Mineure "Calcul Parallèle"
Correction

Eric Goubault

30 mars 1999

On se propose dans ce problème d'implémenter rapidement, sur une machine parallèle, des opérations élémentaires sur des polynômes complexes. On supposera typiquement que l'on dispose pour les implémentations d'un réseau de stations travaillant sous PVM, ayant toujours suffisament de processeurs pour que chaque tâche nouvelle s'exécute sur un processeur nouveau. Dans l'écriture des pseudo-codes PVM demandés, on se contentera d'une écriture raccourcie des principales primitives de communication. On pourra écrire par exemple `spawn(esclave,k)`, `send(esclave(i),p(j)` $|\, 0 \leq j \leq n$`)` et `recv(maitre,q(1))` pour dire respectivement, créer `k` copies du programme `esclave`, envoyer (même sémantique que `pvm_send`) la liste de valeurs `p(0),p(1),...,p(n)` au `i`ème esclave, et enfin, recevoir (même sémantique que `pvm_recv`) une valeur du `maitre` à ranger dans la variable locale `q(1)`.

Les sections 1, 2 et 3 peuvent être traitées indépendament.

1 Multiplication rapide

Soient,

$$P = \sum_{i=0}^{n} p_i X^i$$

$$Q = \sum_{i=0}^{m} q_i X^i$$

$$R = \sum_{i=0}^{n+m} r_i X^i$$

trois polynômes complexes de degrés respectifs n, m et $n+m$ avec $R = PQ$. On a en particulier,

$$r_k = \sum_{j=0}^{k} p_j q_{k-j}$$

Question 1.1: *Proposer un algorithme parallèle simple qui affecte le calcul des r_k sur des processus A_k $(0 \leq k \leq m+n)$ distincts.*
Réponse: Un processus esclave par r_k. Soit l'algorithme:

```
int P[n];
int Q[m];
int R[m+n];

/* send a polynome */
emit(int tid,int P[],int n){
 send(tid,n);
 send(tid,P(i)|0 ≤ i < n);
}

maitre(){
 spawn(esclave,m+n);
 for(i=0; i<m+n; i++) {
  emit(esclave(i),P,n);
  emit(esclave(i),Q,m);}
 for(i=0; i<m+n; i++) recv(esclave(i),R[i]);
}

/* receive a polynome */
copy(int P[],int n){
 recv(maitre,n);
 recv(maitre,P(i)|0 ≤ i < n);
}

A_k: esclave(){
 copy(P,n);
 copy(Q,m);
 for(j=0,R=0; j>k; R=R+P[j]Q[k-j],j++);
 emit(maitre,R,k);
}
```

Question 1.2: *En négligeant les coûts de communication, en supposant que les processus A_k démarrent tous en même temps, et qu'ils s'exécutent exactement à la même vitesse, quel est le temps de calcul de l'algorithme de la question 1.1? (c'est-à-dire quel est le nombre d'opérations flottantes maximal effectué par les A_k ?)*

Réponse 1.2: Si on néglige les coûts de communication, et en supposant que les processus A_k démarrent tous en même temps, et qu'ils s'exécutent exactement à la même vitesse, le temps de calcul de l'algorithme de la question 1.1 est: (avec σ le temps de calcul d'une somme, π celui d'un produit et $\nu = \sigma + \pi$)

$$\begin{aligned} T_k &= k\,\sigma + k\,\pi \\ &= k\,\nu \end{aligned}$$

chaque A_k effectue en parallèle $k * \nu$ opérations flottantes. Le maximum est effectué par A_{m+n} soit

$$T = (m+n)\,\nu$$

car $T = \max_k T_k$.

On va examiner maintenant un autre algorithme, utilisant la transformée de Fourier discrète. Soit $l \in \mathbb{N}$ une puissance de 2 telle que $n+m < l \leq 2(n+m)$. Considérons les vecteurs de $\mathbb{C}^{l+1}$,

$$\mathbf{p} \ = \ \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad \mathbf{q} \ = \ \begin{pmatrix} q_0 \\ q_1 \\ \vdots \\ q_m \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Alors on calcule successivement,

- *a) $\widetilde{\mathbf{p}}$ et $\widetilde{\mathbf{q}}$ les transformées de Fourier (rapides) des vecteurs $\mathbf{p}$ et $\mathbf{q}$,*
- *b) $\widetilde{\mathbf{r}}$ le vecteur dont la jième composante est $\widetilde{\mathbf{r}}_j = \widetilde{\mathbf{p}}_j \widetilde{\mathbf{q}}_j$ $(0 \leq j \leq l)$,*
- *c) la transformée de Fourier inverse $\mathbf{s}$ de $\widetilde{\mathbf{r}}$.*

Question 1.3: *Prouver brièvement que*

$$R = PQ = \sum_{i=0}^{l} s_j X^j$$

(les s_j sont les composantes du vecteur s).
Réponse 1.3: on a (ω_{l+1} racine n+1 ième de l'unité)

$$\widetilde{\mathbf{p}} = \sum_{k=0}^{l} \mathbf{p}(\omega_{l+1}^k) Y^k = FFT(\mathbf{p}, \omega_{l+1})$$

et

$$\widetilde{\mathbf{q}} = \sum_{k=0}^{l} \mathbf{q}(\omega_{l+1}^k) Y^k = FFT(\mathbf{q}, \omega_{l+1})$$

et

$$\widetilde{\mathbf{r}}_j = \widetilde{\mathbf{p}}_j \widetilde{\mathbf{q}}_j = \mathbf{p}(\omega_{l+1}^j)\mathbf{q}(\omega_{l+1}^j)$$

soit

$$\mathbf{r} = FFT^{-1}(\widetilde{\mathbf{r}}, \omega_{l+1}) = \frac{1}{l+1} FFT(\widetilde{\mathbf{r}}, \omega_{l+1}^{-1})$$

Question 1.4: *Etant donné un code PVM (que l'on ne re-écrira pas) **`fft`** qui effectue par la méthode "diviser pour régner" parallèle la transformée de Fourier rapide de vecteurs de $\mathbb{C}^{l+1}$, écrire un pseudo-code implémentant l'algorithme de la question 1.3. Quel en est son temps d'exécution parallèle (avec les mêmes hypothèses que pour la question 1.2)?*
Réponse 1.4: Un processus esclave par fft. Soit l'algorithme :

```
int P[l+1], Q[l+1], R[l+1];
int FP[l+1],FQ[l+1],FR[l+1];

/* FFT de P et Q */
```

```
FP=fft(P,l+1);
FQ=fft(Q,l+1);

/* Produit des FFTs */
for(i=0;i<l+1;i++){
 FR[i]=FP[i]*FQ[i];}

/* FFT inverse du produit*/
R=fft(FR,-(l+1))/(l+1);
```

Temps d'execution parallèle: $O(n \lg n)$

2 Matrices de Toeplitz

Soit M une matrice complexe carrée $n \times n$. M est dite de Toeplitz si pour tout k et l, $2 \leq l, k \leq n$, $M_{k,l} = M_{k-1.l-1}$.

2.1 Produit matrice vecteur

Soient M une matrice de Toeplitz $n \times n$ et $x \in \mathbb{C}^n$ un vecteur complexe de dimension n,

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

On veut calculer efficacement le vecteur $y = Mx$.

<u>Question 2.1.1:</u>
Proposer un pseudo-code PVM permettant de paralléliser simplement le calcul du produit matrice-vecteur Mx. Quel en est le temps global d'exécution en faisant les mêmes hypothèses que dans la section 1?

On remarque qu'en fait, M est déterminée entièrement par un vecteur $m \in \mathbb{C}^{2n-1}$ et que $M_{i,j} = m_{n+i-j}$ On peut alors voir facilement que, pour $1 \leq j \leq n$,

$$y_i = \sum_{j=1}^{n} x_j m_{n+i-j}$$

Donc pour $n+1 \leq i \leq 2n$,

$$y_{k-n} = \sum_{j=1}^{n} x_j m_{k-j}$$

<u>Réponse:</u> Un processus esclave par y_i. Soit l'algorithme:

```
int y[n];
int M[n][n];
int x[n];
```

```
/* send a polynome */
emit(int tid,int P[],int n){
 send(tid,n);
 send(tid,P(i)|0 ≤ i < n);
}

maitre(){
 spawn(esclave,n);
 for(i=0; i<n; i++) {
  emit(esclave(i),M[i],n);
  emit(esclave(i),x,n);}
 for(i=0; i<n; i++) recv(esclave(i),y[i]);
}

/* receive a polynome */
copy(int P[],int n){
 recv(maitre,n);
 recv(maitre,P(i)|0 ≤ i < n);
}

A_k: esclave(){
 copy(M,n);
 copy(X,n);
 for(j=0,Y=0; j<n; Y=Y+M[j]X[j],j++);
 emit(maitre,Y,n);
}
```

Question 2.1.2:

En faisant le parallèle avec la formule donnant r_k en fonction de p_j et q_{k-j} dans la section 1 et en s'inspirant de l'algorithme de la question 1.3 utilisant la transformée de Fourier rapide, décrire brièvement un algorithme parallèle effectuant le calcul de y.

Réponse: On a

$$M[1,n] = m_1 \xrightarrow{j} M[1,1] = m_n \xrightarrow{i} M[n,1] = m_{2n-1}$$

soit

$$y_i = \sum_{j=1}^{n} M_{i,j}\, x_{j=}m[n+i-1..i]\, x = \sum_{j=1}^{n} m_{n+i-j} x_j = \sum_{j=1}^{n} m_{k-j} x_j = FFT^{-1}(FFT(x)\, FFT(m))$$

Un processus esclave par r_k. Soit l'algorithme:

```
int P[n];
int Q[m];
int R[m+n];

/*
```

2.2 Inversion

Ici on cherche, étant donné une matrice de Toeplitz non-singulière triangulaire inférieure $n \times n$ M, à trouver son inverse M^{-1}. On suppose ici que n est une puissance de 2. On va utiliser pour ce faire un algorithme de type "diviser pour régner" parallèle. Pour ce faire, on écrit M par blocs $\frac{n}{2} \times \frac{n}{2}$ comme,

$$M = \begin{pmatrix} M_1 & 0 \\ M_2 & M_1 \end{pmatrix}$$

où M_1 et M_2 sont deux matrices de Toeplitz non-singulières, M_1 étant triangulaire inférieure. Alors on a,

$$M^{-1} = \begin{pmatrix} M_1^{-1} & 0 \\ -M_1^{-1} M_2 M_1^{-1} & M_1^{-1} \end{pmatrix}$$

<u>Question 2.2.1:</u>

Trouver un algorithme récursif parallèle de calcul de M^{-1} dont on écrira le code en pseudo-PVM. On supposera pour cela que l'on a une implémentation de l'algorithme de la question 2.1.2 et on remarquera que comme M_1 et M_2 sont de Toeplitz, $-M_1^{-1}M_2M_1^{-1}$ est de Toeplitz et est donnée par sa première colonne

$$-M_1^{-1}M_2M_1^{-1} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

<u>Réponse:</u> Un processus esclave par r_k. Soit l'algorithme:

```
int P[n];
int Q[m];
int R[m+n];

/*
```

<u>Question 2.2.2:</u>

Quel est le temps d'exécution de l'algorithme 2.2.1 avec les mêmes hypothèses qu'en 1.2?

<u>Réponse:</u>

<u>Question 2.2.3:</u>

Comment cet algorithme se compare-t-il avec l'algorithme de remontée de Gauss du cours (sur des matrices denses)?

<u>Réponse:</u>

3 Division rapide

Soient S et T deux polynômes complexes de degrés respectifs n et m. On suppose $n > m$. On veut déterminer rapidement le quotient Q et le reste R de la division euclidienne de S par T, c'est à dire les uniques polynômes complexes tels que, $S = QT + R$ avec degré de R strictement inférieur à m.

Question 3.1 :

Supposons que l'on a pu calculer Q par quelque procédé. Donner un algorithme parallèle de calcul de R. Quel en est son temps global de calcul (encore une fois sous les mêmes hypothèses que dans la section 1) ?

Réponse:

On écrit,

$$S = \sum_{i=0}^{n} s_i X^i$$

$$T = \sum_{i=0}^{m} t_i X^i$$

$$Q = \sum_{i=0}^{n-m} q_i X^i$$

Alors, on peut remarquer que,

$$\begin{pmatrix} s_n \\ s_{n-1} \\ \vdots \\ s_m \end{pmatrix} = \begin{pmatrix} t_m & 0 & \cdots & 0 \\ t_{m-1} & t_m & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ t_{2m-n} & t_{2m-n+1} & \cdots & t_m \end{pmatrix} \begin{pmatrix} q_{n-m} \\ q_{n-m-1} \\ \vdots \\ q_0 \end{pmatrix}$$

Question 3.2 :

Déduire de cette remarque un algorithme parallèle efficace de calcul de Q.

Réponse:

Question 3.3 (facultative):

Supposons que l'on a pu calculer R par quelque autre procédé. Donner un algorithme parallèle de calcul de Q. Pour ce faire on remarquera que $Q = \frac{S-R}{T}$ et on s'inspirera de l'algorithme de multiplication rapide par FFT de la section 1. Quel en est son temps global de calcul (encore une fois sous les mêmes hypothèses que dans la section 1) ?

Réponse: