Paralelní a distribuované algoritmy

2. projekt

Jakub Svoboda
xsvobo0z@stud.fit.vutbr.cz

18. 3. 2020

1. Teoretická složitost algoritmu

Výpočet časové složitosti algoritmu závisí na několika faktorech:

- Nejprve se každý lichý procesor spojí se svým (sudým) sousedem a zašle mu svoji hodnotu O(n).
- Sudé procesory porovnají své hodnoty O(n)
- Sudé procesory pošlou hodnotu nazpět O(n).

Ve druhém kroku se přechozí operace opakují. Celkově se jedná o posloupnost operací s lineární složitostí, složitost celého algoritmu lze tedy označit jako O(n).

Výpočet ceny lze pak provést následujícím výpočtem:

$t(n) = O(n)$

$p(n) = n$

$c(n) = t(n) * p(n) = O(n) * n = O(n^2)$,

kde t(n) je čas řešení úlohy v jednotkách (krocích), p(n) je počet procesorů potřebných k vyřešení úlohy v závislosti na velikosti vstupu n a c(n) je cena paralelního řešení úlohy.

Z výpočtu je patrné, že časová cena algoritmu je $O(n^2)$, což není optimální. Pro optimální řadící algoritmus by platila cena: $p(n) = 1$, $t(n) = O(n*\log n)$, $c(n) = O(n.\log n)$.

Prostorová složitost algoritmu je rovna velikosti vstupu (n), jelikož se proměnné pouze vyměňují na svých pozicích. Implementace algoritmu může vyžadovat využití lokálních proměnných, toto by nicméně nemělo změnit třídu prostorové složitosti.

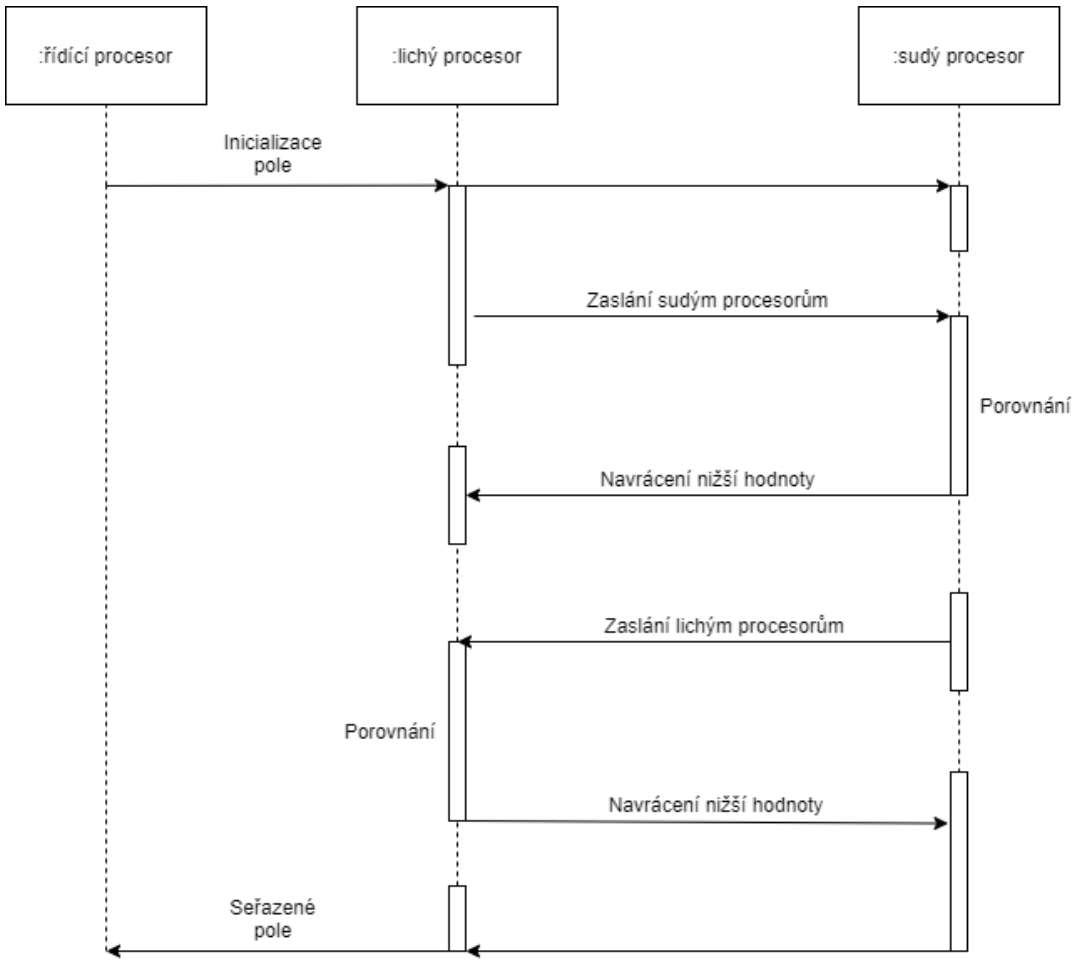
2. Implementace algoritmu

Algoritmus by naiplementován v jazyce C++. Program nejprve provede inicializaci MPI a každý z procesorů si uloží své ID. Proces s ID 0 je označen za řídící proces a při běhu programu pak řeší operace jako načtení vstupního řetězce čísel, shromáždění výsledků a výstup programu. Řídící proces zašle každému procesu číslo pro seřazení o velikosti jednoho Byte a zároveň vypíše celé neseřazené pole. Po přijetí zpráv procesory začíná samotný řadicí algoritmus.

Hlavní cylus provádí [n/2] iterací, kde výpočet v každé iteraci se dělí na dvě hlavní části. V první části nejprve liché procesory zašlou své číslo pomocí MPI zprávy procesoru se sousedním indexem. Tyto sudé procesory porovnají jejich číslo s přijatým číslem, větší z nich si uloží a nižší číslo pošlou MPI zprávou zpět lichému procesoru. V druhé části se výpočet opakuje, jen nejprve svá čísla zasílají sudé procesory a porovnávání provádějí liché procesory. Po [n/2] krocích je pole nutně seřazené.

Řídící proces následně po skončení hlavního cyklu přijímá zprávy od všech ostatních procesorů a vypisuje seřazené pole.

Výměnu zpráv lze ilustrovat následujícím diagramem:



3. Měření a analýza

Testování algoritmu proběhlo na severu Merlin. Pro ověření rychlosti algoritmu byl program modifikován, namísto vypisování seřazeného pole byl vypsán čas, po který běžel hlavní cyklus programu. Měření začátku a konce algoritmu bylo provedeno pomocí funkce MPI_Wtime(). Toto měření bylo provedeno pro vstupy o velikosti 1 až 15 prvků. Horní omezení bylo zvoleno z důvodu, že algoritmus vyžaduje stejný počet procesů, jako je velikost pole a sever Merlin při vyšším počtu procesů program často ukončí.

Pro každou velikost pole bylo měření provedeno dvacetkrát a výsledné časy byly následně zprůměrovány.

Z grafu je patrné, že i po opakovaném měření sice křivka vykazuje lineární závislost, nicméně naměřené hodnoty se často výrazně liší od očekávaných. Při měření se zjistilo, že časy jednotlivých řazení jsou značně proměnlivé, často se naměřené hodnoty lišily o několik řádů při spuštění s identickými parametry. Toto je nejspíše způsobeno rozdílným zatížením serveru při měření a také velmi nízkou třídou časové složitosti algoritmu, kdy řízení programu a režije MPI knihovny zřejmě může zabrat velký podíl z celkového času.

4. Závěr

V projektu byl naimplementován algoritmus odd-even transposition sort v jazyce C++, který je schopný efektivně řadit vstupní pole hodnot. Z měření vyplývá, že algoritmus odpovídá předpovídané složitosti, ačkoliv jednotlivé výsledky jsou obtížně měřitelné vzhledem ke zvolenému algoritmu a limitacím referenčního serveru.