**Résumé**

Motivation : Ce document présente une implémentation du bien connu algorithme de Smith-Waterman pour la comparaison de séquences protéiques et nucléiques, utilisant des instructions vidéo spécialisées. Ces instructions, semblables au SIMD dans leur design (conception), rendent possible la parallélisation de l'algorithme au niveau des instructions.

Résultats : les Points de référence (Bancs d'essai) sur ULTRA SPARC exécuté à 167 MHz montrent un facteur d'accélération de deux comparé au même algorithme d’implémentation avec des instructions d'entier sur la même machine. La performance atteint 18 millions de cellules matricielles par seconde sur un processeur unique, donnant à notre connaissance l’implémentation la plus rapide de l'algorithme de Smith-Waterman sur un poste de travail. La procédure accélérée a été introduite dans LASSAP (Large Scale Sequence compArison Package), un logiciel développé à l’INRIA, qui traite le parallélisme au plus haut niveau. Sur une entreprise SUN, 6000 serveurs avec 12 processeurs, une vitesse de presque 200 millions de cellules matricielles par seconde a été obtenue. Une séquence de longueur 300 acides aminés est parcourue contre SWISSPROT R33 (18531385 résidus) en 29 s. Cette procédure n'est pas limitée au balayage de banque de données. Il s'applique à tous les cas traités par LASSAP (intra et inter comparaisons de banques, le calcul de Z-score, etc).  
Disponibilité : Cette mise en œuvre est disponible par LASSAP paquet http: // www-rocq.inria.fr/genome  
Contact : Courrier électronique : Andrzej. Wozniak@inria.fr

**Introduction**

La performance d'algorithmes de comparaison de séquences protéiques et nucléiques est un des problèmes-clés pour le balayage de banque de données de séquences. En utilisant l'algorithme de Smith-Waterman régulier (SW), les niveaux réels de performance des processeurs utilisés dans des postes de travail varient de 2 à 10 millions de comparaisons par seconde.

Il y a quelques solutions spécialisées de matériel informatique (hardware), comme BIOCCELERATOR (Ltd., 1996) ou SAMBA (Lavenier, 1996), qui peut donner la performance de plus de deux séquences de grandeur plus élevée que des postes de travail, mais ces solutions ont aussi quelques inconvénients : haut coût, difficulté de programmation et débogage, le matériel informatique qui ne peut pas être réutilisé pour d'autres buts.

Ceci fait l’implémentation logicielle alternée, pure, pouvant s’exécuter en parallèles sur plusieurs postes de travail, une solution attractive. C'est le cas de LASSAP développé à l’INRIA (Glémet et Codani, 1996).

Du profil d'exécution de comparaison de séquences de LASSAP avec l'algorithme SW, nous avons constaté que le programme dépense 99 % du temps dans la procédure d'algorithme SW. Donc chaque amélioration de la vitesse de cette procédure donnera presque la même amélioration au niveau application.

Dans la section suivante, nous décrivons notre implémentation (mise en œuvre) de l'algorithme ; la section d’après présente les véritables points de référence (bancs d'essai) de balayage de banque de données. La section finale présente les conclusions.

**Implémentation de l'algorithme**

La figure 1 présente l’algorithme de bas de score affine SW (Smith and Waterman 1981) exprimé en langage C.

Deux séquences de protéine, *seqA* et *seqB*, de longueurs respectivement *lena* et *lenb*, sont comparées en utilisant la matrice de score *matrix* et deux paramètres *gap\_open* et *gap\_ext*, représentant les pénalités pour ouvrir et étendre un gap (trou), respectivement.

On peut voir cet algorithme comme une matrice des calculs de taille *lena x lenb*, où une lettre de la séquence *seqB* (la boucle extérieure) est comparée à toutes les lettres de la séquence *seqA* (la boucle intérieure). Pour chaque comparaison, trois résultats intermédiaires sont produits dans chaque itération : *nogap*, *a\_gap* et *b\_gap*, c'est-à-dire l'alignement parfait et des pénalités pour ouvrir et étendre le gap sur les sséquences *seqA* et *seqB*, respectivement. La valeur finale de score est calculée comme le maximum de toutes les valeurs trouvées dans chaque itération. Seulement deux vecteurs de résultats intermédiaires *nogap*[*lena*] et *b\_gap*[*lena*] doivent être stockés. La complexité globale de l'algorithme SW est *O*(*lena x lenb*).

En raison de la croissance de la taille des banques de données, parcourir en utilisant l'algorithme SW peut prendre des heures sur un poste de travail seul.  
De plus, des analyses à grande échelle émergeantes comme la construction des familles de séquences mènent à inter - et des comparaisons d'intra-banque-de-données. Ce processus peut prendre les semaines de calcul.  
L'accélération d'exécution peut être obtenue en utilisant des multiprocesseurs ou les groupes de postes de travail parce que les comparaisons d'ordre sont indépendantes. Dans cet article, nous présentons une technique parallélisation au niveau d'instruction de processeur à l'exécution d'accélération de l'algorithme SW plus loin.

On peut considérer deux sortes potentielles de parallelization :  
Pour comparaison d'ordre-à-banque et pour comparaison d'ordre-tosequence. Le premier est basé sur le manque de dépendance entre deux comparaisons consécutives. Ainsi, un ordre peut être comparé contre deux ou plus ordres au niveau d'instruction utilisant la même structure de contrôle. Bien sûr, chaque ordre comparé a son propre jeu de variables intermédiaires. Une mise en œuvre utilisant des principes semblables est décrite dans Alpern et d'autres. (1995).  
La susdite approche est limitée au balayage de banque de données pur.  
Ce n'est pas possible, par exemple, mettre en œuvre facilement l'ordre brouillant des procédures pour calculer le Z-grand-nombre (Lipman et d'autres., 1984; Pearson et Lipman, 1988). En outre, il ne peut pas profiter de la programmation de LASSAP'S des capacités qui permettent la combinaison 'instantanément' d'algorithmes par paires à base de comparaison. À titre d'exemple, LASSAP met en œuvre un algorithme appelé blsw qui calcule, pour chaque paire d'ordres, SW des alignements selon les résultats d'explosion (Altschul et d'autres., 1990).  
Notre approche est à parallelize au niveau de comparaison d'ordre seul(simple). Cela préserve l'efficacité dans chaque cas(caisse).  
On peut voir de la Figure 1(du Chiffre 1) qui aboutit le (je;) th l'itération dépendent (de / ' - 1,)), (/-l, j - 1) et (je, j - 1) des valeurs. Ceci peut être visualisé comme des dépendances verticales, diagonales et horizontales (la Figure(le Chiffre) 2).  
Si nous changeons du calcul de la deuxième rangée(dispute) un endroit(place) à droite, des rangées(disputes) 1 et 2 peut être exécuté dans parallèle, avec seulement un petit au-dessus de deux pas(étapes) : un sur le début de la boucle intérieure et le deuxième sur sa fin.

Si la longueur de l'ordre seqB est étrange(impaire), sur le dernier pas(étape) seulement une rangée(dispute) est calculée. Comme les valeurs de nogap [j] et b\_gap (j] calculé dans la première rangée(dispute) sont utilisées par le deuxième, nous pouvons transférer ces valeurs localement, se divisant par deux la largeur de bande de mémoire(souvenir) pour stocker et charger ces valeurs.  
Pour faire l'uniforme de calcul, on peut ajouter un symbole factice au commencement et à la fin de l'ordre seqA et un symbole factice à l'ordre seqB dans le cas où sa longueur n'est pas même. Ce plan(arrangement) peut être prolongé(étendu) au calcul parallèle d'un nombre(numéro) arbitraire de rangées(disputes).  
La figure(Le chiffre) 3 présente une mise en œuvre à quatre rangées(disputes) de l'algorithme dans le pseudocode de C. Les données tapent Vint4 est un vecteur contenant quatre entiers. VMAX et VSHIFT sont des opérateurs vectoriels. Le premier calcule un vecteur de par paires les maximums de vecteurs d'argument(de dispute), les deuxièmes changements ont laissé(quitté) tous les composants de son argument(dispute) vectoriel et mettent à l'endroit(la place) extrême droit son deuxième argument(dispute) d'entier. Un l l '+' et '-' un r e le vecteur ajoute et soustrait des opérateurs, respectivement.  
Bien qu'implementable dans le jeu d'instruction d'entier standard, ce plan(arrangement) de parallelization donne peu d'amélioration de performance(d'exécution) sur la mise en œuvre de base (voir des points de référence(bancs d'essai) sur la Figure(le Chiffre) 6).  
L'accélération substantielle peut être réalisée utilisant spécial, videooriented des instructions comme VIS (le Jeu d'Instruction Visuel) trouvé le processeur SPARC au soleil ULTRA (la Microélectronique solaire, 1996a).  
VIS des instructions sont exécuté dans l'unité de virgule flottante particulièrement améliorée(augmentée) (FPU) et utilisent les registres(enregistreurs) de 64 morceaux(de bit) de le  
FPU. Les instructions opèrent(fonctionnent) sur deux 32 bits, ou quatre données d'entier 16 bits emballées dans un mot double de 64 morceaux(de bit). Ces instructions incluent l'arithmétique AJOUTENT, SOUSTRAIENT, COMPARENT et  
MULTIPLIEZ-VOUS, la logique ET, OU, XOR, PAS et la conversion de données et la manipulation. On peut ajouter une instruction deux jeux de quatre entiers 16 bits et obtenir quatre résultats 16 bits, ou les multiplier de quatre entiers de 8 morceaux(de bit). Les détails peuvent être trouvés dans  
Microélectronique solaire (1996b).

Nous pouvons utiliser des instructions VIS d'exécuter dans quatre rangées(disputes) parallèles de l'algorithme SW. La mise en œuvre de la plupart des opérations de la Figure 3(du Chiffre 3) utilisant VIS l'interface intégrée est directe. Nous remplaçons le type Vint4 par le type vis\_d64 qui est en réalité un vecteur de quatre paquet d'entiers 16 bits dans un mot de 64 morceaux(de bit). VMAX l'opération est remplacé par VIS\_MAX\_INT de la Figure 5(du Chiffre 5). Ajoutez, soustrayez et VSHIFT est remplacé, respectivement, par vis\_fpaddl6, vis\_fpsubl6 et des instructions vis\_faligndata.  
SPARC ULTRA VIS compare les instructions calculent un masque de 4 morceaux(de bit) de résultat, qui peut être utilisé par une instruction de magasin(dépôt) sélective (de quatre mots 16 bits seulement ceux avec le jeu de morceaux(bits) de statut correspondant à 1 sont stockés). Malheureusement, nous avons besoin de beaucoup de comparaisons avec la réutilisation des résultats intermédiaires qui implique rechargent des valeurs stockées. Ceci est le temps et la consommation de largeur de bande de mémoire(souvenir) et diminue la vitesse d'exécution effective(efficace).  
Au lieu de cela, nous utilisons le calcul de valeur d'entier maximal basé sur le code d'entier présenté sur la Figure(le Chiffre) 4. Il calcule le maximum de deux entier signé estime int\_max et int\_val, le résultat est stocké dans int\_max. Ce code n'utilise pas de branches conditionnelles, donc il évite le saut potentiel misprediction et le nettoyage à grande eau de pré-decoded, des instructions partiellement exécutées ou spéculativement exécutées sur une certaine UC.  
Le résultat est correct tant qu'il n'y a pas ne débordent dans le calcul de la différence (int\_max - int\_val) donc la méthode peut être utilisée sans risque sur une gamme limitée d'arguments(de disputes).  
La figure(Le chiffre) 5 montre la mise en œuvre de la susdite méthode pour des entiers 16 bits emballés(entassés) dans des instructions VIS. Il n'y a aucune instruction de changement de morceau arithmétique dans VIS, donc le masque est produit avec l'instruction de multiplication vis\_fmul8ulxl6.  
Comme cette multiplication est exécutée sur une unité séparée et le processeur graphique est un moteur de question(publication) double, il y a peu si n'importe quelle dégradation de performance(d'exécution).  
L'unité graphique SPARC ULTRA est entièrement pipelined et a une latence de trois cycles d'horloge pour la multiplication. Donc le résultat peut être utilisé par une autre instruction seulement trois cycles d'horloge plus tard sans caler l'UC.  
Dans notre mise en œuvre, nous traitons huit rangées(disputes) immédiatement, donc une autre multiplication peut être commencée dans le cycle suivant, gardant l'UC occupée(animée) pour la performance(l'exécution) maximale. Le grand jeu de 32 registres(enregistreurs) de 64 morceaux(de bit) dans les aides d'unité graphiques garde tous les calculs locaux.  
La boucle intérieure est exécutée (l ena+7)/8 des temps. Comme nous avons mentionné auparavant, nous ajoutons sept symboles factices au commencement et à la fin de l'ordre A et l'ordre  
B est prolongé(étendu) à un multiple de huit.

Seize morceaux(Bit) ont signé la précision est acceptable pour la comparaison d'ordres protéiques et nucléiques. En effet, il à permet de calculer un score jusqu'à 32767 qui représente en moyenne un alignement parfait de longueur-5000 avec la matrice BLOSUM62.  
Cette longueur peut être plus grande(super) si les disparités(dissonances) et des écarts(trous) sont présents.  
La mise en œuvre garantit la saturation sur le score de 32767 (voir ci-dessous pour les détails de la mise en œuvre). En cas de la saturation, on peut calculer le score exact avec une mise en œuvre d'entier standard de l'algorithme. En pratique, de tels alignements avec le grand nombre très élevé peuvent être détectés avec des algorithmes plus rapides comme fasta (Pearson et Lipman, 1988) et le besoin, de toute façon, des études plus profondes en raison de leurs longueurs. Nous croyons que notre mise en œuvre couvre la plupart des besoins en utilisant l'algorithme SW.  
Le résultat (c'est-à-dire la valeur de score) est correct jusqu'à 32767 et sature ensuite à cette valeur. La preuve de justesse est comme suit. Le seul cas(caisse) de déborde dans la soustraction de valeurs d'entier signées est quand les deux opérandes ont de signe(panneau) opposé et leur différence est plus grande(super) dans l'ampleur que la valeur représentable maximale (32767,-32 768 pour des entiers 16 bits).  
Dans notre cas(caisse), la plupart de valeur négative d'a\_gap et b\_gap est  
- (Gap\_open + gap\_ext). La valeur de nogap est toujours positive donc la condition d'erreur pourrait surgir seulement quand un argument(dispute) du calcul de valeur maximal s'approche 32767 et l'autre est négatif.  
De l'algorithme dans la Figure(le Chiffre) 1, on peut voir qu'à n'importe quel point du calcul les valeurs de nogap, a\_gap et b\_gap peuvent changer seulement par des pas(étapes) limités basés en buts de valeurs matricielles, gap\_open et gap\_ext. La valeur absolue de la différence entre a\_gap ou b\_gap et les arguments(disputes) last\_nogap de la fonction de MAX (1) est liée par l'expression :  
2 x (Max (maU\x) + gap\_open + gap\_ext) - la minute (la matrice) où Max,/mn (la matrice) est les coûts de substitution extrêmes de la matrice marquante des points. Comme cette valeur est petite (d'habitude < 200) comparée à 32767, les valeurs de nogap, a\_gap et b\_gap sont strictement positives quand le score atteint la valeur de saturation.

**Points de référence (Bancs d'essai)**

Nous avons mis en œuvre l'algorithme SW de la Figure 1(du Chiffre 1) tant dans l'entier des instructions (32 bits) que dans VIS (16 bits) pour un  
Machine de SPARC ULTRA. Nous avons choisi quatre ordres typiques de longueur 104, 319 et 52J7, respectivement. Les résultats de vitesse d'exécution sont présentés dans la Figure(le Chiffre) 6.  
La mise en œuvre VIS de l'algorithme pour un SPARC  
La machine ULTRA est deux fois si rapidement que le code d'entier donnant la figure(le chiffre) de 18 millions de cellules matricielles par seconde. Pour la comparaison, nous donnons la performance(l'exécution) du même code d'entier recompilé et exécuté sur un ALPHA DE DÉCEMBRE 300 MHz (DÉCEMBRE UNIX) et le PENTIUM PRO 200 MHz (LINUX) Ensuite, le code ont été intégrés dans le progiciel du LASSAP'S et la demande(l'application) réelle(vraie) de parcourir un ordre de longueur 300 acides aminés contre SWISSPROT R33 (18531385 résidus) ont été exécutés sur SPARC ULTRA SOLAIRE basé  
Entreprise 6000 serveur avec 12 processeurs fonctionnant(courant) à 167 MHz. La figure(le chiffre) 7 présente les résultats du temps d'exécution réel(vrai) global de la demande(l'application) comme une fonction du nombre(numéro) de processeurs utilisés.  
La balance(Les échelles) de performance(d'exécution) globale(mondiale) tout à fait parfaitement avec le nombre de processeurs utilisés, étendant à 12 processeurs 200 millions de cellules matricielles par seconde. Nous croyons que nous pouvons obtenir 500 millions de cellules matricielles par seconde sur l'entreprise SUN existante 30 serveurs de processeurs. La performance semblable peut être obtenue utilisant des postes de travail SPARC ULTRA connectés par un réseau.  
Ce niveau de performance (d’exécution) peut être comparé aux solutions de matériel(quincaillerie) disponibles dans le commerce spécialisées comme le BIOCCELERATOR (Ltd., 1996) machine. Basé sur le XILINX des tableaux de porte programmables de terrain(des champs) (FPGA), l'accélérateur est composé d'un support contenant jusqu'à 16 puces(chips) d'accélérateur connectées à un poste de travail par l'interface SCSI.  
Une puce calcule 80 millions de cellules matricielles par seconde, donnant 1280 millions de cellules matricielles par seconde dans la configuration maximale.

**Conclusion**

La performance excellente a été obtenue utilisant VIS, bien que le VIS des instructions n'ont pas été destiné initialement au calcul de but général. Nous croyons que la méthode est séduisante(attractive) et peut être prolongée(étendue) à d'autres types d'algorithmes parallelizable donnant plus de puissance de traitement aux mises en œuvre logicielles pures.  
Bien que notre solution offre moins de performance(d'exécution) que le matériel(la quincaillerie) spécialisé, il offre plus de flexibilité. En outre, le principe est applicable à d'autres jeux d'instruction comme  
Le MMX pour INTEL LE PENTIUM et le PENTIUM PRO les processeurs qui seront présentés bientôt (Gwennap, 1996), permettant l'utilisation des groupes de PC comme la puissance de calcul.  
Nous nous attendons que à d'autres familles de processeur comme l'ALPHA DE DÉCEMBRE,  
L'HP PA-RISC et PowerPC suivra la tendance, ajoutant VISlike des instructions à leurs UC. Depuis la balance(les échelles) de performance(d'exécution) avec la vitesse d'horloge d'UC, nous nous attendons à la performance(l'exécution) proportionnellement plus haute sur les machines de 250 MHz SPARC ULTRA annoncées.  
Le plan(L'arrangement) de parallelization présenté dans ce papier(journal) n'est pas limité à la mise en œuvre VIS-SEMBLABLE Sur une UC de 64 morceaux(de bit), on peut emballer trois 21 morceaux(bit) ou quatre entiers 16 bits dans le mot de machine.  
Le vecteur ajoute et soustrait peut être calculé utilisant des masques de morceau(bit) pour éviter portent la propagation entre des entiers emballés(entassés). La preuve de manque de déborde peut être utilisé pour simplifier l'opération VMAX. Sur quelques machines, on peut s'attendre au gain de performance(d'exécution) comparé à la mise en œuvre d'entier pleine(complète).