Algorithme de diffusion génétique pour l'équilibrage de charge dans les grilles de calcul.

DRIF Ahlem
Département informatique
UFA, Sétif, Algérie
starinfonet@yahoo.fr
BOUKERRAM Abdellah
Département informatique
UFA, Sétif, Algérie
boukerram@hotmail.com

Résumé **–L'équilibrage de charge est une clé importante dans le développement d'applications sous des environnements de grille de calcul. La plupart des algorithmes d'équilibrage de charge sont basés sur des approches ou des méthodes heuristiques itératives totalement distribués. L'équilibrage de charge selon le voisinage, implique que les stations de travail éloignées du nœud central ne bénéficient d'une charge supplémentaire qu'après un temps assez long, ce qui diminue le temps de convergence de l'algorithme usité. Ce papier introduit un nouvel algorithme d'équilibrage de charge, s'inspirant du principe des algorithmes de diffusion et de division génétique. La solution proposée apporte une meilleure optimisation du processus d'équilibrage de charge avec une accélération dans la convergence de l'algorithme.**

Mots Clé **-Grille de calcul, équilibrage de charge, réseau hétérogène, algorithme de diffusion, algorithme itératif.**

I. INTRODUCTION

La grille de calcul est une infrastructure répartie qui offre un partage de ressources flexibles, sécurisées, et cohérent entre des collections dynamiques, d'individus, d'institutions et de ressources matérielles et logicielles, à travers des organisations virtuelles (VOs) [1]. Ces dernières (VOs) permettent un partage distribué performant, quelque soit le potentiel des participants, tel que les consortiums industriels du design aérospatial, par exemple.

Avec l'intégration des middlewares des grilles de calcul, une augmentation de la capacité de traitement global est offerte aux utilisateurs. Néanmoins, de nombreuses difficultés restent à surmonter pour avoir un système sûr et fiable. Une des difficultés est le manque de performances obtenues lorsque le travail assigné aux processeurs devient déséquilibré [2]. C'est à ce problème particulier qu'on s'intéresse dans ce papier.

Plusieurs études et algorithmes ont été proposés pour redistribuer les charges des processeurs surchargés sur ceux qui sont moins chargés, et ce, en tenant compte des changements dynamiques dans les réseaux hétérogènes. Dans la littérature on trouve plusieurs algorithmes d'équilibrage de charge, le plus ancien est l'algorithme d'équilibrage direct (client/serveur) mais cette approche dispose de plusieurs inconvénients ce qui a imposé l'utilisation d'une autre approche basée sur les méthodes itératives [3]. Les algorithmes itératifs constituent une solution plus appropriée pour la résolution d'un certain nombre de problèmes. Ces méthodes ont la particularité d'effectuer plusieurs itérations afin d'affiner la solution recherchée. Les algorithmes itératifs s'appuient sur des approximations pour obtenir au bout d'un nombre minimal d'étapes un équilibre optimal. Les algorithmes itératifs asynchrones sont bien adaptés aux problèmes liés aux grilles de calcul, à savoir l'hétérogénéité et les synchronisations globales entre les noeuds de calcul [4].

Un algorithme itératif d'équilibrage de charge effectue, par itérations successives, des transferts de charge entre les différents processeurs voisins afin de converger vers l'équilibre des charges du système. Ce genre d'algorithmes, adapté à l'équilibrage de charge, a été introduit par George Cybenko dans [5]. Le point le plus important dans l'algorithme de diffusion est la détermination de la matrice de diffusion. C'est elle qui définie la vitesse de convergence de l'algorithme.

L'approche locale proposée par Boillat et Cybenko [5] a montré que la matrice de diffusion optimale dépend intégralement de la topologie du réseau à équilibrer. Cette approche présente des inconvénients parce qu'elle opère de proche en proche. Une approche globale a été définie (ODF Optimally-tuned Diffusion) dans [5] qui maximise la vitesse de convergence de l'algorithme de diffusion mais celle-ci nécessite tout de même une étude préalable du réseau considéré.

Les algorithmes de type "Dimension Exchange" sont dérivés de l'algorithme de diffusion. Ils sont basés sur des communications uni-port, c'est à dire qu'un nœud *i* ne peut s'équilibrer qu'avec un seul de ses voisins *j* à un instant *t* donné [6].

L'algorithme "Generalized Dimension Exchange" "GDE" est l'application de l'algorithme de dimension exchange sur des réseaux quelconques [7]. Ce type d'algorithme équilibre les processeurs deux à deux alors que le facteur d'équilibrage de charge est choisi tel que l'équilibre entre les paires de processeurs s'effectue de manière uniforme ce qui influe négativement sur la vitesse de convergence.

Couturier a proposé alors un nouvel algorithme GAE, dérivant de GDE, pour lequel le choix des paires de voisins n'est pas imposé dans l'algorithme. Ce choix doit cependant être réalisé selon une stratégie (aléatoire, arbitraire (GDE) ou plus sophistiquée) respectant les contraintes des algorithmes de type Dimension Exchange[8].

Flavien Vernier a proposé un algorithme de diffusion relaxé, axé sur la diffusion et permet l'accélération par relaxation de la vitesse d'équilibrage de la diffusion [9]. Cet algorithme intègre un paramètre de relaxation qui doit être correctement choisi afin d'accélérer la convergence. Ce choix est limité par plusieurs contraintes, ce qui complique l'application de ce type d'algorithme dans des topologies dynamiques.

Les travaux de recherche et les implémentations performantes en matière d'optimisation de temps de réponse et en matière d'exploitation de ressources inoccupées tirent le bénéfice des algorithmes itératifs qui ont une validité expérimentale. Une infrastructure de gestion de la grille à base d'agents associée à un ordonnanceur de tâche a été développée pour garantir l'équilibrage de charge de la grille locale ; chaque ordonnanceur utilise les données des applications prédictives [2] (predictive application performance data) et un algorithme heuristique itératif pour établir l'équilibrage de charge localement à travers les multiples nœuds de traitement [10].

Cet article propose un nouvel algorithme itératif de diffusion qui élimine l'inconvénient généré par l'équilibrage voisin par voisin en utilisant un principe de diffusion équitable tout en simplifiant le choix des paramètres de la matrice de diffusion. Le temps de réponse est ainsi optimisé et offre une meilleure exploitation des ressources de traitement et de communication dans un environnement distribué.

II. ALGORITHME PROPOSE

Pour étudier les comportements des algorithmes de diffusion et analyser les différents paramètres qui entrent en jeu pendant le processus d'équilibrage, nous nous appuyons sur des résultats de simulation. Des tests sont établis pour démontrer l'évolution des charges, prouver la convergence de l'algorithme proposé ainsi que son efficacité.

II.1 Principe de fonctionnement de l'algorithme de diffusion génétique

Les algorithmes de diffusion diffusent la charge initiale de proche en proche sur les autres nœuds jusqu'à l'obtention d'une charge équitable sur l'ensemble des nœuds (fig. 1.).

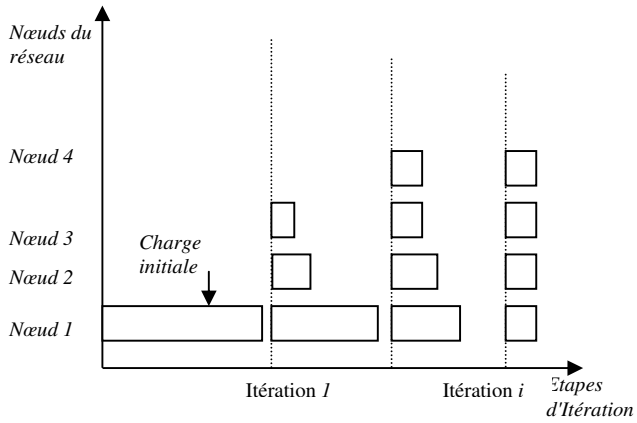

Fig.1. Principe des algorithmes de diffusion.

Dans l'algorithme de diffusion génétique, chaque nœud partage sa charge en deux portions; il garde une portion pour lui et il diffuse l'autre portion à ses voisins d'une façon équitable, selon le principe de la méiose (méiose ; division cellulaire qui, à partir d'une cellule somatique, produit deux cellules filles, toutes deux identiques, et également identiques à la cellule mère) (fig. 2.).

II.2 Modélisation de l'algorithme de diffusion génétique

L'algorithme de diffusion trouve une formulation mathématique comme suit [9] :

$$W_i^{(t+1)} = w_i^{(t)} + \sum \alpha_{ij} (w_j^{(t)} - w_i^{(t)}) + \eta_i^{(t+1)} - c \qquad (1)$$

avec :

$w_i^{(t)}$ la charge du processeur i à l'instant t.

α_{ij} : est un paramètre d'échange dans le cas d'algorithmes d'équilibrage de charge de type diffusion ; il représente une fraction de la différence de charge entre les processeurs i et j.

$\eta_i^{(t+1)}$: est la charge de travail qui est créée sur le processeur i à l'instant t.

c :est une constante représentant la charge de travail consommée par un processeur en une itération.

L'équation (1) peut être simplifiée, si l'on considère que la charge est statique, c'est à dire la charge demeure constante entre chaque itération. ($\sum_{i=1}^{n} w_i^{(t)} = \sum_{i=1}^{n} w_i^{(t+1)}$).

Pour une charge statique, le modèle s'exprime simplement en considérant que $\eta_i^{(t+1)} = 0$ à tout instant t et $c = 0$. Dans ce cas, l'équation (1) peut se mettre sous la forme:

$W^{(t+1)} = MW^{(t)}$ où:

$W(t)$ est le vecteur de dimension n contenant la charge de tous les processeurs à l'instant t.

M est la matrice de diffusion définie par m_{ij} telle que :

$$m_{ij} = \begin{cases} \alpha_{ij}, & \text{si } i \neq j, \\ 1 - \sum_j \alpha_{ij} & \text{si } i = j \quad (i,j = 1..n) \end{cases}$$

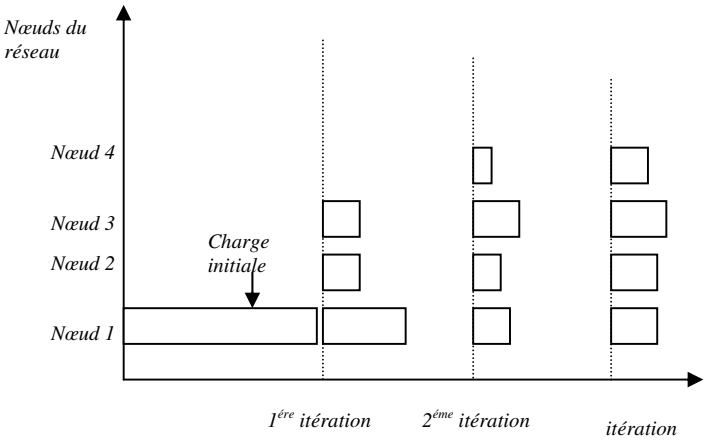

Fig. 2. Principe d'algorithme de diffusion génétique.

II.3 Détermination de la matrice de diffusion génétique

La matrice de diffusion génétique que nous avons définie s'écrit comme suit:

$$m_{ij} = \begin{cases} \alpha_{ij} = \dfrac{1}{2v_i} & Si\ i \neq j \\ \dfrac{1}{2} & Si\ i = j \\ 0 & Si\ l'arc\ (i,j) \neg\exists\ dans\ E \end{cases}$$

Dans la matrice de diffusion génétique, la somme de α_{ij} sur les colonnes donne *1*, ce qui assure une probabilité sur les colonnes $\sum_{i=1}^{n} \alpha_{ij} = 1$, contrairement aux algorithmes classiques de diffusion où la somme de probabilité est égale à *1* sur les lignes.

La matrice de diffusion proposée permet de générer une population initiale d'individus où chacun représente une solution au problème, à savoir une allocation possible. Dans notre cas on prend la charge initiale comme une population initiale et on reproduit cette charge à chaque itération grâce aux éléments $m_{ij.}$.

II.4 Etapes principales de l'algorithme proposé

Les algorithmes de diffusion supposent qu'un nœud d'un graphe peut envoyer et recevoir une charge de/vers tous ses voisins simultanément à chaque itération. Après un certain nombre d'itérations un état équilibré est atteint. Ces algorithmes itératifs sont souvent utilisés en raison de leur caractère totalement distribué et de leur générécité. Ce qui leur permet d'être appliqué sur n'importe quel système nécessitant un équilibrage de charge [11].

Lorsque un déséquilibre survient ou le seuil maximum/minimum est atteint sur un nœud du système, l'algorithme d'équilibrage doit répartir la charge, donc les données, de telle sorte que le temps global d'exécution soit réduit.

On note tout de même que ces algorithmes exigent quelques contraintes d'entrées :

- nombre de nœuds *:* l'information de l'ensemble des processeurs du réseau est nécessaire pour répartir la charge du travail.
- les seuils min et max : la charge très faible ou la charge très haute permet le déclenchement du processus d'équilibrage ainsi que l'arrêt de ce dernier.
- état du système à équilibrer: la répartition de la charge initiale du système à équilibrer est fondamentale pour pouvoir commencer le processus d'équilibrage.

Le diagramme suivant donne les principales phases de l'algorithme proposé:

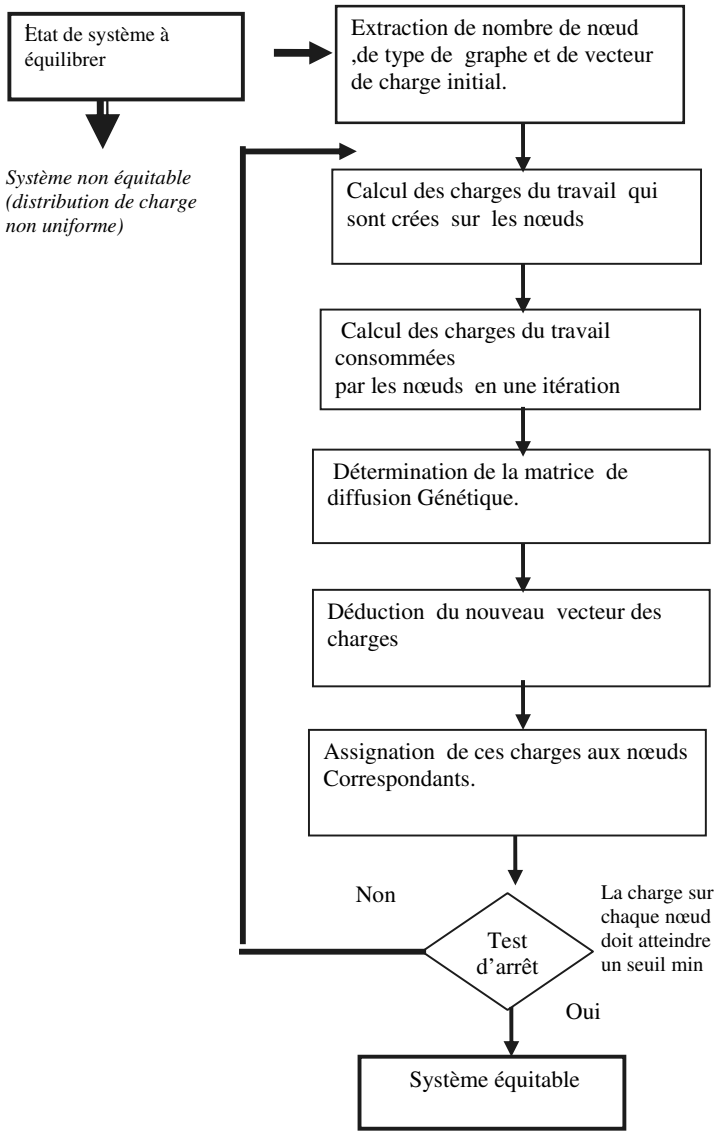

Fig. 3. Les étapes principales de l'algorithme.

II.5 Implémentation

Les méthodes itératives ne sont pas généralement complexes à implémenter et à paralléliser. La parallélisation la plus simple consiste à distribuer le travail effectué dans une itération à l'ensemmble des nœuds de la grille. A chaque nouvelle itération, les processeurs s'échangent les données dont ils ont besoin et déterminent si le critère de convergence est atteint. Afin de tester le comportement de l'algorithme proposé, nous avons effectué plusieurs simulations dans différentes conditions.

Ces tests sont établis pour démontrer l'évolution des charges, la convergence de l'algorithme de diffusion génétique et l'efficacité de l'algorithme proposé qui répond au mieux aux performances souhaitées.

III. RESULTATS DE L'EXPERIMENTATION

On suppose une grille quelconque de nœuds, modélisé par le graphe suivant $G = (V,E)$ (Fig. 4.). Les éléments de la matrice de diffusion génétique (α_{ij}) représentent la fraction de la charge qui va être échangé entre le nœud *i* et le nœud *j*.

Pour initialiser les charges, une charge virtuelle a été placée sur tous les nœuds, de valeur *1000* pour le nœud *1* et de valeur *0* pour tous les autres.

Cette configuration initiale extrême du système permet de voir une évolution complète du vecteur d'équilibrage de charge.

En comparant la matrice de diffusion de l'algorithme de Cybenko [12] (Fig.6) avec celle obtenue avec l'algorithme de diffusion génétique (Fig.5), on remarque la convergence de l'algorithme dès les premières itérations. Cela s'explique bien du fait que la charge initiale se diffuse à chaque itération selon un principe de diffusion équitable.

Dans cette expérimentation, le nombre d'itérations nécessaires pour arriver à un équilibrage de système est de (*17*) itérations alors qu' avec les algorithmes de diffusions on obtient pour un choix de Cybenko *34* itérations et pour un choix de Boillat *34* itérations. On voit bien qu'ici l'algorithme de diffusion génétique améliore la vitesse de convergence.

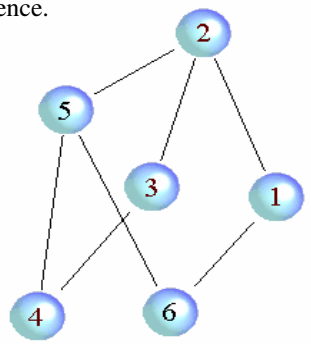

Fig.4. Graphe modélisant une grille de noeuds.

0.500	0.167	0.000	0.000	0.000	0.250
0.250	0.500	0.250	0.000	0.167	0.000
0.000	0.167	0.500	0.250	0.000	0.000
0.000	0.000	0.250	0.500	0.167	0.000
0.000	0.167	0.000	0.250	0.500	0.250
0.250	0.000	0.000	0.000	0.167	0.500

Fig.5. Matrice de diffusion génétique

0.417	0.250	0.000	0.000	0.000	0.333
0.250	0.250	0.250	0.000	0.250	0.000
0.000	0.250	0.417	0.333	0.000	0.000
0.000	0.000	0.333	0.417	0.250	0.000
0.000	0.250	0.000	0.250	0.250	0.250
0.333	0.000	0.000	0.000	0.250	0.417

Fig.6. Matrice de diffusion; choix de Cybenko

III.1 Influence de la topologie

La topologie du réseau a une large influence sur l'évolution des algorithmes d'équilibrage de charge. A cet effet, on va prendre des topologies différentes dont chacune possède le même nombre de nœuds afin que la comparaison soit sensée. Initialement, toute la charge du système est placée sur le processeur 0, tous les autres ayant une charge nulle. Le nombre d'itérations nécessaires afin d'obtenir une répartition uniforme de la charge est donné dans le tableau1 (voir Tableau.1).

Les tests présentent la vitesse de convergence des algorithmes en fonction de la topologie sur laquelle ils sont appliqués. On constate bien que si la topologie est fortement couplée l'équilibrage se fait dans un temps optimal, si elle est faiblement couplée, le nombre d'itérations nécessaire augmente.

	Grille q 6 nœuds	Ligne 9nœuds	Anneau 9 nœuds	Grille q 8 nœuds	Grille 3D 3x3 noeud
α_{Cyb}	34	211	52	32	42
α_{Boi}	34	211	52	40	35
α_{ODF}	//	138	52	//	32
$\alpha_{Diff\ gén}$	17	200	60	28	35

Tableau. 1. Nombre d'itérations selon les différentes topologies.

Ces résultats montrent bien, que la vitesse de convergence de l'algorithme de diffusion génétique est supérieure à celles des algorithmes de diffusion notamment dans les topologies maillées. L'algorithme de diffusion génétique donne une grande partie de la charge aux nœuds qui ont un nombre maximal de voisins. Dans le cas des graphes qui ont une faible connexité (comme le cas d'un brin), les performances de l'algorithme de diffusion génétique diminuent.

III.2 Influence de l'hétérogénéité

L'hétérogénéité du réseau influe sur les paramètres de l'algorithme d'équilibrage. Pour déterminer un α optimal, dans le cas hétérogène, une méthode générale a été proposée. Elle consiste à calculer α comme suit :

$\alpha = \min_i (\alpha_{opt}, 1/L)$

Tel que :

L : laplacien général du graphe et $L=(AA^TC^{-1})$.

A : matrice d'incidence nœud arc du graphe *G*.

C : matrice carrée (n*n) composée des c_i sur la diagonale et de 0 ailleurs.

L'utilisation de la matrice de laplacien permet de rendre efficace l'utilisation des algorithmes de diffusion dans le cas hétérogène. Des détails de la matrice de laplacien sont explicités par Robert Elsässer dans [13].

Dans le cas de l'algorithme de diffusion génétique on peut également déterminer un α optimal pour généraliser l'utilisation de cet algorithme si l'on tient compte des poids de communication.

Dans le cas de grilles de calcul dynamique, certains liens de communication peuvent être temporairement ou non, perdus ou surchargés. Les algorithmes de diffusions ont leurs adaptations à ce type de grille *en* utilisant des conditions de convergences.

IV. Conclusion

L'algorithme de diffusion génétique développé, optimise bien la vitesse de convergence. La diffusion permet une accélération de la convergence d'équilibrage dès les premières itérations. Le critère d'arrêt de l'algorithme basé sur des fractions génétiques, converge bien en un nombre déterminé d'itérations. Cette convergence est assurée par l'intégration des deux seuils minimum et maximum. Les structures de données utilisées assurent la portabilité de cet algorithme sur n'importe quelle type de topologie de grille de calcul.

REFERENCE

[1] Frédéric Vivien, Stéphane Genaud, Arnaud Giersch, *Load-Balancing Scatter Operations for Grid Computing.* Mars 2003,INRIA.

[2] D.J. Kerbyson, J. Harper, A. Craig, G.R. Nudd PACE: A Toolset to Investigate and Predict 5. Performance in Parallel Systems. High Performance Systems Group, University of Warwick.

[3] C.Z. Xu, B. Monien, R. L¨uling, and F.C.M. Lau. First and second order diffusive methods for rapid coarse distributed load balancing.2002

[4] Kamel MAZOUZI. JACE: un environnement d'exécution distribué pour le calcul itératif asynchrone. Décembre 2004 (Doctorat en informatique) Franche-Comté.

[5] Dynamic load balancing for distributed memory multiprocessors. *G. Cybenko.*

[6] C.Z. Xu and F.C.M. Lau. Analysis of the generalized dimension exchange method for dynamic load balancing. Journal of Parallel and Distributed Computing, 1992.

[7] M.Xu, F.C.M. Lau, The Generalized Dimension Exchange Method for Load Balancing in K-ary n-cubes and variants.1995

[8] Raphaël COUTURIER. Algorithmes itératifs asynchrones sur grappes distantes et équilibrage de charge sur topologies dynamiques. Université de Franche-Comté (thèse hdr 2005)

[9] Flavien VERNIER. Algorithmique itérative pour l'équilibrage de charge dans les réseaux dynamiques. Décembre 2004 (Doctorat en informatique) Franche-Comté.

[10] Junwei Cao , Daniel P. Spooner , Stephen A. Jarvis , Subhash Saini , and Graham R. Nudd. Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling. 2003

[11] Rob V. van Nieuwpoort, Jason Maassen. Adaptive Load Balancing for Divide-and-Conquer Grid Applications. Vrije Universiteit, Amsterdam, The Netherlands.2004

[12] J.E. Boillat Load balancing and poisson equation in a graph.

[13] Robert Elsässer. Spectral Méthods of Efficient Load Balancing Stratégies.2002