REPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE
****
UNIVERSITE DE DSCHANG
****
ECOLE DOCTORALE

REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND
****
UNIVERSITY OF DSCHANG
****
POST GRADUATE SCHOOL

DSCHANG SCHOOL OF SCIENCES AND TECHNOLOGY
Unité de Recherche en Informatique Fondamentale, Ingénierie et Application (URIFIA)

PROPOSITION

Présenté par :
TCHIO AMOUGOU Styves daudet
Matricule : CM-UDS-14SCI0251
Licencié en Informatique Fondamentale

Sous la direction de
Dr BOMGNI ALAIN Bertrand
(*Chargé de Cours, Université de Dschang*)

Sommaire

Déclarations

- i = indice du Noeud i ;
- L_i = Liste des voisins du Noeud i ;
- $N_i = |L_i|$ = Nombre de listes de voisins du Noeud i ;
- M_i = Mémoire du noeud i :
- M_{ri} = Mémoire restante du noeud i ;
- M_{mi} = Mémoire supplémentaire requise par le Noeud i
- $Sawp_i$ = Mémoire swap du Noeud i ;
- $T_{i,j}$ = Taille de la liste des voisins du noeud j voisin du noeud i ;
- IN_i = Noeud de stockage probable ;

Déclarations

- $M =$ la Mémoire nécessaire pour l'exécution du protocole ; on a : $M = \sum_i \sum_i T_{i,j}$
- $IN_i = -1$ si le noeud i n'a pas besoin d'espace supplémentaire.

Conditions

Pour pouvoir exécuter le protocole de formation de clique il faut :

- $\sum_i M_{ri} > M$
- pour tout Noeud i du réseau,
  - $\sum_j \sum_k T_{j,k} < \sum_j M_{rj}$, $j \in L_i + \{i\}$ et $k \in L_j$

Notion de supériorité

Pour comparer les différents nœuds, nous définissons une relation " $>^{\alpha}$" sur les noeuds comme suit :

Définition : $i >^{\alpha} j$ si et seulement si :

- $M_i > M_j$ ou
- $M_i = M_j$ et $i > j$

Pour tout noeud i du réseau en manque de mémoire ;

- $\forall j \in L_i$ avec $M_{mj} > 0$, $M_{rj} = 0$ et $j >^{\alpha} i$.
  - si la différence entre la mémoire restant de *INi* et la mémoire manquante de j est positive.
    $M_{rIN_i} - M_{mj} > 0$
    - alors la mémoire restante de INi est réduit de la mémoire manquante de j $M_{rIN_i} := M_{rIN_i} - M_{mj}$
  - sinon la mémoire restante de IN_i passe a zéro.
    $M_{rIN_i} := 0$
- ce premier calcul permet au noeud i de vérifier si les noeuds avec qui il est en conflit et qui sont supérieur lui peuvent tous stocker leur données sur le noeud IN_i.
- si tous les noeuds supérieur a i peuvent stocker leurs données sur IN_i, le noeud i vérifie s'il reste de l'espace sur le Noeud IN_i.

- SI la mémoire restante de IN_i est positive,$M_{rIN_i} > 0$
  - Si la différence entre la mémoire restant de INi et la mémoire manquante de i est positive.
    $M_{rIN_i} - M_{mi} > 0$.
    - le noeud i considère IN_i comme noeud de stockage.
  - Sinon
    - la mémoire manquante du noeud i est réduit du reste de mémoire de IN_i. $M_{mi} = M_{mi} - M_{rIN_i}$,
    - $IN_i =$ noeud second de la mémoire virtuelle

- Dans les cas où le noeud i ne peut stocker aucune de ses listes sur le noeud IN_j
  - le noeud i retire le noeud IN_j de sa mémoire virtuelle.
  - $IN_j =$ noeud second de la mémoire virtuelle

programme

Algorithm 1 Choix des noeuds de stockages

$M_{Vi} = \{j \in L_i | M_{ri} > 0\}$ classées par ordre décroissant.
$S_i = \{j \in L_i | M_{mi} > 0 \text{ et } j >^{\alpha} i\}$ classées par ordre décroissant.
while $M_{rIN_i} > 0$ et $S_i \neq \emptyset$ **do**
  if $M_{rIN_i} - M_{mj} > 0$ **then**
    $M_{rIN_i} \leftarrow M_{rIN_i} - M_{mj}$
    $S_i \leftarrow S_i - j$
  end if
end while
if $M_{rIN_i} > 0$ **then**
  if $M_{rIN_i} - M_{mi} < 0$ **then**
    $M_{mi} \leftarrow M_{mi} - M_{rIN_i}$
    $IN_i \leftarrow M_{vi}[1]$
  end if
end if

Pour tout noeud i du réseau avec un reste de mémoire :

- $S_p = \emptyset$ (Ensemble des noeuds dont i doit stocker les listes)
- $\forall j \in L_i$ avec $M_{mj} > 0$, $M_{rj} = 0$ et $IN_j = i$.
  - si la différence entre la mémoire restant de i est positive. $M_{ri} > 0$
    - alors la mémoire restante de i est réduit de la mémoire manquante de j $M_{ri} := M_{ri} - M_{mj}$
    - $S_p = S_p \cup \{j\}$
  - sinon la mémoire restante de i passe a zéro.$M_{ri} := 0$
- Ce premier calcul permet au noeud i de déterminer les noeuds voisin qui ont un manque de mémoire et dont il peut satisfaire leur demande de stockage.

programme

Algorithm 2 Recherche des noeuds a satisfaire

$S_p = \emptyset$ (Ensemble des noeuds dont i doit stocker les listes)
$S_i = \{j \in L_i | M_{mi} > 0 \text{ et } IN_j = i\}$ classées par ordre décroissant.
for $j \in S_i$ **do**
  if $M_{rIN_i} > 0$ **then**
    $M_{rIN_i} \leftarrow M_{rIN_i} - M_{mj}$
    $S_p \leftarrow S_p \cup \{j\}$
  else
    Break
  end if
end for

Noeud en manque de mémoire

- Tous les noeuds en manque de mémoire qui ont changé de IN_i font une demande de M_{mi} a leur nouveau IN_i

Noeud avec surplus de mémoire

- $S_d = \emptyset$ (ensemble des noeuds dont i peut satisfaire les demandes)
- $\forall j \in L_i$ avec $M_{mj} > 0$, $M_{rj} = 0$, a reçu un demande de M_{dj}.
  - si la différence entre la mémoire restant de i et la mémoire demandée par j est positive.
    $M_{ri} - M_{dj} > 0$
    - alors la mémoire restante de i est réduit de la mémoire manquante de j $M_{ri} := M_{ri} - M_{dj}$
    - $S_d = S_d \cup \{j\}$
  - sinon break

Traitement des demandes

Algorithm 3 Traitement des demandes

$S_d = \emptyset$ (ensemble des noeuds dont i peut satisfaire les demandes)
$S_i = \{j \in L_i | M_{mi} > 0 \text{ et } M_{dj} > 0\}$ classées par ordre décroissant.
for $j \in S_i$ **do**
  if $M_{rlN_i} > 0$ **then**
    $M_{rlN_i} \leftarrow M_{rlN_i} - M_{dj}$
    $S_d \leftarrow S_d \cup \{j\}$
  else
    Break
  end if
end for

envoi des réponses

- Pour les noeuds j qui ont fait une demande de M_{dj} au noeud i,
  - si $j \in S_d$ la demande est approuvée ;
  - sinon envoyer une message de refus au noeud j
- Quand un noeud reçoit un réponse négative a une demande,
  - le noeud i retire le noeud a qui il a fait la demande de sa mémoire virtuelle.
- Par contre si le noeud i ne reçoit aucune réponse il considère que sa demande a été approuvée.

envoi des réponses

Algorithm 4 envoi des réponses

$S_d =$ (ensemble des noeuds dont i peut satisfaire les demandes)
$S_i = \{j \in L_i | M_{mi} > 0 \text{ et } M_{dj} > 0\}$ classées par ordre décroissant.
for $j \in S_i$ **do**
  if $j \notin S_d$ **then**
    envoyer une message de refus au noeud j
  end if
end for

Partage d'informations

- Partage d'informations sur les mémoires restantes du réseau aux différents noeuds du réseau encore en manque de mémoire.
- Recommencer a la première phase.

Remplissage de la table des pages

a la fin de chaque calcul, les mappeurs synchronise les différentes table de pages. pour que chaque noeuds ai les informations complètent sur tout les endroits ou il doit stocker ses données.

Remplissage de la table des pages

Algorithm 5 Synchronisation des tables des pages

S_d = Ensemble des noeuds dont i peut satisfaire les demandes.
S_p = Ensemble des noeuds dont i doit stocker les listes.
index = 0
for $j \in S_p$ **do**
  for $k = 0; k < M_{mj}/T; k++$ **do**
    $index \leftarrow index + k$
    $Tap_i[index].Owner \leftarrow j$
  end for
  envoyer($Tap_i[index - k]$ a $Tap_i[index]$ au noeud j)
end for
for $j \in S_d$ **do**
  for $k = 0; k < M_{dj}/T; k++$ **do**
    $index \leftarrow index + k$
    $Tap_i[index].Owner \leftarrow j$
  end for
  envoyer($Tap_i[index - k]$ a $Tap_i[index]$ au noeud j)

Théorème

Cette phase de calcul peut s'exécuter au maximum d fois avec d = dégrée le plus haut du réseau.

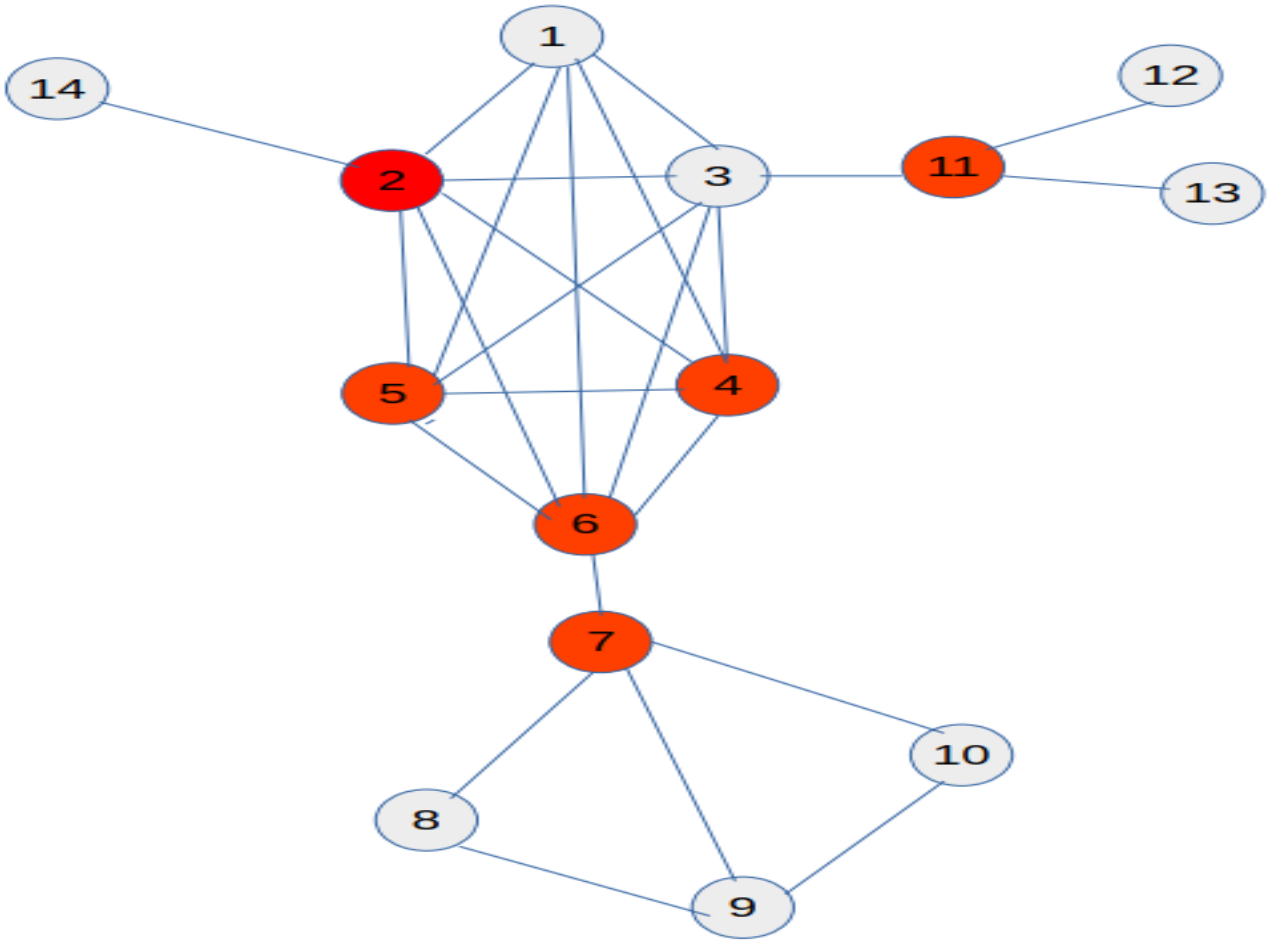
Preuve :

Soit un réseau de N noeuds avec D le dégrée haut du réseau .
Soit i un noeud en manque de mémoire avec $|L_i|$ = dégrée du noeud i.
soit $M_{vi} = \{j \in |L_i||M_{ri} > 0 et M_{mi} = 0\}$
Le noeud i peut donc faire au maximum $|M_{vi}|$ demande. car si la demande de i a $j \in M_{vi}$ est rejeté alors $|M_{vi}| := |M_{vi}| - \{j\}$; et i retirée j de sa mémoire virtuelle. Dans le cas contraire, le noeud i n'est plus éligible a la prochaine phase de calcul.

Théorème

Donc le noeud i peut participe au maximum a $|M_{vi}|$ phases de calcul. $|M_{vi}| < |L_i| \leq D$. D'où Cette étape de calcul peut se faire au maximum d fois.

Réseau



Informations sur les noeuds

Node	Voisins	M_i	M_{ri}	M_{mi}
1	{2,3,4,5,6}	13	8	0
2	{1,3,4,5,6,14}	5	0	1
3	{1,2,4,5,6,11}	12	6	0
4	{1,2,3,5,6}	4	0	1
5	{1,2,3,4,6}	1	0	4
6	{1,2,3,4,5,7}	1	0	5
7	{6,8,9,10}	2	0	2
8	{7,9}	3	1	0
9	{7,8,10}	10	7	0
10	{7,9}	3	1	0
11	{3,12,13}	1	0	2
12	{11}	2	1	0
13	{11}	2	1	0
14	{2}	1	0	0

Informations sur les noeuds

les nœuds en manque de mémoire sont les nœuds : 2, 4, 5, 6, 7 et 11

Mémoire virtuel et IN_i

- Nœud 2

$IN_2 = 1$

M_{r1}	M_{r3}

- Nœud 4

$IN_4 = 1$

M_{r1}	M_{r3}

- Nœud 5

$IN_5 = 1$

M_{r1}	M_{r3}

- Nœud 6

$IN_6 = 1$

M_{r1}	M_{r3}

- Nœud 7

$IN_7 = 9$

M_{r9}	M_{r10}	M_{r8}

- Nœud 11

$IN_{11} = 3$

M_{r3}	M_{r13}	M_{r12}

Pour le Noeud 2

- $L_2 = \{1, 3, 4, 5, 6, 14\}$, Les voisins du noeud 2 en manque de mémoire avec le même $IN_2 = 1$ sont : $\{4, 5, 6\}$
- en classant les noeuds par ordre de grandeur on : 6 > 5> 4> 2.
- La mémoire du noeud 1 ne peut satisfaire que 6 et 5.
- donc le noeud 2 fait une demande 1T au noeud 3 et retire le noeud 1 de sa mémoire virtuelle

Pour le Noeud 4

- $L_4 = \{1, 2, 3, 5, 6\}$, Les voisins du noeud 4 en manque de mémoire avec le même $IN_4 = 1$ sont : $\{2, 4, 5, 6\}$
- en classant les noeuds par ordre de grandeur on : 6 > 5> 4> 2.
- La mémoire du noeud 1 ne peut satisfaire que 6 et 5.
- donc le noeud 4 fait une demande 1T au noeud 3 et retire le noeud 1 de sa mémoire virtuelle

Pour le Noeud 5

- $L_5 = \{1, 2, 3, 4, 6\}$, Les voisins du noeud 5 en manque de mémoire avec le même $IN_5 = 1$ sont : $\{2, 4, 6\}$
- en classant les noeuds par ordre de grandeur on : $6 > 5 > 4 > 2$.
- La mémoire du noeud 1 ne peut satisfaire que 6 et il reste 3T au noeud 1
- le noeud 5 fait une demande de 3T au noeud 1 et de 1T au noeud 3

Pour le Noeud 6

- $L_6 = \{1, 2, 3, 4, 5, 7\}$, Les voisins du noeud 6 en manque de mémoire avec le même $IN_6 = 1$ sont : $\{2, 4, 6\}$
- en classant les noeuds par ordre de grandeur on :$6 > 5 > 4 > 2$.
- noeud 6 est le noeud maximal
- La mémoire du noeud 1 ne peut satisfaire le noeud 6
- le noeud 6 fait une demande de 5T au noeud 1.

Pour le Noeud 7

- $L_7 = \{6, 8, 9, 10\}$, le noeud 7 n'est en conflit avec aucun de ses voisins.
- La mémoire du noeud 9 peut satisfaire le noeud 7
- le noeud 7 fait une demande de 2T au noeud 9.

Pour le Noeud 11

- $L_{11} = \{3, 12, 13\}$, le noeud 11 n'est en conflit avec aucun de ses voisins.
- La mémoire du noeud 3 peut satisfaire le noeud 11
- le noeud 11 fait une demande de 2T au noeud 3.

Pour le Noeud 3

- $S_p = \{11\}$, et $S_d = \{5, 4, 2\}$
- La mémoire du noeud 3 satisfaire les noeuds 5,4,2 et 11

Pour le Noeud 1

- $S_p = \{6, 5\}$, et $S_d = \emptyset$
- La mémoire du noeud 1 satisfaire les demandes des noeuds 5,4,2 et 11

Pour le Noeud 9

- $S_p = \{7\}$, et $S_d = \emptyset$
- La mémoire du noeud 1 satisfaire les demandes des noeuds 7

REPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE
****
UNIVERSITE DE DSCHANG
****
ECOLE DOCTORALE

REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND
****
UNIVERSITY OF DSCHANG
****
POST GRADUATE SCHOOL

DSCHANG SCHOOL OF SCIENCES AND TECHNOLOGY
Unité de Recherche en Informatique Fondamentale, Ingénierie et Application (URIFIA)

Storage Virtualization

Présenté par :
TCHIO AMOUGOU Styves daudet
Matricule : CM-UDS-14SCI0251
Licencié en Informatique Fondamentale

Sous la direction de
Dr BOMGNI ALAIN Bertrand
(*Chargé de Cours, Université de Dschang*)