UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Semestrální práce

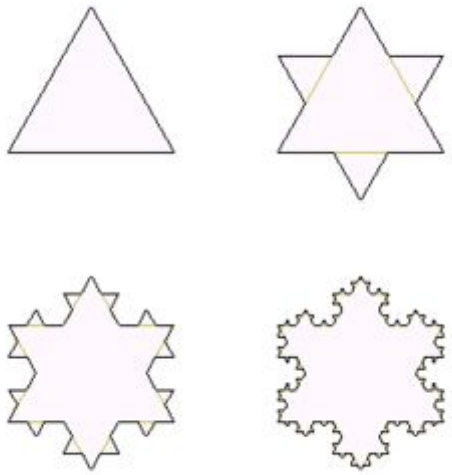
Paralelní programování - Kochova vločka v OpenMP

2017 Jakub Žufánek

ÚVOD

Semestrální práce byla napsána v jazyce C++ s využití technologie OpenMP pro paralizaci vypočtu kochovi vločky a SDL pro vykreslení kochovi vločky.

KOCHOVA VLOČKA



Kochova vločka se skládá rovnostranného trojúhelníku o straně a. Nad středem každé jeho strany sestrojíme opět rovnostranný trojúhelník o straně a1 = a/3 (dostaneme židovskou hvězdu). Nad středem každé její strany (je jich 12) sestrojíme další trojúhelníček o straně a2 = a1/3 = a/9. Vzniklý útvar bude mít 48 stran. Takhle pokračujeme neomezeně dál. Každá další iterace vločky má čtyřikrát víc stran než iterace předchozí a strany jsou třikrát menší.

IMPLEMENTACE

OMP Nastavení

Počet vláken definujeme na řádku 12 konstantou COUNT_THREADS.

Struct Line

Struktura nám drží 4 atributy: double pointX, pointY, length, angle. PointX,pointY reprezentuje souřadnicový systém. Length reprezentuje délku přímky a angle úhel úsečky. Funkce getX2(), getY2() vrací souřadnice konce úsečky. Funkce draw() vykreslí úsečku pomocí SDL.

Funkce Main

V datové struktuře vektor jsou drženy všechny úsečky. OMP dokumentace doporučuje deklarovat pomocné proměnný před #pragma omp, aby se zajistilo, že každé vlákno bude mít vlastní proměnný. V pragma je pak zmíníme pomocí klauzole private, která zajistí, aby každý vlákno mělo vlastní proměnný. Viz následná příloha Main.cpp. Sdílený proměnný považuje původní vektor pro úsečky a nový vektor pro úsečky, který má předem rezervovanou velikost počtu úsečku. Každý vlákno pak vlastní vektor pro uložení vlastních úseček, který prošel ve for cyklu a zároveň je rezervován na velikost původního počtu úseček / počtem vláken. Po skončení prací ve for cyklu je považována spojování vektorů vláken do nového vektoru. Kód je řádně okomentován a čitelný.

SDL

Pro spuštění SDL je potřeba odkomentovate řádek 8 a 9. Je potřeba mít správně nastavené visual studio pro SDL. Návod pro postup zde: https://www.youtube.com/watch?v=QQzAHcojEKg&t=278s
Na řádku 171 je funkce SDL_Delay, která nastavuje po jaký době se má znovu vykreslit plátno.

Měření

Poznámka: měření jsou bez vykreslování externí knihovny SDL

Iterace (počet úseček)	1. vlákno	2. vlákna	3. vlákna
1. (12)	0.00009	0.00021	0.00064
2. (48)	0.00018	0.00032	0.00076
3. (192)	0.00026	0.00045	0.00089
4. (768)	0.00058	0.00234	0.00108
5. (3 072)	0.00131	0.00298	0.00142
6. (12 288)	0.00554	0.00442	0.00481
7. (49 152)	0.01106	0.00965	0.00981
8. (196 608)	0.03836	0.02950	0.02597
9. (786 432)	0.14122	0.11370	0.07824
10. (3 145 728)	0.52306	0.4655	0.29017
11. (12 582 912)	1.91034	1.53156	1.14041
12. (50 331 648)	7.68642	5.81434	4.43660

13. (201 326 592)	114.08904	63.81174	61.8621

Sestava stroje

Procesor: Intel i5-6200U s taktem 2.3GHz
Operační pamět: 8GB
Disk: SSD 223 GB

Závěr

Po mukách a občasném tápání v OMP bylo docíleno výsledku. Problém nastal s kritickou sekcí, kde OMP strašně zpomalil v každé for iteraci, jak bylo původně řešeno. Nakonec bylo vyřešeno skládáním vektorů do jednoho nového vektoru a přesun kritické sekce za for cyklus. Pro autora byla dobrá zkušenost s OMP a řešení paralizace. Ve výsledku měření je vidět, že pro delší běh se hodí nastavení 3 vláken. V prvních iterací běh s více vlákny je pomalejší oproti jednomu vláknu, díky režii operačního systému. Bohužel 14 iteraci nedovolí stroj vykonat, jelikož nabíhá vysoká alokace paměti, kde OS nutí často swapovat na disk a neposlední řadě sestřelí tento proces. Na závěr chce autor dodat, že kód je řádně okomentovaný a čitelný.