FAKULTA INFORMAČNÍCH TECHNOLOGIÍ, VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Architektura výpočetních systémů (AVS 2019)

Počítačové cvičení č. 5 - Pokročilé OpenMP (sekce, zámky, kritická sekce)

Gabriel Bordovský (ibordovsky@fit.vutbr.cz)
Filip Kuklis (ikuklis@fit.vutbr.cz)
Kristian Kadlubiak (ikadlubiak@fit.vutbr.cz)

25. 11. 2019

1 ÚVOD

Cílem dnešního cvičení je procvičit si pokročilé funkce v OpenMP (sekce, zámky, kritická sekce). Cvičení vypracujte na počítač v laboratoři.

2 POPIS ÚLOH

Úlohy v dnešním cvičení se zaměřují na pokročilé funkce v OpenMP. Celkem dvě úlohy se nacházejí přímo ve složce `lab3`. Prostudujte si soubor Makefile, který máte k dispozici. Příkazem *make* jednoduše přeložíte obě úlohy zároveň. Obě dvě úlohy implementují stejný problém, a to tvorbu histogramu z náhodně vygenerovaných hodnot v daném rozsahu. Defaultně se generuje 1000000 náhodných hodnot v rozsahu 0 až 255. Vaším úkolem je provést paralelní implementaci pomocí sekcí následujícím způsobem:

- Pracujte s třídou `ParallelQueue` - je nutné doimplementovat její metody pomocí kritické sekce (viz podkapitolu 2.1) a zámků (viz podkatolu 2.2).
- Tato třída používá C++ kontejner `std::queue` implementující FIFO frontu. Při implementaci jejích metod využijte metody `empty()`, `pop()`, `front()` a `push()` tohoto kontejneru.
- Program bude obsahovat dvě sekce (pozor na rozdíl mezi `#pragma omp section` a `#pragma omp sections`), kde každou bude vykonávat pouze jedno vlákno. V první sekci se budou generovat náhodná čísla pomocí volání funkce `generate()`, v druhé sekci se bude aktualizovat histogram pomocí fuknce `histogramAdd()`.

- Pro názornost a jednoduchost pracují vlákna v sekcích po jednotlivých prvcích, tj. generování prvků po jednom, aktualizace histogramu vždy jedním prvkem. Jedno vlákno tedy generuje náhodné hodnoty na konec fronty, druhé vlákno čte a odstraňuje prvky z vrcholu fronty a pomocí nich aktualizuje histogram. Jakmile první vlákno vygeneruje požadovaný počet hodnot (proměnná `iterations`), vloží na konec fronty hodnotu -1 (proměnná `stopper`). Toto je důležité z toho důvodu, aby se druhé vlákno dozvědělo o tom, že již zpracovalo všechny prvky. Vzhledem k tomu, že vlákna pracují paralelně, může se stát, že druhé vlákno pracuje s prázdnou frontou, i když ještě všechny prvky nebyly zpracovány (vygenerovány).

2.1 Implementace využívající kritickou sekci

Při implemetaci metod třídy `ParallelQueue` využijte kritické sekce (`#pragma omp critical`). Kritická sekce je tu z toho důvodu, aby v jednu chvíli modifikovalo nebo četlo prvky fronty pouze jedno vlákno. Kritickou sekci nezapomeňte pojmenovat. Implementaci proveďte do souboru `sections-critical.cpp`.
Vaším úkolem je tedy naimplementovat metody třídy využívající kritické sekce a doplnit implementaci ve funkci `main` používající sekce.

2.2 Implementace využívající OpenMP zámky

Jistě jste si všimli, že ve třídě `ParallelQueue` máme navíc proměnnou `mLock` typu `omp_lock_t`. Je to z toho důvodu, že nyní nebudeme při implementaci třídních metod využívat kritické sekce, ale zámků. Využijte tedy funkcí `omp_set_lock` a `omp_unset_lock`. Zde je vaším úkolem naimplementovat i konstruktor a destruktor třídy, kde je nutné inicializovat a destruovat zámek (vhodné funkce zkuste tentokrát najít sami nebo využijte podkladů k přednáškám). Implementaci sekcí ve funkci `main` můžete převzít z předchozího bodu. Implementaci proveďte do souboru `sections-locks.cpp`.

2.3 Otázky

Zkuste spustit kód s jedním vláknem a porovnejte. Je tady nějaký problém? Jak byste tento problém řešili?