Implementace algoritmu

Carry Look Ahead Parallel Binary Adder

Autor: Anna Ostroukh (xostro00)
Datum: 02.05.2016

1. Popis algoritmu Carry Look Ahead Parallel Binary Adder

Cílem algoritmu Carry Look Ahead Parallel Binary Adder je sečíst dvě binární čísla v log(n) krocích a to pomocí předvýpočtu bitů přenosu (carry) C_{n-1} .. C_0, aby bylo možné přímo spočítat $Z = X_i+Y_i+C_i \mid i = 0 .. n-1$
Výpočet carry bitů se provádí v následujících krocích:

- Výpočet pole D, které bude obsahovat hodnoty stop, generate nebo propagate: Výpočet se provádí na základě hodnot dvou bitů X a Y:
  - Jsou – li obě hodnoty 1, D = g;
  - Jsou – li obě 0, D = p;
  - Jinak, D = s.
- Výpočet scan pole D na základě binární operace $\odot$, čímž dostaneme všechny bity přenosů v logaritmickén čase.

Výpočet scan pole D

Výpočet scan pole D se provádí na základě operace prescan na paralelní stromové architektuře, ale s drobnými úpravami, které jsou implementovány v současné práci.

Úpravený algoritmus scan má následující kroky:

- UpSweep algoritmus provede reduce hodnot D (aplikuje pro káždý uzel stromu, který má děti, operaci $\odot$ na jich hodnoty) v log(n) krocích. Každý uzel si bude pamatovat svou vypočítanou hodnotu pro následující krok DownSweep, na rozdíl od obyčejné operace reduce;
- DownSweep přířadí kořenu neutrální prvek p a v log(n) krocích počínaje kořenem, směrem k listům procesory paralelně vykonávají následující operaci:
  - Uzel dá svému levému synovi hodnotu právého syna $\odot$ svou hodnotu a pravému synovi dá svoji hodnotu.

Teoretická složitost algoritmu

Složitost algoritmu Carry Look Ahead Parallel Binary Adder se skladá ze složitostí jeho jednotlivých částí:

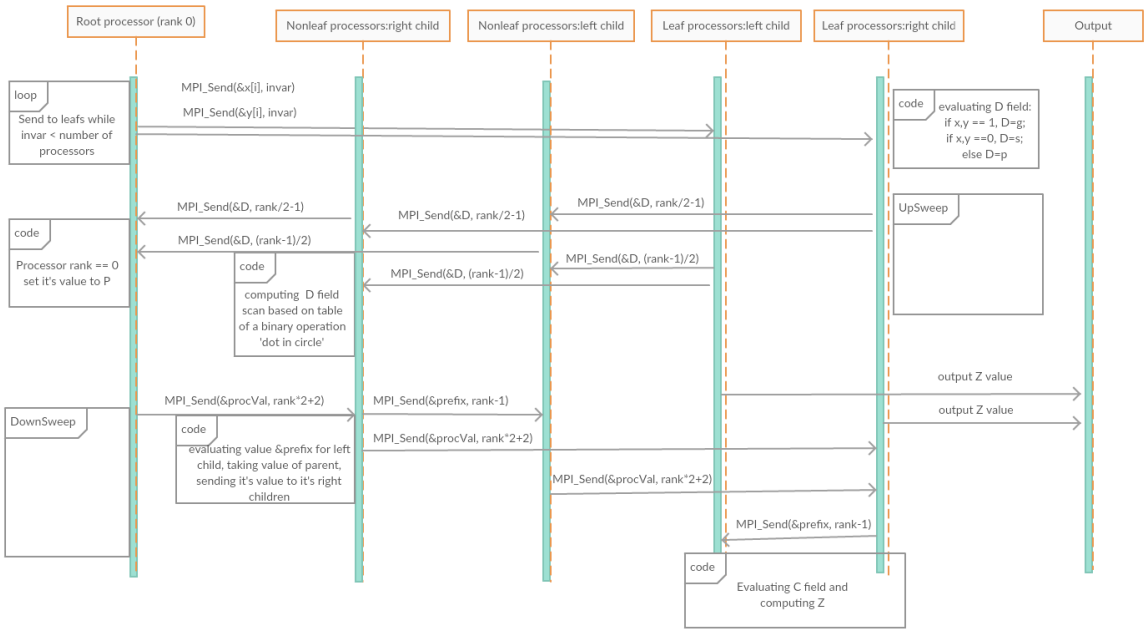
- Vypočet pole D se provede v konstantním čase – O(n)
- UpSweep se provede v logaritmickém čase – O(log n)
- DownSweep se provede v logaritmickém čase – O(log n)
- Výpočet Z se provede v konstantním čase – O(n)

Výsledná časová složitost t(n) je logaritmická O(log n).
Počet procesorů p(n), potřebný pro výpočet je 2n – 1.
Výsledná cena: c(n) = p(n)*t(n) = O(n*log n).

2. Implementace algoritmu

Implementace algoritmu je provedena pomocí jazyka C++ a knihovny pro paralelní programování *Open MPI* (*Message Passing Interface*).
Algoritmus je schopen fungovat na vstupní posloupností dvou bitů ze souboru s názvem *numbers*, který je uložen ve složce projektu. Je-li posloupnost čísel jednoho z bitu není rovná mocnině dvojky, program dorovná jej zleva nulami. Algoritmus je implementován tak, že skript *test.sh* na základě určené délky posloupnosti bitů sam výpočitá správný počet procesorů.
Procesor s rankem 0 (kořen stromu) načítá posloupností bitů ze souboru, rozdělí na jednotlivá čísla a rozešle hodnoty listovým procesorům.
Nejlevejší listový procesor označuje signalizace přetečení klíčovým slovem *overflow*.

3. Komunikační protokol



Obrázek 1 – Sekvenční diagram implementovaného algoritmu

4. Experimenty

Pro měření výsledků použita standardní funkce knihovny MPI *MPI_Wtime()*.
Měření bylo provedeno jen pro samotné sečtení. Čas na náčtení hodnot ze souboru *numbers* a rozesílání hodnot ostatním procesorům nezapočítán.

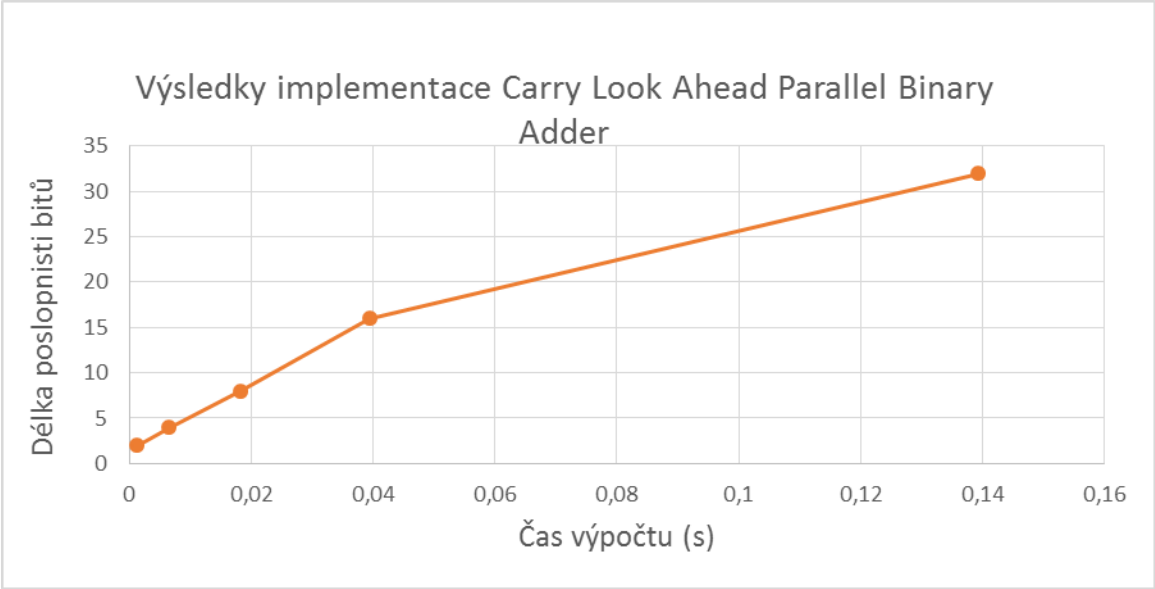
Měření bylo prováděno 10-krát pro každou hodnotu, pak byl najden aritmetický průměr.

V následující tabulce jsou uvedeny hodnoty času výpočtu algoritmu pro různé délky vstupních bitů.

Délka posloupnosti bitů	Čas výpočtu (s)*
2	0,001314522

4	0,006535095
8	0,018321389
16	0,039377356
32	0,1393549

**Měření probíhalo na virtuálním stroji*



Obrázek 2 – Výsledky měření výpočtu algoritmu sečtení

5. Závěr

Jak víme podle grafu na obrázku 2, graf experimentálních výsledků je obdobný grafu n*log n, co potvrzuje teoretickou složitost.