Modèle Arborescent pour l'Equilibrage de Charge dans les Grilles de Calcul

YAGOUBI Belabbas

Université d'Oran Es Sénia,
Département d'Informatique,
ORAN, ALGERIE.
byagoubi@yahoo.fr

RÉSUMÉ. Afin d'obtenir de meilleures performances dans un système parallèle ou distribué, il est primordial d'équilibrer la charge de travail à travers toutes ses ressources. En d'autres termes il est recommandé d'éviter toute situation où certaines ressources sont surchargées alors que d'autres le sont légèrement ou sont complètement libres. La majorité des travaux relatifs à l'équilibrage de charge se sont focalisés sur des systèmes où les ressources et les réseaux de communication sont homogènes.
Les dernières évolutions dans le calcul distribué ont conduit à l'apparition de nouvelles infrastructures appelées *grilles de calcul* dont les principales caractéristiques sont l'*hétérogénéité*, la *dynamicité* et le *passage à l'échelle*, ce qui rend le problème d'équilibrage indispensable et plus complexe.
Dans cet article, nous proposons un modèle arborescent d'équilibrage de charge pour les grilles de calcul. Ce modèle possède les propriétés suivantes : (i) il est *hiérarchique*;(ii) il supporte l'*hétérogénéité* et le *passage à l'échelle*; (iii) il est totalement *indépendant* de toute topologie de grille particulière.

ABSTRACT. In order to get a better performance in distributed systems, it is essential to distribute the workload among their resources. In other words, it is desirable to avoid the situation where some resources are overloaded with a backlog of jobs to be serviced while other are lightly loaded or even idle. To achieve this goal several load balancing strategies and algorithms have been proposed. Generally, these strategies suppose that resources and networks are homogeneous in terms of capabilities. Recent advances in networks and architectures have led to a new type of computing, namely the *grid computing*. Key properties of a grid are *heterogeneity*, *dynamicity* and *scalability*. These properties require more complex load balancing techniques.
In this paper, we propose a load balancing model based on a tree structure. This model is characterized by three elements: (i) it's *hierarchical*; (ii) it supports *heterogeneity* and *scalability* of grid's; and, (iii) it's totally *independent* from any physical topology of a grid.

MOTS-CLÉS : Grilles de calcul, Equilibrage de charge, Modèle arborescent, Charge de travail, Performances.

KEYWORDS : Grid computing, Load balancing, Tree model, Workload, Cost.

1. Introduction

Depuis un certain nombre d'années, nous assistons à l'émergence d'applications de plus en plus exigeantes en capacité de calcul et de stockage pour lesquelles les architectures actuelles s'avèrent insuffisantes [1]. Comme exemples , nous pouvons citer les applications de météorologie, de calcul intensif, de datamining, de recherche de séquences ADN, etc... Une des solutions adoptée, pour satisfaire les besoins de ces applications , consiste à agréger les ressources disséminées à travers le monde en utilisant des infrastructures existantes tels que Internet. Ceci a permis le développement d'un nouveau concept de calcul appelé *calcul de grille* ou *grid computing*. Les principales caractéristiques des architectures supportant ce type de calcul, appelées *grilles de calcul*, sont l'*hétérogénéité*, le *passage à l'échelle* et la *dynamicité* [3]. Ce type d'infrastructure peut être facilement déployé en utilisant comme ressources de calcul les réseaux de PC à une échelle planétaire. Cependant la gestion de ces ressources pose évidemment des problèmes beaucoup plus complexes que ceux posés par les systèmes distribués traditionnels, et ce à cause notamment de leur hétérogénéité et de leur dimension dynamique. Parmi ces problèmes, la répartition de la charge de travail sur les diverses ressources disponibles d'une grille s'avère être un véritable défi. Les objectifs d'une politique d'équilibrage de charge peuvent être de trois types : (i) minimisation du temps moyen de réponse des applications ; (ii) maximisation du degré d'occupation des ressources ; et, (iii) réduction des coûts de communication [4, 5]. Dans ce cadre, nous proposons, dans cet article, un modèle arborescent pour l'équilibrage de charge avec comme objectif la réduction du temps de réponse moyen des tâches présentes dans une grille de calcul. Les principales caractéristiques de ce modèle sont : (i) la transformation simple, unique et bijective de toute architecture de type grille en arbre ; (ii) la prise en charge, au niveau du modèle proposé, des propriétés de dynamicité, d'hétérogénéité et de passage à l'échelle d'une grille ; et, (iii) l'indépendance totale du modèle par rapport à toute topologie physique d'une grille.

Le reste de cet article est organisé comme suit : la section 2 décrit les aspects essentiels du problème de l'équilibrage. La section 3 présente le modèle arborescent proposé pour traiter le problème de l'équilibrage de charge dans les grilles de calcul. La stratégie d'équilibrage ainsi que l'algorithmique associée sont définies dans la section 4. Dans la section 5, nous présenterons et discuterons quelques résultats expérimentaux relatifs à l'implémentation de l'instance 1/1/M du modèle proposé. Enfin, la section 6 conclut cet article et présente quelques perspectives futures de recherche.

2. Problème de l'équilibrage

2.1. Politiques d'équilibrage de charge

L'équilibrage de charge couvre l'ensemble des techniques permettant une distribution équitable de la charge de travail parmi les ressources disponibles d'un système. L'objectif consiste essentiellement à optimiser le temps de réponse moyen d'un ensemble de tâches, ce qui revient souvent à maintenir une

charge proportionnellement équivalente sur l'ensemble des ressources de calcul. Les méthodes d'équilibrage de charge se distinguent généralement par cinq politiques [2, 7] : (i) la *politique d'information* qui consiste à définir l'information de charge, l'instant de sa collecte et à partir de quelle (s) ressource(s) elle émane ; (ii) la *politique de déclenchement* qui détermine le moment opportun pour entamer une opération d'équilibrage de charge ; (iii) la *politique de participation* qui partitionne l'ensemble des ressources en *sources* (surchargées), *destinations* (sous-chargées) ou *neutres* (équilibrées) ; (iv) la *politique de correspondance* qui détermine les paires convenables (source,destination) ; et, (v) la *politique de sélection* qui choisit les tâches à faire migrer à partir des ressources *sources* vers les ressources *destinations*.

2.2. Problématiques particulières liées aux grilles de calcul

Les spécificités des grilles de calcul font que le problème d'équilibrage de charge devient beaucoup plus complexe que dans le cas des systèmes parallèles et distribués traditionnels, qui peuvent offrir une certaine *homogénéité* et une *stabilité* des ressources qui les composent. Ces deux facteurs font malheureusement défaut dans le cas des grilles de calcul. Par exemple, le passage à l'échelle peut être à la fois important et brusque (augmentation ou diminution significative des ressources de calcul). D'autre part, les réseaux d'interconnexion au niveau des grilles présentent des performances très diversifiées en ce qui concerne les largeurs des bandes passantes. Enfin, les tâches soumises au système peuvent être de nature très irrégulières. Ces différentes caractéristiques montrent qu'il est difficile, voire impossible, de définir un système d'équilibrage qui puisse intégrer tous ces facteurs. Pour cela, nous serons amenés à faire un certain nombre d'hypothèses et de restrictions afin que le modèle proposé puisse être viable.

En tout état de cause, tout système d'équilibrage de charge devra en premier lieu permettre d'estimer l'état de charge instantané de chaque ressource. Cet état constitue l'information clé où il s'agira alors de préciser [6] :
(i) Quels critères retenir dans la définition de la charge d'une ressource ?
(ii) Comment mesurer cette charge ?
(iii) Comment éviter les fluctuations subites ?
(iv) Comment prendre en compte l'hétérogénéité des ressources pour obtenir une moyenne représentative de la charge instantanée de tout le système ?

3. Modèle proposé

D'un point de vue topologique, nous supposons qu'une grille est composée de **G** *Grappes* (ou *Clusters*) C_k reliées par des passerelles. Chaque grappe est elle-même composée de **S** sites S_{jk} interconnectés par des switchs. Chaque site est à son tour composé de **M** Eléments de Calcul EC_{ijk} et de **N** Eléments de Stockage ES_{ljk}, qui communiquent à travers un réseau local. L'ensemble de ces éléments et moyens de communications peuvent être *hétérogènes* sur le plan des architectures, des systèmes d'exploitation et des bandes passantes.

3.1. Modèle générique associé à une grille : *le modèle G/S/M*

Pour représenter une grille de calcul, nous proposons de la transformer de manière univoque en un arbre d'interconnection virtuel. Cet arbre nous donne un modèle de représentation générique, noté **G/S/M**, où **G** représente le nombre de grappes (clusters), **S** le nombre de sites et **M** le nombre d'éléments de calcul. Ce modèle peut être instancié sous trois formes : **G/S/M**, **1/S/M** ou **1/1/M**, illustrées par la *figure* 1. L'arbre de représentation de la grille est composé de quatre niveaux :

– **Niveau 0** : Ce niveau, qui correspond à la racine de l'arbre, est constitué d'un nœud associé à toute la grille. Appelé *Gestionnaire de grille*, il a pour rôle de : (i) maintenir l'information de charge sur l'ensemble de la grille ; (ii) estimer le seuil de charge moyen ; (iii) décider d'un équilibrage global entre les clusters de la grille ; et, (iv) envoyer les décisions d'équilibrage aux clusters pour exécution.

– **Niveau 1** : Ce niveau est rattaché aux clusters de la grille où chaque nœud est appelé *Gestionnaire de sites*. Il permet de maintenir l'information de charge sur chacun de ses sites, d'estimer sa charge moyenne, de décider d'un équilibrage local et d'envoyer les décisions d'équilibrage aux sites qu'il gère.

– **Niveau 2** : Ce niveau est composé de nœuds d'arbre qui correspondent aux sites de la grille. Appelé *Gestionnaire d'éléments de calcul*, chaque nœud a pour rôle de : (i) gérer l'information de charge relative à ses EC's , (ii) maintenir son état de charge,(iii) décider de déclencher un équilibrage local, et, (iv) informer les EC's pour mettre en oeuvre un équilibrage de charge.

– **Niveau 3** : Chaque nœud de ce niveau représente un EC ayant pour fonction de maintenir à jour l'information sur son état de charge, de l'envoyer périodiquement à son gestionnaire et d'exécuter l'opération d'équilibrage décidée par son gestionnaire.

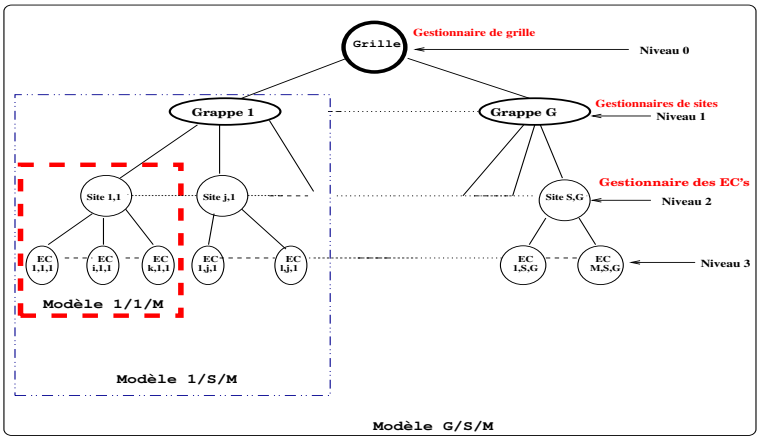

Figure 1. *Modèle générique de représentation d'une grille*

Le modèle *G/S/M* décrit ci-dessus est une agrégation de **G** modèles *1/S/M* , qui à leur tour sont des agrégations de **S** modèles *1/1/M*.

4. Algorithme d'équilibrage

La structure d'arbre que nous avons proposé nous permet de définir trois niveaux d'algorithmes d'équilibrage : intra-site, intra-cluster et inter-clusters.

Notations : Les notations suivantes sont utilisées dans la description des algorithmes [8].
$C_k = k^{ieme}$ Cluster ; $S_{jk} = j^{ieme}$ site du k^{ieme} cluster ; $EC_{ijk} = i^{ieme}$ élément de calcul du j^{ieme} site du k^{ieme} cluster ; T = facteur de tolérance de charge exprimé en (%) ; L_{ijk} = charge courante de EC_{ijk} ; L_{jk} = charge courante de S_{jk} ; L_k = charge courante de C_k ; $Seuil$ = charge moyenne de la grille ; sit - max = seuil de déclenchement d'un équilibrage intra-cluster ; clu-max = seuil de déclenchement d'un équilibrage inter-clusters ; $card(E)$ = cardinal de l'ensemble E.

Equilibrage Intra-Site : Cas d'un site S_{jk}

Début
Pour chaque EC_{ijk} **ET** Périodiquement **faire**
  Mise à Jour de la charge actuelle L_{ijk} de EC_{ijk}
  Envoi vers le gestionnaire des éléments de calcul S_{jk}
Fin Pour
- Mise à Jour de la charge L_{jk} du site S_{jk}
- Envoi vers le gestionnaire de sites C_k
- Réception du seuil de charge moyen à partir de C_k
Si ($\frac{|L_{jk}-Seuil|}{L_{jk}} > T$) **Alors** État de déséquilibre **Sinon** Retourner **Fin Si**
[Partitionnement des EC's de S_jk en surchargés, sous-chargés et équilibrés]
ECS$\leftarrow \emptyset$;ECR$\leftarrow \emptyset$;ECN $\leftarrow \emptyset$
Pour Chaque EC_{ijk} de S_{jk} **faire**
  Selon que
    $L_{ijk} > Seuil + T : ECS \leftarrow ECS \cup \{EC_{ijk}\}$
    $L_{ijk} < Seuil : ECR \leftarrow ECR \cup \{EC_{ijk}\}$
    $Seuil \leq L_{ijk} \leq Seuil + T : ECN \leftarrow ECN \cup \{EC_{ijk}\}$
  Fin Selon que
Fin Pour
Tant que (ECS $\neq \emptyset$ **ET** ECR $\neq \emptyset$) **faire**
  - Trier ECS par ordre décroissant des charges courantes L_{ijk}
  - Trier ECR par ordre croissant de charges courantes L_{ijk}
  - $EC_{sjk} \leftarrow$ EC le plus surchargé ; $EC_{rjk} \leftarrow$ EC le moins chargé
  - Charge offerte par $CE_{rjk} = Seuil - L_{rjk}$
  - Phase de migration de tâches de EC_{sjk} vers EC_{rjk}
  - Mise à Jour des charges de EC_{sjk}, EC_{rjk} et des ensembles ECS et ECR
Fait
Si ($card(ECS) \geq sit\text{-}max$) **Alors**
  Défaut d'Equilibrage Intra-Site ; recours à un équilibrage intra-cluster
Sinon
  Equilibrage réalisé avec succès
Fin Si
Fin.
Commentaire : Cet algorithme est mis en oeuvre quand le gestionnaire des EC's constate qu'il y a un déséquilibre de charge entre les EC's qui sont sous son contrôle. Pour faire ce constat, le gestionnaire reçoit, de manière périodique, des EC's leur information de charge. A partir de ces informations et d'un seuil de charge moyen, il analyse de manière régulière la charge du site. En fonction du résultat de cette analyse, soit il décide de déclencher un équilibrage local (entre EC's du même site) en cas de déséquilibre, soit d'avertir son gestionnaire du niveau supérieur de sa charge actuelle.

4.1. Algorithmes d'équilibrage intra-cluster & inter-clusters

Equilibrage Intra-Cluster : Cas du cluster C_k

Début
Pour chaque site S_{jk} de C_k **ET** Périodiquement **faire** Mise à jour de la charge actuelle L_{jk} et Envoi vers le gestionnaire des sites C_k **Fin Pour**
- Mise à jour de la charge L_k de C_k et Envoi vers le gestionnaire de la grille
- Réception du seuil de charge moyen à partir du gestionnaire de la grille
Si (Nombre de sites surchargés $\geq$ *sit-max*) **Alors** le cluster C_k est déséquilibré **Sinon** Retourner **Fin Si**
[Partitionnement des sites de C_k en surchargés, sous-chargés et équilibrés]
SS $\leftarrow \emptyset$; SR $\leftarrow \emptyset$; SN $\leftarrow \emptyset$
Pour Chaque S_{jk} de C_k **faire**
  Selon que
    $L_{jk} > Seuil + T : SS \leftarrow SS \cup \{S_{jk}\}$
    $L_{jk} < Seuil : SR \leftarrow SR \cup \{S_{jk}\}$
    $Seuil \leq L_{jk} \leq Seuil + T : SN \leftarrow SN \cup \{S_{jk}\}$
  Fin Selon que
Fin Pour
Tant que (SS $\neq \emptyset$ **ET** SR $\neq \emptyset$) **faire**
  - Tri de SS par ordre décroissant de charges courantes L_{jk}
  - Tri de SR par ordre croissant de charges courantes L_{jk}
  - $S_{lk} \leftarrow$ Site le plus surchargé ; $S_{rk} \leftarrow$ Site le moins chargé
  - Charge offerte par $S_{rk} = Seuil - L_{rk}$
  - Décision de migration de tâches de S_{lk} vers S_{rk}
  - Déclenchement d'un équilibrage intra-site
Fait
Si ($card(SS) \geq$ *clu-max*) **Alors** Défaut d'équilibrage Intra-Cluster ; Recours à un équilibrage Inter-Clusters **Sinon** Equilibrage réalisé avec succès **Fin Si**
Fin.

Equilibrage Inter-Clusters(GLOBAL)

Début
Pour chaque C_k **ET** Périodiquement **faire** Mise à jour de la charge actuelle L_k et envoi vers le gestionnaire global de la grille **Fin Pour**
Mise à jour de la charge globale de la grille ; calcul du seuil de charge moyen et diffusion à tous les clusters
Si (Nombre de clusters surchargés $\geq$ *clu-max*) **Alors** Grille déséquilibrée **Sinon** Retourner **Fin Si**
[Partitionnement des clusters en surchargés, sous-chargés et équilibrés]
CS $\leftarrow \emptyset$; CR $\leftarrow \emptyset$; CN $\leftarrow \emptyset$
Pour Chaque C_k **faire**
  Selon que
    $L_k > Seuil + T : CS \leftarrow CS \cup \{C_k\}$
    $L_k < Seuil : CR \leftarrow CR \cup \{C_k\}$
    $Seuil \leq L_k \leq Seuil + T : CN \leftarrow CN \cup \{C_k\}$
  Fin Selon que
Fin Pour
Tant que (CS $\neq \emptyset$ **ET** CR $\neq \emptyset$) **faire**
  - Tri de CS par ordre décroissant et de CR par ordre croissant de charges L_k
  - $C_s \leftarrow$ Cluster le plus surchargé ; $C_r \leftarrow$ Cluster le moins chargé
  - Décision de migration de tâches de C_s vers C_r
  - Déclenchement d'un équilibrage intra-Cluster
Fait
Fin.

5. Expérimentations

Afin de tester le fonctionnement et d'évaluer les performances de notre modèle, nous avons développé un simulateur de grilles de calcul en Java à l'aide duquel nous pouvons : (i) générer le fichier de configuration d'une grille test (nombre de sites, nombre d'éléments de calcul, leurs caractéristiques, période d'envoi des informations de charge, largeur de bandes, ...) ; (ii) générer un ensemble de tâches avec toutes les données associées (date de soumission, temps d'exécution estimé, taille, priorité,...). Comme indice de charge, nous avons adopté le taux d'occupation des éléments de calcul. En ce qui concerne les mesures de performance, nous nous sommes intéressés au temps de réponse moyen de l'ensemble des tâches soumises durant une période déterminée.

Pour obtenir des résultats qui soient les plus fiables possibles, nous avons réitéré les mêmes expériences plus de dix (10) fois. L'ensemble de ces expériences ont été réalisées sur un PC Pentium IV de 2.8 GHz, doté d'une mémoire de 256 Mo et fonctionnant sous Windows XP.

Pour des contraintes de limitation du nombre de pages de l'article, nous nous contenterons de présenter les résultats des expériences portant sur l'implémentation de l'algorithme intra site. Nous mettrons en évidence le temps de réponse d'une part en fonction du nombre de tâches et d'autre part en fonction du nombre d'éléments de calcul. Les résultats obtenus sont résumés dans les tableaux 1 et 2.

Nombre d'EC's	**Temps de réponse(sec)**			Com (Sec)
	Avant	Après	Gain(%)	
50	817	730	10.65	41
100	409	353	13.69	50
150	273	230	15.75	49
200	126	99	21.43	47
250	164	139	15.24	49

Tableau 1. *Temps de réponse VS nombre d'EC's (nombre de tâches fixe = 2000)*

Nombre de tâches	**Temps de réponse(sec)**			Com (Sec)
	Avant	Après	Gain(%)	
1000	113	95	15.93	22
1250	134	114	14.93	33
1500	145	122	15.86	37
1750	156	132	15.38	47
2000	164	139	15.24	49

Tableau 2. *Temps de réponse VS nombre de tâches (nombre d'EC's fixe = 250)*

Où **EC's** représente les éléments de calcul ; **Avant** : avant équilibrage ; **Après** : après équilibrage ; **Gain** : profit réalisé (en %) et **COM** : coût de communication (en secondes).

Interprétation des résultats :

1. Pour un nombre d'EC's fixé à **250** et en faisant varier le nombre de tâches de **1000** à **2000** par pas de **250**, nous avons réalisé un gain variant de **14.93%** à **15.93%** sur le temps de réponse moyen avec un surcoût relativement négligeable ;
2. Pour un nombre de tâches fixé à **2000** et en faisant varier le nombre d'EC's de **50** à **250** par pas de **50**, nous avons obtenu un profit variant de **10.65%** à **21.43%** sur le temps de réponse.
3. Les meilleurs gains sont obtenus lorsque le système est dans un état stable (ni trop surchargé, ni trop sous-chargé).

6. Conclusion et perspectives

Nous avons proposé un modèle d'équilibrage adapté aux grilles de calcul qui tient compte surtout de l'hétérogénéité des ressources et qui est totalement indépendant de toute architecture physique. Partant d'une structure arborescente du modèle proposé, nous avons développé une stratégie algorithmique hiérarchique privilégiant, quand cela est possible, un équilibrage de charge local, ce qui évite le recours au réseau de communication. Ainsi, cette stratégie d'équilibrage nous permet de réduire le nombre de communications ainsi que le volume de données échangées (messages d'équilibrage et migration de tâches). Afin d'implémenter et de valider le modèle proposé, nous avons développé un simulateur de grilles qui permet de générer, de manière aléatoire, des éléments de calcul et des tâches avec différentes caractéristiques. Les premiers résultats d'expérimentations sont très encourageants dans la mesure où nous arrivons à diminuer sensiblement le temps de réponse moyen, avec un overhead négligeable par rapport au temps de réponse total . Pour mesurer l'efficacité du simulateur , nous envisageons de comparer ses performances avec des simulateurs de grilles tels que *GridSim* et *SimGrid* . Nous envisageons également de l'implanter sur une grille grandeur nature en utilisant l'environnement *GLOBUS*.

7. Bibliographie

[1] A. CHERVENACK AND E. DEELMAN AND C. KESSELMAN AND B. ALLCOCK AND I. FOSTER AND V. NEFEDOVA AND J. LEE AND A. SIM AND A.SHOSHANI AND B. DRACH AND D. WILLIAMS AND D. MIDDLETON. « High performance remote access to climate simulation data : a chalenge problem for data grid technologies. » In *Proc. of 22th Parallel Computing*, Vol. 29 Issue 10, pages 13–35, 1997.

[2] E. CARON AND V. GARONNE AND A. TSAREGORODTSEV. « Evaluation of meta-scheduler architectures and task assignment policies for high throughput computing. » Technical Report C2005-27, École Normale Supérieure de Lyon, May 2005.

[3] I. FOSTER AND C. KESSELMAN. *The Grid : Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[4] DANIEL J. HARVEY. « Load Balancing Techniques for Distributed Processing Environments. » PhD Thesis, Faculty of the Graduate School of The University of Texas at Arlington, August 2001.

[5] M. ARORA AND S.K. DAS AND R. BISWAS. « A de-centralized scheduling and load balancing algorithm for heterogeneous grid environments. » In *Workshop on Scheduling and Resource Management for Cluster Computing*, Vancouver, Canada, pages 499–505, Aug. 2002.

[6] R.F. DE MELLO AND L.C. TREVELIN AND M.S.V. DE PAIVA AND L.T. YANG. « Comparative analysis of the prototype and the simulator of a new load balancing algorithm for heterogeneous computing environment. » *Internatinal journal of high performance computing and network (IJHPCN)*, vol. 1, Issue 1/2/3, 2004.

[7] N.G. SHIVARATRI AND P. KRUEGER AND M. SINGHAL. « Load distributing for locally distributed systems. » In *IEEE Computer*, vol. 25(12), pages 33–45, December 1992.

[8] B. YAGOUBI. « Dynamic Load Balancing for Beowulf Clusters. » *International Arab Conference on Information Technology , December* $6-8^{th}$*, 2005 ; Israa University, Jordan, Journal ISSN :1812-0857, Pages 394-401*.