RENPAR'15 / CFSE'3 / SympAAA'2003
La Colle sur Loup, France, 15 au 17 octobre 2003

Filtrage de bases de données sur le prototype RDISK

Stéphane Guyetant, Dominique Lavenier

IRISA, Université de Rennes 1,
Campus de Beaulieu,
35042 Rennes Cedex – France
{sguyetan,lavenier}@irisa.fr

Résumé
Le prototype RDISK est une architecture spécialisée dans les recherches par le contenu sur de grandes bases de données. Son principe est de filtrer à la volée les données dès la sortie du dispositif de stockage. Nous présentons son architecture et montrons comment elle répond aux exigences de la comparaison de séquences génomiques.

Mots-clés : bases de données génomiques, parallélisation des entrées/sorties, filtrage

---

1. Introduction

Les bases de données de séquences biologiques et leurs données associées (annotations, méta-données) sont en croissance exponentielle: depuis 1982, la taille de la banque européenne de données génomiques EMBL-bank double en moyenne tous les ans pour atteindre en mai 2003 plus de 39 milliards de bases [11]. Genbank fait partie de l'INSDC, collaboration internationale des bases de données de séquences de nucléotides, avec la banque américaine Genbank et la banque japonaise DDBJ. Toutes trois échangent leurs nouvelles entrées chaque jour et figent une version tous les deux mois.
Bien qu'un très grand nombre de génomes d'espèces animales et végétales soient en cours de séquençage, seuls l'homme, la souris et le rat représentent l'essentiel du volume de ces banques de données, ce qui laisse penser que leur croissance va continuer avec la même frénésie, surtout compte tenu de l'intérêt porté au domaine par la recherche biomédicale et l'industrie pharmaceutique.
En effet, la fonctionnalité biologique d'une protéine dépend beaucoup de sa forme tridimensionnelle et de ses sites actifs, normalement définis par sa séquence d'acides aminés. Lorsque l'on recherche la fonction d'un gène ou d'une protéine inconnue, on essaye de retrouver des similarités entre sa séquence d'acides aminés ou de ses bases d'ADN et celles de molécules connues et étudiées.
De nombreuses solutions logicielles existent pour retrouver des alignements – c'est à dire des ressemblances qui peuvent être approximatives – ayant un intérêt biologique potentiel entre une séquence requête et une base de données génomiques. Les premiers algorithmes, comme celui de Smith-Waterman [10] en 1981, utilisant des techniques de programmation dynamique, sont d'une complexité quadratique, si bien qu'il furent rapidement délaissés avec l'explosion de la taille des banques au profit de méthodes plus rapides. A partir de 1990 apparurent les algorithmes de comparaison de séquence basés sur des heuristiques, dont BLAST [1] est incontestablement le plus connu et le plus utilisé.
Si BLAST est devenu la référence en exploration de base de données génomiques, c'est qu'il permet d'exécuter dans un temps raisonnable l'alignement d'une requête contre de grands génomes sur une station de bureau. Son principe est de dire que tout alignement entre deux séquences possède certainement un très court fragment d'identité, ce qui est calculatoirement très rapide à repérer. Dans un premier temps, l'heuristique consiste donc à ne chercher que ces mots identiques de W caractères. De part et d'autre de ces mots, qui servent de point d'ancrage, on applique un algorithme classique de recherche d'alignement, dont on a globalement restreint l'espace de recherche (cf figure 1).
Pendant une dizaine d'années, l'accélération des traitements a surtout été une conséquence du calcul réparti sur des clusters de stations ou de l'ajout d'accélérateurs matériels en tant que périphériques

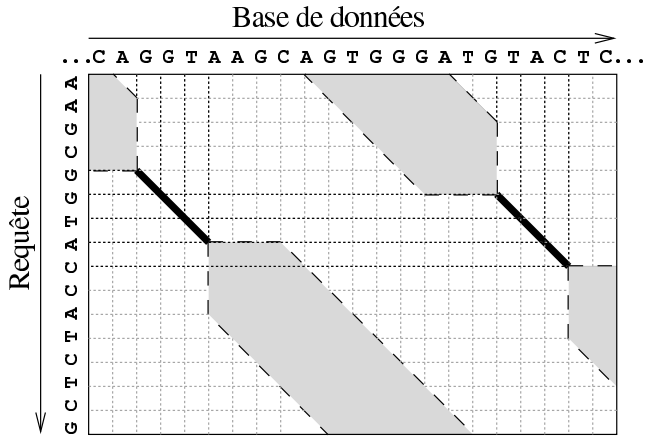

FIG. 1 – *Représentation de l'espace de recherche de similarités entre une séquence requête et une base de données: sur cet exemple, l'heuristique a trouvé deux ancrages:* GGTA *et* GTAC*; il ne reste donc à parcourir que les parties grisées. Le gain en temps d'exécution est le rapport de la surface de la matrice sur la surface grisée.*

de l'unité centrale. Plus récemment, de nouveaux travaux [7, 3] ont tenté de faire des heuristiques qui produisent moins de points d'ancrage, réduisant ainsi d'autant l'espace de recherche final – on dit que l'algorithme est plus sélectif – et de plus les points d'ancrage trouvés prédisent plus souvent un alignement intéressant – on dit que l'algorithme est plus sensible. Le problème est que plus l'étape heuristique de l'algorithme est complexe, pour produire des points d'ancrage de qualité, moins le gain de temps est évident parce que toute la base de données est traitée par l'heuristique.
C'est pour se libérer de ce compromis que l'architecture prototype RDISK [4, 6] a été définie: l'heuristique est considérée comme une étape de filtrage des données directement en sortie du disque dur. L'implémentation du filtrage dans un composant reconfigurable proche du disque permet d'une part de garantir un débit maximal mais surtout d'exécuter une recherche d'ancrage câblée en matériel, qui donc n'introduit pas de délai même pour un ancrage bien plus complexe.
Dans la suite de l'article nous présentons le prototype, de son fonctionnement global à une étude détaillée d'un noeud puis nous calculons l'équilibrage sur l'application de recherche de similitude dans les séquences d'ADN.

2. Présentation du prototype RDISK

L'objet du prototype est donc le filtrage à la volée et par le contenu de grandes bases de données. L'idée est de déporter le filtrage au plus près du lieu de stockage des données (cf figure 2) pour s'affranchir du goulot d'étranglement que constituent les entrées/sorties et de paralléliser l'accès aux données pour augmenter le débit agrégé. L'approche est d'autant plus profitable que l'application possède un prétraitement des données qui est parallélisable et s'applique indépendamment à l'ensemble des données. Les banques de données sont donc réparties sur l'ensemble des disques; lorsque l'hôte reçoit une requête, il détermine le type de filtre à implémenter dans le composant FPGA et le diffuse à l'ensemble des cartes. Dès que les cartes sont configurées l'hôte envoie la requête à chaque filtre qui démarre la recherche sur le sous-ensemble de la base qui lui est associé. Les résultats potentiels sont alors envoyés à la station hôte où des traitements de haut niveau finalisent les résultats de la requête.
Les disques sont montés sur un circuit imprimé (cf figure 3) avec un composant FPGA qui contient le filtre: ce couplage fort évite les pertes de qualité du signal (diaphonie, rebond à la masse) que l'on a avec les nappes à haute fréquence. Le FPGA est un composant économique Xilinx Spartan-II équivalent à 200000 portes logiques.

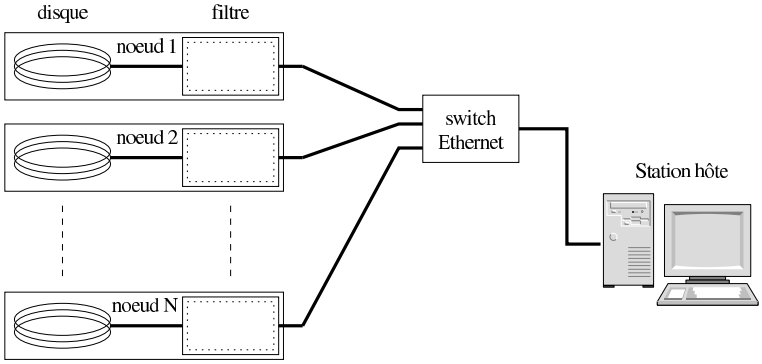

FIG. 2 – *Architecture globale du prototype* RDISK*: un grand nombre de noeuds comprenant un disque dur et un dispositif de filtrage sont reliés à une station hôte par un réseau en étoile.*

Définition du protocole réseau

Les points principaux de spécification du protocole réseau étaient la facilité de mise en oeuvre et un coût modéré. Ainsi, toutes les cartes sont reliées entre elles et avec la station hôte par un réseau Ethernet 100Mbit/s. La couche supérieure du protocole est dérivée de UDP (*User Datagram Protocol*); elle est donc assez générique pour faciliter le développement, mais plus légère: cela permet de doubler le débit que l'on aurait avec TCP/IP (*Transmission Control Protocol / Internet Protocol*). En effet l'agencement hiérarchique de notre réseau système supprime les contrôles d'intégrité et de gestion de paquets et les communications carte à carte ne sont pas prévues.

3. Étude d'un noeud: aspects système

Configuration du composant FPGA

A chaque application, voire à chaque requête correspond une configuration différente du FPGA. Une solution de configuration à base d'EEPROM ne peut donc convenir, vu le grand nombre de configurations à stocker. Une possibilité consisterait à envoyer les données de configuration par le réseau depuis l'hôte, mais les communications réseau sont gouvernées par un module dans le FPGA. La solution retenue utilise l'espace de stockage gratuit du disque: au démarrage de la carte, un microcontrôleur de type 8051 lit une configuration générale sur une partition dédiée du disque et charge le FPGA. Ensuite, il libère le contrôle du disque et s'endort en attente d'une interruption.

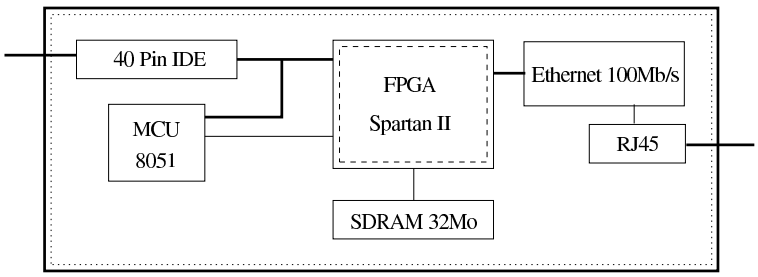

FIG. 3 – *Les noeuds s'organisent autour du FPGA: les données provenant du disque par l'interface IDE sont filtrées puis envoyées sur le réseau après création des trames dans le contrôleur Ethernet. Le microcontrôleur de type 8051 gère les reconfigurations du FPGA.*

Description du système sur puce reconfigurable

Une fois configuré, le FPGA contient un système sur puce autonome: un processeur RISC 16 bits configure ses périphériques, charge son code en mémoire depuis le disque et contrôle les flux en entrée et sortie du filtre et de la mémoire. Nous disposons d'un compilateur C adapté de LCC pour le programmer. Lorsque le débit en sortie du filtre est très faible, il prend également en charge le formatage des trames réseau. Les périphériques sont les contrôleurs ATA-IDE vers le disque dur, SDRAM vers la mémoire vive, Ethernet vers le contrôleur réseau, EPP vers le port parallèle de développement et bien sûr le filtre.

Scénario de reconfiguration

Une requête d'un type nouveau nécessite la reconfiguration de toutes les cartes. Elle n'est pas pénalisante, car elle s'opère en 800ms,temps négligeable comparé aux quelques minutes que prend la lecture d'une banque. Si la configuration est déjà sur les disques, il suffit d'envoyer la commande de configuration avec son identifiant. Sinon on commande son écriture sur le disque juste avant. Le FPGA communique au microcontrôleur l'identifiant de la prochaine configuration. Ce dernier le réinitialise et procède à une configuration comme décrit ci-dessus; ainsi, pendant un court instant, la carte n'est plus présente sur le réseau. En cas d'erreur lors du processus, la configuration de démarrage est reprise pour que la carte puisse se signaler à l'hôte.

4. Équilibrage du prototype

Le fonctionnement nominal du prototype n'est atteignable que si le prototype est équilibré; cela signifie qu'aucun de ses composants n'est ni en attente ni en saturation. Dans cette partie, nous allons estimer les performances du prototype en se basant sur l'application de comparaison de séquences d'ADN. Considérons le point de vue "flot de données", c'est à dire le canal que va suivre la donnée atomique (un fragment de séquence), depuis son stockage quelque part à la surface du disque jusqu'à l'obtention d'un résultat valide.

Débit soutenu du disque dur

Le disque est un Seagate Barracuda IV de 40 Giga-octets choisi pour son prix, sa fiabilité et sa bonne vitesse de lecture séquentielle. En effet, si les disques de la norme IDE Ultra-DMA mode 5 promettent des débits de crête jusqu'à 100Mo/s, il s'agit de transferts depuis le buffer du disque, depuis une mémoire cache de 2Mo; mais la valeur qui nous intéresse est le débit soutenu, qui tombe de 41 à 24Mo/s selon que la donnée soit écrite respectivement à l'extérieur ou à l'intérieur du disque, avec une chute très rapide au delà de la mi-capacité [8]. Notons que ce débit soutenu touche les limites technologiques, et que l'amélioration à prévoir est faible (+7% par an environ), la seule solution consistant à augmenter le nombre de plateaux, ce qui revient au même que de paralléliser des disques dans un système RAID.

Choix du contrôleur de disque dur

Le disque peut fonctionner avec deux protocoles principaux: l'un, PIO, pour *Programmed IO*, permet d'atteindre 16Mo/s avec un contrôleur simple et nécessitant peu de ressources. L'autre mode, Ultra-DMA, permet d'atteindre le débit théorique de 50Mo/s en lecture avec un contrôleur plus complexe, ayant une gestion plus fine des contraintes de temps. Aussi pour maximiser la logique disponible pour le module de filtrage dans le composant FPGA, nous utilisons le mode PIO avec les données compressées à 3,9 bases par octet, soit un débit D de 66 millions de bases par seconde.

Sélectivité du filtrage

Le filtre correspond à la fin du premier étage, parce qu'en entrée, le filtre est "glouton", c'est à dire toujours en attente de données, et qu'une très faible quantité en ressort. L'utilisateur peut choisir le filtre qu'il veut implémenter: le plus simple consiste en une recherche exacte d'un mot, ou alors de quelques courts mots proches, mais des résultats de meilleure qualité sont obtenus avec des motifs disjoints permettant d'introduire un taux d'erreur acceptable. Une deuxième passe permet d'augmenter la sélectivité du filtrage en étudiant sommairement le voisinage immédiat du mot précédemment trouvé. Retenons que si les applications les plus rapides filtrent à environ 0,1%, les plus limitantes ne devront

pas dépasser 1% du volume des données traversant le filtre. En notant T ce taux, la quantité de données à traiter en sortie du filtre est de $D \cdot T$.

Convergence des débits

Toutes ces données sont rassemblées depuis les N cartes vers la station hôte par le réseau. Nous considérons que le réseau ne saturera jamais parce que ce ne sont pas les séquences en elles-mêmes qui transitent mais leurs identifiants et la position des ancres trouvées. En effet un calcul pire-cas sur notre lien Ethernet 100Mbit/s montre que 12 cartes peuvent envoyer des réponses à plein régime sur le réseau, or ceci n'arrive que si il y a une très forte homogénéité des données localement, donc a priori sur un seul disque. Ainsi, la convergence ne fait qu'introduire la multiplication du débit moyen, on traite donc à l'étape suivante $T \cdot N \cdot D$ bases par seconde.

Puissance de calcul dans l'hôte

Tout d'abord on vérifie que le stockage de masse local peut fournir les données au rythme ou le réseau les appelle, ce qui ne pose pas de problème particulier en dessous d'une centaine de cartes. On évalue la quantité de calculs à effectuer en nombre de mises à jour de cellules de l'espace de recherche (MCPS pour *matrix cell update per second*). Cet espace est représenté figure 1: à chaque colonne correspond une base provenant de la base de données et chaque ligne est une base de la séquence requête. Avec une requête longue de L bases, nous devons traiter $\rho \cdot L \cdot T \cdot N \cdot D$ MCPS. Une utilisation courante des logiciels d'alignement est de rechercher un EST (*Expressed Sequence Tag*, désignant des fragments d'ARN messager qui sont la partie active d'un gène) de longueur environ 500 bases. Le coefficient ρ correspond à une restriction supplémentaire de l'espace de recherche (cf figure 4) que l'on évalue à 0,2. L'équilibrage consiste à fournir alors au processeur hôte le nombre de MCPS qu'il peut traiter: l'algorithme Paralign [9] exécuté sur une station de bureau à base de Pentium IV cadencé à 1,5 GHz permet de calculer 1646 MMCPS. Sur cet exemple d'application, nous aurons donc un prototype équilibré pour $N = 16$ cartes. Néanmoins, l'ajout d'un coprocesseur spécialisé nous permettra d'une part d'utiliser des séquences requête plus longues et d'autre part de paralléliser plus de cartes pour diminuer le temps de parcours global de la base de données. Par exemple, le processeur SWASAD [5], conçu en 2002 en technologie 0,5 micron, annonce une performance de 3200 MMCPS. Prenons ce chiffre avec réserve, car la performance maximale est atteinte avec des calculs très réguliers dans le réseau systolique, ce qui n'est pas le cas dans la figure 4, en b) et encore moins en c).

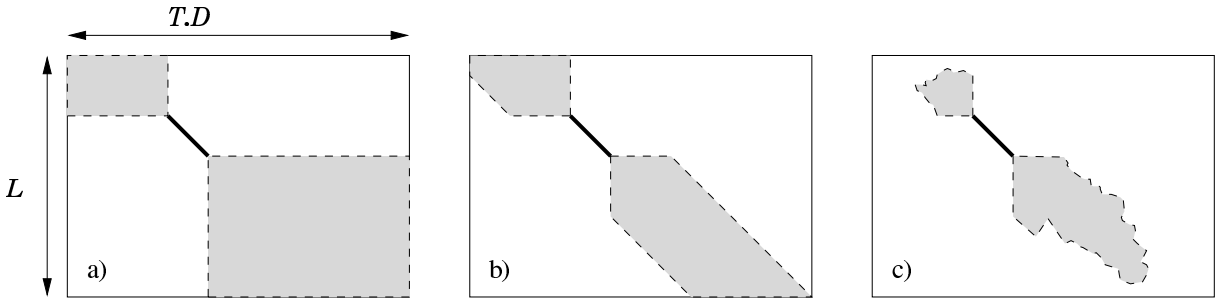

FIG. 4 – *La zone grisée représente l'espace à explorer par l'algorithme exhaustif (sur la station hôte) pour un ancrage repéré (cf Fig. 1). En a), ρ vaut environ 0,5. En b), on limite le gap maximal – l'insertion de bases d'ADN dans l'une des séquences – pour diminuer ρ à 0,2. En c), on évalue le score d'alignement en tout point et on arrête la recherche en dessous d'un seuil. ρ est plus faible mais l'évaluation dynamique du score prend plus de temps.*

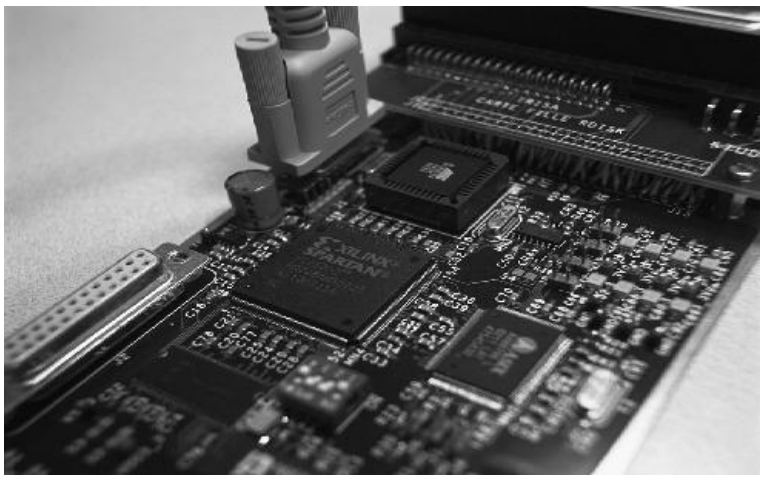

FIG. 5 – *Vue d'une des cartes du prototype: au premier plan, le contrôleur Ethernet et le FPGA. Au fond, on distingue le disque dur sur son support. A gauche, une liaison série supervise l'écriture d'une configuration sur le disque pendant le développement.*

5. Conclusion

Nous avons présenté l'architecture du prototype RDISK, spécialisé dans le filtrage de grandes bases de données. Nous avons conçu une carte de base (cf figure 5) qui est entièrement fonctionnelle, sur laquelle les premiers tests de performance et de faisabilité ont pu être conduits. Le prototype complet avec 48 cartes est actuellement en cours de réalisation. Dans ce papier, nous nous sommes concentrés sur l'étude de la recherche de similarité dans les banques d'ADN; l'essai sur le prototype permettra de fixer les paramètres de sélectivité des filtres et diminuer T pour équilibrer l'architecture avec plus de cartes.
De plus, toute application de parcours de très grandes bases de données possédant une règle de filtrage systématique peut être envisagée sur le prototype. Pour la validation de notre architecture, nous proposerons également une recherche de motifs complexes exprimés par des automates pondérés dans les bases génomiques ainsi qu'une recherche d'image par le contenu.
Cette dernière se prête particulièrement bien à une implémentation sur RDISK grâce à la technique des *descripteurs locaux* [2]: plutôt que de stocker des millions d'images dans la base de données, on représente ces images au moyen d'un ensemble de descripteurs locaux, représentatifs des parties pertinentes de l'image. Ils sont idéalement robustes aux changements d'échelle, aux variations de luminosité et contraste, au bruit, aux rotations, aux redimensionnements.
La recherche d'images commence par le calcul de distance entre descripteurs, qui prend plus de 99% du temps de calcul total en logiciel. Cet opérateur se prête particulièrement bien à une implémentation matérielle parallélisée, d'où l'intérêt de l'expérimentation sur RDISK, réalisée conjointement avec l'équipe TEX-MEX de l'IRISA. Une utilisation possible par les agences de photographies, dont les besoins de stockage sont de plus de 300 Giga-octets de données, est de vérifier les copyrights des images qu'elles distribuent. Cette vérification qui prend actuellement plusieurs heures pourrait être réduite à quelques secondes.

Bibliographie

1. Altschul (Stephen F.), Gish (Warren), Miller (Webb), Myers (Eugene W.) et Lipman (David J.). – Basic local alignment search tool. *Journal of Molecular Biology*, vol. 215, n° 3, 1990, pp. 403–410.
2. Amsaleg (Laurent), Gros (Patrick) et Mezhoud (Rim). – Mise en base d'images indexées par des descripteurs locaux: problèmes et perspectives. *Publication interne IRISA*, 2000.
3. Brudno (Michael) et Morgenstern (Burkhard). – Fast and sensitive alignment of large genomic sequences. *In : Proc. Bioinformatics Conference (CSB'02)*, éd. par CSP (IEEE), p. 138.

4. Guyetant (Stephane), Derrien (Steven) et Lavenier (Dominique). – Architecture parallèle reconfigurable pour la génomique. *In : Sympa'8*.
5. Han (Tony) et Parameswaran (Sri). – Swasad: An asic design for high speed dna sequence matching. *In : Proc. 15th Int. Conf. on VLSI Design*, éd. par CSP (IEEE).
6. Lavenier (Dominique), Guyetant (Stephane), Derrien (Steven) et Rubini (Stephane). – A reconfigurable parallel disk system for filtering genomic banks. *In : Proc. Int. Conf. ERSA'03*.
7. Ma (Bin), Tromp (John) et Li (Ming). – Patternhunter: faster and more sensitive homology search. *Bioinformatics*, vol. 18, n° 3, 2002, pp. 440–445.
8. Prieur (Marc). – Comparatif: 27 disques ide 7200tpm. – http://www.hardware.fr, novembre 2002.
9. Rognes (Torbjorn). – Paralign: a parallel sequence alignment algorithm for rapid and sensitive database searches. *Nucleic Acids Research*, vol. 29, n° 7, 2001, pp. 1647–1652.
10. Smith (T.) et Waterman (M.). – Identification of common molecular subsequences. *Journal of Molecular Biology*, vol. 221, 1981, pp. 403–420.
11. Stoesser (Guenter), Baker (Wendy) et al. – The embl nucleotide sequence database: major new developments. *Nucleic Acids Research*, vol. 31, n° 1, 2003, pp. 17–22.