SLOVENSKÁ TECHINCKÁ UNIVERZITA V BRATISLAVE

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

ZADANIE 2

Paralelizácia pomocou OpenMP a MPI v jazyku C

PARALELNÉ PROGRAMOVANIE

Samuel Bubán

ID: 102879

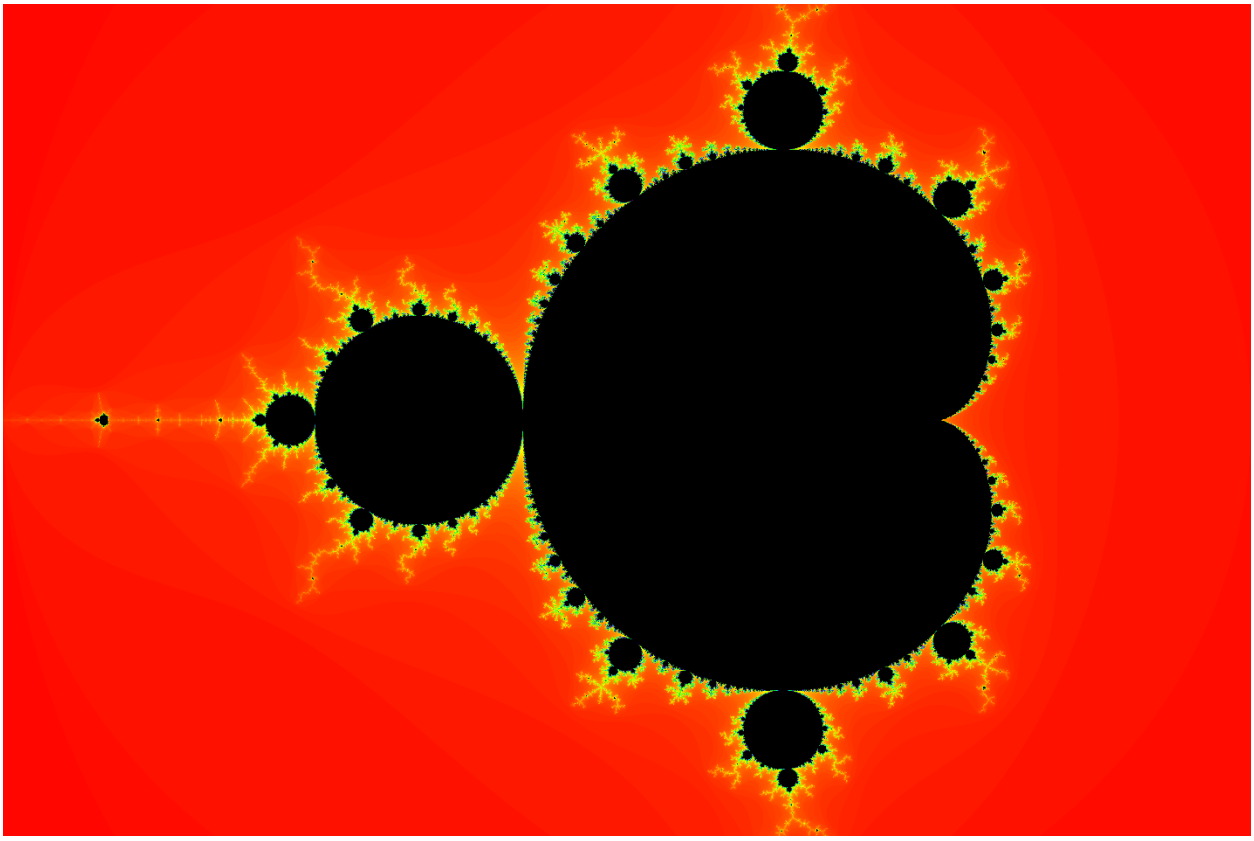
24. 11. 2021

PROBLÉM

Ako problémový algoritmus, ktorý budem pomocou paralelizácie urýchlovať som si zvolil fraktál Mandelbrot set. Jeho výpočet sa dá robiť pomocou rekurzie, ale aj iteratívne. Riešenie bolo prevzaté zo stránky: https://rosettacode.org/wiki/Mandelbrot_set#C a následne adaptované.

Pôvodné riešenie generovalo čierno-biele výsledky, preto som ho upravil na farebné. Na konverziu farieb medzi farebnými modelmi som použil prevodový algoritmus zo stránky: https://stackoverflow.com/questions/3018313/algorithm-to-convert-rgb-to-hsv-and-hsv-to-rgb-in-range-0-255-for-both/6930407#6930407

Implementácia sa nachádza v súbore mandelbrot.c



Raz som nechal tento algoritmus bežať niekoľko hodín, aby vytvoril obrázok s rozlíšením 120 000 x 80 000 pixelov. Veľkosť tohto obrázku bola takmer 30GB (ak by to bolo .png, mal by asi 5GB). Bola to najväčšia veľkosť aká sa mi podarila, ináč som narazil na problém s príliš veľa číslami vo výstupe (.ppm formát ich toľko nepodporoval).

OPENMP

Pomocou tejto knižnice som vytvoril paralelizované počítanie pôvodného algoritmu, kde som na paralelizáciu využil direktívu for.

Ešte bolo potrebné vytvoriť nové globálne pole namiesto priameho zapisovania do súboru, lebo ináč vznikali nežiadúce výsledky (kedy viacero vlákien zapisovalo na rovnaké miesto).

Implementácia je v mandelbrot_OMP.c

MPI

Pomocou tejto knižnice som tiež vytvoril paralelizované počítanie algoritmu. Tu som na paralelizáciu využil dynamické vytvorenie pamäte pre jednotlivé vlákna, aby malo každé vlastnú pamäť, a nebola alokácia príliš veľká, a teda sa predišlo zbytočnému plytvaniu pamäťou.

Vypočítal som veľkosť, koľko cyklov bude vykonávať každé vlákno (rozdelil som jeden beh for cyklu). Každé vlákno si zapisovalo výsledky do svojho poľa.

Po vykonaní všetkých výpočtov sa zhromaždia všetky údaje v hlavnom root vlákne (0) pomocou gather. Následne sa dopočítajú zvyšné údaje (ak nemohol byť for cyklus rozdelený rovnomerne, a zostalo ešte niekoľko iterácii potrebných na doplnenie) a zapíše sa všetko do súboru.

Implementácia je v mandelbrot_MPI.c

TESTOVANIE

Na porovnanie všetkých implementácii som pridal výpis času, ako dlho trval výpočet. Do tohto času sa nezapočítava výpis do súboru (nakoľko je to u všetkých rovnaké). Pri MPI sa čas zastaví, až keď sa dopočítajú aj zvyšné údaje, ktoré neboli počítané paralelne, ale len v hlavnom vlákne.

Každá implementácia bola spustená 3x, na dvoch rôznych rozlíšeniach.

Rozlíšenie 600 x 400 pixelov:

- default: 0.135s
- OMP: 0.24s
- MPI: 0.4s

Rozlíšenie 6000 x 4000

- default: 13.4s
- OMP: 2.4s
- MPI: 2.45s

Z týchto výsledkov je vidieť, že pri menšej veľkosti sa neoplatí paralelizovať tento fraktálový algoritmus. Dôvodom je, že niektoré oblasti trvajú omnoho dlhší čas ako ostatné, a preto pri menšej veľkosti môže jedno vlákno bežať príliš dlho a ostatné už majú dávno všetko vypočítané.

Ale pri vyššom rozlíšení je paralelizované riešenie omnoho rýchlejšie ako sériové (5.5x).

Pre nižší počet vlákien pri OMP riešení to trvalo 3.95s pre 6 vlákien a 7.77s pre 3 vlákna.

VIACERO UZLOV

Pri MPI implementácii som si tiež otestoval spustenie na viacerých uzloch. Najprv som skúšal spustenie cez LAN, ale nedostal som sa cez firewall, preto spustenie na dvoch zariadeniach sa mi nepodarilo.

Podarilo sa mi ale emulovať viacero uzlov na jednom zariadení. Na to bolo potrebné spustiť službu smpd (pomocou run_mpi_server.bat) a následne spustiť MPI na 2 uzloch pomocou run_mpi_nodes.bat. V tomto súbore sa nachádza na pevno zadaná IP adresa môjho zariadenia a počet vlákien, koľko má na každom uzle bežať.

Ja som ho spustil na 2 uzloch, každý po 6 vlákien, a úspešne mi vytvorilo výsledný súbor. Čas behu bol porovnateľný s jednouzlovým MPI (keďže nebola potrebná komunikácia cez sieť).

Pre 2 uzly, každý po 3 vlákna trvalo vypočítanie 4s. Pre dva uzly po jednom vlákne to trvalo 6.61s.

OPENMP + MPI

Kombinácia týchto dvoch spôsobov vytvorila ešte rýchlejšie riešenie, konkrétne 1.5s – 1.7s na vyššom rozlíšení. Skúsil som ho najprv s 12 OMP vláknami až po 3, ale veľký rozdiel to nespravilo (ako bolo vyššie spomenuté, len 0.2s).

Táto implementácia sa nachádza v mandelbrot_COMB.c