Paralelização do Crivo de Erastótenes com OpenMP

Modificações no código

As modificações feitas no código foram muito triviais, bastando utilizar o OpenMP para paralelizar o loop for que marca os primos no crivo. Como o loop que calcula o proximo valor primo não possui um número determinado de iterações e tem objetivo inerentemente sequencial (encontrar o primeiro valor que é primo), ele não foi paralelizado.

Procedimento Crivo Paralelizado

```
void Crivo(int *primos, long indMax) {
  long sqrtMax;
  long indBase;
  long base;
  long i;

  // raiz quadrada do ultimo inteiro eh o maior fator primo

  sqrtMax = (long) sqrt((long) ind2num(indMax));

  // primeira base

  indBase=0;
  base=3;

  // para todas as bases

  do {

    // marca como nao primo os multiplos da base a partir do seu quadrado

    #pragma omp parallel for
    for (i=num2ind(base*base); i<=indMax; i+=base) {
      primos[i]=0;
    }

    // avanca a base para o proximo primo

    for (indBase=indBase+1; indBase<=indMax; indBase++)
      if (primos[indBase]) {
        base = ind2num(indBase);
```

```
        break;
      }

    // enquanto base nao superar o final de primos e
    //               nao superar a raiz quadrada do ultimo inteiro

  }
  while (indBase <= indMax && base <= sqrtMax);
}
```

Tempos de execução e Speed-ups sem otimizações

Threads	Tempo (s)	Speed-Up
1	56.309921	-
2	27.88221	2.0195644821554675
4	21.747289	2.58928462301669
8	11.184777	5.0345144118653415
12	7.734047	7.2807833983941395
16	6.130707	9.1848984138371
20	5.162663	10.907146370003233
24	4.497375	12.520619472470052

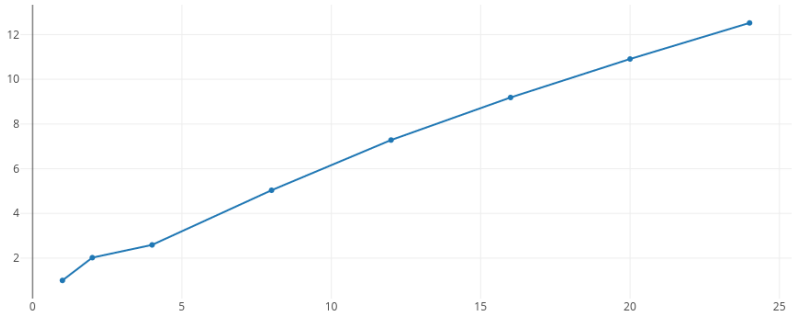

Figure 1: Speed-up para os diferentes numeros de threads sem flags de otimização

Análise

O crescimento do speedup com o numero de threads é aproximadamente linear (especificamente sendo a metade do numero de threads usadas).

Tempos de execução e Speed-ups com otimizações

Threads	Tempo (s)	Speed-Up
1	10.105286	-
2	5.233001	1.931068998458055
4	3.737081	2.704058595465284
8	2.812288	3.5932614298393335
12	2.669879	3.784922837327085
16	2.627346	3.8461953621639475
20	2.60681	3.8764950264883136
24	2.500464	4.041364322781692

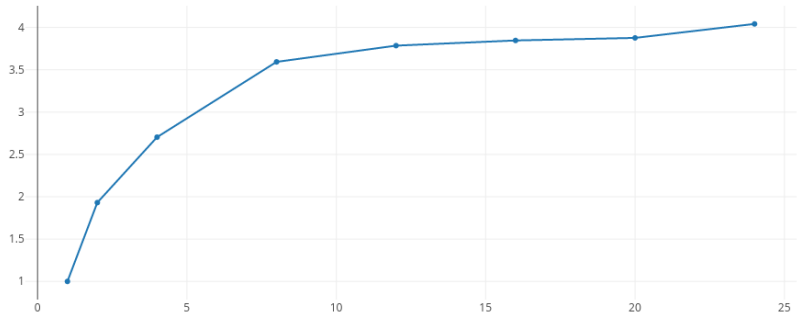

Figure 2: Speed-up para os diferentes numeros de threads com a flag de otimização -O3

Análise

Comparado com o speed-up sem a flag de otimização, aumentar o numero de threads possui um efeito menor, variando muito pouco a partir de 8 threads.

Conclusões

Percebe-se que em alguns casos, as otimizações mais agressivas (-o3) podem causar speedups por si só maiores que o uso de multiplas threads. Essas otimizações em geral focam em diminuir o numero de instruções e utilizar a memória mais eficientemente, mostrando que para tornar o paralelismo ainda melhor, seria possível modificar o algoritmo de crivo para se utilizar do cache de forma mais inteligente.