Implementace algoritmu "Enumeration sort"

PRL: 2. Projekt

9. 4. 2017

Jakub Pastuszek, xpastu00@stud.fit.vutbr.cz

Analýza algoritmu

Algoritmus „Enumeration sort“ pracuje s lineárním polem procesorů $p(n) = n + 1$, kde „n“ je počet prvků určených k seřazení a „$+1$“ značí poslední procesor, který načítá vstupní data a distribuuje je ostatním procesorům.

Optimální sekvenční algoritmus zvládne posloupnost hodnot seřadit v $t(n) = n.log\ n$ čase. Algoritmus „Enumeration sort“ z pohledu časové složitosti vypadá následovně. V první--distribuční fázi (viz Komunikační protokol – Distribution) je složitost $O(n)$ – procesor „$n+1$“ postupně zašle vstupní hodnoty do registru X každému z procesorů (každý má svojí hodnotu). Tuto fázi předchází ještě fáze nastavení počáteční hodnoty registru výsledné pozice C, která je v čase konstantní. V další fázi-řadící (viz Komunikační protokol – Sorting) je složitost $O(2n)$ – a to tak, že každá ze vstupních hodnot se musí postupně zaslat do registru Y na první procesor (n kroků) a poté se musí i ta poslední hodnota postupně zaslat až na poslední procesor (n-1 kroků). Poslední fáze – výstupní (viz Komunikační protokol – Output) má složitost $O(n)$ – nejprve se po fázi řazení musí přemístit hodnoty na správný procesor do registru Z, a to podle hodnoty v registru C – toto vyhodnocení, kam zaslat danou hodnotu se vyhodnotí v konstantním čase a následně se musí hodnoty postupně zaslat k prvnímu procesoru (který je zašle „$n+1$“ procesoru k výpisu), což trvá právě n kroků. Výsledkem je časová složitost $t(n) = O(4n)$.

Cena paralelního algoritmu je dána násobkem časové složitosti a počtu procesorů, v tomto případě $c(n) = O(n^2)$, což je horší než časová složitost optimálního sekvenčního řadícího algoritmu, proto není algoritmus „Enumeration sort“ optimální.

Implementace

Každý pracovní procesor (vyjma posledního „$n+1$“ procesoru) obsahuje 4 pracovní registry označené C, X, Y a Z. Počáteční hodnota registru C každého procesoru je 0, tato hodnota označuje počet vstupních hodnot, které jsou menší než hodnota registru X daného procesoru. Počáteční hodnotou ostatních registrů je hodnota taková, která se nemůže ve vstupní množině hodnot objevit (zavedeno pro rozpoznání, zda již hodnota do daného registru byla načtena).

Distribuční procesor (procesor „$n+1$“) postupně rozešle vstupní hodnoty každému procesoru do registru X, vždy však právě jednu danému procesoru. Při každém odeslání hodnoty do registru X se také zašle tato hodnota do registru Y prvnímu procesoru a pak také každý procesor, který má platnou hodnotu ve svém registru Y ji zašle svému následníkovi a novou hodnotu příjme od předchůdce (odeslání hodnoty neprování poslední „n“ procesor). Po obdržení hodnoty do registru Y každý procesor tuto hodnotu porovná s hodnotou v registru X a pokud je menší, tak registr C inkrementuje o jedna. Toto zasílání pokračuje do té doby,

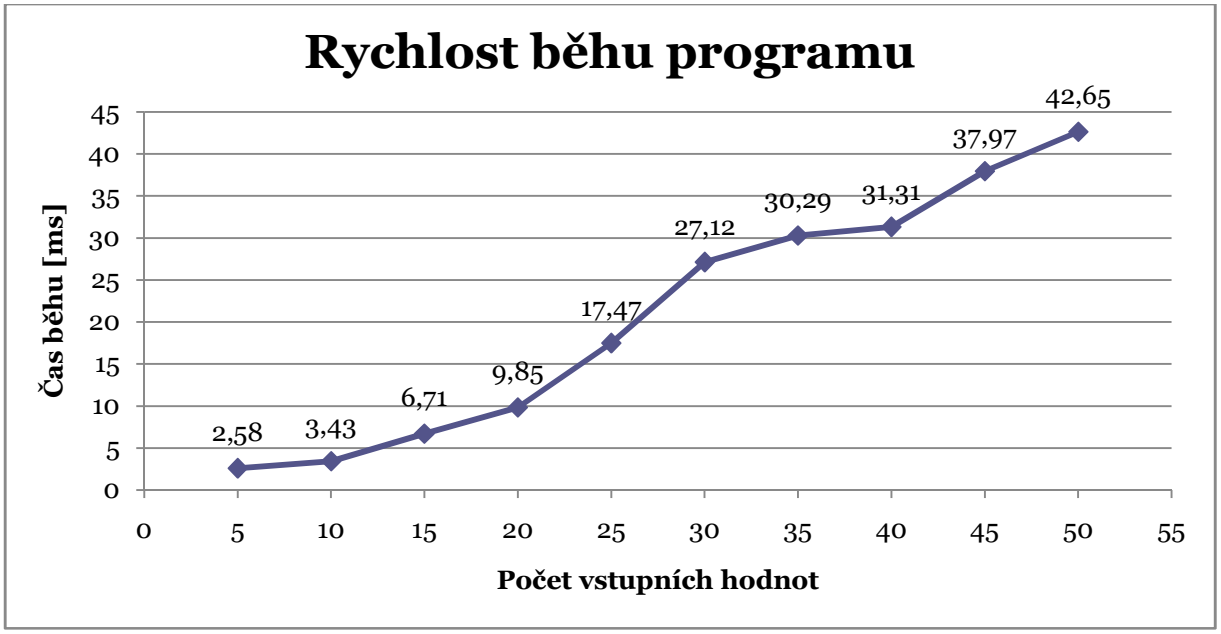
než se poslední vstupní hodnota postupně přepošle až poslednímu procesoru, který ji porovná se svojí hodnotou v registru X.

Po posledním porovnání hodnot má již každý procesor ve svém registru C hodnotu, která značí výstupní pozici hodnoty registru X mezi danými procesory. Poté již jen proběhne zaslání této hodnoty na daný procesor do registru Z. Po naplnění registru Z všech procesorů proběhne konečně výpis seřazené posloupnosti tak, že každý procesor zašle hodnotu svého registru Z předchozímu procesoru do registru Z, kdežto první procesor svou hodnotu zašle procesoru „*n+1*“, který následně danou hodnotu zapíše na výstup.

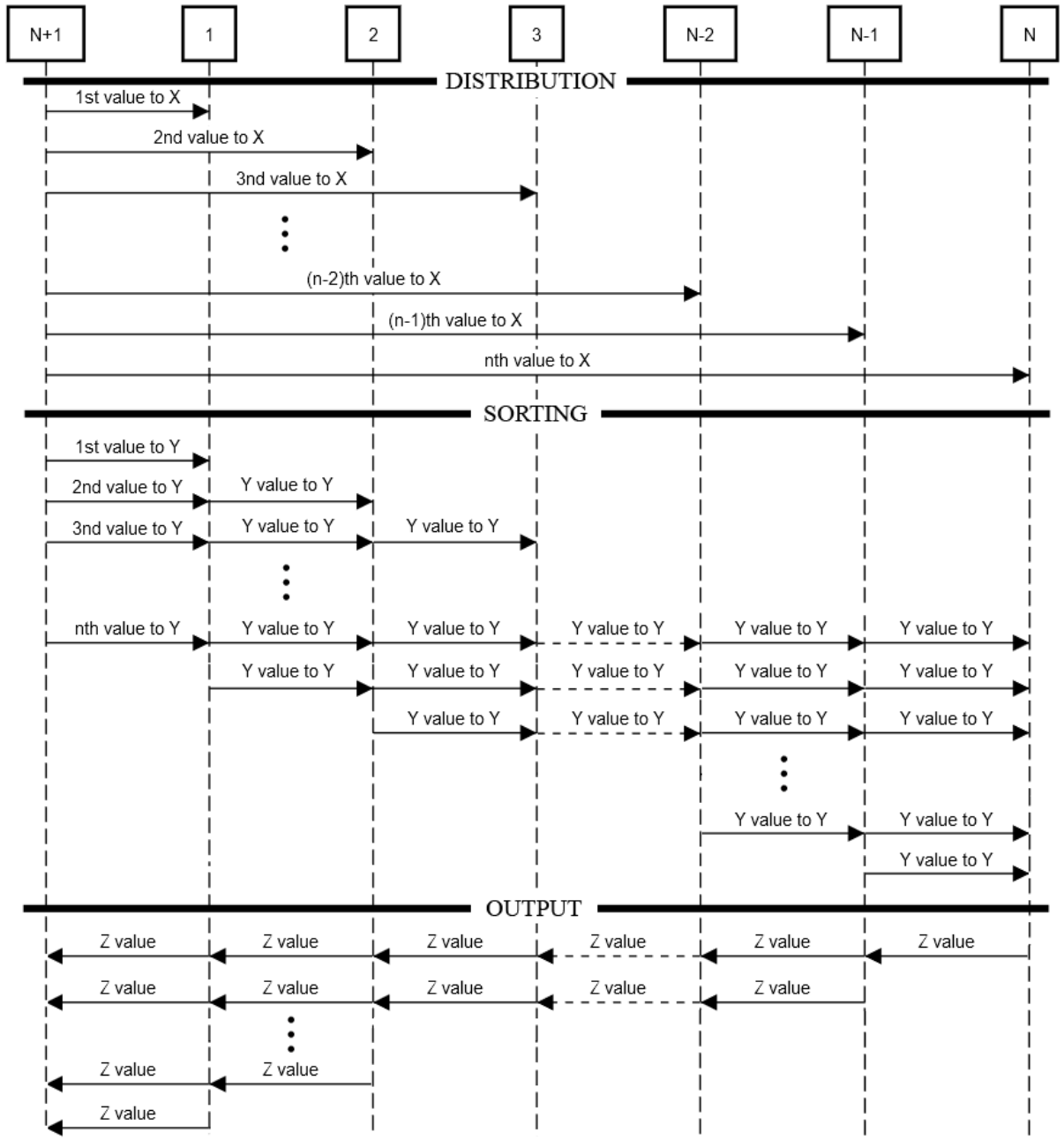
Experimenty

Testování doby běhu programu probíhalo v závislosti na velikosti vstupní množiny hodnot a to po krocích velikosti 5 od 5-ti do 50-ti hodnot. Celkem teda proběhlo 10 měření. Testování probíhalo na školním serveru Merlin.

Hodnoty naměřené při testování jsou v souladu s teoretickou časovou složitostí algoritmu, proto lze označit implementaci za správnou.



Komunikační protokol



Závěr

Projekt byl úvodem do oblasti paralelního programování a to za pomocí zasílání zpráv mezi procesory. Implementací a následným testováním algoritmu „Enumeration sort“ na „*n+1*“ procesorech se potvrdila jeho teoretická časová složitost t(n) = *O(n)*. Paměťově je tento algoritmus málo náročný, protože každý procesor si potřebuje pamatovat pouze 4 hodnoty. Úkony jednotlivých procesorů nejsou nikterak složité, procesor provádí pouze porovnávání dvou celočíselných hodnot a případně inkrementaci o jedna. Poté už přichází na řadu jen zasílání a přijímání zpráv.

Rychlost algoritmu není ovlivněna vstupními hodnotami, to znamená, že při stejně velké množině vstupních hodnot algoritmus vždy vykoná stejné kroky (viz sekvenční diagram), bez ohledu na to, zda jsou všechny hodnoty stejné, již seřazené nebo zcela náhodné.