Travaux Pratiques de Calcul Distribué n°3

Cours d'Informatique de l'École Centrale d'Électronique

—Ingénieur Quatrième Année—

Cette feuille est une étude d'une méthode récente de calcul de π, méthode due à David Bailey, Peter Borwein et Simon Plouffe. L'algorithme a l'avantage de pouvoir être distribué puisqu'il permet de calculer une hexadécimale de π sans calculer les autres.

Jusqu'à présent, on croyait que calculer la i-ème décimale d'un nombre transcendent comme π était aussi difficile que de calculer π lui-même. En fait, il n'en est rien comme viennent de le montrer David Bailey, Peter Borwein et Simon Plouffe. En effet, ils ont pu calculer le 10 milliardième chiffre du développement hexadécimal de π en utilisant la formule :

$$\pi = \sum_{n=0}^{\infty} \left(\frac{4}{8n+1} - \frac{2}{8n+4} - \frac{1}{8n+5} - \frac{1}{8n+6} \right) \left(\frac{1}{16} \right)^n .$$

Cette formule présente cet immense avantage de permettre de calculer la i-ème hexadécimale de π sans être obligé de calculer toutes les précédentes.

▸ **Exercice 1.** (**algorithme de Bailey, Borwein et Plouffe**) *Selon Bailey, Borwein et Plouffe, partant de cette formule, on peut déduire un algorithme ne nécessitant pas d'arithmétique multi-précision, n'utilisant que peu de mémoire et dont les performances sont intéressantes par rapport à l'ordre de l'hexadécimale désirée, pour calculer le développement hexadécimal de π. Puisque nous pouvons calculer une hexadécimale de π sans calculer les autres, nous pouvons distribuer le calcul de n hexadécimales sur plusieurs machines, chaque machine calculant par exemple un intervalle de digits hexadécimaux.*

Nous avons développé un programme réalisant effectivement la distribution du calcul de π sur un réseau de machines hétérogènes regroupées en une machine parallèle virtuelle à l'aide de `PVM` *:*

```
$ pi_master 780 5

  3.243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89452821E638D013
    77BE5466CF34E90C6CC0AC29B7C97C50DD3F84D5B5B54709179216D5D98979FB1BD1310BA698DF
    B5AC2FFD72DBD01ADFB7B8E1AFED6A267E96BA7C9045F12C7F9924A19947B3916CF70801F2E285
    8EFC16636920D871574E69A458FEA3F4933D7E0D95748F728EB658718BCD5882154AEE7B54A41D
    C25A59B59C30D5392AF26013C5D1B023286085F0CA417918B8DB38EF8E79DCB0603A180E6C9E0E
    8BB01E8A3ED71577C1BD314B2778AF2FDA55605C60E65525F3AA55AB945748986263E8144055CA
    396A2AAB10B6B4CC5C341141E8CEA15486AF7C72E993B3EE1411636FBC2A2BA9C55D741831F6CE
    5C3E169B87931EAFD6BA336C24CF5C7A325381289586773B8F48986B4BB9AFC4BFE81B66282193
    61D809CCFB21A991487CAC605DEC8032EF845D5DE98575B1DC262302EB651B8823893E81D396AC
    C50F6D6FF383F442392E0B4482A484200469C8F04A9E1F9B5E21C66842F6E96C9A670C9C61ABD3
```

Une partie du programme vous est donnée. En particulier, on donne le fichier `pi_slave` *qui implémente l'algorithme de calcul et permet ainsi de produire sur demande un intervalle de 13 hexadécimales.*

Le fichier source `pi_slave.c` *n'est pas fourni mais le protocole de calcul correspond à l'extrait de programme suivant :*

```
void main(void)
{ int mytid, pos, master, i, j=0, k, decitranch, result_pos;
  double y, s1, s2, s3, s4, ss, pi_decimal;
  char pi_hexa[2*PART_SIZE];
  int buffer_id, bytes, msgtag, tasktid;
  char deci_hexa[16]={'0','1','2','3','4','5','6','7',  /* hexa. digits   */
                      '8','9','A','B','C','D','E','F'};

  struct rusage used_time;                              /* time structure */
  double user_sys_time=(double)0.0;                     /* user time      */

  mytid=pvm_mytid();                                    /* enrolling PVM  */
  master=pvm_parent();                                  /* parent's tid   */

  decitranch=(int)(PART_SIZE/PARTS_DIVISION)+1;

  while(1)                                              /* infinite loop  */
  { buffer_id=pvm_recv(master, -1);                     /* blocking recv  */
    pvm_bufinfo(buffer_id, &bytes, &msgtag, &tasktid);  /* info. about    */
                                                        /* received msg.  */

    switch(msgtag)
    { case TAG_POSITION_SLOW :                          /* new position   */
      case TAG_POSITION_FAST :                          /* new position   */

                        ...                             /* computing the  */
                        ...                             /* execution time */

                        pvm_upkint(&pos, 1, 1);
                        result_pos=pos;
                        for(k=0;k<PARTS_DIVISION;++k) /* the series     */
                        { s1=s2=s3=s4=0;
                          Serie(pos);
                          ss=4*s1-2*s2-s3-s4;
                          pi_decimal=ss-(int)ss+1.;
                          y=fabs(pi_decimal);
                          for(i=j;i<j+decitranch;i++)
                          { y=16*(y-(floor(y)));
                            pi_hexa[i]=deci_hexa[(int)floor(y)];
                          }
                          j+=decitranch;
                          pos+=decitranch;
```

```
                        }
                        j=0;
                        pvm_initsend(PvmDataDefault);

                        ...                         /* computing the  */
                        ...                         /* execution time */

                        pvm_pkdouble(&user_sys_time, 1, 1);
                        pvm_pkint(&result_pos, 1, 1);
                        pvm_pkbyte(&pi_hexa[0], PART_SIZE, 1);
                        pvm_send(master, (TAG_POSITION_SLOW==msgtag)?
                                 TAG_RESULT_SLOW:TAG_RESULT_FAST);
                        break;
      case TAG_KILL     : pvm_exit();
                        exit(0);
    }
  }
}
```

1. *Écrire un programme* `pi_master.c` *permettant de distribuer le calcul des hexadécimales de π. Le programme devra suivre un protocole de communication en accord avec le protocole correspondant au programme* `pi_slave`*. En particulier, le programme devra accepter des arguments selon deux formats :*

   - *nombre d'hexadécimales, nombre de tâches : afin de calculer les premières hexadécimales de π en utilisant un nombre donné de tâches ;*
   - *première hexadécimale, dernière hexadécimale, nombre de tâches : pour calculer les hexadécimales comprises entre deux bornes en utilisant un nombre fixé de tâches.*

   Le programme commencera par calculer les positions des débuts des tranches de `TAILLE_TRANCHE = 13` *hexadécimales à calculer, puis il lancera les tâches* `pi_slave` *et répartira les tranches à calculer entre ces différentes tâches. Le programme calculera les tranches d'hexadécimales en partant de la position la plus grande jusqu'à la position la plus petite.*

2. *Utiliser le programme* `pi_master` *pour calculer les 5000 premières hexadécimales de π en séquentiel.*

3. *Utiliser le programme* `pi_master` *à partir de* `XPVM` *pour calculer les 5000 premières hexadécimales de π en parallèle avec une, deux, trois, etc. tâches* `pi_slave` *lancées sur la machine parallèle virtuelle.*

4. *Intégrer au programme* `pi_master.c` *des instructions permettant de sauver sur disque les temps de calcul de chacune des tranches de l'intervalle demandé. On choisira un format de sortie compatible avec un logiciel de visualisation de données afin de pouvoir*

tracer des courbes par la suite (on pourra choisir un format brut et utiliser `gnuplot` *par exemple).*

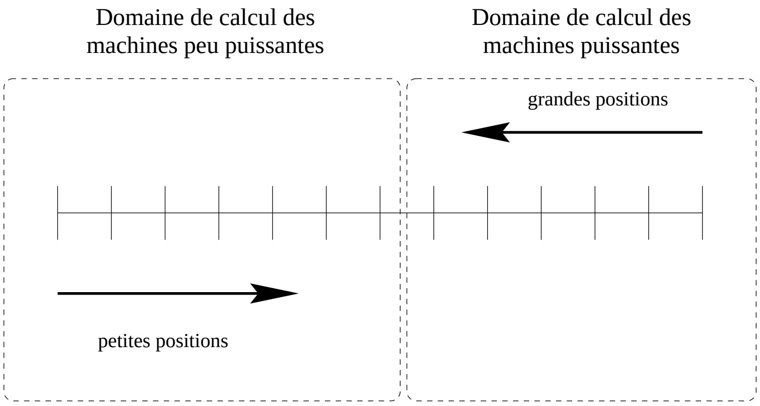
5. *Tracer la courbe donnant le temps de calcul séquentiel en fonction de la position p d'une tranche d'hexadécimales.*

6. *En déduire une estimation de la complexité en temps de l'algorithme en fonction de la position p d'une tranche d'hexadécimales.*

7. *Estimer le temps de calcul de la tranche d'hexadécimales commençant à la position 10^6 en séquentiel.*

8. *En déduire une estimation du temps de calcul des 10^6 premières hexadécimales de π sur la machine virtuelle.*

9. *Nous souhaitons désormais modifier le programmme* `pi_master` *afin qu'il distribue une seule tâche par machine. Par ailleurs, il serait judicieux de distribuer les calculs des premières tranches sur des machines peu puissantes, les machines plus efficaces se chargeant de calculer les tranches correspondant aux positions les plus grandes. Quel paramètre du fichier de configuration de la machine virtuelle permettrait de gérer cette notion de "puissance des machines" ?*



10. *Modifier le programme* `pi_master.c` *afin qu'il intègre les idées précédentes de distribution des calculs des tranches en fonction de la puissance des machines. En particulier, on positionnera le paramètre de la question précédente afin d'avoir autant de machines dites performantes que de machines moins efficaces.*

11. *Tracer en fonction de n les courbes des temps de calcul parallèle sur k machines des n premières hexadécimales.*

12. *Utiliser* `XPVM` *et* `PGPVM2` *pour visualiser et analyser le comportement du programme.*