Aprendendo a programar com memória compartilhada

EP 1 – Entrega 30/09/2012

Computação Paralela e Distribuída – MAC5742/MAC0431

Segundo Semestre de 2012

Prof. Alfredo Goldman (*gold at ime.usp.br*)

1. Introdução

Nesse exercício programa vamos explorar a computação paralela com memória compartilhada, para isso serão usados dois padrões principais: OpenMP e Pthreads. Este primeiro EP é bastante simples, mas é composto por três fases distintas.

Em uma primeira fase, o desafio é usar a concisão da sintaxe do OpenMP, mostrando que com poucos caracteres um programa já pode ser paralelizado, ainda mais se a estrutura do código ajudar. Na segunda fase, o desafio será ir além da simples paralelização, de forma a evitar problemas clássicos no uso de memória compartilhada, como falso compartilhamento de cache e correto uso dos *cores* disponíveis. Finalmente, em uma terceira fase, vamos explorar as diferenças entre OpenMP e Pthreads, verificando as possibilidades de obtenção de desempenho.

O trabalho **pode** e **deve** ser feito por grupos de uma a duas pessoas. A entrega deve ser feita pelo Paca, através de um arquivo **ZIP** ou **TAR.GZ** com o **nome dos componentes**. Exemplos:

- `MarianaAlfredo.zip`
- `MarianaBravoAlfredoGoldman.tar.gz`

Nada de iniciais, por favor! Esse arquivo deve conter o **programa desenvolvido** (se for um projeto exportado do Eclipse, ajuda) e mais um **relatório** explicando detalhes sobre utilização e implementação do programa. Note que desta vez, como recebemos vários EPs sem nome será descontada nota se o padrão não for seguido.

As dúvidas devem ser resolvidas através do fórum da disciplina no Paca.

2. Fase 1

Nesta primeira etapa, o seu objetivo é paralelizar o programa abaixo usando o menor número de caracteres para criar uma versão paralela (OpenMP) do programa abaixo.

```
static long num_steps = 1000000;
double step;
void main (){
  int i; double x, pi, sum = 0.0;
  step = 1.0/(double) num_steps;
  for (i=0;i< num_steps; i++){
    x = (i+0.5)*step;
    sum = sum + 4.0/(1.0+x*x);
```

```
  }
  pi = step * sum;
  printf("Pi = %g\n", pi);
}
```

3.Fase 2

Para a segunda fase, será necessário otimizar o código de forma a obter bons desempenhos em OpenMP, é importante documentar cada recurso usado. Para a segunda e terceira fase, num_steps deve ser passado como parâmetro em linha de comando.

4. Fase 3

Finalmente, o mesmo algoritmo deverá se implementado usando a linguagem C e Pthreads, de forma a também obter o melhor desempenho possível.

5. Critérios de correção

Deverão ser entregues, além dos códigos relativos às fases, um relatório comparando as facilidades e desempenhos das versões das fases 2 e 3. Os resultados dos programas (variando o número de passos) serão comparados com versões corretas do código.

Para obter o prêmio (média 10), é necessário que um EP ganhe ao menos duas fases do EP, em ao menos duas máquinas diferentes (a serem divulgadas). Para ganhar a primeira fase, é necessário que além de ter o número mínimo de caracteres, o programa tenha um desempenho no mínimo entre os cinco melhores da mesma categoria (fase 1).