Dokumentácia projektu 3 do predmetu Paralelní a distribuované algoritmy

Pokorný Fridolín
xpokor32@stud.fit.vutbr.cz

30. apríla 2014

1 Zadanie

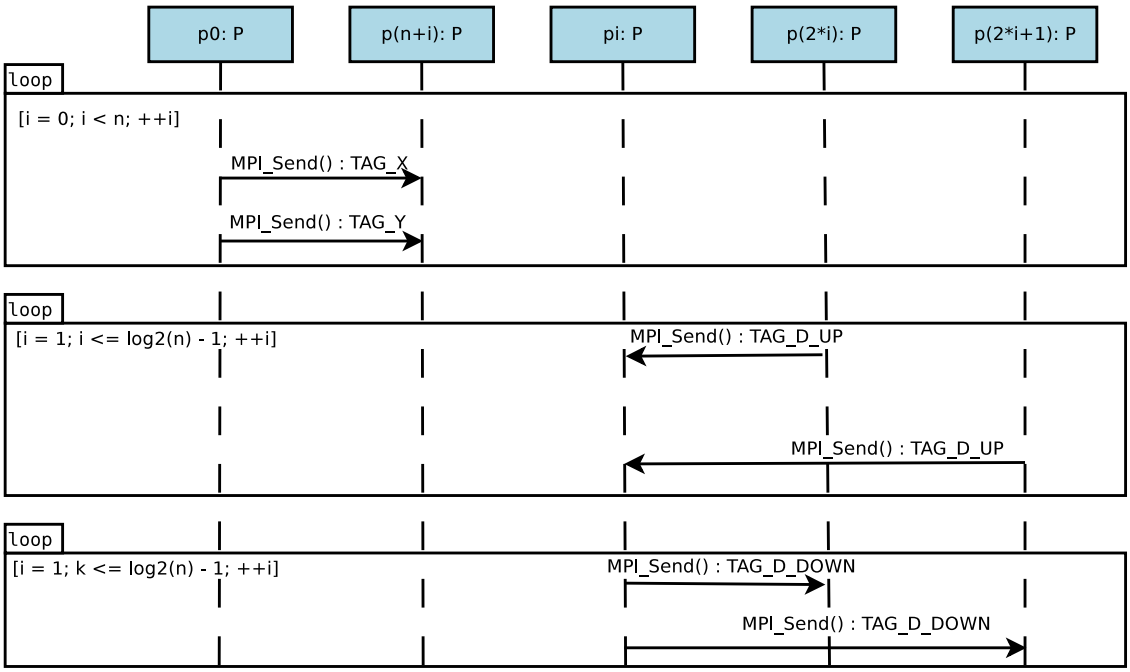
Cieľom projektu bolo implementovať algoritmus *Carry Look Ahead Parallel Binary Adder* v prostredí *OpenMPI*. Implementovaný program pracuje nad súborom „`numbers`“, ktorý obsahuje práve dve čísla v binárnej podobe zapísané na samostatných riadkoch. Konce riadkov sú ukončené znakom LF (ASCII 10). Prácu implementovaného programu riadi *BASH* skript „`test.sh`“, ktorý sa stará o overenie základnej korektnosti vstupu, vypočítanie počtu potrebných procesorov pre výpočet a samotné zahájenie výpočtu spustením programu s príslušnými parametrami.

2 Implemntácia algoritmu

Algoritmus v základnej podobe využíva pre výpočet celkovo $2 \cdot n - 1$ procesorov. Implementovaný algoritmus využíva $2 \cdot n$ procesorov, kde jeden procesor neprevádza výpočet, ale stará sa o načítanie a overenie vstupu s následnou distribúciou hodnôt ostatným procesorom. Načítanie vstupu a distribúcia hodnôt však vzhľadom na sekvenčné čítanie vstupu má lineárnu časovú zložitosť, čo má zlý vplyv na celkovú časovú náročnosť algoritmu. Vzhľadom na sekvenčné prostredie (prístup k datám na disku, neasociatívny prístup k datám, plánovanie a počet dostupných procesorov pre paralelný výpočet, ...) je však veľmi náročné nasimulovat skutočne paralelný výpočet.

Po distribúcii hodnôt procesorom s ID 1 (príp. aj paralelne), je vypočítaný obsah registra `d` a prevádza sa akcia *up – sweep*, ktorá je podobná operácii *reduce* uvedenej v prednáškach, ale každý uzol si poznamenáva medzisúčet pre ďalší výpočet. Binárna operácia je definovaná pomocou funkcie `operation_prop()`, ktorá na základe parametrov vypočíta výsledok pre aktuálne počítaný uzol. Vzhľadom na blokujúce operácie `MPI_Send()` a `MPI_Recv`, nie je potrebná explicitná synchronizácia procesorov.

Po výpočte *up – sweep* je zahájený výpočet *down – sweep* (prípadne do istej miery je možné prevádzať *down-sweep* paralelne pre niektoré uzly). Koreňu je priradený neutrálny prvok „*PROPAGATE*“ a každý uzol priradí svojmu pravému synovi svoju hodnotu a svojmu ľavému synovi hodnotu vypočítanú na základe operácie $\oplus$, kde operandy sú hodnota registra `d` pravého syna a svoja hodnota registra `d`. Nutné je podotknúť, že operácia je pozmenená (zrkadlovo otočená) než pôvodná verzia uvedená v prednáškach, nakoľko listový procesor s najnižším ID uchováva najvýznamnejší bit. Nakoľko operácia nie je komutatívna, prvý operand sa berie hodnota registra `d` pravého syna, druhým je hodnota svojho registra `d`.



Obr. 1: Sekvenčný diagram zasielania správ.

$\oplus$	*STOP*	*PROPAGATE*	*GENERATE*
STOP	*STOP*	*STOP*	*STOP*
PROPAGATE	*STOP*	*PROPAGATE*	*GENERATE*
GENERATE	*GENERATE*	*GENERATE*	*GENERATE*

Tabuľka 1: Binárna operácia $\oplus$ implementovaná vo funkcii `operation_prop()`.

3 Zložitosť algotitmu a experimentalne výsledky

Asymptotická zložitosť algoritmu:

$$\mathcal{O}(n) + \mathcal{O}(1) + \mathcal{O}(\log_2 n) + \mathcal{O}(\log_2 n) = \mathcal{O}(n)$$

kde n je počet bitov sčítaných čísiel. Jednotlivé zložky vyjadrujú:

$\mathcal{O}(n)$ – distribúcia hodnôt
$\mathcal{O}(1)$ – výpočet hodnoty registra `d`
$\mathcal{O}(\log_2 n)$ – operácia „*up – sweep*“
$\mathcal{O}(\log_2 n)$ – operácia „*down – sweep*“

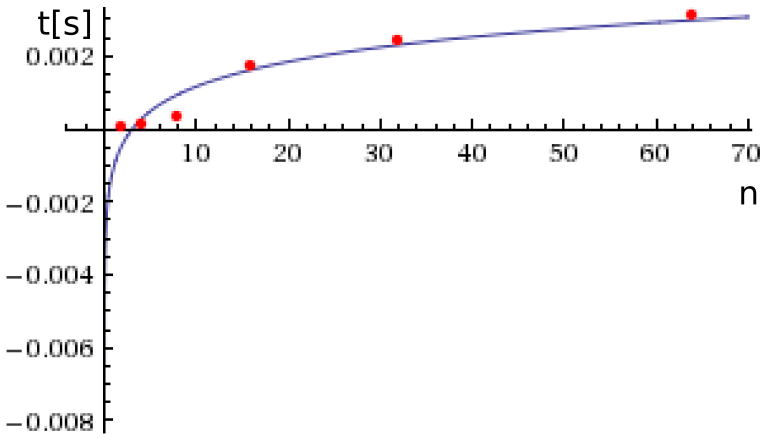
Vzhľadom na cieľ projektu, ktorý bol implementácia algoritmu *Carry Look Ahead Parallel Binary Adder* bol v testoch zanedbaný čas distribúcie hodnôt na jednotlivé procesory. Preto bola implementovaná funkcia `get_time()`, ktorá získa aktuálny systémový čas pre porovnanie experimentálnych výsledkov. Pre získanie požadovaného výstupu je nutné definovať konštantu preprocesoru „`DEBUG`“ odkomentovaním príslušného riadku na začiatku zdrojového kódu. Získavanie času je realizované s ohľadom na zanedbanie počiatočnej distribúcie hodnôt jednotlivým procesorom (v hardware je táto operácia s konštantnou časovou zložitosťou) ako aj na prípadné synchronizovanie jednotlivých uzlov. Pri zanedbaní distribúcie hodnôt by mala výsledná asymptotická časová zložitosť degradovať na logaritmickú, čo koreluje so získanými výsledkami uvedenými v tabuľke 2. Táto tabuľka reprezentuje

namerané časy výpočtu. Pre každú bitovú šírku bolo uskutočnených päť testov a z nameraných výsledkov bol vybraný medián pre zohľadnenie vedľajších efektov (plánovanie operačným systémom a pod.).

Počet bitov čísla	Nameraný čas výpočtu $[s]$
2	2.40803e-05
4	0.000106096
8	0.000286102
16	0.00172311
32	0.00242608
64	0.003101602

Tabuľka 2: Namerané hodnoty pri časových testoch.

Ako je zrejmé z obrázka 2, namerané hodnoty približne aproximujú logaritmickú funkciu. Pri nižších hodnotách počtu bitov však namerané hodnoty rastú skôr lineárne. Dôvodom môže byť príliš malý počet procesorov a tým náročné zmeranie presnejších hodnôt s ohľadom na réžiu spojenú so zasielaním a prijímaním správ v prostredí *OpenMPI*.



Obr. 2: Logaritmická aproximácia mediánov nameraných hodnôt pri časových testoch.

4 Zhrnutie

Experimentálne výsledky uvedené v sekcii 3 potvrdzujú logaritmickú triedu asymptotickej zložitosti. Komunikáciu medzi procesormi znázorňuje sekvenčný diagram v sekcii 2, v ktorej je uvedený aj podrobný popis implementovaného algoritmu. Algoritmus bol úspešne implementovaný a otestovaný.