Compression parallèle de données textuelles avec contraintes de positionnement

Jacques Chauche*, Frédéric Guinand**, Mathieu Lafourcade*
** LIRMM - 161 rue Ada - 34392 Montpellier cedex 5, France*
{Jacques.Chauche,Mathieu.Lafourcade}@lirmm.fr
*** (Orateur) LIH - Université du Havre, 25 rue Philippe Lebon 76600 Le Havre, France*
Frederic.Guinand@univ-lehavre.fr

Mots clés : compression de données, parallélisme, dictionnaire multilingue, superchaîne.

On dispose d'une importante base de données regroupant un très grand nombre de mots. Ces mots sont utilisés pour la constitution de dictionnaires spécialisés. Sur requête d'un utilisateur, à partir d'un sous-ensemble de la base de données, un nouveau dictionnaire est construit. La consultation en ligne des entrées de ce dictionnaire nécessite son chargement intégral dans la mémoire du processeur. Une utilisation confortable requiert donc un volume de données et un temps de construction du dictionnaire raisonnables.

La construction aussi rapide que possible d'une représentation aussi compacte que possible de ce dictionnaire constitue l'objectif de ce travail. On souhaite d'autre part, que la reconstruction d'une entrée du dictionnaire puisse se faire sans traiter la totalité des données. Cette contrainte rend inadaptées les algorithmes de compression pour l'archivage du type Ziv-Lempel[1]. Dans ce contexte, la construction d'une superchaîne, associée à un codage idoine des articles, représente un bon compromis volume de données / temps de traitement. La solution retenue pour le codage des articles est la suivante :

$$\text{article} = (< attribut >, < valeur\_bis >)^+$$
$$\text{valeur\_bis} = \{(index\_depart, longueur\_chaine)\}$$

Le codage des index de départ est variable. La minimisation du codage des index associés aux mots les plus fréquents, nécessite l'apparition de ces mots au plus tôt dans la superchaîne.

Bien que plus complexe, du fait de la prise en compte du positionnement des mots, ce problème s'apparente à celui de la recherche de la plus courte superchaîne, dont le problème de décision associé est NP-complet[2].

Pour des raisons de performances, nous voulons résoudre ce problème en parallèle. À ce jour, il semble n'exister aucun algorithme parallèle qui prenne en compte ce double objectif.

[1] J. Ziv and A. Lempel. *Compression of Individual Sequences via Variable-Rate Coding.* IEEE Transactions on Information Theory 24, 5 (1978) 530-536

[2] M. Garey and D. Johnson. *Computers and Intractability* A Guide to the Theory of NP-Completeness. Freeman and Co (page 228)

L'algorithme d'approximation que nous proposons considère les deux aspects du problème, construction d'une superchaîne et positionnement des mots dans la superchaîne en fonction de leur utilisation présumée. Celle-ci étant représentée par un indice de fréquence. Nous étudions une version simplifiée du problème en supposant certaines hypothèses sur les données de départ. Ces hypothèses sont les suivantes :

- Tous les symboles de l'alphabet nécessitent le même nombre de bits pour être codés.
- L'ensemble des mots est noté $\mathcal{D}$. Tous les mots de $\mathcal{D}$ sont différents deux à deux, et aucun mot n'est infixe d'un autre.
- Les mots sont triés par ordre décroissant d'indice de fréquence.
- Le mot le plus long est de longueur fixé $<<$ cardinal de $\mathcal{D}$.
- Étant donné le mode d'accès aux mots dans la superchaîne $(index, longueur)$, les mots ne sont pas partitionnables.

L'algorithme proposé se décompose en quatre principales phases :

- 1. Construction de l'arbre des infixes. Cette étape consiste à générer une arborescence dont chaque noeud N_i possède trois attributs :
  - sc : une sous-chaîne correspondant à un infixe de mots de $\mathcal{D}$,
  - lp : la liste des mots $\in \mathcal{D}$ tels que sc est un préfixe de ces mots,
  - ls : la liste des mots $\in \mathcal{D}$ tels que sc est un suffixe de ces mots.
- 2. Nettoyage de l'arbre. Les chemins $n_i..n_f$, tels que n_f est une feuille et dont tous les noeuds sont tels que lp (resp. ls) est vide, sont éliminés de l'arbre.
- 3. Formation des superchaînes. Pour un noeud donné, dont l'attribut sc est l'un des plus long, les superchaînes sont formées en appariant les mots appartenant à ls avec les mots appartenant à lp. L'indice de fréquence de cette superchaîne est égal à une fonction des indices de fréquence de la paire de mots le constituant. Une fonction triviale pouvant être la somme. Les superchaînes ainsi construites sont insérées dans l'arbre. Les listes ls et lp contenant les mots constituant les superchaînes sont mises à jour.
- 3 bis. Retour à l'étape 2, jusqu'à ce que l'arbre ne contienne plus aucune branche.
- 4. Lorsque toutes les branches de l'arbre ont été éliminées, seules restent les superchaînes qui ne se recouvrent pas. On les trie alors par ordre décroissant d'indice de fréquence, puis on en fait la concaténation.

L'algorithme proposé repose sur une architecture de type maître/esclave. Nous effectuons une parallélisation par phases, en suivant les phases de l'algorithme.

Pour la phase expérimentale, nous avons considéré un dictionnaire multilingue français-anglais-malais de 20000 entrées. Un tel dictionnaire comprend environ 400000 mots, dont le nombre est ramené à 40000 lorsque sont éliminés les doublons et les mots infixes d'autres mots.