Implementace algoritmu

Minimum Extraction sort

Autor: Anna Ostroukh (xostro00)
Datum: 30.03.2016

1. Zadání

Pomocí knihovny Open MPI implementujte v jazyce C/C++ algoritmus Minimum Extraction sort.

2. Popis algoritmu Minimum Extraction sort

Algoritmus pracuje nad binárním stromem. V listech se nachází řazené hodnoty. Celkem strom má n listů a 2n-1 procesorů. Nelistové procesory jsou schopny porovnávat hodnoty svých dětí, odebírat menší z nich a přeposílat svému rodiči. Pak v následujících krocích tato hodnota postupue výš až dosáhne kořene. Na to potřebuje (log n) + 1 kroků. Další hodnoty potřebují jen 2 kroky – jeden krok na porovnání kořenovým procesorem, druhý na uložení do výstupního seznamu seřazených hodnot. Po 2n + (log n) – 1 krocích všechny hodnoty budou ve výstupním seznamu.

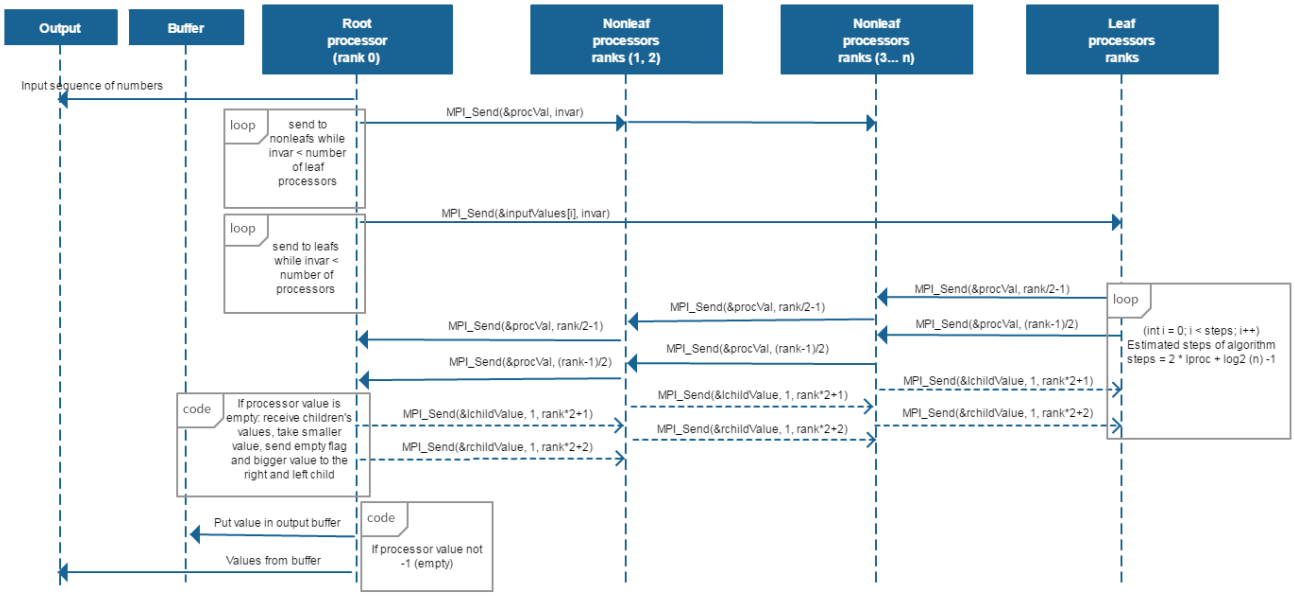
Teoretická složitost algoritmu

Jak bylo zmíněno výše, první hodnota bude ve výstupní posloupnosti po (log n) + 1 krocích (počet úrovní binárního stromu). Další n-1 hodnoty potřebují po 2 kroky, aby se dostaly do výsledné posloupnosti. Celkový počet kroků t(n) je tedy 2(n - 1) + (log n) + 1 = 2n + (log n) – 1. To znamena, že čas výpočtu algoritmu má lineární závislost na počtu hodnot vstupní posloupnosti, což je t(n) = O(n). Počet procesorů p(n) je 2n – 1.
Tedy cena c(n) = t(n)*p(n) = $O(n^2)$ – což není optimální.

3. Implementace algoritmu

Implementace algoritmu je provedena pomocí jazyka C++ a knihovny pro paralelní programování *Open MPI* (*Message Passing Interface*).
Algoritmus je schopen řadit vstupní posloupnost čísel, která není rovná mocnině dvojky. Proto musíme poznamenat listové procesory, které jsou prázdné.
Pomocí utility *dd* se provádí generování náhodných čísel v soubor s názvem *numbers.* Program ten soubor načítá a ověří, jestli obsahuje počet hodnot rovný mocnině dvojky. Pokud ne, doplní posloupnost hodnot na nejbližší vyšší hodnotu 2^n znakem prázdnosti, což je hodnota -1.
Procesor s rankem 0 (kořen stromu) načítá posloupnost náhodných čísel, dodá znaky prázdnosti pro listové procesory a rozešle hodnoty ostatním procesorům. Procesory provádí výpočet algoritmu v 2n + (log n) – 1 krocích, kde n je počet listových procesorů.

4. Komunikační protokol



Obrázek 1 – Sekvenční diagram implementovaného algoritmu

5. Experimenty

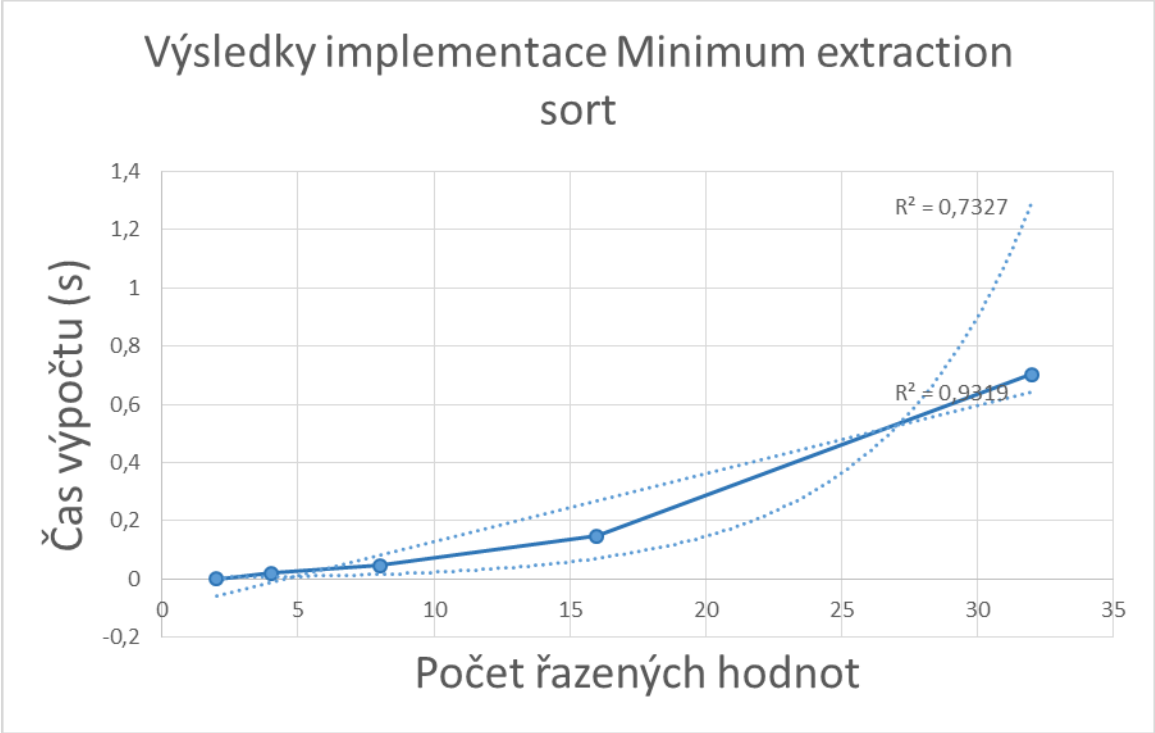
Pro měření výsledků použita standardní funkce knihovny MPI *MPI_Wtime()*.
Měření bylo provedeno jen pro samotné řazení. Čas na náčtení hodnot ze souboru *numbers* a rozesílání hodnot ostatním procesorům nezapočítán.

Měření bylo prováděno 10-krát pro každou hodnotu, pak byl najden aritmetický průměr.

V následující tabulce jsou uvedeny hodnoty času výpočtu algoritmu řazení pro různé počty vstupních hodnot.

Počet hodnot	Počet procesorů	Čas výpočtu (s)*
2	3	0,000710201
4	7	0,0202322
8	15	0,04728493
16	31	0,1482095
32	63	0,7019583

**Měření probíhalo na virtuálním stroji*



Obrázek 2 – Výsledky měření výpočtu algoritmu řazení

6. Závěr

Jak víme podle grafu na obrázku 2, graf experimentálních výsledků vypadá jako střední mezi exponenciální a lineární závislostí a je to proto, že ve skutečnosti nemáme tolik fyzických procesorů, kolik potřebujeme na řazení prvků. Ale jak je vidět po aproximaci trendy, výsledný graf je lepší aproximovatelný liněárním trendem co potvrzuje teoretickou složitost.