Calcul distribué de π

(projet de parallélisme, maîtrise informatique 1995-96)

Jean-Yves Thibon, Sébastien Veigneau

Résumé

```
3.243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89452821E638D01377BE5
  466CF34E90C6CC0AC29B7C97C50DD3F84D5B5B54709179216D5D98979FB1BD1310BA698DFB5AC2FFD72
  DBD01ADFB7B8E1AFED6A267E96BA7C9045F12C7F9924A19947B3916CF70801F2E2858EFC16636920D87
  1574E69A458FEA3F4933D7E0D95748F728EB658718BCD5882154AEE7B54A41DC25A59B59C30D5392AF2
  6013C5D1B023286085F0CA417918B8DB38EF8E79DCB0603A180E6C9E0E8BB01E8A3ED71577C1BD314B2
  778AF2FDA55605C60E65525F3AA55AB945748986263E8144055CA396A2AAB10B6B4CC5C341141E8CEA1
  5486AF7C72E993B3EE1411636FBC2A2BA9C55D741831F6CE5C3E169B87931EAFD6BA336C24CF5C7A325
  381289586773B8F48986B4BB9AFC4BFE81B6628219361D809CCFB21A991487CAC605DEC8032EF845D5D
  E98575B1DC262302EB651B8823893E81D396ACC50F6D6FF383F442392E0B4482A484200469C8F04A9E1
  F9B5E21C66842F6E96C9A670C9C61ABD388F06A51A0D2D8542F68960FA728AB5133A36EEF0B6C137A3B
  E4BA3BF0507EFB2A98A1F1651D39AF017666CA593E82430E888CEE8619456F9FB47D84A5C33B8B5EBEE
 06F75D885C12073401A449F56C16AA64ED3AA62363F77061BFEDF72429B023D37D0D724D00A1248DB...
```


1 Introduction

Considérant un cercle de rayon 1, nous savons que son aire est $\pi = 3.14159265\ldots$, les points signifiant que les chiffres ne s'arrêtent pas mais continuent à l'infini.

Ces chiffres du développement décimal de π ont fasciné de nombreuses personnes depuis des siècles pourtant aucune régularité n'a été trouvé dans ces chiffres. Bien qu'il n'y ait pas vraiment d'application pratique à la connaissance de plusieurs milliards de chiffres après la virgule de π, les gens tentent quand même de les calculer pour le plaisir, ou pour le record...

Voici une manière très très lente de calculer π. Inscrivons un cercle de rayon 1 dans un carré de côté 2. Suspendons le total sur un mur et lançons complètement aléatoirement des fléchettes dans le carré. Supposons que sur les N fléchettes qui heurtent l'intérieur du carré, M d'entre elles se situent à l'intérieur du cercle. Les lois de la probabilité nous disent qu'il est raisonnable d'espérer que

$$\frac{M}{N} \quad \text{est approximativement égal à} \quad \frac{\pi}{4}\,,$$

car l'aire du cercle vaut π et celle du carré vaut 4.

Des années durant, les gens ont essayé de mettre au point des algorithmes pour calculer π, plus efficaces que l'algorithme décrit ci-dessus. Cela donna lieu à de nombreux résultats parfois mathématiquement très sophistiqués et souvent compliqués à programmer sur un ordinateur.

Maintenant, si nous changeons un peu la question pour désormais se demander quelle peut être la i-ième décimale de π plutôt que de chercher à les calculer toutes, le problème est tout autre. Jusqu'à présent, on croyait généralement que calculer la i-ième 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 viennent de calculer le 10 milliardième chiffre du développement hexadécimal de π en utilisant la formule :

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

Cette formule permet de calculer la i-ième "décimale" de π sans être obligé de calculer toutes les précédentes. Personne n'avait auparavant seulement conjecturé qu'un tel algorithme d'extraction de décimale pour π était possible. La formule (1) donne le développement de π en base 16 et il est trivial de convertir un tel développement en binaire ou en décimal. D'un autre côté, la question suivante reste évidemment en suspens: existe-t-il des polynômes $a(n)$ et $b(n)$ à coefficients entiers tels que :

$$\pi \quad = \quad \sum_{n=0}^{\infty} \frac{a(n)}{b(n)} \left(\frac{1}{10} \right)^n \; ?$$

2 Implémentation

Selon Simon 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 seraient linéaires par rapport à l'ordre de la décimale désirée, pour calculer le développement hexadécimal (voire décimal) de π. On peut donc espérer vérifier sans trop de difficulté que la 10 milliardième décimale de π est 9 sur une simple station de travail, en quelques jours. En fait, Bailey, Borwein et Plouffe ont utilisé leur formule dans un programme FORTRAN pour trouver cette 10 milliardième décimale de π sur un ordinateur très rapide de la NASA. Cela a pris seulement 96 heures pour réaliser ce travail qui constitue d'ores et déjà un nouveau record en la matière. Le point fort n'est pas ce nouveau record mais la méthode utilisée.

Nous nous intéressons *a priori* à de plus modestes résultats dans l'immédiat. L'objet de ce projet est de distribuer sur plusieurs machines le calcul des décimales de π à l'aide du logiciel `PVM`—faute de machine multi-processeurs accessible l'Université.

Puisque nous sommes désormais capables de calculer une décimale de π sans calculer les autres, nous pouvons distribuer le calcul de n décimales de π sur plusieurs machines, chacune calculant par exemple un intervalle précis. Si chaque machine est apte en un temps raisonnable à calculer 10000 décimales de π, nous devrions pouvoir sans peine calculer à l'aide de 20 machines, les premières 100000 décimales de π. Tout ceci paraît raisonnable dans la mesure où *Maple* est capable de sortir instantanément les 10000 premières décimales de π à l'aide de l'instruction :

```
> evalf(Pi, 10000);
```

En fait, pour la petite histoire *Maple* (depuis la version V) calcule π par la formule de Chudnovsky :

$$\frac{1}{\pi} = \frac{6541681608}{640320^{3/2}} \sum_{k=0}^{\infty} \Big(\frac{13591409}{545140134} + k \Big) \frac{(6k)!}{(3k)!(k!)^3} \frac{(-1)^k}{640320^{3k}} \,.$$

Nous nous intéressons plutôt à la nouvelle formule de Bailey, Borwein et Plouffe et proposons un programme *Maple* réalisant cet algorithme. L'objectif du projet s'inscrit plus dans le cadre d'une distribution intelligente du calcul de π que dans la traduction de la formule (1).

L'algorithme suivant permet de calculer une tranche de **12 chiffres consécutifs** après la virgule de π, à partir d'une position donnée.

Programme Maple de calcul des chiffres après la virgule du développement hexadécimal de π.

On positionne le nombre de chiffres après la virgule des décimaux en *Maple*.

```
Digits:=24;
```

`VersHex` est une fonction prenant en paramètre un réel `x`. Elle retourne la table des chiffres de `x` correspondant aux 12 premiers chiffres recherchés, en base 16. La fonction *Maple* `trunc` prend en paramètre un réel `x` et retourne le plus grand entier plus petit que `x`. La fonction *Maple* `frac` prend en paramètre un réel `x` et retourne sa partie fractionnaire, *i.e.* `x`− `trunc`(`x`).

```
VersHex:=proc(x)
  local
    y,       # valeur absolue de x...
    i,       # variable de boucle...
    pi_hexa; # table des 12 chiffres...
  y:=abs(x);
  for i from 1 to 12 do
    y:=16*frac(y);
    # on sauve le i-ieme chiffre (sa valeur hexadecimale)...
    # de trunc(y), i.e. 0, 1, ...., 9, A, ..., F.
    pi_hexa[i]:=convert(trunc(y), hex)
  od;
  # on retourne la table des chiffres...
  RETURN(eval(pi_hexa))
end;
```

Expm : le nom de cette fonction fait référence à un calcul d'exponentielle modulaire, c'est-à-dire dans notre cas de 16^p modulo ak. Deux cas se présentent (en *Maple*) à cause de la fonction spéciale Power de calcul intelligent de $16^p \bmod$ ak; ces deux cas sont ak $= 1$ et ak > 1. Dans tous les cas, l'enjeu est de calculer intelligemment $16^p \bmod$ ak, *i.e.* de ne pas calculer 16^p puis d'effectuer le modulo...

```
Expm:=proc(p, ak)
  if (ak<=1) then RETURN(0) else RETURN( Power(16, p) mod ak ) fi
end;
```

Serie calcule un petit morceau du développement de la somme.

```
Serie:=proc(m, pos)
  local
    ak,      # un module...
    epsilon, # un epsilon pour arreter le calcul...
    somme,   # le calcul de la somme partielle...
    t1,      # variable de calcul...
    t2,      # variable de calcul...
    k;       # variable de boucle...
  epsilon:=10^(-17);
  somme:=0.0;
  ak:=m;
  for k from 0 to pos-1 do
    t1:=Expm(pos-k, ak);
    somme:=frac(somme+t1/ak);
    ak:=ak+8
  od;
  t1:=1;
  for k from pos to pos+100 do
    t2:=1/(t1*ak);
    if (t2<epsilon) then
      break
    fi;
    somme:=frac(somme+t2);
    t1:=16*t1;
    ak:=ak+8
  od;
  RETURN(somme);
end;
```

Dig est la fonction de calcul d'un intervalle de 12 chiffres à partir la position pos. Par exemple, pour avoir les 12 premiers chiffres après la virgule en base 16 de π, il suffit de lancer `Dig(0);` . Pour avoir successivement, les 12 premiers puis les 12 suivants, lancer les deux commandes : `Dig(0); Dig(12);` . La limitation sur l'argument de la fonction Dig est la position pos $= 2^{24} = 16777216$

```
Dig:=proc(pos)
  local
    s1, s2, s3, s4, pi_decimal, pi_hexa, i;
  # calcul de pi en decimal...
  s1:=Serie(1, pos);
  s2:=Serie(4, pos);
  s3:=Serie(5, pos);
  s4:=Serie(6, pos);
  pi_decimal:=frac(evalf(4*s1-2*s2-s3-s4))+1;
  # conversion en hexadecimal...
  pi_hexa:=VersHex(pi_decimal, 12);
  # affichage du resultat : mise sous forme d'une chaine de caracteres.
  print(Pi[pos..pos+11] = cat(seq(pi_hexa[i], i=1..12)))
end;
```

Vous aurez donc à traduire cet algorithme en langage *C* en ne tenant pas compte des quelques instructions concernant la programmation en *Maple*. Par exemple, la première ligne `Digits:=24;` n'est là que pour fixer le nombre de chiffres après la virgule des décimaux, mais bien entendu, en *C* cette information n'a pas de sens puisque ce nombre de chiffres est fixe et de plus, machine-dépendant. Ensuite, vous aurez à distribuer astucieusement ce calcul en essayant le plus possible de ne pas calculer plusieurs fois les mêmes choses. La lisibilité du source et sa rapidité seront des facteurs importants : lisibilité parce que le source sera court, rapidité parce que la performance est importante dans ce contexte de calcul de π.

3 Divers

Le source du programme *Maple* ci-dessus est disponible sur simple demande par courrier électronique à `sv@univ-mlv.fr`. En ce qui concerne le logiciel PVM, voici quelques adresses ou *URL Internet* utiles :

1. Ftp, `ftp netlib2.cs.utk.edu`, `cd pvm3/book`, `get pvm-book.ps`
2. Email, `netlib@ornl.gov`, message : `send pvm-book.ps from pvm3/book`
3. Http, `http: //www.epm.ornl.gov/pvm/pvm_home.html`
4. Http, `http: //www.netlib.org/pvm3/book/pvm-book.html`