# Gestion de serveurs de calcul avec *PowerShell*

Christophe Bontemps[1] et Olivier de Mouzon[1]

**Résumé.** Nous présentons ici deux programmes *PowerShell* destinés à gérer les ressources de serveurs de calcul dédiés à une communauté de 250 chercheurs, enseignants-chercheurs et doctorants. Sur chaque serveur, le premier programme, propose une régulation active en appliquant des fermetures de programmes et de sessions suivant des règles établies. Ce programme envoie également des informations (boîtes de dialogue *pop-up* ou courriel) lorsqu'une action est réalisée. Un second programme permet à chaque utilisateur d'obtenir des informations sur l'état de la mémoire sur la session en cours et sur les ressources disponibles. Ce système est en place depuis avril 2012 sur les quatre serveurs de L'UMR Gremaq de Toulouse, gérés par le Pôle Informatique du Gremaq (PIG).

**Mots-clés :** serveurs, calcul, régulation, mémoire, MS-Windows, *Powershell*

## Introduction

Nos serveurs de calcul fonctionnent sous Windows à la fois pour des raisons historiques (culture commune des chercheurs), pragmatiques (les compétences des informaticiens sont sous Windows) et d'utilisation de logiciels statistiques (certains logiciels statistiques, comme Eviews, ne fonctionnent que sous Windows). Ce choix n'ayant jamais été vraiment remis en cause, nous avons été amenés à réfléchir au problème de la gestion d'une ressource partagée sur un ou plusieurs serveurs fonctionnant donc sous Windows (Windows server 2008 pour la plupart). L'un des problèmes majeurs de ce partage de ressources concerne l'utilisation et la répartition de la mémoire (RAM) de chacun de ces serveurs. Les autres ressources (processeurs, disque) ont été jugées moins "critiques" puisqu'une surutilisation ou un "mauvais partage" n'entraînent pas forcément de défaillance du serveur, celui-ci étant simplement ralenti (trop de cœurs sollicités) ou moins efficace (utilisation de disques réseau au lieu des disques du serveur). Les questions de la répartition du temps d'utilisation, du nombre d'utilisateurs connectés simultanément et des règles d'accès à ces serveurs ont donc été abordées comme des contraintes propres résultant des contraintes de répartition de la mémoire. La prise en compte de ces contraintes a pour objectif d'assurer une pérennité du service rendu et de garantir la disponibilité des ressources pour tous sans altérer le système et sans qu'aucun crash du système lié à une utilisation excessive de la mémoire ne soit possible. En outre, notre approche devait permettre à tous de se connecter aux serveurs, et de proposer différents services attractifs.

Nous avons donc conçu deux programmes en utilisant *Powershell* nous permettant, d'une part, de réguler les ressources mémoire de chaque serveur, et, d'autre part, d'informer les usagers de leur utilisation du serveur et des ressources disponibles.

[1] *GREMAQ-INRA, UMR 1291, Manufacture des tabacs, 21 allée de Brienne, F-31015 Toulouse cedex 6, (Christophe.bontemps@toulouse.inra.fr et Olivier.deMouzon@toulouse.inra.fr).*

## Choix de *Powershell*

*PowerShell* est un interpréteur de commande et un langage de script permettant d'utiliser les "objets" de l'environnement Windows. Il est présent, ou proposé en mise à jour facultative, sur toutes les versions de Windows sorties depuis fin 2006. C'est en quelque sorte le successeur de l'outil MS-Dos *command.com* dans une version nettement plus complète et proposant une programmation orientée objet. C'est donc le candidat idéal comme langage de script complètement intégré proposant des fonctionnalités similaires à celles existant sous Unix (et Linux).

## 1. Définition des règles d'utilisation des serveurs et conséquences pour les utilisateurs

### 1.1 Principes

Il n'est pas facile de définir ce que peut ou ce que doit être une règle « juste et équitable » pour l'accès à une ressource partagée. Cette répartition, ce partage dépend fortement des objectifs visés par la mise à disposition de cette ressource et de la politique scientifique ou opérationnelle proposée. On pourrait par exemple proposer une allocation de la ressource au cas par cas en fonction des projets de recherche des uns et des autres, ou, au contraire, proposer à chacun un quota strict de temps de calcul et de mémoire, ou bien laisser la ressource en libre-service sur le principe du « premier arrivé, premier servi » en ne gérant que l'intégrité du système, ou encore mettre aux enchères entre les utilisateurs cette ressource. Chacun de ces principes ayant des avantages (simplicité, équité, etc.) et des inconvénients (complexité, arbitraire, ressources inutilisées, etc.).

### 1.2 Règles

Dans notre unité, il a été clairement défini que la régulation des serveurs devait exploiter au mieux les ressources disponibles, proposer une offre de calcul attractive, disponible à tous et permettre un ajustement à la demande (on est chez des économistes quand même !). Le tout devant reposer sur des règles claires connues de tous et compréhensibles par tous[2]. Ceci n'exclue évidemment pas des actions (qui sont toujours associées à un retour informatif à l'utilisateur concerné) de fermeture de programmes ou de session si les ressources venaient à manquer et si le système devenait instable. Ces règles, qui sont au nombre de quatre, suffisent à définir les principes opératoires que nous avons mis en œuvre dans notre algorithme de gestion des serveurs :

- Tout membre de l'unité peut se connecter sur les serveurs pourvu qu'il ait un compte répertorié sur le système de l'unité[3].

- Une fois connecté au serveur, chaque utilisateur a accès à :
  - un quota minimum de mémoire (***MemG***) et
  - une période de temps de calcul garantie (***TimeG***).

- Si le système est saturé et ne permet pas à un nouvel utilisateur de se connecter au serveur et de bénéficier des ressources minimales garanties, un **accès dégradé** est proposé.

---

[2] *Des actions de communication (les « PIG sessions ») ont été proposées auprès des chercheurs avant la mise en place du système.*

[3] *Chaque utilisateur du système informatique a un compte utilisateur (stocké dans l'Administrative Directory (AD)).*

- Si le système devient instable (parce que le total de la mémoire demandée s'approche du total de la mémoire physiquement disponible), des **actions de régulation automatique** sont effectuées sur les utilisateurs excédant le quota garanti de temps de calcul ou excédant leur quota de mémoire garantie.

Il est à noter que rien dans ces règles n'interdit de dépasser, même de beaucoup, son quota et que la régulation n'intervient que si les ressources viennent à manquer pour un nouvel utilisateur. On autorise ainsi l'utilisation de toutes les ressources du serveur si la demande est faible, tandis que chaque utilisateur sera fortement contraint si, à l'inverse, la demande de ressources est forte et le nombre d'utilisateurs élevé.

### 1.3 Actions

Les actions de fermeture forcée de programmes s'appliqueront dans un ordre précis à chaque utilisateur[4] en fonction de son utilisation réelle de la mémoire du système (au-delà ou en-deçà de la mémoire garantie *MemG*) et de la durée réelle d'utilisation du système (session durant plus longtemps ou moins longtemps que la durée garantie *TimeG*). Plus précisément, les utilisateurs sont dans l'un des états décrits dans le **Tableau 1** ci-dessous et illustrés par les icônes[5] qui seront visibles sur le serveur (voir section 2).

**Tableau 1.** *Etats des utilisateurs au regard du temps de connexion et de l'utilisation de la mémoire (illustrés par les icônes du programme d'information*

| | | **Mémoire réelle utilisée par l'utilisateur** | |
|---|---|---|---|
| | | *< MemG* | *> MemG* |
| **Durée de la session de l'utilisateur** | *> TimeG* | **Etat C**  | **Etat A**  |
| | *< TimeG* | **Etats D-E**  **(D) (E)** | **Etat B**  **($B_D$) ($B_E$)** |

Lorsque le système devient instable, et afin de récupérer de la mémoire pour un utilisateur supplémentaire, des actions sont entreprises et une information (boîte *pop-up* ou courriel) est envoyée à l'utilisateur. L'ordre des actions entreprises automatiquement est le suivant :

[4] *L'utilisateur "root" sera exempté de ces règles afin de garantir aux administrateurs du serveur de pouvoir intervenir à tout moment.*

[5] *Les icônes en forme de petits cochons (le Pôle Informatique du Gremaq s'appelant le PIG…) adoptent un code couleur et d'autres repères visuels pour savoir immédiatement ce qui pourrait poser problème à l'utilisateur. Ainsi, quand tout va bien, l'icône est verte. Quand le temps est dépassé, le cochon passe en jaune, avec 2 aiguilles rouges qui rappellent une horloge. Quand la mémoire est dépassée, le cochon passe en bleu avec le haut du crâne rouge (pour schématiser un mal de tête). Quand temps et mémoire sont dépassés, le cochon est rouge et ressemble à une tête de mort, car c'est le pire cas du point de vue du risque d'avoir un programme arrêté par le système de régulation du serveur ; on retrouve le mal de tête (de couleur violette = bleu + rouge) et les aiguilles de l'horloge (de couleur orange = jaune + rouge).*

- On clôturera le programme (*process*) le plus gourmand en mémoire de l'utilisateur connecté depuis le plus longtemps se trouvant dans l'état **A**. On répète ce processus jusqu'à avoir récupéré suffisamment de mémoire. Lorsqu'il n'y a plus de personnes dans l'état **A**, et s'il faut encore récupérer de la mémoire, on passe à l'action suivante.

- On clôturera le programme le plus gourmand en mémoire de l'utilisateur dont l'ensemble des programmes mobilise le plus de mémoire parmi les utilisateurs dans l'état **B**. On répète ce processus jusqu'à avoir récupéré suffisamment de mémoire, et s'il faut encore récupérer de la mémoire, on passe à l'action suivante.

- On clôturera le programme[6] le plus gourmand en mémoire de l'utilisateur connecté depuis le plus longtemps se trouvant dans l'état **C**. La session pourra être fermée si plus aucun programme ne tourne sur cette session. On répète ce processus jusqu'à avoir récupéré suffisamment de mémoire, et s'il faut encore récupérer de la mémoire, on passe à la dernière action (cf. état **E**, ci-après).

- Aucune action ne sera effectuée sur les programmes des utilisateurs se trouvant dans l'état **D**, qui est le cadre garanti.

- Enfin, les personnes ayant eu la possibilité de se connecter en mode dégradé se trouvent dans l'état **E**. S'il faut encore libérer de la mémoire, on supprimera le programme le plus gourmand de l'utilisateur le plus récent[7] dans l'état **E**. La session pourra être fermée si plus aucun programme ne tourne sur cette session.

On remarquera qu'aucune de ces actions n'est entreprise s'il n'y a pas rareté de la ressource et/ou instabilité potentielle du système. En effet, d'un point de vue du système, qu'il y ait trois très gros programmes ou 70 petits, aucune action ne sera réalisée si la mémoire disponible, qui est LE facteur limitant ici, reste suffisamment importante pour accueillir un nouvel utilisateur. L'offre de calcul s'adapte donc aux besoins réels, ce qui permet d'utiliser le plus possible les ressources disponibles et de contraindre le moins possible les utilisateurs.

### 1.4 Cycle des états de l'utilisateur

La **Figure 1** présente le cycle des états dans lequel l'utilisateur peut se trouver[8]. Les flèches pointillées représentent les actions du programme de régulation qui peuvent généralement amener à un changement d'état. Elles sont vertes pour l'attribution de ressources et rouges pour la récupération de ressources. L'action irréversible du temps est donnée par les flèches pleines jaunes unidirectionnelles. Les actions de l'utilisateur sur la mobilisation ou la libération de mémoire pour ses programmes sont représentées par des flèches bleues bidirectionnelles.

Ainsi, lorsque l'utilisateur se connecte, il se trouve généralement dans l'état **D** (exceptionnellement en **E**, mode dégradé, lorsque des ressources sont disponibles, mais

---

[6] *Les processus du système nécessaires au bon fonctionnement de la session ne sont pas considérés comme programmes (plus de détails sur la note de bas de page n°18, section 3.1).*

[7] *En réalité, les utilisateurs connectés depuis moins d'une minute ne sont pas concernés, afin qu'ils aient au moins le temps de visualiser le message d'accueil les informant de leur accès en mode dégradé et du faible espace mémoire disponible.*

[8] *L'utilisateur connecté pourra rapidement connaître l'état dans lequel il se trouve en regardant, dans la barre des tâches de sa session, l'icône correspondant à son état.*

peuvent être exigées par d'autres utilisateurs ; dans ce cas, la régulation peut lui fermer des programmes, voire sa session, s'il n'a plus de programme en cours).
Si l'utilisateur mobilise plus de mémoire que ce qui lui est habituellement garanti, il passe dans l'état **B** (on distingue donc un état $\mathbf{B_D}$, pour l'utilisateur en mode standard, et un état $\mathbf{B_E}$, pour l'utilisateur en mode dégradé). Si le programme de régulation doit fermer un programme, l'utilisateur[9] peut rester dans l'état **B** (s'il lui reste d'autres programmes qui au total mobilisent plus de mémoire que la garantie habituelle), ou revenir dans l'état initial (**D** ou **E**).
Un utilisateur initialement connecté en mode dégradé peut passer en mode standard si le programme de régulation a la possibilité de lui garantir sa mémoire habituelle (passage des états **E** à **D** ou $\mathbf{B_E}$ à $\mathbf{B_D}$). Ceci est possible quand d'autres utilisateurs se déconnectent du serveur ou dépassent leur temps garanti. Dans ce cas, l'utilisateur qui passe en mode standard a sa mémoire habituelle garantie tant que la durée de sa connexion n'excède pas le temps garanti (même si une partie de cette période était en mode dégradé).
Lorsque le temps habituellement garanti est terminé, l'utilisateur change d'état (flèches jaunes : de **D** ou **E** en **C**, ou encore de **B** en **A**). Il n'y a alors plus aucune distinction entre utilisateur connecté en mode standard ou dégradé : cet utilisateur peut avoir ses programmes, voire sa session, fermés par la régulation, si cela était nécessaire pour le maintien des performances du serveur. Une fois révolu le temps garanti, la seule distinction concerne la mémoire mobilisée : en-dessous de la garantie habituelle (état **C**) ou au-delà (état **A**).
Comme le programme de régulation priorise les actions à effectuer, l'utilisateur dans l'état **A** peut voir un ou plusieurs de ses programmes fermés par la régulation, jusqu'à arriver dans l'état **C**. Si le programme de régulation a épuisé les actions sur les utilisateurs en états **A** et **B**, l'utilisateur en état **C** peut également voir ses programmes fermés, voire sa session (mais uniquement s'il n'a plus de programme en cours).

Pour terminer, l'utilisateur qui souhaite minimiser les fermetures de ses programmes (par le programme de régulation) prendra soin de ne pas utiliser de mémoire au-delà de sa garantie habituelle et libèrera aussi rapidement que possible la mémoire qui serait utilisée ponctuellement au-delà, afin de revenir très vite dans les états **D-E** ou **C.** Seul l'état **D** donne une immunité par rapport au programme de régulation.
La seule possibilité pour l'utilisateur en état C de revenir à l'état **D** est de se déconnecter et de se reconnecter. Cependant, la régulation peut lui attribuer l'état **E** lors de la reconnexion.
L'utilisateur dans l'état **E** ne peut migrer vers l'état **D** que si des utilisateurs quittent les états **D** ou $\mathbf{B_D}$ (il faut donc que ces utilisateurs suivent une transition jaune, action du temps, ou se déconnectent d'eux-mêmes avant l'expiration de leur temps garanti).

[9] *Rappelons que, pour les actions sur l'état B, le programme de régulation choisit l'utilisateur le plus gourmand, indifféremment qu'il soit en mode standard ou dégradé.*

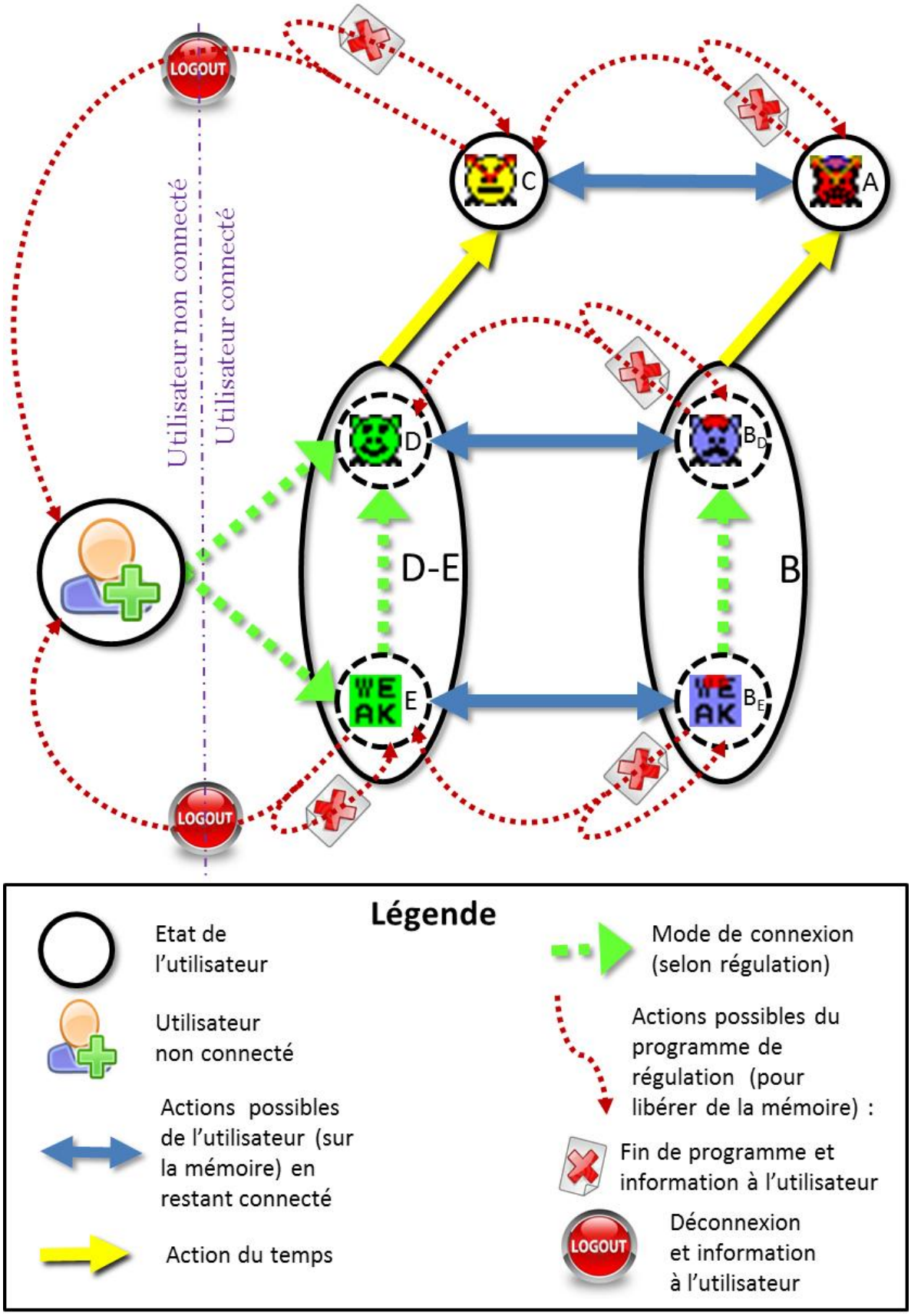


**Figure 1.** *Cycle des états d'un utilisateur connecté sur un serveur*

## 2. Mise en œuvre et illustration

Notre UMR possède sept serveurs (cf. **Annexe 2**) dont quatre sont ouverts à l'ensemble d'une communauté de 250 chercheurs, enseignants-chercheurs, ingénieurs et doctorants de toutes nationalités[10]. Chaque serveur est doté d'une configuration spécifique et possède des caractéristiques propres en termes de mémoire, nombre et type de cœurs, taille et vitesse de rotation des disques durs, sans parler des spécificités d'architecture interne (bus, chipsets, etc.). Chaque utilisateur est libre de se connecter sur n'importe quel serveur (parmi les quatre ouverts à toute la communauté), aucune restriction ne s'appliquant sur le nombre maximum de sessions ouvertes sur différents serveurs qui travaillent de manière complètement indépendante. La régulation proposée doit donc être paramétrée pour chaque serveur. Le **Tableau 2** ci-dessous récapitule, dans les deux dernières colonnes, les valeurs garanties pour les deux paramètres limitant l'utilisation, *MemG* et *TimeG*. Ces valeurs ont été décidées et validées par le Conseil Informatique de notre Unité regroupant chercheurs, ingénieurs et informaticiens de cette même Unité.

**Tableau 2.** *Mémoire et temps garantis pour les différents serveurs*

| Serveurs | Mémoire installée | Cœurs | Utilisation Type | *MemG* | *TimeG* |
|---|---|---|---|---|---|
| **Hobbes** | **192 Go** | **24 cœurs rapides (3,33 GHz)** | **Stata, Matlab, simulations** | **10 Go** | 24 h |
| **Athena** | **32 Go** | **8 cœurs, vitesse moyenne (3.17 Ghz)** | **Serveur d'applications, calcul parallèle** | **4 Go** | 24 h |
| **Hal** | **192 Go** | **16 cœurs très rapides (3.60 Ghz)** | **Stata, Matlab, R simulations,** | **10 Go** | 24 h |
| **Cassio** | **292 Go** | **80 cœurs lents (2.40 Ghz)** | **calcul parallèle, R & Matlab** | **15 Go** | 48 h |

### 2.1 Programme de régulation

Le programme de régulation est invisible pour l'utilisateur, il n'apparaît pas dans la liste des processus en cours sur le serveur et ne peut donc être arrêté par un utilisateur autre que l'administrateur du serveur (*root*).

Après chaque action de fermeture de programme, l'utilisateur concerné est averti de deux manières :

- une boîte de dialogue (*pop-up*) s'ouvre sur l'écran de l'utilisateur. Il peut ainsi être prévenu immédiatement, s'il est connecté au moment de la fermeture de son programme. Cette boîte, lui indique les éléments qui ont motivé la fermeture du programme, rappelle les valeurs de *MemG* et *TimeG* garantis pour sa session, précise le programme fermé (avec sa date de lancement, qui peut être utile dans le cas où plusieurs programmes de même nom ont été lancés en parallèle) ainsi que le serveur sur lequel cette fermeture s'est produite. Par exemple, la boîte représentée **Figure 2** indique que le programme R (*Rgui.exe*) a été fermé sur le serveur CASSIO, l'utilisateur dépassant son quota garanti de mémoire et de temps. Cette fermeture a permis de libérer 11,79 Go de mémoire sur le serveur). La boîte indique également si des éléments restent critiques (i.e. cette action n'a pas permis de revenir à l'état D – celui de l'utilisation garantie en temps et mémoire vive). Dans l'exemple de la **Figure 2**,

[10] *L'ensemble des informations transmises et des boîtes affichées est donc en anglais.*

la mémoire libérée permet de passer sous la limite des 15 Go garantis (18,12-11,79 = 6,33 < 15 Go), mais le temps garanti est déjà écoulé depuis longtemps ; seule la fermeture de la session peut permettre de démarrer une nouvelle session avec un temps garanti. Enfin, si l'utilisateur ne ferme pas la boîte, elle disparaît automatiquement au bout d'un moment (2 minutes), pour éviter de garder des informations trop anciennes ;


**Figure 2.** *Boîte s'affichant lors de la fermeture d'un programme*

- un courriel est également envoyé systématiquement lors de la fermeture d'un programme, afin de laisser une trace et de garantir qu'un utilisateur, qui n'aurait pas eu le temps de lire la boîte de dialogue (en particulier s'il n'était pas connecté au serveur à ce moment-là) ou qui l'aurait fermée trop rapidement, ait accès à cette information. Le message reprend en détail les informations de la boîte de dialogue. Dans l'exemple ci-dessous, il s'agit d'un programme Matlab qui a été fermé. Un soin particulier a été apporté à l'entête du message qui permet à l'utilisateur de savoir ce qui s'est passé à la lecture seule de l'entête "objet" du message. Dans l'exemple ci-dessous, les parties en rouge sont relatives à la session de l'utilisateur, celles en bleue au serveur (cf. **Tableau 2**) et celles en violet au programme de la session qui a été fermé pour la stabilité du serveur.

Object: [CASSIO] MATLAB.exe: Process Halted! (due to excessive memory and time use)

Dear Olivier de Mouzon,

CASSIO was becoming unstable due to excessive memory use!
MATLAB.exe (started 2012/03/20 12:45:15) has been halted on 20/10/2012 12:54 to maintain other users granted memory space.

You were using >= 72.14 GB, when only 15 GB are granted.
Your session has lasted >= 288.15 hours, when only 48 hours are granted.

Halting this process has freed 4.26 GB.
Note that your other processes on this session may still be halted as you have exceeded the granted time...
Note that your session is still using more memory than what is granted to each user, so it places your processes in the top list of processes to be halted if the system was to become unstable again.

The PIG Team.

**Figure 3.** *Courriel envoyé lors de la fermeture d'un programme*

## 2.2 Programme d'information

Le programme d'information s'exécute au démarrage de chaque session et se présente comme un ensemble d'icônes et de boîtes diffusant de l'information. Lors de l'ouverture d'une session sur le serveur, une boîte éphémère s'affiche dans la zone de notification du bureau (en bas à droite) et donne les informations sur l'état du serveur (**Figure 4**).

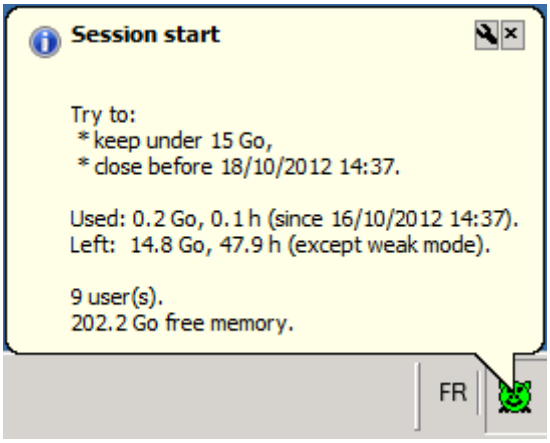


**Figure 4.** *Boîte (info-bulle) d'information au démarrage d'une session*

On y trouve le type de connexion (standard ou dégradé), le nombre d'utilisateurs déjà connectés sur le serveur (ici, neuf), la mémoire disponible sur le serveur (ici 202,2 Go), la date de fin de période garantie (quand *TimeG* sera écoulé, ici le 18/10/2012 à 14h37), la mémoire garantie sur ce serveur (*MemG*, ici 15 Go), la mémoire utilisée (ici 0,2 Go), le temps écoulé (ici 0,1 h) depuis l'ouverture de la session (dont la date est rappelée : ici le 16/10/2012 à 14h37), la mémoire garantie restante (ici 14,8 Go) et le temps garanti restant (ici 47,9 h).

Une icône en forme de petit cochon[11] permet d'informer l'utilisateur de sa situation en termes de ressources garanties et de visualiser immédiatement dans quel état il se situe. L'icône prenant alors différentes couleurs/attributs (voir le **Tableau 1**) en fonction de l'état de l'utilisateur.

**Etat D**
*(Mode garanti)*

**Etat C**
*(Temps dépassé)*

**Etat B**
*(Quota mémoire dépassé : Etat $B_D$)*

**Etat A**
*(Temps et mémoire dépassés)*


**Etat E**[12]
*(Mode dégradé)*


*(Mémoire dépassée & connexion dégradée : Etat $B_E$[12])*

**Figure 5.** *Icônes illustrant l'état de chaque utilisateur du serveur*

A chaque changement d'état (i.e. changement[13] d'icône, cf. **Figure 5**), une boîte (bulle) d'information éphémère apparaîtra au niveau de l'icône, sur le modèle de celle présentée **Figure 4**. Cette bulle d'information est assez verbeuse (limitée à 255 caractères) pour souligner les changements d'état.
Par ailleurs, si l'utilisateur souhaite connaître le détail de l'état courant, il lui suffit de passer la souris sur l'icône de son état. Une petite boîte rectangulaire lui donne alors des informations courtes (limitées à 64 caractères) mais précises (cf. **Figure 6**). Dans l'exemple de la **Figure 6**, l'information indique que l'utilisateur dépasse la mémoire garantie de 4,2 Go et qu'il a également dépassé son temps de plus de 19 jours (468,7 h). L'icône du petit cochon indique clairement que l'utilisateur est dans l'état A et compte tenu de la mémoire disponible sur le serveur (2 Go), il est possible qu'un de ses programmes soit fermé par le programme de régulation si un utilisateur vient à demander une ouverture de session (même pour des programmes peu gourmands en mémoire) ou si la demande des quatre autres utilisateurs actuellement connectés augmente de plus de 2 Go.


**Figure 6.** *Information "à la volée"*

---

[11] *Rappelons que le Pôle Informatique du Gremaq s'appelle au quotidien le PIG (cochon en anglais) et que chaque icône possède son code couleur et autres éléments mnémotechniques pour indiquer rapidement et clairement l'état de l'utilisateur (cf. note de bas de page n°5, section 1.3).*
[12] *En mode dégradé, l'icône clignote pour mieux attirer l'attention sur le mode dégradé. On alterne ainsi entre l'icône du mode dégradé et l'icône standard (située au-dessus dans la* ***Figure 5****).*
[13] *Pour mémoire, les transitions possibles sont représentées sur le graphe des états,* ***Figure 1****.*

### 2.3 Implémentation sous *PowerShell*

Les programmes de régulation et d'information se basent sur plusieurs fonctions *PowerShell* (V2) qui permettent de récupérer les informations sur chaque processus et chaque session. Ces informations sont stockées dans des fichiers temporaires qui peuvent ensuite servir à une analyse fine des besoins des utilisateurs.

En particulier, la fonction « *get-wmiobject win32_process* » permet d'accéder aux informations de l'ensemble des processus qui tournent sur le serveur. Ici, nous nous focalisons sur les attributs suivants :

- *WS* (WorkingSpace) : l'espace mémoire utilisé par le programme,
- *SessionID* : le numéro de session (permettant de relier le programme à l'utilisateur qui l'a lancé),
- *ProcessID* : le numéro d'identification du programme (permettant de l'arrêter, si nécessaire),
- *ProcessName* : le nom du programme (utilisé lors de la fermeture, cf. boîte *pop-up* **Figure 2** et exemple de courriel **Figure 3**),
- *CreationDate* : la date à laquelle le programme a été lancé (et qui sert aussi en retour à l'utilisateur lors de la fermeture).

Une autre fonction, « *query user* » permet de récupérer des informations sur la session de chaque utilisateur[14]. En particulier, on récupère pour chaque identifiant de session, l'utilisateur (login) associé et la date d'ouverture de la session.

Enfin, pour le programme d'information, « *whoami* » permet d'identifier l'utilisateur courant pour l'informer sur son état.

Ces fonctions permettent d'alimenter la procédure d'analyse de l'état de chaque utilisateur et de vérifier les paramètres de régulation. On connaîtra le total de mémoire utilisé par l'ensemble des utilisateurs, et on pourra déterminer dans quelle situation se trouve chaque utilisateur. En cas d'instabilité du serveur (le total de mémoire utilisé dépasse un total fixé), les règles de fermeture de programmes seront appliquées dans l'ordre défini et pour chaque état : on commence par les utilisateurs dans l'état **A** (Temps et mémoire dépassés) puis dans l'état **B** (mémoire dépassée), puis **C** (temps dépassé) et enfin dans l'état **E** (accès dégradé), cf. section 1.3.

Le programme de régulation utilise deux fonctions pour les actions entreprises :

- *« taskkill » pour arrêter un programme ;*
- *« logoff » pour fermer une session (sur laquelle plus aucun programme ne tourne).*

Dans le cas d'un arrêt de programme, la boîte de dialogue qui s'affiche est générée par la commande « msg ». Afin de gérer une liste d'événements (fermetures de programme) à afficher (pendant au plus 2 minutes chacun), le programme de régulation utilise un système de tâches organisés par un dictionnaire de tâches-utilisateurs et un système de sémaphore par fichiers temporaires (« [*System.IO.Path*]::*GetTempFileName* », « [*System.IO.File*]::*Exists* » et « [*System.IO.File*]::*Delete* »). Les commandes *PowerShell* spécifiques des tâches que nous utilisons sont « *Start-Job* », « *Wait-Job* », « *Get-Job* », « *Receive-Job* » et « *Remove-Job* ». Afin de transmettre des fonctions à la tâche lancée, nous utilisons aussi « *invoke-expression* »

[14] *Ces informations seront associées avec l'Active directory pour pouvoir envoyer un courriel aux utilisateurs si besoin est.*

(autrement les fonctions définies dans le programme de régulation ne sont pas visibles depuis la tâche). Enfin, afin d'attendre que l'ordre d'arrêt d'un programme soit bien effectif (autrement, le programme de régulation pourrait aller trop vite et demander plusieurs fois l'arrêt du même programme), on utilise « *Get-Process –ID* » (plus légère que « *get-wmiobject win32_process* ») dans une boucle d'attente.

Pour le programme d'information, on utilise une librairie qui offre l'interface graphique souhaitée : « [*void*] [*System.Reflection.Assembly*] :: LoadWithPartialName ("*System. Windows.Forms*") ». Cela permet de définir différentes icônes « *New-Object System.Drawing.Icon* » et l'objet d'interface souhaité « *New-Object System.Windows. Forms.NotifyIcon* », auquel on affectera différents attributs (que le programme met à jour régulièrement) :

- « Text » : les informations concises qui s'affichent dans la boîte rectangulaire quand la souris passe sur l'icône (cf. **Figure 6**) ;
- « Icon » : le choix de l'icône dans la barre des tâches, sur la droite, correspondant à l'état courant (cf. **Figures 4, 5 et 6**) ;
- et les éléments pour l'info-bulle (cf. **Figure 4**) :
  - « *BalloonTipIcon* » (« *Info* », « *Warning* » ou « *Error* ») : le petit logo à gauche du titre de l'info-bulle (cf. **Figure 4** pour celui correspondant à « *Info* ») ;
  - « *BalloonTipTitle* » : le titre de l'info-bulle (« *Session Start* », dans l'exemple) ;
  - « *BalloonTipText* » : le contenu de l'info-bulle (limité à 255 caractères) ;
  - « *ShowBalloonTip* » : la commande qui permet d'afficher l'info-bulle (voire d'en limiter l'affichage dans la durée).

Enfin, d'autres fonctions *PowerShell* classiques ont été utilisées :

- « *Get-Date* » et « *New-Timespan* » pour tout ce qui concerne les dates et durées ;
- « *gc env:computername* » pour récupérer le nom du serveur (utile pour déployer le même programme sur plusieurs serveurs) ;
- « *start-sleep* » pour les pauses entre deux cycles de vérifications ;
- « *Send-MailMessage* » pour l'envoi des informations par mél (avec notamment des options de priorité du message et copie au service informatique).

L'ensemble des ressources sur lesquelles nous avons pris appui pour implémenter les deux programmes sous *PowerShell* figure dans la liste des références bibliographiques.

## 3. Algorithmes implémentés

Les algorithmes proposés ici sont des versions simplifiées de ceux codés en *PowerShell* et implémentés sur nos serveurs. Une procédure d'installation de ces deux programmes a été rédigée conjointement à ces programmes à l'attention des administrateurs système des serveurs (cf. **Annexe 1**). Les versions simplifiées de ces programmes permettent d'avoir un aperçu rapide de l'ordre des actions réalisées[15], chacune ayant fait l'objet de la programmation d'une fonction spécifique. En particulier les actions de tri, de fermeture de processus, d'envoi de message sont des fonctions programmées spécifiquement pour ces programmes qui utilisent les fonctions *PowerShell* décrites précédemment.

[15] *Les ACTIONS qui apparaissent dans les algorithmes sont ici en majuscule.*

### 3.1 Algorithme du programme de régulation

# Début
REPETER INDEFINIMENT
- RECUPERER les informations sur le serveur (MemTotale, NbUtilisateurs, etc.)
- RECUPERER les informations de la session utilisateur (MemUtilisée, TempsSession, etc.)

SI il existe des utilisateurs ET SI mémoire totale utilisée > total mémoire disponible[16]
ALORS
TANT QUE mémoire totale utilisée > total mémoire disponible
SI il existe des utilisateurs dans l'**état A** *# mémoire et temps dépassés*
- IDENTIFER l'utilisateur en **état A** le plus ancien
- IDENTIFER le processus le plus gros[17] de cet utilisateur
- FERMER ce processus
- ENVOYER message (boîte+courriel)

FIN
SINON SI il existe des utilisateurs dans l'**état B** *# mémoire dépassée, temps OK*
- IDENTIFER l'utilisateur en **état B** le plus gourmand
- IDENTIFER le processus le plus gros[17] de cet utilisateur
- FERMER ce processus
- ENVOYER message (boîte+courriel)

FIN
SINON SI il existe des utilisateurs dans l'**état C** *# mémoire OK, temps dépassé*
- IDENTIFER l'utilisateur en **état C** le plus ancien
- IDENTIFER le processus le plus gros (hors système[18]) de cet utilisateur

SI pas de processus en cours
- FERMER la session
- ENVOYER message (courriel)

FIN
SINON # il existe un processus en cours
- FERMER ce processus
- ENVOYER message (boîte+courriel)

FIN
FIN
SINON SI il existe des utilisateurs dans l'**état E** *# connexion dégradée*
- IDENTIFER l'utilisateur en **état E** le dernier connecté (depuis au moins 1 min)
- IDENTIFER le processus le plus gros (hors système[18]) de cet utilisateur

SI pas de processus en cours

---

[16] *Le total disponible pour les utilisateurs est donné pour chaque serveur. Actuellement, c'est le total du serveur (cf. deuxième colonne du **Tableau 2**) privé de 2 Go. Cela permet d'éviter qu'il y ait des swaps mémoire qui font perdre du temps de calcul, tout en laissant un petit espace pour les administrateurs (non comptés dans l'espace mémoire utilisé).*

[17] *Ici, on n'est pas obligé d'exclure les processus système, car ils utilisent peu de mémoire et ne sont donc jamais concernés par les actions de fermeture dans les états A et B, contrairement aux états C et E (où ces processus doivent être exclus des actions de fermeture de programme, cf. note de bas de page n°18 ci-dessous).*

[18] *Les processus suivants ne sont jamais fermés (sauf en cas de fermeture de session) car ils sont nécessaires au bon fonctionnement de la session : csrss.exe, winlogon.exe, dwm.exe, explorer.exe, rdpclip.exe, taskeng.exe, taskhost.exe, conime.exe, conhost.exe, cmd.exe, powershell.exe. Les trois premiers ne peuvent tout simplement pas être arrêtés. Les trois derniers sont nécessaires au lancement du programme d'information et ne doivent donc pas être fermés par le programme de régulation.*

```
                - FERMER la session
                - ENVOYER message (courriel)
            FIN
            SINON                   # il existe un processus en cours
                - FERMER ce processus
                - ENVOYER message (boîte+courriel)
            FIN
        FIN
        ECRITURE d'un fichier d'enregistrement des actions
        ACTUALISER la liste des processus
    FIN TANT QUE
  FIN
  Pause avant prochain cycle de vérification
FIN
```

### 3.2 Algorithme du programme d'information

Ce programme a pour but d'afficher les informations sur les ressources disponibles sur le serveur d'une part, et d'autre part sur l'état de l'utilisateur concernant sa consommation de ressources (mémoire et durée de connexion). Ces informations sont dispensées en temps réel par l'intermédiaire d'une icône dont l'apparence change suivant l'état (cf. **Figures 5 et 1**) dans la zone de notification des serveurs, en bas à droite du bureau. Si on passe la souris sur l'icône, des informations courtes mais plus précises sur l'état de l'utilisateur et du serveur sont affichées dans un petit rectangle (cf. **Figure 6**). Une boîte éphémère de notification plus verbeuse apparaît également, sous la forme d'une bulle, lors de la connexion et à chaque changement d'état de l'utilisateur (cf. **Figure 4**).

```
# Début
Connexion = TRUEMode = Standard
NouveauMode = Degradé
Mémoire = OK
Temps = OK
NouveauTemps = OK
REPETER INDEFINIMENT
  - RECUPERER les informations sur le serveur (MemTotaleUtilisée, NbUtilisateurs, MemTotaleAGarantir sauf pour les utilisateurs connectés après l'utilisateur courant)
  - RECUPERER les informations sur la session utilisateur (MemUtilisée, TempsSession)
  SI MemTotaleAGarantir <= total mémoire disponible
    NouveauMode = Standard
  FIN
  SI MemUtilisée > mémoire garantie
    NouvelleMemoire = KO
  FIN
  SINON
    NouvelleMemoire = OK
  FIN
  SI TempsSession > temps garanti
    NouveauTemps = KO
  FIN
```

- METTRE A JOUR les informations courantes (icône de l'état[19], informations courtes, informations détaillées avec icône associée au nombre de points soulevés dans la bulle[20]) en fonction des informations récupérées (mémoire habituellement garantie totale et restante, temps habituellement garanti restant, temps passé, date de début de session, date de fin des garanties, nombre d'utilisateurs, mémoire actuellement disponible sur le serveur) et du nouvel état (Connexion, NouveauMode, NouvelleMemoire, NouveauTemps)
SI CHANGEMENT (i.e. Connexion ou NouveauMode <> Mode ou NouvelleMemoire <> Memoire ou NouveauTemps <> Temps)
- AFFICHER la boîte d'informations détaillées pendant quelques (3) secondes
FIN
Connexion = FALSE
Mode = NouveauMode
Memoire = NouvelleMemoire
Temps = NouveauTemps
- ECRITURE d'un fichier d'enregistrement des états
SI Mode = Standard
Pause (60 secondes) avant prochain cycle de mise à jour des informations
FIN
SINON # Mode = Dégradé
PENDANT le temps de pause (60 secondes) avant le prochain cycle de mise à jour des informations FAIRE
- AFFICHER l'icône standard pendant la demi-période de clignotement (1 seconde)
- AFFICHER l'icône « WEAK » pendant la demi-période de clignotement (1 seconde)
FIN
FIN
FIN

## Conclusion

La régulation proposée ici se base sur des règles simples connues des utilisateurs qui sont informés en temps réel de leur état et de l'état du système. L'information, qui est primordiale dans un système de régulation automatique, est ainsi au cœur de ce système qui fonctionne de manière autonome sans intervention humaine. A partir de ces règles et des contraintes d'information que nous nous sommes fixées collectivement, nous avons construits les outils d'une régulation simple et efficace pour une gestion *a minima* de nos serveurs de calcul. Cette régulation permet une bonne répartition des ressources, notamment en mémoire, en limitant les contraintes sur les utilisateurs et en maximisant l'utilisation des ressources disponibles. Un utilisateur pourra en effet utiliser 90% de la mémoire disponible si cela ne gêne personne et tant que les autres utilisateurs peuvent bénéficier d'un accès à la ressource. Si des utilisateurs se connectent et que la mémoire vient à manquer, c'est sur la session de cet utilisateur que le programme de régulation agira pour fermer ses programmes et libérer de la ressource pour ses collègues.

Les outils développés ici restent évidemment perfectibles. En effet, ces programmes se concentrent sur un paramètre vital des serveurs – la mémoire – à l'exclusion d'autres entités

---

*[19] Sauf en mode dégradé (cf. note de bas de page n°12, section 2.2) où seront choisies deux icônes de la* ***Figure 5*** *pour un effet clignotant : celle du mode standard (première ligne) et celle qui lui correspond (2 è ligne) avec le mot clef « WEAK ».*

*[20] Trois icônes sont disponibles et sont attribuées aux situations suivantes : « Info » pour l'état D (où tout va bien), « Warning » pour les états E, C et $B_D$ (où un seul élément doit attirer l'attention), « Error » pour les états $B_E$ et A (où deux éléments doivent attirer l'attention*).

importantes (cœurs, disques) qui pourraient être gérées également. On pourrait imaginer, comme cela se fait sur de gros serveurs de calcul, que les ressources garanties par utilisateur soient étendues du duo *mémoire-temps* au triplet *mémoire-temps-cœur*. Chaque utilisateur pouvant ainsi compter sur un quota garanti de mémoire, de temps d'utilisation et sur un nombre de cœurs. Cette pratique nous a semblé trop difficile à mettre en œuvre et à calibrer sur les besoins de nos chercheurs dont les calculs nécessitent parfois beaucoup de mémoire (grosses matrices de données), parfois un nombre élevé de réplications ou de simulations sur différents cœurs (simulation de type *Monte-Carlo* ou de *bootstrap*). La gestion des disques durs propres aux serveurs est une question qui semble plus facile à aborder (en définissant des quotas par utilisateur) mais qui n'a pas encore été mise en place.
L'une des extensions évidente consiste à remplacer le paramétrage actuel de chaque serveur qui est inclus dans chacun des deux programmes, en un fichier externe facilement modifiable. L'intérêt serait de pouvoir ainsi modifier facilement les ressources garanties (*MemG* et *TimeG*). Un autre avantage serait de pouvoir définir facilement des projets prioritaires échappant aux règles de partage communes en venant prendre directement un quota de mémoire sur le serveur, en amont du processus de régulation[21] et donc avec des garanties de mémoire et de temps étendues.

Enfin, l'un des éléments, qui nous semblent importants à mettre en œuvre pour compléter notre gestion de ces serveurs, concerne l'analyse des activités des serveurs (*monitoring*), afin de connaître précisément la demande réelle de nos chercheurs. Une prochaine étape consistera à développer un outil permettant de recenser et de compiler l'utilisation des différentes ressources, les problèmes rencontrés, les logiciels les plus utilisés, etc. Les outils et fonctions *PowerShell* utilisés ici permettent tout à fait de construire des outils de supervision adaptés à notre cadre en vue de visualiser les périodes de pointe, les pics de mémoire et ainsi d'ajuster l'offre (la capacité mémoire de certains nos serveurs est encore loin d'être saturée) à la demande de nos chercheurs.

## Références bibliographiques

Don Jones, "Windows PowerShell : Cours intensif de script", http://technet.microsoft.com/fr-fr/magazine/hh551144.aspx.
Robin Lemesle & Arnaud Petitjean, "Windows PowerShell (versions 1 et 2) – Guide de référence pour l'administration système", Editions ENI, 2010.
Forum PowerShell Scripting, http://powershell-scripting.com/.
Eric Vernié, “Introduction au développement sous PowerShell”, http://msdn.microsoft.com/fr-fr/visualc/bb906067.aspx

## Remerciements

Ce travail a été réalisé avec le support actif et les encouragements de l'ensemble des membres du PIG. Nous tenons à remercier très chaleureusement nos collègues Thibault Laurent, Alain Lavergne, Alexandre Martins et Valérie Orozco pour leur précieuse aide dans l'implémentation et dans les tests de ces outils. L’ensemble des programmes développés ici sont disponibles sur demande auprès des auteurs (Christophe.Bontemps@toulouse.inra.fr et Olivier.deMouzon@toulouse.inra.fr).

[21] *Evidemment, lorsque cette mémoire réservée ne serait pas utilisée, d’autres utilisateurs y auraient accès (soit pour dépasser ponctuellement leur mémoire garantie, soit pour accéder au serveur en mode dégradé).*

# Annexe 1 : Programme d'installation

*---*

*Titre : Manuel d'installation des programmes de régulation*

*Auteur : Olivier de Mouzon (TSE, Gremaq, INRA)*

*Date : 28/03/2012*

*---*

*1) Copier ce répertoire MonitorPIG (et tout ce qu'il contient) dans le répertoire C:\ProgramData du serveur à réguler.*

*2) Adapter la mémoire max du serveur, ainsi que la mémoire et le temps garantis par utilisateur :*

*Ces paramètres sont à modifier dans les 2 programmes : GlobalMonitorV??.ps1 (actuellement V1_4) et LocalMonitorV??.ps1 (actuellement V1_0).*

*A noter que ces versions peuvent évoluer sans que le manuel soit modifié (donc prendre la dernière version disponible qui va avec les fichiers GlobalMonitor.bat et LocalMonitorInvisible.bat ; sinon pb de cohérence possible).*

*3) Permettre les fichiers PowerShell locaux de s'exécuter sur le serveur (si ce n'est pas déjà le cas ; donc sur un nouveau serveur) :*

*Lancer Windows PowerShell en tant qu'administrateur [via clic droit sur l'icône correspondant dans la barre de lancement rapide, par exemple] et exécuter la commande suivante :*

*Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope LocalMachine -Force*

*Puis fermer la fenêtre Windows PowerShell.*

*4) Copier le raccourci du programme d'information dans les lancements au démarrage de la session :*

*Placer une copie de LocalMonitorPIG.lnk dans "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup" du serveur [on peut par exemple l'atteindre via "Démarrer" -> "Tous les programmes" -> Clic droit sur "Démarrage" -> "Ouvrir tous les utilisateurs"].*

*[Rq : si les utilisateurs souhaitent voir l'icône en permanence, il faut qu'ils personnalisent leur tray en conséquence : clic sur la double flèche verticale pour voir les icônes cachés -> clic sur "Personnaliser" -> Choisir "Afficher l'icône et les notifications" sur la ligne correspondant au logo du PIG OU clic sur la boîte qui permet de toujours tout afficher (en bas à gauche).]*

*5) Depuis le compte ROOT du serveur :*

*Lancer GlobalMonitor.*

# Annexe 2 : Architecture du système