PROJETO TARTARUGA - FINAL

Grupo:

https://github.com/GabrielOgun/Os-Crias-e-o-Augusto-Paralelos/tree/main/Projeto/ProjetoFINAL

Nomes:

Artur Valladares Giacummo 32129221

Augusto Esteves Carrera 32114842

Gabriel Marques Gonçalves Almeida 32066724

Versão final 8min 300.000 casas decimais corretas.

Como resolvemos o problema:

Dividimos o programa em duas threads: a primeira divide o cálculo do fatorial em duas partes, de 0 a N/2 e de N/2 + 1 a N. Assim que o cálculo de 0 a N/2 é finalizado a 2a thread é iniciada para fazer a parte da conta da seria de Euler. Enquanto isso, a primeiro thread calcula a segunda parte do fatorial e já utiliza para realizar a segunda metade da conta do cálculo de e.

Qual foi o speedup da última versão e como eles fizeram para melhorar:

```
staation@DESKTOP-CN9BCOP:~$ gcc -O3 teste6.c -o teste6.exe -pthread -lgmp
staation@DESKTOP-CN9BCOP:~$ time ./teste6.exe >>300kfinaltrue4.txt

real    8m3.542s
user    13m51.468s
sys     0m4.730s
staation@DESKTOP-CN9BCOP:~$ nano testeserial.c
staation@DESKTOP-CN9BCOP:~$ gcc -O3 testeserial.c -o testeserial.exe -lgmp
staation@DESKTOP-CN9BCOP:~$ time ./testeserial.exe >> testeserial.txt

real    13m20.761s
user    13m10.156s
sys     0m10.600s
staation@DESKTOP-CN9BCOP:~$
```

Como podemos observar a versão, levando em consideração o tempo real, a versão paralela em relação a serial temos a diminuição do tempo em 5min

Resultando em um speedup de aproximadamente 1.6625 (13.3/8)

Se considerarmos o tempo de usuário o tempo é de 0.95155459146 (13,16/13,83)

```
staation@DESKTOP-CN9BCOP:~$ time ./testep1.exe >> testep1.txt

real    13m30.620s
user    13m17.612s
sys     0m12.878s
staation@DESKTOP-CN9BCOP:~$
```

Já em relação ao speedup da última versão paralela tivemos = 1.6875 (13,5/8)

Considerações sobre a estratégia de paralelismo adotada e como isso auxiliou na escala do problema.

A estratégia de paralelismo que adotamos, usando threads e semáforos, foi de certa forma eficiente para lidar com o problema, ao dividir o trabalho de calcular o fatorial e a série de Taylor em si pudemos ganhar mais velocidade na execução apesar de não conseguirmos mais casas decimais e o uso de semáforos nos auxiliou a um cálculo mais preciso sem "coredump" evitando que os dois threads acessem o mesmo espaço de memória não compartilhada.

Referencias e artefatos:

https://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread.h.html

Semaforo:

Sem_t https://pubs.opengroup.org/onlinepubs/7908799/xsh/semaphore.h.html

Sem_wait: https://pubs.opengroup.org/onlinepubs/7908799/xsh/sem_wait.html

Sem_init: https://pubs.opengroup.org/onlinepubs/7908799/xsh/sem_init.html

Sem_destroy: https://pubs.opengroup.org/onlinepubs/7908799/xsh/sem_destroy.html

Tipo de dado GMP: https://gmplib.org/manual/Nomenclature-and-Types

(Operacoes Artimeticas) https://gmplib.org/manual/Integer-Arithmetic

(Divisao) https://gmplib.org/manual/Integer-Division

(Iniciar) https://gmplib.org/manual/Initializing-Integers

(Atribuir Valor) https://gmplib.org/manual/Assigning-Integers

(Iniacilizacao e definir precisao) https://gmplib.org/manual/Initializing-Floats

(Atribuir valor) https://gmplib.org/manual/Assigning-Floats

(Operacoes Matematicas): https://gmplib.org/manual/Float-Arithmetic

https://stackoverflow.com/questions/2125410/gmp-convert-mpz-to-mpf

https://replit.com/join/gyfviztlze-arturvalladares