Násobení matic metodou Divide and Conquer

Matěj Boxan

5. prosince 2018

Úvod a popis zadání

Úkolem mnou vybraného zadání semestrální práce z předmětu PJC bylo implementovat algoritmus násobení matic. Zvolil jsem metodu Divide and conquer, která pomocí dělení matice na čtvrtiny a jejich skládání poskytuje dobrou výchozí pozici pro implementaci algoritmu na více vláknech. Pro zjednodušenní zadání jsem uvažoval následující podmínky:

- Matice obsahuje pouze celá čísla integer
- Matice je čtvercová
- Strana matice je mocninou čísla 2

Část 1 Implementace

Pro řešení jsem použil vlastní implementaci matice, kterou reprezentuje třída Matrix1D. Ta nahrazuje původní třídu Matrix, kterou jsem v řešení ponechal, ale je nefunkční. Na jednodimenzionální reprezentaci matice ve třídě Matrix1D jsem přešel z důvodu zvýšení rychlosti algoritmu.

Matice má atribut *m_strategy*, reprezentují zvolenou strategii násobení matic. Zde lze volit mezi naivním algoritmem, algoritmem Divide and conquer a algoritmem Divide and conquer na více vláknech (pokud to hardware počítače umožňuje).

K ovládání aplikace slouží několik parametrů, které je možné zobrazit pomocí přepínače --help (ekvivalentní k přepínačům -h a -H). Mezi přepínače patří:

- -h -H --help
  Vypíše seznam použití a paremetrů
- -t -T --test [n][lowBound][upperBound][filename]
  Spoustí n-krát testovací skript pro matice o velikost od lowBound do upperBound, přičemž lowBound a upperBound musí být mocniny 2. Průměrné výsledky pro jedno i vícevláknový Divide and conquer uloží do souboru, specifikovaného pomocí *filename*.
- -f -F --file [filename]{implementation}
  Načte matice ze souboru *filename* a dle zvolené implementace provede test rychlosti algoritmů násobení matic.
- -r -R --random [n]{implementation}
  Náhodně vygeneruje čtvercové matice o straně *size* a dle zvolené implementace provede test rychlosti algoritmů násobení matic.

Přepínače implementací jsou:

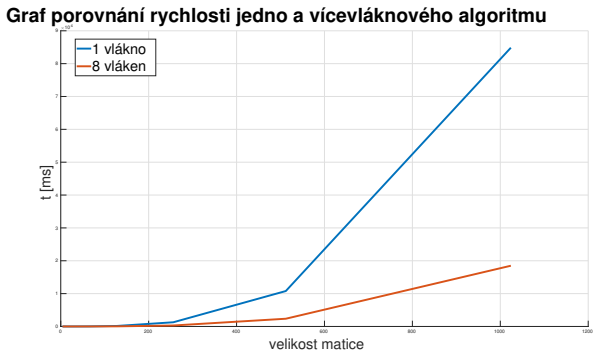
- -p -P --parallel
  Otestuje rychlost paralelního algoritmu.
- -s -S --single
  Otestuje rychlost jednovláknového algoritmu.
- -c -C --compare
  Porovná rychlosti jednovláknového a paralelního algoritmu.

V případě, že není implementace specifikována, je vybrána implementace --compare.

Část 2 Naměřené výsledky

K naměření časů jsem použil výše zmíněný parametr --test v konfiguraci --test 10 4 1024 TEST.txt, tedy pro čtvercové matice o straně mocniny dvou v intervalu od 4 do 1024 jsem vždy nechal provést 10 měření. Výsledky měření obsahuje tabulka v podobě průměrného času pro danou velikost matice a implementaci.

size	single thread[ms]	parallel[ms]
4	0	0.5
8	0	0.3
16	0	0
32	3	0.6
64	20	4.4
128	156	37.4
256	1276.6	292.5
512	10782	2354.5
1024	84849	18486.7



Obrázek 1: 1 vlákno a 8 vláken pro velikost matice v intervalu $\langle 4; 1024 \rangle$

Z grafu a z tabulky je zřejmé, že použití více vláken zrychlí běh programu zhruba 4násobně. Měření jsem provedl na počítači s 8-jádrovým Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz ve vývojovém prostředí CLion v release módu. Měření bylo provedeno vůči hash commit: 033eb1626674f23867339f7c544bd2c01a832af2.

Závěr

V semestrální práci se mi podařilo implementovat funkční Divide and conquer algoritmus pro násobení matic, pracující na jednom i více vláknech. Tento algoritmus je vzhledem k rozdělování matic vhodný pro implementaci na více vláknech, nicméně co do složitosti je na tom stejně, jako naivní řešení pomocí 3 for cyklů (složitost $\Theta(n^3)$. Lepších výsledků lze dosáhnout například pomocí Strassenova algoritmu se sníženým násobením a složitostí $\Theta(n^{2.807})$. Vypracovaná práce umožňuje jeho pozdější doplnění.