Ladislav Šulák [xsulak04@stud.fit.vutbr.cz]
Projekt do predmetu Paralelní a distribuované algoritmy
Minimum Extraction sort
3/2016, FIT VUT v Brně

Rozbor a analýza algoritmu

Algoritmus vykonáva výpočty nad binárnym stromom, pričom uzly reprezentujú jednotlivé procesory. Koreňový uzol na začiatku distribuuje čísla listovým uzlom. Počet listov, teda aj čísel musí byť mocninou čísla 2 a počet procesorov potom $p(n) = (2 * n) - 1$.

Ostatné nelistové uzly prijímajú vždy hodnoty zo svojich dvoch potomkov, porovnávajú ich a hodnota menšieho z nich je poslaná rodičovskému uzlu. Takto sa po $log(n) - 1$ krokoch (čo je vlastne výška binárneho stromu) dostane na výstup prvá, najmenšia hodnota. Postupne do koreňa vstupujú ďalšie minimálne hodnoty, teda pre zvyšné hodnoty je to $2*n$.

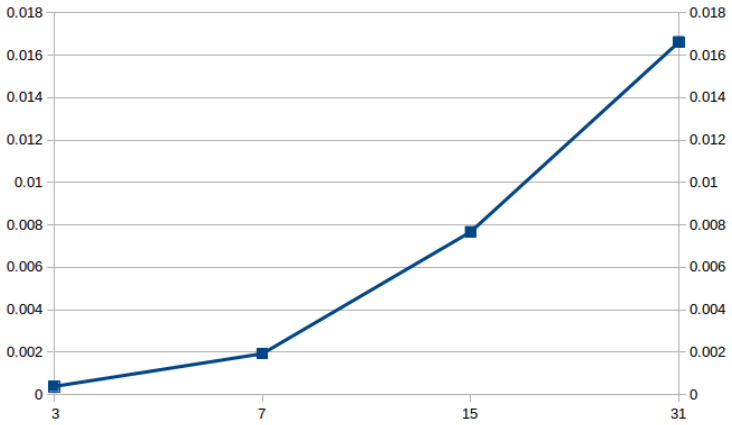
Čo sa týka teoretickej časovej zložitosti, tak je kvôli tomuto postupu rozdelená na logaritmickú a lineárnu časť, čiže $2 * n + log(n) - 1$. Časová zložitosť je teda celkovo lineárna. Priestorová je $2 * n - 1$ a celková cena je daná súčinom týchto zložitostí, teda kvadratická, čo nie je optimálne.

Implementácia

Algoritmus bol implementovaný v jazyku C++ s využitím knižnice Open MPI v prostredí Ubuntu 15.10, kde bol aj testovaný. Množina vstupných čísel bola vždy náhodne generovaná z $\backslash dev\backslash urandom$, čo zabezpečoval skript napísaný v skriptovacom jazyku Bash. Ten generuje dočasný vstupný súbor obsahujúci postupnosť čísel. Program výslednú zoradenú postupnosť vypíše na štandardný výstup.

Experimenty

Experimenty boli realizované pomocou skriptu v jazyku Python, ktorý opakovane spúšťal algoritmus s rôznym počtom náhodne vygenerovaných hodnôt a vďaka funkcii `MPI_Wtime()`, ktorá merala čas vykonávania procesu. Počet procesorov bol volený 3, 7, 15 a 31 a každý experiment bol vykonávaný 10 krát, dokopy teda 40 meraní. Časová zložitosť je zachytená na grafe 1. Pri experimentoch bolo nutné pracovat s hodnotami, ktoré sú mocninami 2, kvôli povahe algoritmu. To je jeden z dôvodov, prečo je graf značne skreslený a nie je úplne lineárny. Ďalším dôvodom je, že s narastajúcim počtom procesorov rástla réžia s tým spojená.



Obr. 1: Časová zložitosť algoritmu. Na X-ovej ose je znázornený počet procesorov a na Y-ovej ose aritmetický priemer časov behov každého procesu v danom experimente.

Komunikačný protokol

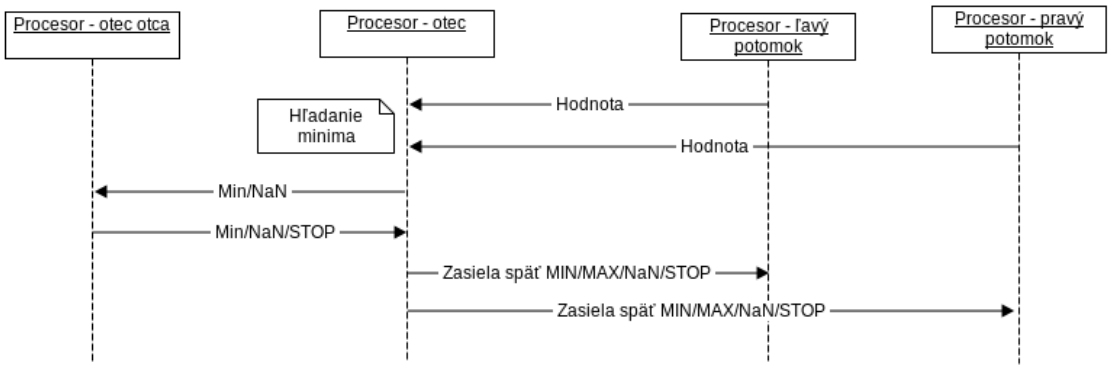
Komunikačný protokol, ktorý bol navrhnutý a pomocou ktorého je realizovaný zadaný algoritmus je znázornený na obrázku 2. Protokol funguje približne nasledovne:

- dva uzly, napríklad listové, zasielajú svoje hodnoty vyššie nadradenému procesoru v binárnom strome,
- ten ich porovná a zašle toto minimum svojmu otcovi,
- vyššie nadradený procesor pošle späť svojmu potomkovi buď toto číslo, alebo značku indikujúcu, že dané čisło bolo aktuálne najviac minimálne a bude poslané o ďalšiu úroveň vyššie (prípadne na výstup, ak sa jedná o koreň). Toto číslo je v grafe znázornené ako `NaN`, čo môže byť aj prázdna hodnota, čím sa simuluje, že procesory medzi sebou už nevymieňajú žiadne hodnoty. Okrem tohto môže byť poslaná ešte hodnota `STOP`, ktorá indikuje, že nižšie úrovňové uzly môžu byť ukončené, pretože ich hodnoty sú už na výstupe a nebudú potrebné. `STOP` značky zasiela vždy koreňový uzol svojim potomkom, tí pošlú túto značku opäť svojím potomkom až po listové uzly,
- v grafe znázornený uzol ako `procesor-otec` pošle svojím dvom synom hodnoty. Aké, to záleží od toho, čo mu poslal jeho otec.

Záver

Táto dokumentácia popisuje algoritmus Minimum extraction sort, ktorý bol úspešne implementovaný v paralelnom prostredí knižnice Open MPI s využitím jazyka C++. Po implementácií bola vykonávaná rada experimentov, ktoré mali porovnať teoretickú zložitosť so skutočnou. Pri väčšom počte zadaných procesorov algoritmus nedosahuje očakávané výsledky, kvôli tomu, že bol projekt testovaný na nízkom počte fyzických jadier, konkrétne na dvoch[1]. S menšími hodnotami nameraná zložitosť odpovedá očakávanej, pri vyšších počtoch procesorov, napríklad 63 alebo 127, zložitosť výrazne narastala.

[1] `http://ark.intel.com/products/67355/Intel-Core-i5-3210M-Processor-3M-Cache-up-to-3_10-GHz-rPGA`



Obr. 2: Komunikačný protokol implementovaný v tomto projekte pre paralelný výpočet znázorňuje komunikáciu medzi procesorom, jeho otcom a 2 potomkami v binárnom strome.