LES CAHIERS DE L'ANALYSE DES DONNÉES

K. BEN SALEM

Associativité de la fusion et parallélisme dans les algorithmes de tri

Les cahiers de l'analyse des données, tome 15, n° 2 (1990), p. 133-138

<http://www.numdam.org/item?id=CAD_1990__15_2_133_0>

Les Cahiers de l'Analyse des Données
Vol XV - 1990 - n° 2 - pp. 133 138.

ASSOCIATIVITÉ DE LA FUSION ET PARALLÉLISME DANS LES ALGORITHMES DE TRI

[ASS. FUS. TRI.]

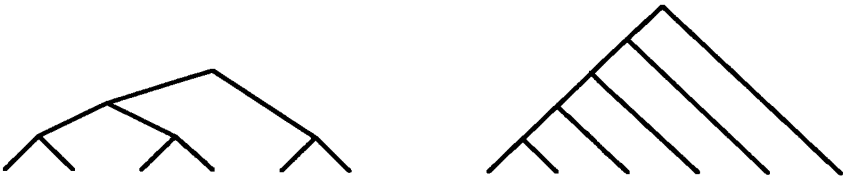
*K. BEN SALEM**

1 Parallélisme et associativité

Considérons une opération associative, telle que l'addition des nombres réels. Il est bien connu, par exemple, que l'addition de 6 nombres peut s'effectuer suivant autant de schémas de parenthèses, c'est-à-dire d'association des arguments, qu'il y a d'arbres binaires ayant 6 éléments terminaux. Le graphique propose deux réalisations possibles, correspondant aux formules:

$$(((a + b) + (c + d)) + (e + f)) \quad \text{et} \quad (((((a + b) + c) + d) + e) + f) \ ;$$

mais sans spécifier l'opération d'addition, car le schéma vaut aussi pour toute



(((a, b), (c, d)), (e, f)) (((((a, b), c), d), e), f)

deux schémas éqivalents pour effectuer des opérations associatives

opération associative telle que la multiplication des nombres; ou encore l'addition de vecteurs appartenant à un même espace, la composition d'applications linéaires d'un espace vectoriel dans lui-même; et même la composition d'applications reliant des espaces différents; à condition que pour deux applications consécutives de la suite à composer, par exemple *c* et *d*, la source de celle qui est à gauche soit le but de celle qui est à droite.

(*) Maître assistant à la Faculté des Sciences de Tunis.

Il est classique de définir la profondeur de la pile requise pour réaliser un schéma: cette profondeur n'est autre que la longueur maxima d'une suite de nœuds se développant, à partir du sommet, vers la gauche (ou vers la droite, selon le sens dans lequel on parcourt l'expression à calculer).

D'autre part, le traitement parallèle étant à l'ordre du jour, on doit aussi considérer la profondeur *de haut en bas*, qui n'est autre que la longueur maxima d'une suite de nœuds se développant, à partir du sommet, vers la gauche *ou* vers la droite. Pour la première des expressions proposées ci-dessus, cette profondeur vaut 3; pour la seconde, elle vaut 5.

Si l'on dispose d'autant de processeurs qu'on le veut pour exécuter simultanément plusieurs opérations, il est clair que le temps de calcul de l'expression globale suivant un schéma n'est autre que le maximum du total obtenu en additionnant les temps de calcul afférents aux opérations situées sur les nœuds d'une branche issue du sommet. Si toutes les opérations à effectuer prennent le même temps, ce qui est le cas pour l'addition des nombres réels, le temps de calcul, suivant un schéma, est proportionnel à la profondeur de l'arbre *de haut en bas*; et le premier schéma apparaît préférable au second.

Les considérations ci-dessus ne prennent de l'intérêt que dans le cas où le calcul à effectuer a pour arguments et pour résultat des êtres mathématiques de format général; avec, en chaque nœud, deux arguments dont la taille n'est pas nécessairement la même; et, par conséquent, des temps d'exécution différents suivant les nœuds. Nous croyons que réduire les algorithmes à de tels schémas associatifs est un exercice mathématique utile à la programmation et nous proposons de traiter l'exemple qu'annonce le titre du présent article en montrant que le *tri par fusion* et le *tri par insertion* ne sont que la réalisation, suivant deux schémas différents, d'un système d'opérations de fusion dont la composition est associative.

2 Tri de fichiers et associativité de la fusion

Le principe du tri par fusion est bien connu; et il a, récemment, dans *CAD*, (cf. [LING. TRI], §1, in Vol XV, n°1) fait l'objet d'un exposé qui débute ainsi:

"Dans sa version la plus simple, l'algorithme de tri par fusion part de deux listes données, dont chacune est déjà rangée (par exemple dans l'ordre croissant), et en crée une troisième, rangée également, et comprenant les éléments des deux premières."

Pour démontrer la propriété générale d'associativité que nous avons en vue, l'hypothèse que les deux listes données soient bien rangées est inutile: cette hypothèse sert seulement à démontrer que le résultat de la fusion est une liste bien rangée. Et plutôt que de *tri par fusion*, nous parlerons ici d'une opération de *fusion* (ou, si l'on préfère, d'une procédure de fusion) pour laquelle les deux arguments, ainsi que la valeur, sont des listes.

Dans l'article cité, cette opération est présentée comme suit:

"Deux files d'hommes, fA et fB, se présentent à un guichet: dans chacune des files, les hommes sont rangés par taille croissante, les petits devant, les grands ensuite. Le guichetier fait passer l'homme de tête de la file fA, s'il est plus petit que celui de fB (ou de même taille que lui); sinon, il fait passer l'homme de tête de fB. Il est clair qu'ainsi, après le passage du guichet, les individus de fA et de fB constituent une file unique fC, bien rangée."

Comme nous omettons, en général, l'hypothèse que fA et fB soient bien rangées, la conclusion pour fC ne vaut pas, en général, non plus; mais l'opération reste ce qu'elle est; et l'on peut écrire:

fC = fus (fA, fB) .

Soit maintenant à trier une liste de 6 éléments, qu'on notera {a,b,c,d,e,f}, en reprenant les mêmes lettres qu'au §1. On peut considérer que chacun de ces éléments forme, à lui seul, une liste élémentaire, laquelle est évidemment bien rangée. On notera {a}, {b}, ..., ces listes élémentaires. Ceci posé, pour cette liste à 6 éléments, l'exécution de l'algorithme de tri par fusion peut se reformuler, en termes d'opérations de fusion, suivant le 1-er des schémas proposés ci-dessus:

fus (fus (fus ({a}, {b}), fus ({c}, {d})), fus ({e}, {f}));

autrement dit: on classe les trois paires (a, b), (c, d) et (e, f); on interclasse les paires résultant des deux premiers classements; et le quadruplet ainsi obtenu est interclassé avec le résultat du classement de la paire (e, f).

Appliquons maintenant à ces mêmes données le deuxième des schémas proposés au §1; on a:

fus (fus (fus (fus (fus ({a}, {b}), {c}), {d}), {e}), {f});

autrement dit: on classe la paire (a, b); on insère {c} dans le résultat; puis on insère {d} dans le triplet {a, b, c} bien rangé; etc... Il est clair qu'on a ainsi réalisé un très classique *tri par insertion*. En effet, la fusion d'une liste quelconque avec une liste à un seul élément n'est autre chose que l'insertion, dans la première liste, de l'unique élément dont se compose la seconde.

Quant au temps de calcul, on notera que l'exécution d'une fusion consiste en des comparaisons de paires dont le nombre ne peut dépasser la longueur de la liste résultante (plus exactement cette longueur diminuée de 1; et ne lui est inférieure que si l'une des listes se trouve épuisée avant que la fusion ne soit achevée). Sur ce principe, repose l'estimation du temps de calcul correspondant aux différents schémas possibles; sous les diverses hypothèses qu'il n'y a qu'un seul processeur; qu'il y en a un nombre illimité; ou qu'il y en a un nombre

inférieur à ce qu'on pourrait désirer pour tirer parti du parallélisme permis par le schéma.

3 Associativité de la fusion

Interpréter comme l'exécution répétée de la seule opération de fusion, sur les mêmes arguments mais suivant deux schémas différents, les procédures usuelles du tri par fusion et du tri par insertion, c'est poser implicitement le problème de l'associativité de la fusion. Seule l'associativité, en effet, assure que la liste résultat est la même, quel que soit le schéma choisi; et cela, même si les arguments de base ne sont pas des listes élémentaires , mais des listes quelconques, bien rangées ou non. La formule à démontrer est donc:

$$\forall\ fA, fB, fC : fus\ (fus\ (fA, fB), fC) = fus\ (fA, fus\ (fB, fC))\ ;$$

formule où les arguments fA, fB, fC sont des listes quelconques.

Pour plus de précision, on posera qu'une liste est une suite finie ordonnée d'éléments d'un ensemble totalement ordonné X; ou, plus généralement, de paires (x, y) formées d'un élément x de X et d'un élément y d'un ensemble Y quelconque: en effet, dans les applications usuelles, il s'agit, e.g., de trier des documents y d'après leur date x; ou encore (cf. [LING. TRI], §2.2) des occurrences d'un mot x pour chacune desquelles est précisée une adresse ou un contexte y. Dans la suite de l'exposé, on fera abstraction de y, et supposera tacitement qu'il suit x dans la création d'une liste par fusion.

Afin de soutenir l'intuition, nous supposons que la fusion se réalise avec deux fichiers fA et fB ouverts en lecture et un fichier fE ouvert en écriture. La longueur, ou nombre d'enregistrements, de chaque fichier sera notée: nA, nB, nE; le p-ème enregistrement d'un fichier f sera noté f(p); pour abréger, on écrira xA, xB pour fA(nA), fB(nB), qu'on appellera *éléments de tête*. Pour la généralité des raisonnements, on doit introduire le fichier vide, ∅, de longueur 0.

Quand s'exécute la fusion de fA et fB pour créer fE, la somme des trois nombres (nA + nB + nE) reste constante: cette somme sE est aussi la longueur du fichier fE créé, quand l'opération de fusion est achevée. Au départ, nE = 0; à chaque pas, on décide de celui des deux fichiers, fA ou fB, qu'on doit raccourcir pour allonger fE: si inf(xA, xB) = xB, nB est diminué de 1; sinon (même si xA = xB) c'est nA qui est changé en nA-1. Il faut prendre garde que, dans l'état final de fE, l'élément inf(xA, xB) aura pour rang (nA + nB); on notera donc:

$$fE(nA + nB) = inf(xA, xB) = inf\ (fA(nA), fB(nB));$$

même si, présentement, la longueur de fE n'est que (sE+1-(nA+nB)).

Le premier élément inscrit dans fE est l'élément de tête, fE(sE); le dernier mis en place est fE(1). En cours d'exécution, les nE éléments de fE se trouvent ainsi numérotés de sE à sE+1-nE (rang du dernier élément inscrit).

Pour prouver l'associativité de la fusion, nous croyons commode d'introduire un terme intermédiaire, noté fus (fA, fB, fC), dont il faudra montrer qu'il est égal à l'un et l'autre des fichiers, créés par deux fusions successives:

fus (fus (fA, fB), fC) et fus (fA, fus (fB, fC));

il est, en effet aisé de définir la fusion entre un nombre quelconque de fichiers ouverts en lecture, par exemple trois: (fA, fB, fC): il suffit de poser qu'à chaque pas est copié, dans fE, inf(xA, xB, xC); et que, si l'inf est réalisé par deux des *x* à comparer ou les trois, est raccourci celui des fichiers offrant cette valeur qui est le premier dans l'ordre alphabétique des trois lettres (A, B, C). On dira, en bref, que *x* est pris dans *le plus petit* des fichiers qui l'offre. Comme dans le cas de deux fichiers, dans l'état achevé de fE, *x* a pour rang (nA + nB + nC); on écrit donc:

fE(nA + nB + nC) = inf(xA, xB, xC).

On notera, au passage, que du fait que l'inf peut être réalisé plusieurs fois, la fusion n'est pas une opération commutative; de plus, il faut, en toute rigueur, prévoir le cas où est vide l'un des fichiers ouverts en lecture (arguments fA, fB fC de fus): on prend alors, pour calculer l'inf, les éléments de tête de fichier non vides; à la fin de la fusion, il ne reste plus, nécessairement, qu'un seul fichier non vide.

Ceci posé, reste à établir les égalités:

fus (fA, fB, fC) = fus (fA, fus (fB, fC)) ;
fus (fus (fA, fB), fC) = fus (fA, fB, fC)) ;

les deux étant analogues, il suffit de considérer la première.

On procédera par récurrence sur la somme sE = (sA + sB + sC) des longueurs initiales des trois fichiers arguments, somme qui est, répétons-le, la longueur du fichier à créer. Le départ de la récurrence est simple: si les 3 fichiers arguments sont vides, il en est de même du résultat, de quelque manière qu'on procède; ce qui correspond au fait que ∅ est élément neutre pour l'opération de fusion.

Dans le cas général notons:

fE = fus (fA, fB, fC) ; fF = fus (fB, fC) ; fG = fus (fA, fF) :

il faut démontrer que fE = fG, pour une valeur donnée de sE, si cela est vrai pour toute valeur inférieure prise par cette somme.

Le premier élément, fE(sE), écrit dans fE, n'est autre que inf(xA, xB, xC) (calculé sur les valeurs de tête des fichiers argument dans leur état initial); et, en

cas d'égalité, cette valeur x est prise dans le plus petit fichier qui l'offre. Que fG(sE) = fE(sE) résulte simplement de l'associativité de l'inf; car on a:

fE(sE) = inf(fA(sA), fB(sB), fC(sC)) ;
fG(sE) = inf(fA(sA), fG(sB + sC)) = inf(fA(sA), inf(fB(sB), fC(sC))) .

De plus, si, pour fE, x est pris dans fA, il en est de même pour fG; et si, pour fE, x est pris dans l'un des deux fichiers fB ou fC, alors pour fG, x est pris dans fF, dont il est l'élément de tête; et cet élément de tête x a été pris dans le même fichier, fB ou fC, que pour fE.

Ceci posé, on voit que fE, privé de son élément de tête, est un fichier fE' qui s'obtient par fusion de trois fichiers (fA', fB', fC') qui ne sont autres que (fA, fB, fC) dont l'un a été privé de son élément de tête. D'autre part, fG' (i.e. fG privé de son élément de tête) s'obtient par fusion de deux fichiers fA' et fF': qui ne sont autres que fA et fF, modifiés en racourcissant celui des deux qui a fourni la tête de fG; et si c'est fF qui est raccourci, alors fF' résulte de la fusion de deux fichiers fB', fC'; et celui des deux fichiers fB et fC qui est raccourci est le même que dans l'expression de fE'.

On peut donc écrire dans tous les cas:

fE' = fus (fA', fB', fC') ; fG' = fus (fA', fus (fB', fC')) ;

avec les mêmes fichiers (fA', fB', fC') dans l'expression de fE' et dans celle de fG'. Il résulte alors immédiatement de l'hypothèse de récurrence que fE' = fG'; et donc que fE = fG (car ceux-ci ont même élément de tête); *quod erat monstrandum.*

Nous conclurons en conjecturant que le cas exemplaire des algorithmes de tri n'est pas un cas unique; et que l'associativité, au sens général où on l'entend dans le présent exposé, est une propriété dont il importe de tenir compte dans la conception des algorithmes; que l'on procède sur une seule voie ou sur plusieurs en parallèles.