Optimização 3dfluid: OpenMP

Gonçalo Brandão
Departamento Engenharia Informática
Universidade do Minho
Braga, Portugal
pg57874@alunos.uminho.pt

Henrique Pereira
Departamento Engenharia Informática
Universidade do Minho
Braga, Portugal
pg57876@alunos.uminho.pt

Maya Gomes
Departamento Engenharia Informática
Universidade do Minho
Braga, Portugal
pg57891@alunos.uminho.pt

I. Intrução

Analisando o novo *solver* cedido pela equipa docente, percebemos que as dependências dos elementos pares e dos elementos ímpares foram removidas. Também foi adicionado um ciclo `while` de modo a parar a função `lin_solve` quando esta converge mais cedo do que as 20 iterações. Nesta fase, o tamanho do `size` foi aumentado de 42 para 84.

II. Análise

A. Identify

No início da análise do novo solver, voltamos a fazer um *profiling* do mesmo. Analisando o nosso novo *call graph*, percebemos que a `lin_solve` representava 42,65% do tempo de execução, sendo o nosso *hotspot* e o ponto onde decidimos começar a implementação de diretivas de paralelismo no nosso código.

B. Analyse

Analisando o novo solver, percebemos que as operações no ciclo mais interno eram agora independentes entre iterações, também alteramos a ordem dos ciclos de forma a aumentar a localidade espacial. De seguida decidimos analisar possíveis *data races*, sendo que estas estariam presentes no *max_c*. Com esta análise percebemos que clausulas de *Data Sharing* teriam de ser implementadas, para as variáveis `old_x` e *change* que iram ser inicializadas antes das possíveis zonas em paralelo.

Decidimos implementar duas zonas paralelas, a zona "vermelha" e a zona "preta". Estas duas zonas são muito semelhantes, sendo que uma calcula as iterações pares e outra as ímpares. Esta duas zonas possuem 3 ciclos *for's* cada uma, sendo o seu trabalho de tamanho semelhante.

C. Select

As nossas escolhas para explorar o paralelismo foram as seguintes: decidimos implementar um `#pragma omp reduction(max:max_c)` para lidar com as *data races* nesta variável. Esta cláusula garante que o maior valor de `max_c` é o valor da variável no final da zona partilhada, sendo possível assim convergir corretamente o `lin_solve` antes das 20 iterações.

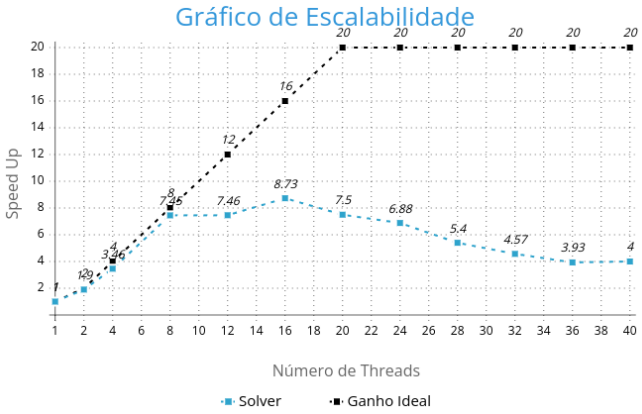
Para o *Data Sharing*, decidimos utilizar o *firstprivate*, iniciando as variáveis a 0 no sentido de que o lixo na memória não iria afetar a nossa execução. Nas zonas paralelas, decidimos fazer um `#pragma omp for` para os dois ciclos mais externos, definindo isto com um `collapse`. Adicionamos ainda a cláusula `nowait` para garantir que as threads não esperavam umas pelas outras.

Percebemos depois que todas as operações são muito parecidas, tendo um tempo de execução semelhante, por isso decidimos utilizar um *loop schedule static*, por omissão, garantindo o menor *overhead* possível. Pensamos em implementar um *schedule dynamic*, contudo, verificamos que este seria melhor para iterações que têm um tempo de execução diferente, logo, na nossa implementação só resultava em tempos de execução mais lentos.

D. Implement and Optimise

Após a implementação e testes na participação de Cpar, percebemos o que nosso tempo de execução baixou consideravelmente, decidimos voltar a fazer o *profiling*, percebendo que a `lin_solve` não representava mais hotstop mas sim a função `project`. Onde realizamos o mesmo processo que fizemos na `lin_solve`. Após repetir este processo optimizamos também a função `advect` e `set_bnd`.

E. Measue



Analisando o gráfico de escalabilidade percebemos que o nosso *speed up* máximo foi com 16 *threads* com 8,73. Contudo este *speed up* fica longo do *speed up* ideal teórico, sendo que isto é causado pelas operações que tem de ser executadas obrigatoriamente de forma sequencial, principalmente na função `set_bnd`.

III. Anexos

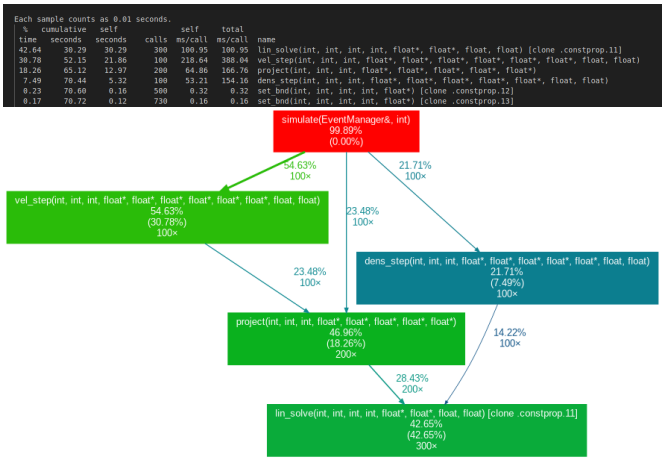


Fig. 1. Profilling Original

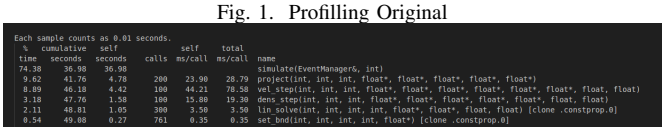


Fig. 2. Profilling após optimização `lin_solve`

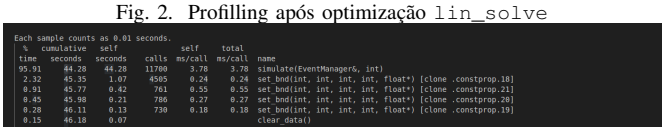


Fig. 3. Profilling Final `lin_solve`

```
Running with OMP_NUM_THREADS=8
Run 1: 0m2.030s
Run 2: 0m2.088s
Run 3: 0m2.006s
Average time for OMP_NUM_THREADS=8: 2.0413333333333333333 seconds

Running with OMP_NUM_THREADS=12
Run 1: 0m1.627s
Run 2: 0m1.647s
Run 3: 0m1.436s
Average time for OMP_NUM_THREADS=12: 1.5700000000000000000 seconds

Running with OMP_NUM_THREADS=16
Run 1: 0m1.205s
Run 2: 0m1.453s
Run 3: 0m1.361s
Average time for OMP_NUM_THREADS=16: 1.3396666666666666666 seconds
```

Fig. 4. Resultados Obtidos

Os ficheiros completos encontram se na pasta Anexos.