Título
MAC0438 - Programação Concorrente

André Meneghelli Vale - 4898948
andredalton@gmail.com

Marcello Souza de Oliveira - 6432692
mcellor210@gmail.com

1 Cálculo por série infinita de π

1.1 Fórmula:

A fórmula escolhida foi a *Fórmula de Bellard*[1]. Esta escolha se deve ao fato de poder ser calculado um enésimo dígito de pi na base 2 sem a necessidade de se calcular qualquer um dos termos anteriores, o que torna o programa possível de ser totalmente paralelizável. Esta fórmula foi desenvolvida a partir da fórmula *Bailey–Borwein–Plouffe*[2] (`BBP`) que por sua vez foi inspirada na série infinita para o cálculo de uma aproximação da inversa da arcotangente.

$$\pi = \frac{1}{2^6}\sum_{n=0}^{\infty}\frac{(-1)^n}{2^{10n}}\left(-\frac{2^5}{4n+1} - \frac{1}{4n+3} + \frac{2^8}{10n+1} - \frac{2^6}{10n+3} - \frac{2^2}{10n+5} - \frac{2^2}{10n+7} + \frac{1}{10n+9}\right) \tag{1}$$

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

1.2 Desafios:

Alguns problemas fizeram com que o algorítimo, embora com conversão incrível, não tenha cumprido a sua função. Como a fórmula, diferente da descrição nos relatórios que encontramos retorna o n-ésimo termo de pi sem a necessidade de cálculo dos termos anteriores e não o n-ésimo algarismo.

Como este algorítimo gera uma somatória de números racionais, que por muitas vezes podem ser dízimas, fica muito dificil trabalhar com a precisão de um double em linguagem C muito menos usar o vetor de armazenamento de inteiros que foi proposto na descrição inicial do trabalho.

Embora o armazenamento da informação calculada seja perdido na segunda dezena de precisão é possível calcular os termos até que eles atinjam o valor máximo de precisão de uma variável double. No entanto isso ocorre no termo 107, e o algoritmo termina no termo 108.

Devido a isso não foi possível fazer a medição de tempo adequada para rodar em diversas maquinas (o método desenvolvido para a medição no primeiro ep é muito pouco preciso para este caso). Foi testada em um Acer Aspire com 8Gb de RAM e processador AMS Phenom II X4 Mobile (2GHz). E os resultados obtidos, calculando até o termo de maior precisão possível para uma variável long double foi de:

- `SEQUENCIAL`: 5155175 ns
- `NORMAL`: 4238274 ns
- `UNLIMITED`: 1157542 ns

1.3 Resultados:

Foi possível observar que a velocidade de conversão tem uma boa melhora na versão concorrente, no entanto a versão sem sincronização a conversão ocorre de maneira muito mais veloz. A idéia original de salvar as casas decimais assim como o algoritmo promete não é viável, já que na segunda dezena de casas decimais a precisão já é perdida. E os valores dos termos, embora calculados corretamente, tenham suas dízimas arredondadas e faça com que o cálculo fique incorreto.

Acredito que os resultados estão de acordo com o esperado. E o desempenho do algoritmo é surpreendente. Devido a não ter completado os métodos de tomada de tempo em nanossegundos ficou impossível fazer os testes em mais máquinas (já que seriam insuficientes até então) e também fazer o gráfico de desempenho adequado.

Referências

[1] `http://en.wikipedia.org/wiki/Pi`

[2] `http://en.wikipedia.org/wiki/Bellard%27s_formula`

[3] `http://en.wikipedia.org/wiki/Bailey-Borwein-Plouffe_formula`