Carry Look Ahead Parallel Binnary Adder

Petr Dvořáček – xdvora0n@stud.fit.vutbr.cz

Tento dokument pojednává o algoritmu Carry Look Ahead (CLA) Parallel Binary Adder. Text se především zaměřuje na jeho implementaci a experimentováním s časovou složitostí.

1 Popis algoritmu a jeho implementace

CLA Parallel Binary Adder je úplná paralelní binární sčítačka, která sečte dvě n-bitová binární čísla v logaritmickém čase, $t(n) = log(n)$. K tomu je však zapotřebí $2n-1$ procesorů ve stromové topologii, $p(n) = 2n-1$. Této časové složitosti lze docílit tím, že nejdříve předpočítáme všechny bity přenosů $c_n - 1 \dots c_0$. Ty pak použijeme k výpočtu výsledné hodnoty z.

Princip algoritmu lze shrnout následovně:

1. Každý procesor vypočítá hodnotu d_i podle hodnot x_i a y_i a tabulky:

x	y	Hodnota d	Zkr.	Popisek
1	1	GENERATE	g	(přenos nastane)
0	0	STOP	s	(přenos nenastane)
1	0	PROPAGATE	p	(přenos může nastat)
0	1	PROPAGATE	p	(přenos může nastat)

2. Spočte se suma prefixů podle operace $scan()$ a hodnot d_i jednotlivých procesorů. Tento krok se skládá ze dvou fází.

   - Upsweep – Jedná se o průchod z listů ke kořeni. Pomocí stromové topologie se vypočte hodnota reduce $scan(d_0, d_1, d_2 \dots d_n)$ takovým způsobem, že si každý uzel pamatuje svůj mezisoučet.
   - Downsweep – Jedná se o průchod z kořene k listům. Kořenu stromu se nejdřív přiřadí neutrální prvek (PROPAGATE). Pak levému synovi se předá hodnota $scan(d_i, d_{rson})$, pravému synovi se předává svá hodnota mezisoučtu.

   Operace $scan()$ je definována takto:

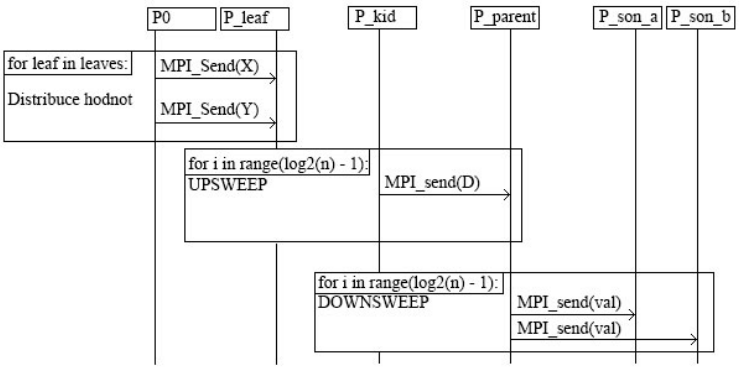
x	s	p	g
s	s	s	s
p	s	p	g
g	g	g	g

3. Pomocí výsledků sumy prefixů se nastaví hodnota c a vypočte hodnota z.

Celková složitost algoritmu je zřejmá. První krok algoritmu se provede v konstantním čase $O(1)$. Upsweep prochází stromem v logaritmickém čase $O(log(n))$. Downsweep je rovněž průchod binárním stromem a jeho složitost je taktéž logaritmická $O(log(n))$. Výpočet výsledné hodnoty je konstantní $O(1)$. Výsledná časová složitost je tedy $t(n) = O(2 * log(n) + 2) = O(log(n))$. Na začátku této kapitoly je vysvětleno, že je zapotřebí $p(n) = 2 * n - 1 = n$ procesorů, které jsou zapojeny ve stromové architektuře. Tudíž celková cena algoritmu je rovna $c(n) = t(n) * p(n) = n * log(n)$.

2 Komunikační protokol

V implementaci bylo předávání zpráv řešeno tímto způsobem. Proces s indexem 0 rozesílá zprávy listům přímo. Pro fungování algoritmu každý proces zná indexy synů a rodiče. Index rodiče lze vypočíst podle rovnice: $parentidx = (myidx–1)/2$, kde operace dělení zaokrouhluje dolů na celých číslech. Index levého syna získáme dle rovnice: $leftsonidx = myidx * 2 + 1$ a index pravého syna dostaneme podle rovnice: $rightsonidx = myidx * 2 + 2$. Princip této komunikace je pak znázorněn na následujícím obrázku:



3 Experimentální vyhodnocení

Pro ověření časové složitosti programu byla použita funkce `gettimeofday()` z modulu `<sys/time.h>`. Začátek respektive konec měření se prováděl po inicializaci procesů respektive před jejich ukončením. Bylo zvoleno vždy 10 běhů pro pole o délkách 4, 8 a 16, jejíchž časové hodnoty se pak vypisovaly na standardní chybový výstup (`stderr`). Měření proběhlo na školním serveru Merlin. Následující tabulka pak zachycuje naměřené hodnoty, z nichž lze vyčíst, že algoritmus se chová logaritmicky.

Velikot pole	Průměrný čas (us)
4	293.0
8	559.6
16	947.2