SpeedSiteRank: Algorithme parallèle pour un PageRank distribué en sites

Saint-Jean A. O. Djungu[1] Pierre Manneback Fabien Mathieu

Université de Kinshasa
R. D. Congo
sj_djungu@yahoo.fr

Faculté Polytechnique de Mons
Belgique
Pierre.Manneback@fpms.ac.be

France Télécom R&D
France
fabien.mathieu@normalesup.org

RÉSUMÉ. Un classement global et centralisé de pages web exige un coût de calcul assez important et donc ne favorise pas une mise à jour régulière de la base d'index. Pour pallier ce problème, nous proposons dans cet article un algorithme parallèle asynchrone, nommé *SpeedSiteRank*, susceptible de calculer le vecteur PageRank par site. Les résultats de tests réalisés sur un cluster composé de 10 noeuds bi-opteron ont démontré l'efficacité de notre algorithme.

ABSTRACT. A complete and centralized classification of web pages requires a rather costly computation, making regular updates of the index base not easy. To circumvent this problem, we propose in this article an asynchronous parallel algorithm, named *SpeedSiteRank*, which calculates the PageRank vector site by site. Tests were performed on cluster of 10 bi-opteron nodes, and the results showed the efficiency of our algorithm.

MOTS-CLÉS : PageRank, SpeedSiteRank, site, graphe, matrice

KEYWORDS : PageRank, SpeedSiteRank, site, graph, matrix

1. Boursier doctorant financé par la Coopération Technique Belge, dans le cadre d'une co-tutelle entre la Faculté Polytechnique de Mons et l'Université de Kinshasa.

1. Introduction

L'exploitation de la structure des liens hypertextes existant entre les différentes pages web a permis d'améliorer considérablement les performances des moteurs de reherche. Conçu en 1998, par Larry Page et Sergey Brin [PB98], le fameux moteur de recherche *Google* classe les pages web au moyen de la combinaison de multiples facteurs, dont le plus connu porte le nom de *PageRank*. Ce dernier utilise le nombre de liens pointant sur une page web pour lui affecter un indice de popularité. PageRank est aussi l'une des mesures utilisées pour permettre aux crawlers d'explorer le contenu du web [APC03].

L'exploration systématique du web est devenue cependant une tâche délicate soumise à de fortes contraintes. En effet, les aspects techniques du web tels que le trafic réseau et la bande passante constituent des facteurs pénalisants dans le rapatriement de pages web. De plus, la taille actuelle du web, estimée à plusieurs milliards de pages, contrarie lourdement la progression du crawler en prolongeant significativement le temps nécessaire à l'achèvement d'un cycle d'exploration [CR04]. Cela exige par la suite un temps considérable pour la mise à jour de l'index ainsi que le calcul de l'indice de popularité. Il est devenu difficile de mettre en oeuvre des explorations ayant pour but de visiter la totalité des pages web et donc de mettre à jour régulièrement la base d'index. Face à ces contraintes, même la parallélisation classique, basée sur la centralisation d'index, a montré ses limites [DM06]. Ainsi, il est devenu impérieux de penser aux systèmes distribués et collaboratifs [AW03].

Dans cette contribution, nous proposons une approche, basée sur la méthode d'agrégation/désagrégation de Vantilborgh [CS85], qui permet de calculer le vecteur PageRank par site. Elle se distingue des travaux antérieurs, [KHM+03, BLM+04, WD04, ZYL05, MV06], dans lesquels l'estimation du vecteur PageRank se fait d'abord par site et ensuite affinée par la prise en compte des informations extérieures au site. Dans notre approche, le calcul du vecteur PageRank intègre préalablement toutes les informations extérieures au site. Cela permet d'éviter de calculer deux fois le vecteur Pagerank par site.

Cet article est organisé comme suit : la section 2 présente un certain nombre de définitions et notations. Dans la section 3, nous donnons la formulation de SpeedSiteRank, en indiquant la façon dont se fait l'équilibrage de charge ainsi que les communications entre processeurs. La section 4 est consacrée à la présentation de résultats expérimentaux et à l'analyse de performance. Enfin nous concluons et présentons des perspectives pour des travaux futurs.

2. Notations et définitions

Dans cette section, nous regroupons un certain nombre de définitions et notations qui seront utilisées tout au long de cet article.

Définition 1 [KCN06] : Un *site* est un ensemble de pages web inter-reliées et identifiées par un nom commun (domaine, serveur, répertoire, etc).

Soit $\delta = (S_1, ..., S_k)$ un ensemble de k sites, avec $k > 1$. La taille de chaque site S_i est égale à n_i et $\sum_{i=1,...,k} n_i = n$.

Définition 2 : Le *graphe étendu du site* S_i est défini par le couple $G_i = (V_i, E_i)$, où V_i est un ensemble de n_i pages web de S_i et E_i l'ensemble de couples de pages web dont la première composante (élément de V_i) possède un lien hypertexte vers la deuxième

composante (élément de V_i ou pas). La matrice de transition associée au graphe G_i est une matrice $A_{i*} \in \mathcal{M}_{n_i,n}(R)$ définie par l'équation (1) ci-dessous.

$$A_{i*} = (a_{j,k}),\ \text{avec} a_{j,k} = \begin{cases} \frac{1}{d_j} & \text{si } j \to k \\ 0 & \text{sinon} \end{cases} \tag{1}$$

où d_j est le nombre de liens sortants de la page $j \in V_i$.

Définition 3 : Soient A_{ii} la *matrice intra-site* S_i et A_{ij} $(j \neq i)$ la *matrice inter-site* S_i et S_j. La matrice A_{i*}, donnée par l'égalité (1), peut encore s'écrire comme :

$$A_{i*} = [A_{i1}, ..., A_{ii}, ..., A_{ik}] \tag{2}$$

En agrégeant les graphes de sites, on obtient le graphe du web $G = (V, E)$, où l'ensemble des pages web est $V = \cup_{i=i}^{k} V_i$ et les liens hypertextes est $E = \cup_{i=i}^{k} E_i$. La matrice de transition associée au graphe G est donc

$$A = \begin{bmatrix} A_{11} & A_{12} & . & . & . & A_{1k} \\ A_{21} & A_{22} & . & . & . & A_{2k} \\ . & & . & & & . \\ . & & & . & & . \\ . & & & & & . \\ A_{k1} & A_{k2} & . & . & . & A_{kk} \end{bmatrix}_{n \times n} \tag{3}$$

Soit x un vecteur de taille n. En tenant compte des tailles des différents sites, le vecteur x peut s'écrire comme suit :

$$x = (x_1, ..., x_k)^T \tag{4}$$

où chaque x_i est un vecteur-ligne de taille $n_i, i = 1, ..., k$.

Soit

$$x^* = (x_1^*, ..., x_k^*)^T \tag{5}$$

un vecteur avec $x_i^* = \frac{x_i}{\|x_i\|_1}$ un vecteur de taille n_i.

Soient $L \in \mathcal{M}_{n,k}(R)$ et $S \in \mathcal{M}_{n,k}(R)$ les matrices associées aux opérateurs de communications entre les sites définies respectivement comme suit [CS85] :

$$L(x^*) = \begin{bmatrix} x_1^* & 0 & . & . & . & 0 \\ 0 & x_2^* & . & . & . & 0 \\ . & & . & & & . \\ . & & & . & & . \\ . & & & & & . \\ 0 & 0 & . & . & . & x_k^* \end{bmatrix}_{n \times k} \tag{6}$$

et

$$S = \begin{bmatrix} e_1 & 0 & . & . & . & 0 \\ 0 & e_2 & . & . & . & 0 \\ . & & . & & & . \\ . & & & . & & . \\ . & & & & & . \\ 0 & 0 & . & . & . & e_k \end{bmatrix}_{n \times k} \tag{7}$$

avec e_i vecteur unitaire de taille n_i.

Soit $B(x^*)$ la matrice de transition inter-sites : pour deux sites i et j donnés, on veut que $B_{i,j}(x^*)$ mesure la probabilité que, partant du site i suivant la distribution x_i^*, l'on

atteigne j en suivant au hasard une arête de G_i. En utilisant les relations (6) et (7), nous obtenons l'expression de la matrice B à l'aide des matrices A, L et S.

$$B(x^*) = L(x^*)^T AS \tag{8}$$

La matrice B hérite de toutes les propriétés de la matrice A [MM98].

3. Algorithme SpeedSiteRank

En général, la matrice A définie par la relation (3) n'est pas irréductible. Pour forcer l'irréductibilité, nous considérons la normalisation de A en prenant la moyenne pondérée par α de A et de ez^T.

$$Q = \alpha A + (1-\alpha)ez^T \tag{9}$$

où z est une distribution de zap uniforme et α[1] le facteur de zap [PB98]. L'expression en sites de la matrice Q s'écrit comme indiquée dans (10).

$$\begin{cases} Q_{ii} = \alpha A_{ii} + (1-\alpha)e_i z_i^T \\ Q_{ij} = \alpha A_{ij} + (1-\alpha)e_i z_j^T \end{cases} \tag{10}$$

où e_i et z_i sont des vecteurs de taille n_i.

Proposition 1

Soient z une distribution de zap, e^* un vecteur unité de taille k (nombre de sites) et $b = [\|z_1\|_1, ..., \|z_k\|_1]^T$. Soit $V = e^* b^T$ une matrice carrée de taille k.

1) L'expression de la matrice inter-sites est donnée par :

$$B^* = \alpha B + (1-\alpha)V \tag{11}$$

2) L'importance relative de sites est un vecteur solution de l'équation (12).

$$w = \text{SpeedRank}(B, \alpha, (1-\alpha)b) \tag{12}$$

où $w = \text{SpeedRank}(B, \alpha, (1-\alpha)b)$ vient de $w = \alpha B^T w + (1-\alpha)b$, avec $w > 0$.

Proposition 2

Une approximation du vecteur PageRank, x_i, relative aux pages web du site S_i est donnée par :

$$\begin{cases} y_i = \text{SpeedRank}(A_{ii}, \alpha, r_i) \\ x_i = w_i y_i \end{cases} \tag{13}$$

où $r_i = (1-\alpha)(2-w_i)z_i + \alpha \sum_{j \neq i} w_j x_j^{*T} A_{ji}$ est un vecteur de taille n_i.

Nous avons maintenant tous les ingrédients pour décrire notre algorithme, que nous proposons de nommer *SpeedSiteRank*. Le parallélisme requis par *SpeedSiteRank* est mieux exploité si le nombre de sites (k) est égal au nombre de processeurs (p). Dans la pratique, le nombre de processeurs est souvent inférieur au nombre des sites. D'où la nécessité de faire le groupement de calcul de PageRanks de sites sur un processeur.

1. Probabilité de cliquer au hasard sur un de liens sortants d'une page web.

Soient $F(i)_{1,...,p}$ la fréquence du i^{eme} processeur et $Nnz(j)_{1,...,k}$ le nombre d'éléments non nuls de la matrice du j^{eme} site. L'algorithme glouton 1, permet de déterminer la distribution statique optimale de la charge de calcul de PageRanks sur les p processeurs.

Algorithme 1 : fonction DC = distribution_charge(Nnz, F, k, p)

Début

Trier Nnz par ordre décroissant

$tempsEx(i) = 0$, pour $i = 1, ..., p$

Pour $i = 1$ à k **faire**

$j = arg\ \ min_{1 \leq t \leq p}(tempsEx(t) + \frac{Nnz(i)}{F(t)})$

$DC(i) = j$

$tempsEx(j) = tempsEx(j) + \frac{Nnz(i)}{F(j)}$

Fin

Fin

L'étape 7 de l'algorithme 2 indique comment se fait la communication entre les sites. Le site S_i, par exemple, envoi à tous les autres sites un vecteur $K_{i,*}$. Ce dernier contient toutes les informations concernant les liens hypertextes entre le site S_i et tous les autres sites. L'exploitation du paradigme maître-esclave, par exemple, exige la redistribution de vecteurs w et f (étapes 9 et 10) sur les processeurs esclaves.

4. Expériences numériques

4.1. Matrices de test

Pour tester notre algorithme, nous avons considéré deux matrices issues du projet WebGraph[2]. La partition en sites de ces deux matrices a été faite sur base d'un tri lexicographique sur les urls. Nous avons effectué une partition en serveurs pour la matrice cnr-2000 et en domaines pour in-2004. Le tableau 1 reprend les principales caractéristiques de ces matrices.

	Année de crawl	**#Pages**	**#Liens**	**Niv. de partition**	**#Sites**
cnr-2000	2000	325557	3216152	serveur	143
in-2004	2004	1382908	16538959	domaine	13

Tableau 1. *Caractéristiques des matrices*

2. Ces matrices ont été téléchargées à partir de la page http ://law.dsi.unimi.it/ (dernière consultation le 03 janvier 2008).

algorithme 2 : SpeedSiteRank : estimation de PageRank en sites

Données

* Un ensemble $\delta = (S_1, ..., S_k)$ et $n_i = |S_i|$;
* Matrice de transition A associée à δ ;
* Matrice S associée à l'opérateur de communications entre sites ;
* Une distribution de zap z ;
* Un coefficient de zap $\alpha \in]0, 1[$;
* Une distribution de charge DC

Résultat

* Une estimation du vecteur PageRank distribué en sites

Début

1. $x^{(0)} = (\frac{z_1}{\|z_1\|_1}, ..., \frac{z_k}{\|z_k\|_1})^T$
2. $b = (\|z_1\|_1, ..., \|z_k\|_1)^T$
3. $B = 0_{k \times k}$
4. **Pour** chaque site S_i de δ selon DC **faire**
5. $\quad K_{i,*} = x_i^{(0)T} A_{i*}$
6. **Fin**
7. $K =$ CollecteGlobale$(K_{i,*})$
8. $B = KS$
9. $w =$ SpeedRank$(B, \alpha, (1-\alpha)b)$
10. $f = wK$
11. **Pour** chaque site S_i de δ selon DC **faire**
12. $\quad r_i = (1-\alpha)(2-w_i)z_i + \alpha(f_i - w_i x_i^{(0)T} A_{ii})$
13. $\quad y_i =$ SpeedRank(A_{ii}, α, r_i)
14. $\quad x_i = w_i y_i$
15. **Fin**

Fin

Ecrit en C et MPI, notre code a été testé sur un cluster Linux à 10 noeuds bi-opteron 244 cadencés à 1.8 GHz interconnectés en Gigabit Ethernet. Chaque noeud dispose de 2 Go de RAM et 76 Go de HDD en 10000 RPM SCSI. Seuls 8 noeuds ont été utilisés pour faire les tests.

4.2. Analyse de performance

Compte tenu de la nature asynchrone de SpeedSiteRank et du fait que les graphes de sites n'ont pas le même nombre de liens hypertextes, les temps d'exécution par processeur sur un ensemble de noeuds du cluster se trouvent dans un intervalle $[temps_{min}, temps_{max}]$. Pour permettre d'apprécier le gain obtenu, nous proposons une présentation où seul le temps moyen ($(temps_{max} - temps_{min})/2$) est indiqué (voir tableau 2).

# Noeuds	1	2	3	4	5	6	7	8
cnr-2000 (en sec)	53	30	22	19	17	14	13	12
in-2004 (en sec)	256	131	88	67	56	48	43	38

Tableau 2. *Temps moyen d'exécution de SpeedSiteRank sur cnr-2000 et in-2004 par rapport au nombre de noeuds ($\alpha = 0.85$)*

La figure 1 montre la variation du temps d'exécution en fonction du nombre de noeuds utilisés. On remarque que la scalabilité obtenue est fonction de la taille et de la nature de

la matrice utilisée. La matrice in-2004 présente une meilleure scalabilité par rapport à cnr-2000.

4.3. Corrélation entre PageRank et SpeedSiteRank

Dans cet article, nous avons recouru à la distance de Kendall normalisée pour comparer les classements obtenus en utilisant respectivement PageRank et SpedSiteRank.

Définition 4 : La distance de Kendall normalisée entre deux listes X et Y de taille n est donnée par la relation (14).

$$KDist(X,Y) = \frac{\sum_{\{i,j\}\in P} K_{ij}(X,Y)}{n(n-1)/2} \tag{14}$$

où

– P est l'ensemble des paires désordonnées d'éléments distincts dans X et Y

– $K_{ij}(X,Y) = 0$ si i et j sont dans le même ordre dans X et Y

– $K_{ij}(X,Y) = 1$ si i et j sont dans un ordre opposé dans X et Y.

Partant de la définition 4, nous avons trouvé respectivement 0.0189 et 0.0306 comme distances de Kendall normalisées moyennes pour les matrices cnr-2000 et in-2004. Ainsi, les classements obtenus en utilisant PageRank et SpeedSiteRank sont correlés à plus de 98% pour cnr-2000 et 97% pour in-2004.

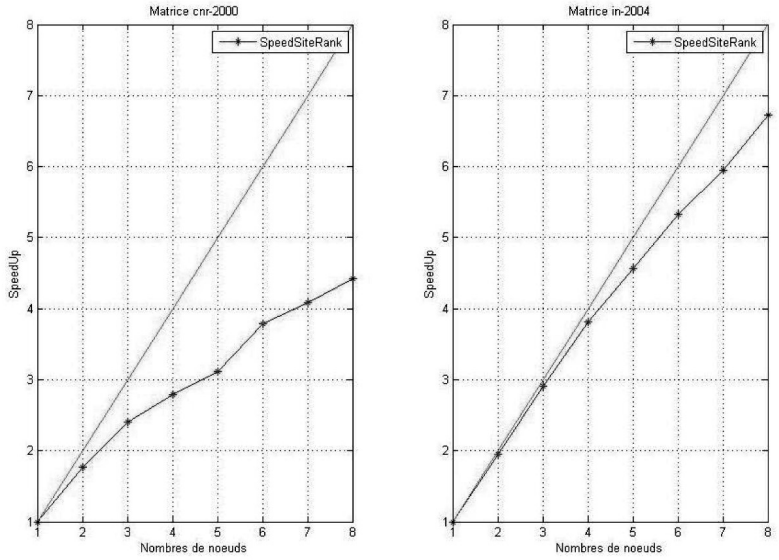

Figure 1. *Speed-up de SpeedSiteRank pour les matrices cnr-2000 et in-2004*

5. Conclusion

L'objectif de ce travail était de trouver un moyen non centralisé de calculer le vecteur PageRank. L'algorithme SpeedSiteRank développé est non seulement une réponse à la

préoccupation du départ, mais se revèle un moyen puissant pour permettre au webmaster de mettre à jour sa base d'index sans chercher à chahuter tout le web.

Nos travaux futurs consisteront à profiter des résultats obtenus dans [DMM06] pour utiliser EramRank[3] ou GmresRank[4] comme paramètres d'entrée de SpeedSiteRank à la place de SpeedRank.

6. Bibliographie

[APC03] S. Abiteboul, M. Preda et G. Cobena. *Adaptive on-line page importance computation*. Dans Proceedings of the twelfth international conference on World Wide Web, pages 280-290, ACM Press, 2003.

[AW03] K. Aberer et J. Wu. *A framework for decenralized ranking in web information retrieval. Dans Web Technologies and Appliations : Proceedinfs of th Asia-Pacific Web Conference*. AP-Web 2003 volume LNCS 2642, pages 213-226, Xi'an, China, September 2003. Springer-Verlag. September 27-29, 2003.

[BLM+04] A. Broder, R. Lempel, F. Maghoul, and J. Pedersen. *Efficient pagerank approximation via graph aggregation*. In Proc. of the WWW'04 Conf., pages 484 - 485, 2004.

[CS85] W. Cao and W. J. Stewart. *Iterative aggregation/disaggregation techniques for nearly uncoupled markov chains*. ACM Press, 702 - 719. 1985.

[CR04] J. Cho et S. Roy. *Impact of Web search engines on page popularity*. Dans Proceedings of the Thirteenth International WWW Confernce, 2004.

[DM06] S.-J. Djungu et P. Manneback. *Mise en oeuvre parallèle sur cluster d'algorithmes itératifs de PageRank*. Dans les Actes de 17ème Rencontres francophones du Parallélisme. Perpignan/France, du 3 au 6 octobre 2006.

[DMM06] S.-J. Djungu, P. Manneback et F. Mathieu. *Etude comparative des méthodes de calcul de PageRank*. Dans les Actes de la 8ème Conférence Africaine sur la Recherche en Informatique, Cotonou/Bénin, November 6-9, 2006.

[KCN06] C. Kohlschutter, P. Chirita, and W. Nejdl. *Efficient Parallel Computation of PageRank*, 2006.

[KHM+03] S. Kamvar, T. Haveliwala, C. Manning et G. Golub. *Exploiting the Block Structure of the Web for Computing PageRank*. Technical Report, Stanford University, 2003.

[MM98] I. Marek et P. Mayer. *Convergence analysis of an aggregation / disaggregation iterative method for computation stationary probability vectors of stochastic matrices, Numer*. Linear Algebra Appl. 5, 253-274, 1998.

[MV06] F. Mathieu and L. Viennot. *Aspects of the Global Ranking of Web Pages*. In Proceedings of I2CS 2006, 2006.

[PB98] L. Page et S. Brin. *The anatomy of a large-scale hypertextual Web search engine*. Computer Networks and ISDN Systems, 33(3) :107-117, 1998.

[WD04] Y. Wang and D. J. DeWitt. *Computing PageRank in a distributed internet search system*. In Proceedings of the 30th VLDB Conference, 2004.

[ZYL05] Y. Zhu, S. Ye, and X. Li. *Distributed pagerank computation based on iterative aggregation-disaggregation methods*. In Proc. of the 14th ACM international conference on Information and knowledge management, 2005.

3. PageRank basé sur ERAM (Explicitly Restarted Arnoldi Method).
4. PageRank basé la méthode Gmres(Generalized Minimum Residual).