous Linux, et palcé automatiquement dans le fichier de configuration du Worker ; par contre, sous Unix ceci ne peut être déterminé simplement, et le nombre de processeurs indiqué est alors pris par défaut, c'est-à-dire 1.

Les programmes *server*, *monitor*, *worker* et *submitter* sont ensuite installés sur votre machine (dans *$HOME/ali/bin* par défaut).

Les fichiers de configuration sont placés par défaut dans le répertoire *$HOME/ali/conf*. Dans le cas du worker, la liste des items disponibles est générée automatiquement par le script *conf_items.sh* qui tente de détecter toutes les applications connues afin de déterminer les items disponibles.