Computação Paralela 2018/19

Aula de laboratório – OpenMP – 12/11/2018

Material base e instruções

Para resolver os exercícios desta folha, comece por descarregar `openmp_lab2.zip` disponível na página da disciplina. Depois de descomprimir o ficheiro, poderá verificar a existência de subdirectórios `01` a `03`, identificando o número do exercício. Pode usar as várias `Makefiles` em conjunto com os habituais comandos `make` ou `make all` para compilar, e `make clean` para limpar os ficheiros gerados.

Em todos os exercícios está em causa a paralelização de código sequencial usando OpenMP. Recomenda-se que proceda da seguinte forma para cada exemplo `X.c` nos vários directórios:

— Copie o ficheiro `X.c` para `X-par.c`.

— Edite a `Makefile` do directório para habilitar a geração do ficheiro `X-par.bin`. Por exemplo em `01` altere `PROGRAMS=mm.bin # mm-par.bin` para `PROGRAMS=mm.bin mm-par.bin`

— Execute `make all` para gerar ambos os binários.

— Á medida que for editando `X-par.c` para paralelisar o código pode ir comparando o output de `X.bin` (programa sequencial original) com `X-par.bin` (programa paralelo). **Nota: se os outputs forem os mesmos isso não quererá dizer necessariamente que o código esteja correcto. Experimente executar o programa paralelo várias vezes e aumentar os tamanhos dos vectores / números de threads envolvidos para uma validação mais precisa.**

Exercícios

1. Paralelise em `mm-par.c` o algoritmo de multiplicação tradicional para matrizes de valores inteiros.

2. Paralelise em `min_max_sum-par.c` o algoritmo dado para encontrar o mínimo, máximo e soma dos elementos numa matriz de inteiros.

3. Em `code1-par.c`, `code2-par.c` e `code3-par.c` está em causa a paralelisação de ciclos com dependências "loop-carried".

- Comece por identificar precisamente as dependências de dados no programa, o seu tipo (de fluxo, anti-dependência, ou de output), e quais são "loop-carried" ou não.
- Modifique de seguida o programa para conseguir paralelisar o código da forma mais eficiente possível.