Le crible de la roue en distribué

Gabriel Paillard, Christian Lavault

Le crible de la roue en distribué

Gabriel Paillard* Christian Lavault
LIPN UMR CNRS 7030 – Université Paris 13
99, Av. Jean-Baptiste Clément 93430 - Villetaneuse, France
{gap, lavault}@lipn.univ-paris13.fr

Résumé : Cet article aborde la génération des nombres premiers inférieurs à *N* en utilisant le crible de la roue de façon distribuée [Pritchard, 1982]. Les algorithmes de crible peuvent être un moyen très efficace de déterminer la primarité des entiers appartenant à un intervalle fini donné *[1..N]*, si celui-ci est assez grand et que le test de primarité s'effectue sur tous les nombres de l'intervalle [Crandall, 2001]. Cette distribution du crible de la roue est la première realisée. Pour l'introduire, nous expliquons le crible classique d'Eratosthène et nous montrons une implémentation que nous avons realisé en séquentiel de celui-ci, puis une implémentation en distribué. L'objectif primordial de la parallélisation de ce genre d'algorithme est, soit de repousser les limites de la génération des nombres premiers, soit d'atteindre les limites desirées dans un délai de temps inférieur. Nous montrons les résultats expérimentaux des algorithmes du crible d'Eratosthène et du crible de la roue, aussi bien en séquentiel qu'en distribué.

Mots Clés : Crible, roues, systèmes distribués.

1. INTRODUCTION

Les nombres premiers se caractérisent par une propriété simple, qui est leur divisibilité par deux nombres, à savoir eux mêmes et *1*. Comme conséquence, nous avons deux moyens d'obtenir les nombres premiers : soit en ajoutant unité par unité et testant si le nouveau nombre engendré est premier ou non [1], soit en engendrant aléatoirement un nombre pour procéder ensuite à un test de primarité. En decouvrant des nouveaux nombres premiers, nous pouvons engendrer encore de nouveaux nombres, de façon plus rapide et directe. Pour cela, au lieu de l'addition (unité par unité), nous utilisons la multiplication, qui ne garantit qu'un ordre partiel de l'ensemble des entiers [Tenenbaum, 1995]. Les nombres que nous ne pouvons pas obtenir en utilisant la multiplication sont des nombres premiers. On le remarque par le théorème fondamental de l'arithmétique, qui établit que chaque nombre peut être decomposé en une suite ordonnée unique de nombres premiers [Delahaye, 2000]. Le premier algorithme de génération de nombres premiers, fut élaboré par Eratosthène. De nos jours cette application des nombres premiers peut-être employée pour faire des mesures de performance d'une nouvelle machine parallèle, puisque cet algorithme nécessite un usage intensif de la mémoire principale et du processeur [Bokhari, 1987].

Les nombreuses applications des nombres premiers à la cryptologie [Singh, 1998 ;Coutinho, 1997] constituent un autre champ d'investigation . Dans ce domaine, l'utilisation des nombres premiers permet d'engendrer une clef publique qui est en fait le produit de deux grands nombres premiers. Cette caractéristique est le point fort de ce système de cryptographie, puisque, pour pouvoir accéder au message protégé, il faut obtenir la clef privée par factorisation de la clef publique (ses deux facteurs premiers).

Dans cet article nous verrons comment obtenir des réponses plus rapides en distribuant le problème. Nous distribuons d'abord le crible simple d' Eratosthène d'une manière naturelle et, ensuite grâce au crible de la roue. Pour finir nous montrons les résultats expérimentaux obtenus.

2. LE CRIBLE D'ÉRATOSTHENE

Cet algorithme a été créé il y a plus de 2000 ans par Eratosthène de Cyrène. Il consiste à éliminer tous les nombres premiers de l'intervalle *[2..N]*, où *N* est majorant donné des nombres premiers à engendrer. L'algorithme simple d'Eratosthène nécessite ainsi un nombre de pas de calculs de l'ordre de *NloglogN* (additions uniquement) [Pritchard, 1982 ;Bokhari, 1987]. (Noter que la complexité en espace mémoire de ces cribles est très grande.)

.1 Algorithme Séquentiel

La version séquentielle de cet algorithme consiste tout d'abord à engendrer tous les nombres entre *2* et *N*, et allouer en mémoire un vecteur ou une liste doublement chainée. Ensuite on élimine tous les entiers *n* possédant un diviseur en dehors de 1 et *n*. Le crible

[1] Ce test peut être effectué en utilisant un algorithme de divisions par tentatives, qui tente d'obtenir les diviseurs de ce nombre.

* L'auteur est en parti subventionné par le gouvernement brésilien, sous le contrat : 1522/00-0, Capes, Ministére de l'éducation, Brésil.

d'Eratosthène effectue cette tâche en prenant le premier nombre de la liste, qui est *2*, et en cherchant ses multiples sur toute la liste. Ensuite l'algorithme procède à une élimination élément par élément. Le prochain nombre supérieur à *2* sera le prochain nombre premier, et nous procédons de la même façon jusqu'à ce que le prochain nombre premier soit inférieur ou égal à $\sqrt{N}$.

Ce crible est d'une extrême simplicité et est très employé à titre pédagogique [Djikstra, 1972 ; Misra, 1981 ; Cosnard, 1989]. Il n'est cependant pas linéaire en temps d'exécution, puisque certains nombres peuvent être éliminés plusieurs fois. D'autres algorithmes ont été dévelopés pour obtenir une complexité en temps inférieure au crible d'Eratosthène [Pritchard, 1981 ; Misra 1978].

Dans le tableau 1 nous pouvons observer les mesures de temps obtenus avec notre implémentation.

N	Temps (secondes)
10000	0.00
50000	0.02
100000	0.05
200000	0.12
300000	0.22
400000	0.34
500000	0.45
600000	0.55
700000	0.70
800000	0.82
900000	0.94
1000000	1.08

Tab. 1 – Temps déxécution du crible d'Eratosthène en séquentiel en fonction de *N*

.2 Algorithme distribué

Nous utilisons *9* machines équivalentes en puissance de calcul et mémoire, pour distribuer le crible d'Eratosthène. L'environnement utilisé pour la distribution du programme en question est le *LAM-MPI 6.5.9* ("*Local Area Multicomputer Message-Passing Interface*"), qui suit le paradigme d'une communication par passage de messages, et où les processus sont symétriques [Lavault, 1995]. Le schéma de distribution consiste à considérer un processeur maître et *N-1* processeurs esclaves. Le maître a pour fonction de passer au divers processeurs esclaves leur intervalle de calcul (qui définit la suite des nombres sur lesquels le processeur exécutera l'algorithme) et le nombre premier actuel, et lui même aussi possède son intervalle de calcul, et de lui dépend la terminaison de l'algorithme. Dans le tableau 2 nous pouvons observer les mesures de temps obtenues avec cet algorithme d'Eratosthène distribué.

N	Temps (secondes)
10000	0.003
50000	0.010
100000	0.011
200000	0.017
300000	0.026
400000	0.037
500000	0.048
600000	0.061
700000	0.073
800000	0.087
900000	0.102
1000000	0.116

Tab. 2 – Temps déxécution du crible d'Eratosthène en distribué en fonction de *N*

La figure 1 montre les courbes des mesures de temps des deux algorithmes, séquentiel et distribué, du crible d'Eratosthène. On peut constater que la courbe de l'algorithme distribué, est bien meilleure que celle de l'algorithme séquentiel.

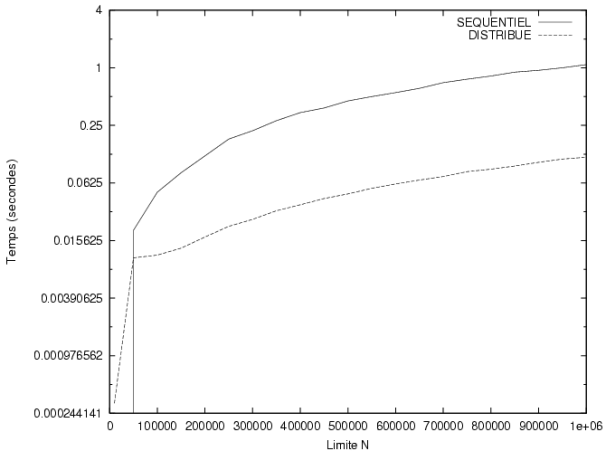

Fig. 1 – Performances respectives des algorithmes séquentiel et distribué du crible d'Eratosthène simple

3. LES ROUES

Afin d'améliorer le temps d'exécution d'un programme de crible, il serait intéressant d'essayer de réduire le nombre d'éléments engendrés plusieurs fois au cours de la génération des nombres premiers. Pour cela il est possible d'utiliser des concepts mathématiques (appartenant à la théorie des nombres) comme les roues.

Les roues sont un ensemble d'éléments, déjà criblé jusqu'à un certain point. Les éléments appartenant à une certaine roue sont dénommer quasi-premiers[2]. La définition des roues est la suivante :

$$W_k = R(\prod_k)$$

où $\prod_k$ est le produit des *k* premiers nombres premiers et la fonction $\Re$ retourne tous les membres entre 1 et $\prod_k$ qui sont premiers entre eux (leur plus grand commun diviseur est 1). En étendant cette définition (qui apllique cette définition de $\Re$ à un nombre $>\prod_k$), les roues représentent un motif qui se répète modulo $\prod_k$. Cela signifie que nous pouvons, en utilisant une roue de taille quelconque, engendrer tous les nombres quasi-premiers

[2] Ce sont des nombres qui ne sont pas multiples des nombres premiers précédemment engendrés.

jusqu'à une limite donnée. Nous pouvons donc estimer quelle roue nous devons engendrer. Mais le plus simple est de construire une roue à partir des roues précédentes, c'est-à-dire par récurrence. C'est l'objet de la définition suivante [Pritchard, 1982] :

$$W(\prod_k . p_{k+1}) = \{\prod_k . y_1 + p_{k+1} . y_2 \bmod \prod_k . p_{k+1} \mid y_1 \in R(p_{k+1}), y_2 \in W(\prod_k)\}$$

Son désavantage est que les éléments engendrés précédemment pour $W(\prod_k)$ sont à nouveau engendré pour $W(\prod_k . p_{k+1})$. Le théorème qui suit, permet la génération des éléments de W_{k+1} sans celles des éléments appartenant à la roue W_k :

$$W_{k+1} = W_k \cup \{a.\prod_k + b \mid a \in \{1 \ldots p_{k+1} - 1\}, b \in W_k\}$$

On peut donc construire la roue W_{k+1} à partir de la roue W_k sans répéter d'éléments en dehors de W_{k+1}.

4. LE CRIBLE DE LA ROUE

Le crible de la roue consiste à soustraire de l'ensemble final W_{k+1}, défini dans le dernier théorème de la section précédente, les multiples de p_{k+1}, c'est-à-dire l'ensemble :

$$\{p_{k+1} . b \mid b \in W_k\}$$

En utilisant ce théorème, nous pouvons implémenter l'algorithme de la roue avec une preuve formelle de sa terminaison [Pritchard, 1982]. Nous présentons la troisième roue sur la figure 2. Sur la figure 3 on peut voir la construction de la quatrième roue à partir de la troisième.

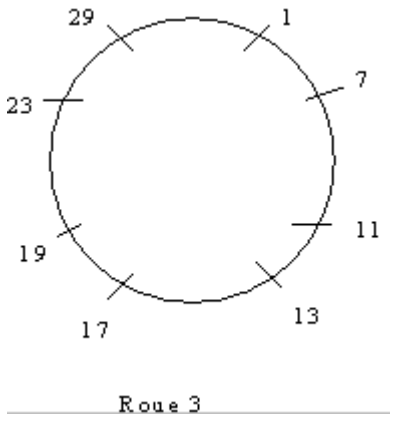

Fig. 2 – Troisième roue

La prochaine roue (W_{k+1}) aura comme périmètre le produit de p_{k+1} par $\prod_k$. L'approche géométrique (pour visualiser la génération des roues) facilite beaucoup la compréhension de cet algorithme.
Si nous roulons cette roue jusqu'à une limite donnée, nous obtiendrons une répétition de motifs sur cet intervalle ($\prod_{k+1}$). La figure 4 nous montre le roulement de la roue 3 pour un certain intervalle qui appartient à la roue 4.

Mais, pour obtenir les nombres premiers dans l'intervalle *[1..N]*, nous sommes contraints, après avoir engendré la roue W_k, d'éliminer tous les nombres quasi-premiers.

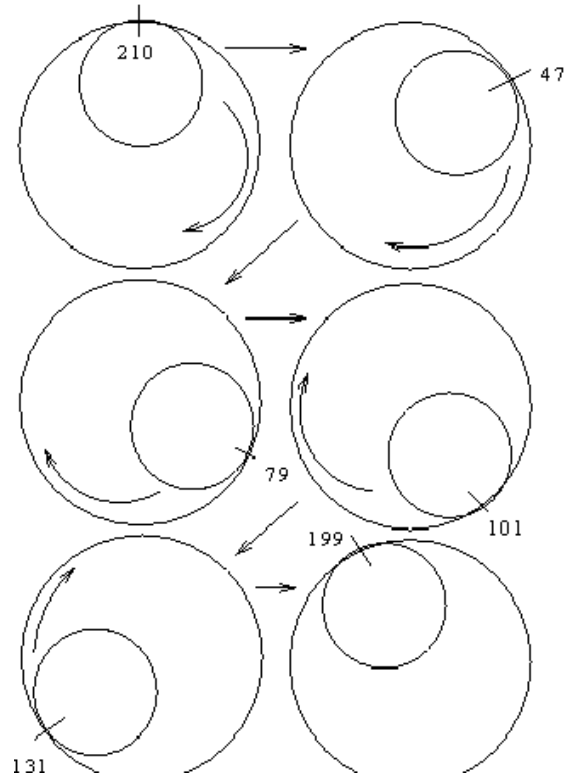

Fig.3 – Génération de la quatrième roue

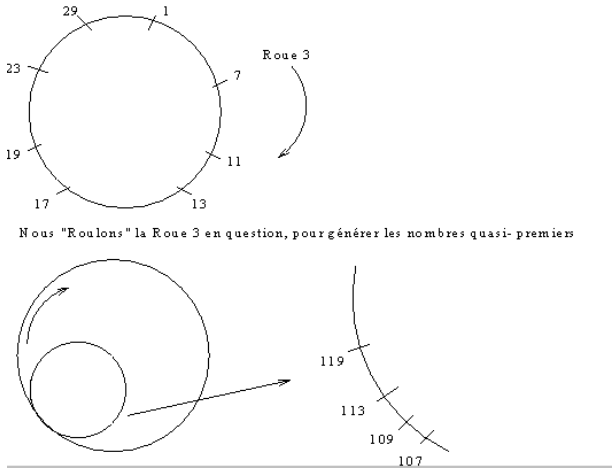

Fig.4 – Application du motif de la roue 3

En utilisant l'algorithme introduit par *Paul Pritchard* [Pritchard, 1982], nous avons implémenté de façon

séquentielle cet algorithme. Les mesures de temps obtenues, sont données dans le tableau 3.

N	Temps (secondes)
10000	0.01
50000	0.25
100000	1.37
200000	9.86
300000	25.60
400000	47.82
500000	76.90
600000	111.57
700000	152.94
800000	200.33
900000	254.81
1000000	313.69

Tab. 3 – Temps déxécution du crible de la roue en séquentiel en fonction de *N*

Si après avoir engendré la roue W_k , on ne peut plus construire une autre roue à cause de la limite *N*, alors on continue l'éxécution de l'algorithme en utilisant seulement le crible de la roue. Lorsque le prochain nombre premier p_{k+1} sera supérieur ou égal à *N*, l'algorithme s'arrêtera. Comme contrainte additionnelle il y a le fait que, pour un nombre comme 35, l'algorithme doit procéder au crible en dehors de la boucle principale, puisque la racine carrée du nombre p_{k+1} est plus grande que *N* alors que 35 n'est pas premier.

En séquentiel, nous observons une grande différence entre les mesures de temps de l'algorithme du crible d'Eratosthène et celles de l'algorithme de la roue. Cela est dû au grand nombre de contraintes à gérer, aussi bien celles liées a la génération de W_{k+1} à partir de W_k (même si cela se fait par des simples additions), que celles inhérentes à la génération des multiples de p_{k+1} à éliminer. On doit ainsi gérer deux listes qui peuvent être de grande tailes, celle des nombres appartenant à la roue W_{k+1} et celle des nombres qui sont multiples de p_{k+1}.

5. DISTRIBUTION DE LA ROUE

Le crible distribué de la roue est moins naturel que le crible d'Eratosthène, même si l'objectif commun est de distribuer de façon équitable les éléments à engendrer. Dans le cas du crible de la roue, nous devons suivre la logique introduite dans l'algorithme original (séquentiel) du crible de la roue, où nous avons un seul processeur pour gérer une structure de donnée ordonnée. Cependant, lorsque nous distribuons le crible de la roue, nous pouvons facilement remarquer que chaque noeud (processeur) possède sa propre liste ordonnée, mais qu'un ordonnancement global de tout les éléments doit être maintenu. Cela s'impose, par exemple, au moment de connaître le prochain nombre premier (p_{k+1}). Comme pour le crible d'Eratosthène, nous utilisons un maître et des esclaves qui coopèrent d'une façon coordonnée, comme nous pouvons le constater à la figure 5 (topologie en forme d'étoile). Le maître a pour objectif de trouver les nombres p_{k+1}, $\prod_{k+1}$ et de les distribuer entre les esclaves. À chaque itération de l'algorithme, le maître reçoit le prochain p_{k+1}, de l'un de ses esclaves connu auparavant.

Les esclaves engendrent les nombres quasi-premiers en sommant à leurs nombres (ceux qui ont déjà été auparavant engendré), la valeur de $\prod_k$ reçue du processeur maître. Une des difficultées d'implémentation de cet algorithme est le fait que nous devons savoir sur quel processeur se trouve le prochain nombre premier et, comme sur chaque processeur nous avons plusieurs éléments, il est difficile de déterminer un ordre croissant entre les nombres des divers processeurs esclaves. Ce nouveau nombre premier p_{k+1} sera utilisé pour engendrer le prochain $\prod_k$ et aussi les prochains multiples de p_{k+1}. Ce contrôle exercé par le maître nécessite un grand nombre de messages. Mais le temps d'attente des processeurs est quasi inexistant car les processeurs esclaves sont souvent en train d'éliminer les multiples de p_{k+1} , ou en train d'engendrer les nouveaux nombres quasi-premiers de la roue W_{k+1}.

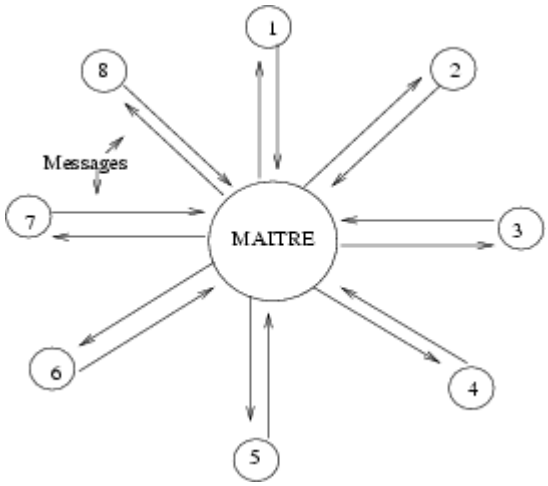

Fig. 5 – Distribution des processeurs (Topologie)

Simultanément le maître poursuit la création des multiples de p_{k+1}, de cette façon le temps inactif des processeurs est faible.

La terminaison de l'algorithme par le maître ce fait en calculant un entier maximal à partir de p_{k+1}. Si celui-ci dépasse $\sqrt{N}$, alors l'algorithme se termine.

Dans le tableau 4 nous pouvons observer les mesures de temps obtenues avec notre implémentation en distribué. Le schéma de distribution est le même, soit 9 machines équivalentes en puissance de calcul et mémoire. La topologie entre le maître et les esclaves est celle d'une étoile, où nous avons des communications par messages entre les esclaves et le processeur maître, mais pas entre esclaves.

N	Temps (secondes)
10000	2.49
50000	7.36

100000	9.61
200000	17.96
300000	25.30
400000	31.86
500000	35.35
600000	51.88
700000	58.90
800000	66.68
900000	90.72
1000000	125.12

Tab. 4 – Temps déxécution du crible de la roue en distribué en fonction de *N*

La figure 6 donne les courbes de performance, des algorithmes séquentiel et distribué, du crible de la roue. On constate que la courbe de l'algorithme distribué a un facteur d'accélération[3] de l'ordre de 2 lorsque *N* est suffisament grand [Singh, 1999]. Donc, en dépit du grand nombre de messages échangés entre le processeur maître et ses esclaves, l'algorithme distribué du crible de la roue est plus performant que l'algorithme séquentiel.

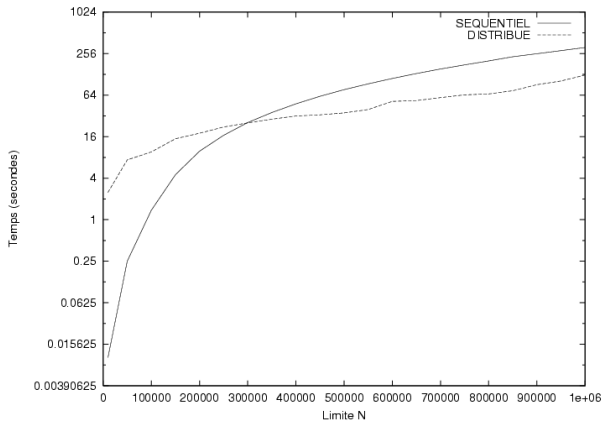

Fig. 6 – Performances des algorithmes séquentiel et distribué du crible de la roue en fonction de *N*

6. CONCLUSION

Pour résumer, nous pouvons dire que l'algorithme distribué du crible de la roue est un succès, dans la mesure où il permet d'obtenir de meilleurs résultats que la version séquentielle. Nous espérons pouvoir améliorer les performances de l'algorithme distribué du crible de la roue en diminuant le nombre de messages échangés, par exemple. Par ailleurs, une étude théorique sur la complexité de l'algorithme du crible de la roue est en cours, afin de mieux comprendre son comportement asymptotique.

[3] Le temps du calcul séquentiel divisé par le temps d'exécution en distribué

BIBLIOGRAPHIE

[Pritchard, 1982] Pritchard, P. : "Explaining the wheel sieve". Acta Informatica, Vol. 17, p. 477-485 (1982).

[Crandall, 2001] Crandall, R., Promerance, C. : "Prime numbers". Springer-Verlag (2001).

[Tenenbaum, 1995] Tenenbaum, G. : "Introduction to analytic and probabilistic number theory". Cambridge University Press (1995).

[Delahaye, 2000] Delahay, J.-P. : "Merveilleux nombres premiers". Éditions pour la science/Belin, Paris (2000).

[Bokhari, 1987] Bokhari, S.H. : "Multiprocessing the sieve of Eratosthenes". IEEE Computer, Vol. 20, N°4, p. 50-58 (1987).

[Singh, 1998] Singh, S. : "Le dernier théorème de Fermat". Éditions Jean-Claude Lattès, p. 118 (1998).

[Coutinho, 1997] Coutinho, S.C. : "Números Inteiros e Criptografia RSA". IMPA/SBM, Rio de Janeiro (1997).

[Djikstra, 1972] Djikstra, E.W. : "Notes on structured programming". In "Structured programming", Academic Press, p. 1-82, New York (1972).

[Misra, 1981] Misra, J. : "An exercise in program explanation". ACM Trans. Program. Lang. Syst., Vol. 3, , p. 104-109 (1981).

[Cosnard, 1989] Cosnard, M., Philippe, J.-L.: "Génération de nombres premiers en parallèle". La lettre du transputer, Janvier (1989).

[Lavault, 1995] Lavault, C. : "Evaluation des algorithmes distribués". Hermès, Paris (1995).

[Misra, 1978] Misra, J., Gries, D. : "A linear sieve algorithm for finding prime numbers". Comm. ACM, Vol. 21, p. 999-1003 (1978).

[Pritchard, 1981] Pritchard, P. : "A sublinear additive sieve for finding prime numbers". Comm. ACM, Vol. 24, p. 18-23 (1981).

[Singh, 1999] Singh, J. P., Culler, D. E.: "Parallel Computer Architecture". Morgan Kaufmann publishers (1999).