



## Fiche projet - grant

AFF065-2009-NOT338

\$Rev: 494 \$

2011-04-03 13:23:15+02:00

Sébastien DEVAUX

dexter@detexia.com

**Résumé :** La constitution de grille de calcul de grande envergure bute sur le problème de sa supervision et de la charge importante induite par le seul processus de répartition de charge. Le projet grant propose de bâtir une grille de calcul à l'échelle d'internet lui-même, et comme ce dernier, capable de fonctionner sans avoir recours à un système de supervision global. Le processus de répartition de charge est distribué sur l'ensemble des noeuds de calcul qui composent la grille selon une stratégie inspirée par celle mise en oeuvre par une colonie de fourmis pour la recherche de nourriture. Ce mode de fonctionnement permet de remplacer un processus global par des processus locaux coopératifs capables de prendre des décisions à partir d'information incomplète pour, malgré tout, accomplir collectivement la fonction d'affectation des tâches d'une manière remarquablement efficace à l'échelle de la grille entière.

**Mots-clé :** répartition de charge, grille de calcul, processus distribué, processus adaptatif, intelligence collective

## Table des matières

1 Grille à répartition de charge adaptative.....	1
2 Transport des requêtes par fourmis artificielles.....	2
3 Mise en oeuvre.....	3
4 Résultats expérimentaux.....	3
4.1 Réseau simulé.....	3
4.2 Démonstrateur réduit.....	4
5 Travaux à venir.....	4
Table des URL .....	4

### 1 Grille à répartition de charge adaptative

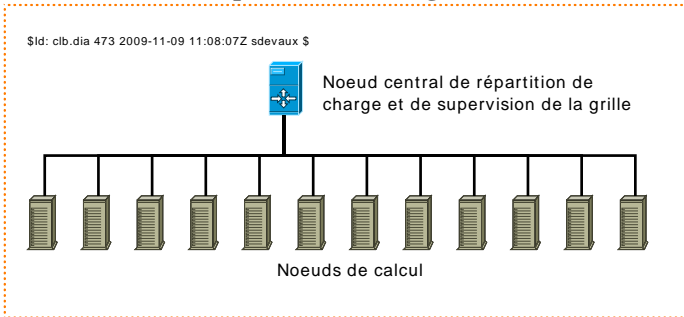
Constituer une grille de calcul efficace à l'échelle d'internet lui-même est un défi de taille. Les systèmes de répartition de charge classiques reposent sur un allocateur centralisé qui a besoin d'une information la plus complète possible pour prendre judicieusement ses décisions. Et plus la taille de la grille augmente plus le système de répartition risque lui-même de s'engorger. De plus la difficulté sera considérablement accrue si on admet dans la grille des noeuds de calcul de natures différentes :

- Les mesures de performance seront toujours plus ou moins dépendantes de la nature du noeud observé.

Il est toujours difficile d'avoir de bon référents qui permettraient de comparer objectivement deux noeuds quelconques.

- L'information utilisée pour déterminer à quel noeud affecter une tâche est en pratique toujours incomplète. Elle est basée sur une ou plusieurs mesures instantanées que le temps de transport réseau rend obsolète dès leur transmission. En outre, la connaissance du passé n'est en soit pas utile directement, on souhaiterait idéalement connaître l'état à venir durant l'exécution de la tâche à affecter, ce qui n'est évidemment que partiellement prédictible.
- Une bonne répartition de charge devrait également prendre en compte la nature des tâches. Une tâche sera d'autant mieux affectée qu'on a une bonne connaissance des ressources nécessaires à sa réalisation. Une telle connaissance ne pourra être obtenue que pour des tâches totalement isolées, c'est à dire n'ayant aucune interaction avec des éléments extérieurs (interaction avec un opérateur, lecture/écriture de flux réseaux, taille des données à traiter non connue a priori).

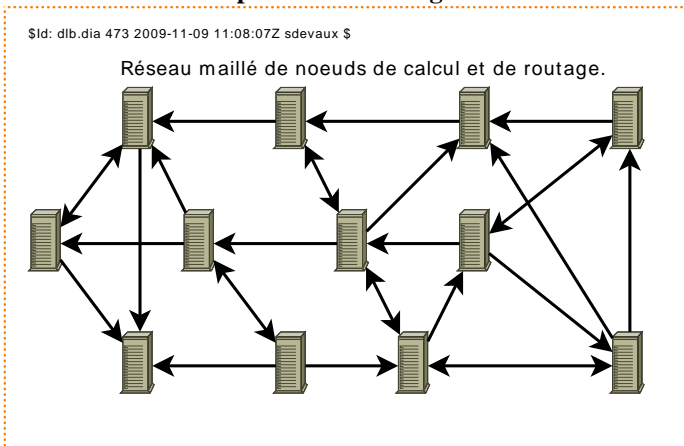
### Grille à répartition de charge centralisée



Grant<sup>[url1]</sup> propose la mise en oeuvre d'une grille de calcul à l'aide d'algorithmes biomimétiques<sup>[url2]</sup>. Ces algorithmes sont de bons candidats pour contourner les difficultés énumérées plus haut : ils sont conceptuellement adaptatifs et capables de fonctionner à partir d'informations incomplètes (erreurs de mesures, latences) et changeantes.

La grille est composée d'un ensemble de noeuds qui forment un réseau connexe selon une relation de voisinage. Chaque noeud ne permet d'atteindre qu'un nombre limité de noeuds voisins mais il doit exister au moins un chemin pour se rendre d'un noeud quelconque à un autre. Chaque noeud peut soumettre des tâches à un autre qui lui-même pourra les relayer à un troisième, etc... Le problème fondamental de la répartition de charge devient assimilable à du routage. Une version biomimétique en est la collecte de nourriture par les fourmis : les noeuds sont des réservoirs plus ou moins importants de nourriture et les tâches à faire exécuter par les noeuds sont des fourmis. La relation de voisinage n'est pas une relation hiérarchique et tous les noeuds sont fonctionnellement rigoureusement équivalents. L'absence de tout superviseur centralisé augmente la robustesse du système. La défaillance individuelle d'un noeud n'a aucun impact à l'échelle de la grille tant que cela ne brise pas la connexité du graphe de voisinage.

### Grille à répartition de charge distribuée



## 2 Transport des requêtes par fourmis artificielles

Tous les noeuds de la grille sont d'un point de vue fonctionnel strictement identiques. Chacun est capable de prendre en charge le traitement d'une tâche ou de relayer une requête vers un autre noeud. Chaque noeud connaît un nombre limité de noeuds voisins et la répartition de charge agit localement pour déterminer à quel noeud confier une requête : soi-même ou l'un des noeuds voisins connus. Evidemment le noeud choisi, voyant arriver la requête fait de même et éventuellement dirige

lui-même la demande vers un troisième noeud, et ainsi de suite. Affecter une tâche à un noeud dans un tel réseau s'apparente à du routage, on parcourt le réseau noeud par noeud jusqu'à la destination avec la particularité suivante : dans notre cas la destination n'est pas déterminée à priori.

Le modèle des fourmis est transposable à la répartition de charge dès que celle-ci peut également être traitée comme un problème de routage. La non connaissance de la destination pour chaque requête ne pose pas de problème puisque le modèle utilisé répond au même critère en réalité : les sources de nourriture ne sont au départ pas connues des fourmis. Elles sont découvertes au fur et à mesure et apparaissent ou disparaissent selon des événements non maîtrisés.

Le réseau de noeuds constitue l'environnement dans lequel les fourmis sont plongées. Les fourmis à la recherche de nourriture sont les tâches à la recherche d'un processeur pour être exécutées. Les fourmis parcourent d'abord au hasard le réseau jusqu'à aboutir à un noeud qui les accepte. Lorsque le travail est terminé, le chemin ayant permis d'atteindre le noeud est marqué. Ce marquage plus ou moins éphémère servira ensuite aux autres fourmis à trouver un chemin menant à un processeur dans le réseau. Afin de répartir efficacement la charge, le marquage doit être proportionnel à la quantité de nourriture, c'est à dire à la puissance de calcul disponible sur un noeud. Si un noeud devient trop chargé, les tâches-fourmis en reviennent moins rapidement, le marquage vers ce noeud s'atténue et il apparaît alors comme un moins bon candidat.

Tout le problème de la répartition de charge se réduit maintenant à une gestion appropriée du marquage des liens inter-noeuds. Ce marquage peut être élémentaire. Un incrément fixe pour chaque terminaison de tâche est suffisant pour un renforcement des routes menant à de bons noeuds. Il est toutefois possible d'accélérer l'apprentissage du réseau en utilisant un marquage lié à la puissance constatée. Ainsi les incidences de situations trompeuses aléatoires (par exemple tâche très courte affectée à un mauvais noeud) seront limitées. Le problème n'est pourtant pas trivial : il n'y a pas d'indice de puissance absolue. Il est déjà difficile de classer des processeurs selon un axe de puissance unique. La mesure de la puissance réelle disponible peut s'appuyer sur un ou plusieurs de ces critères :

- longueur de la route : cette mesure permet de privilégier les noeuds les plus proches de soi (selon la topologie de routage qui peut être différente de la topologie physique du réseau.)
- Taux d'occupation du processeur : cette information est accessible sur la plupart des plateformes mais son acquisition est spécifique à chacune.
- Temps d'exécution : il s'agit de la mesure la plus impartiale et la plus significative pour l'utilisateur. Elle devrait toutefois être pondérée par la quantité de code sinon toute tâche d'une certaine importance signalerait avoir été exécutée sur un mauvais noeud.
- Quantité de code exécutée à l'issue de la tâche : permet de reconsidérer avantageusement les critères précédents. Ainsi le calcul de consommation des ressources sera plus juste en tenant compte du travail réalisé. Toutefois se posent les problèmes de l'unité élémentaire de calcul et son acquisition requière que chaque tâche soit instrumentée à cet effet.

La fonction d'évaluation n'a toutefois pas besoin d'être totalement absolue sur tout le réseau, il suffit qu'elle le soit pour tous les éléments pris en compte lors du routage d'une

tâche. La fonction d'évaluation peut donc être locale à chaque noeud routeur si elle est appliquée de façon identique à tous les noeuds voisins connus.

### 3 Mise en oeuvre

L'ambition d'une grille déployable sur internet à l'échelle mondiale implique la capacité d'y accueillir des noeuds de toutes sortes : machines et systèmes d'exploitation variés. Ceci nous pousse à choisir des technologies portables pour le transport des requêtes et l'exécution des tâches :

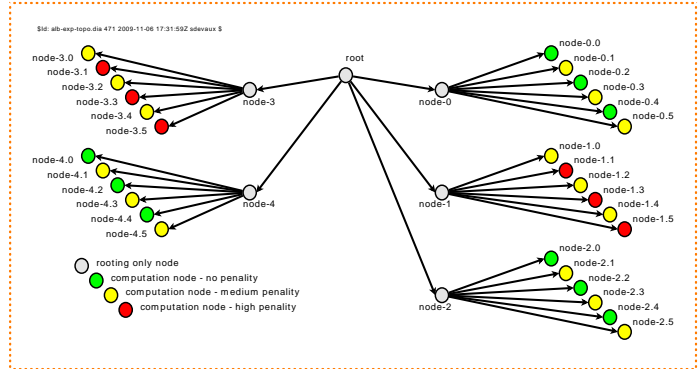
- Java [url3] : pour couvrir le besoin de portabilité binaire, chaque tâche doit être exécutable sur n'importe quel type de noeud.
- XML-RPC [url4] : protocole de communication "passe-partout", qui repose sur l'échange de messages XML via le protocole HTTP.

Dans la première phase d'évaluation de l'algorithme de routage des tâches, la mise en oeuvre se réduit à un environnement hétérogène simulé sur un unique poste : ceci essentiellement pour des raisons économiques et pratiques. Le travail sera transposable directement à un véritable environnement réparti par une simple révision de la mise en oeuvre des interfaces de communication entre noeud :

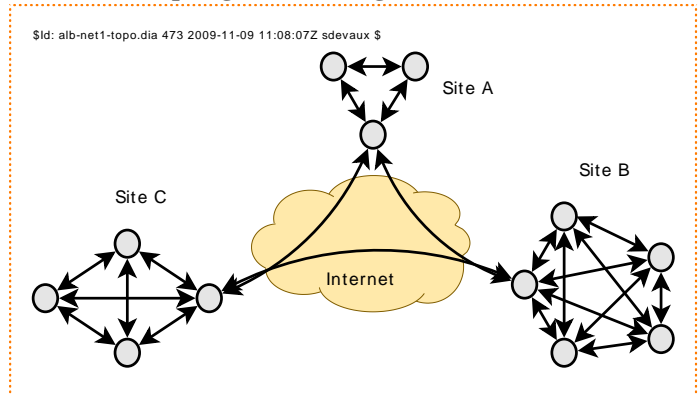
- Le noeud de simulation est un noeud fonctionnel qui ajoute un délai au temps de traitement de la tâche. Ce délai est fixe pour un noeud donné mais varie d'un noeud à l'autre. Il simule la force de calcul plus ou moins importante d'un noeud par un retard artificiel de la fin de traitement d'une tâche. Il s'ajoute un délai supplémentaire proportionnel à la charge du noeud, c'est à dire le nombre de tâches en cours d'exécution sur le noeud. Ce dernier point simule une baisse de performance en fonction du nombre de tâches simultanément actives.
- Les tâches soumises se résument à un délai attribué aléatoirement afin de tester la répartition de tâches hétérogènes dont le temps de traitement n'est pas connu a priori. La performance globale de la grille sera simplement mesurée par l'écart entre le temps d'exécution réel de toutes les tâches et la somme des délais de chaque tâche qui représente le temps d'exécution optimal.
- Le réseau testé contient 30 noeuds de calcul répartis sur une topologie arborescente : toutes les tâches sont soumises à un premier noeud racine qui répartit sur une couche de 5 noeuds routeurs qui eux même répartissent chacun sur 6 noeuds de calcul.
- Il y a trois types de noeuds : fort, moyen et faible, soit trois valeurs de temporisation fixe possibles. Chaque branche finale ne contient que deux types de noeuds : soit faible/moyen, soit moyen/fort.

Dans la deuxième phase, le système est mis en oeuvre sur plusieurs ordinateurs reliés à internet. La communication émulée dans la simulation est remplacée par le protocole de communication XML-RPC et un problème facilement parallélisable (le calcul de l'image de l'ensemble de mandelbrot [url5]) a été confié à la grille.

#### Topologie de voisinage du réseau simulé



#### Topologie de voisinage du réseau réel



## 4 Résultats expérimentaux

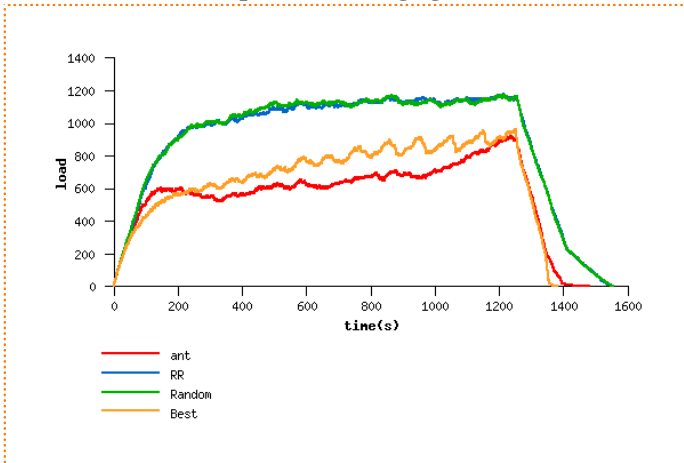
### 4.1 Réseau simulé

Notre répartition de charge a été confrontée à trois autres systèmes à titre de comparaison :

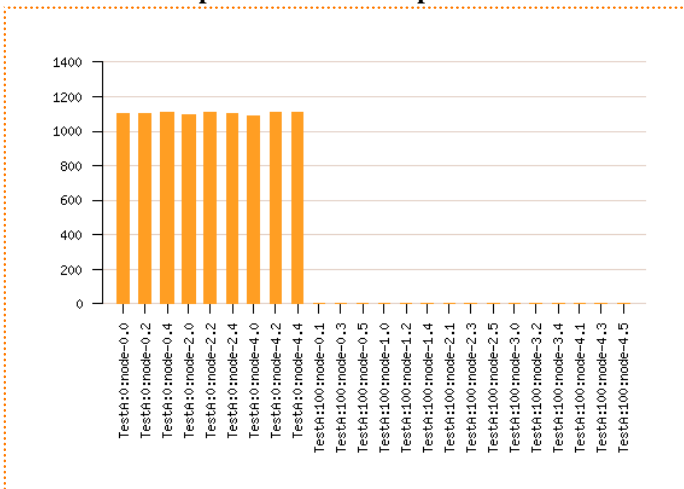
- "Random" : chaque noeud est choisi de façon totalement aléatoire. Il est bon de vérifier que le système proposé est capable de faire mieux que le hasard.
- "Round Robin" : chaque noeud est choisi successivement sans tenir compte de la charge ou des performances. Il s'agit du système de répartition le plus simple à mettre en oeuvre et qui l'est effectivement très souvent. Il est très efficace dans un environnement où noeuds et tâches sont homogènes.
- "Best" : sélection du meilleur noeud à l'instant de l'affectation. Cette méthode est très simple et permet des performances proches de l'idéal. Ce n'est pourtant pas le choix optimal car il faudrait en réalité considérer le meilleur noeud pour toute la durée de la réalisation de la tâche à affecter donc être capable d'anticiper les variations de charge de tous les noeuds.

Grant parvient à répartir la charge aussi bien que le meilleur algorithme déterministe testé (Best). Même s'il y a un surcote induit par la première phase d'exploration assez aléatoire, il parvient dans une seconde phase à mieux transférer la charge vers des noeuds de plus faible capacité quand les noeuds les plus puissants deviennent surchargés.

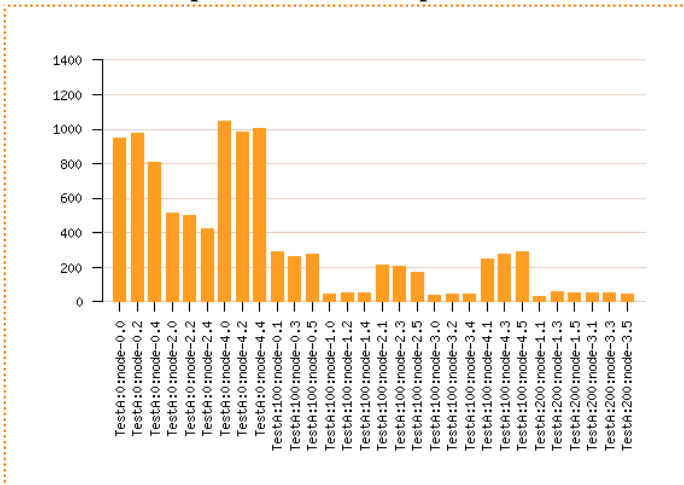
### Comparaison charge globale



### Répartition des tâche par 'Best'



### Répartition des tâches par Grant



## 4.2 Démonstrateur réduit

Le calcul de l'ensemble de Mandelbrot a été réalisé par une petite grille d'une dizaine d'ordinateurs de type PC répartis sur trois sites. Ont participé simultanément des PC animés par Linux (plusieurs distributions) et Windows XP. Le seul objectif de l'expérience a été de vérifier le bon fonctionnement du système sur un cas concret. Chaque noeud a pris en charge une portion du problème et le calcul d'une image de 10000 pixels de coté a été segmenté par le calcul de 100 images de 1000 pixels de cotés sur l'ensemble de la grille et le travail a été

accompli environ 8 fois plus rapidement que sur le PC témoin qui a assumé seul l'intégralité du même calcul.

## 5 Travaux à venir

- Evaluation à plus grande échelle : constituer une grille de plusieurs dizaines de noeuds et lui confier des problèmes un peu plus représentatifs d'une utilisation pratique tels que la factorisation de grands nombres, ou la résolution de systèmes linéaires de grande taille.
- Topologie de voisinage dynamique : pour le moment la relation de voisinage est établie une bonne fois pour toute de façon arbitraire individuellement pour chaque noeud. A terme la liste de voisinage ainsi définie au démarrage devrait évoluer selon un processus imitant la sélection naturelle sur la base du marquage des liens qui constitue un bon critère d'évaluation de chaque voisin connu. La découverte de nouveaux voisins est possible par l'examen de l'origine de chaque requête reçue.
- Généraliser le principe d'allocation ici exposé à d'autres types de ressources que le temps processeur.
- Diffusion de tâches d'étalonnage pour estimer l'état de la grille même en cas d'absence de requête.
- Comparaison de fonctions d'évaluation et de sélection du voisinage. Etude de l'influence d'un changement local du critère de sélection sur le comportement global de la grille.

## Table des URL

[url1] *Grant* : <http://dexter.detexia.net/grant/>

[url2] *algorithmes biomimétiques* : [www.antsearch.univ-tours.fr/publi/azzag04survol.pdf](http://www.antsearch.univ-tours.fr/publi/azzag04survol.pdf)

[url3] *Java* : <http://java.sun.com/>

[url4] *XML-RPC* : <http://www.xmlrpc.com/>

[url5] *ensemble de mandelbrot* : [http://fr.wikipedia.org/wiki/Ensemble\\_de\\_Mandelbrot](http://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot)