Rozbor a analýza algoritmu

Algoritmus pracuje na lineárním poli n procesorů, které jsou doplněny společnou sběrnicí, schopnou přenést v každém kroku právě jednu hodnotu. Každý procesor obsahuje 4 registry, jejichž význam je následující:

- Registr X_i – obsahuje prvek x_i
- Registr Y_i – postupně obsahuje prvky $x_1 – x_n$
- Registr C_i – obsahuje počet prvků, které byly menší než x_i
- Registr Z_i – obsahuje seřazený prvek Y_i

Fungování algoritmu lze rozdělit do 3 následujících samostatných kroků:

1) Všechny registry C se nastaví na hodnotu 1
2) Následující akce se opakují 2n krát $1 \leq k \leq 2n$:
   - Pokud vstup není vyčerpán, vstupní prvek x_i se pomocí sběrnice vloží do registru X_i a lineárním spojením do registru Y_1. Dále se obsah všech registrů Y posune doprava
   - Každý procesor s neprýzdnými registry X a Y je porovná. Pokud je X > Y, tak inkrementuje obsah registru C.
   - Pokud k > n (po vyčerpání vstupu), procesor P_{k-n} pošle sběrnicí obsah svého registru X procesoru P_{ck-n}, který jej uloží do svého registru Z.
3) V následujících n cyklech procesory posouvají obsah svých registrů Z doprava a procesor P_n produkuje seřazenou posloupnost

Analýza složitosti:

- 1. krok je proveden v konstantním čase c
- 2. krok je proveden v lineárním čase 2n
- 3. krok je proveden v lineárním čase n

Z toho plyne, že časová složitost algoritmu **$t(n) = c + 2n + n = O(n)$**.
Prostorová složitost algoritmu **$p(n) = n$**.
Cena algoritmu **$c(n) = t(n) \times p(n) = O(n) \times n = O(n^2)$**.

Implementace

Při implementaci jsem vycházel z výše popsaného algoritmu. Aplikace je spuštěna s n + 1 procesory, kde procesor s id 0 řídí celý výpočet. Řídící procesor nejprve zpracuje vstupní soubor a načtené hodnoty uloží do vectoru. Tyto hodnoty následně vypíše do jednoho řádku na stdout. Registry jsou reprezentovány datovým typem Int a současně každý procesor obsahuje dvě proměnné typu bool na ověření prázdnosti registrů X a Y.

První krok algoritmu je implementován jako prosté přiřazení hodnoty 1 do registru C. Toto přiřazení provede i řídící procesor, avšak na chování algoritmu to nemá žádný vliv.

Druhý krok je implementován jako smyčka, která se provede 2n krát. Pokud jsou registry X a Y neprázdné, tak se provede jejich komparace. Pokud je X větší, inkrementuje se hodnota registru C. Následně se provede posuv registru Y, kde každý procesor, který má hodnotu Y neprázdnou, odešle tuto hodnotu vedlejšímu procesoru s id o 1 větším (kromě posledního procesoru). Poté řídící procesor (pokud není vstup vyčerpán) odešle vstupní prvek x_i procesoru P_i, který uloží prvek do registru X a současně procesoru P_1, který uloží prvek do registru Y. Po vyčerpání vstupu a získání správné hodnoty

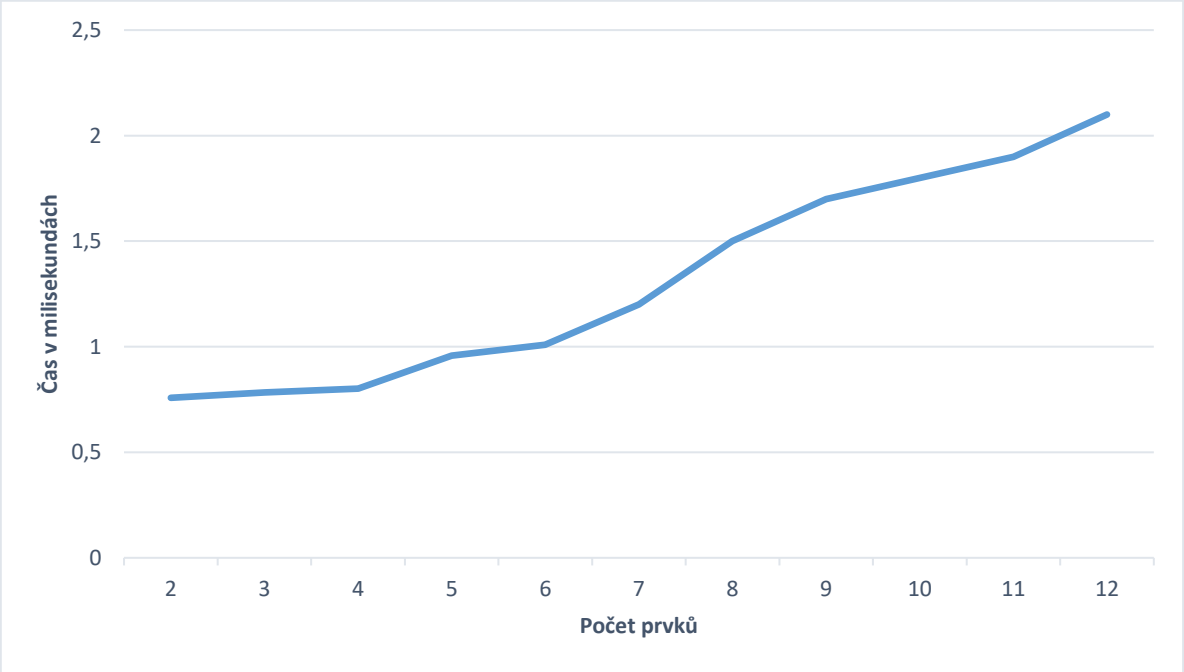
registru C následuje odeslání hodnot příslušným procesorům. V této části je využit pomocný registr zCount, díky kterému je algoritmus schopen setřídit stejné hodnoty. Tato část algoritmu funguje následovně:

1) Procesor s příslušným id odešle pomocí broadcast zprávu všem procesorům. Tato zpráva obsahuje id procesoru, který má obdržet hodnotu registru X.
2) Příjemce zprávy odpoví procesoru zpět a odešle obsah registru zCount, po odeslání registr zCount inkrementuje (hodnota registru zCount začíná na 0).
3) Procesor s příslušným id obdrží zprávu zpět a přičtě obdrženou hodnotu k registru C a pomocí broadcast odešle zprávu s novým příjemcem a tuto hodnotu příjemci posléze odešle.
4) Příjemce obdrží hodnotu a uloží ji do registru Z.

Třetí krok je implementován jako smyčka, která se provede n krát. Každý procesor odesílá hodnotu svého registru Z vedlejšímu procesoru (s id o 1 větším). Poslední procesor odesílá hodnotu registru Z řídícímu procesoru a ten tuto setříděnou posloupnost ukládá do vectoru. Nakonec řídící procesor vypíše setříděnou posloupnost na stdout.

Testování

Aplikaci jsem testoval pro vstupní data o velikostech 2 – 12 prvků. Pro každý procesor (kromě řídícího) jsem zaznamenal čas před zahájením algoritmu a po skončení algoritmu. Každý test jsem spustil vícekrát a spočítal průměr nejdelších časů, jenž byly potřebné k seřazení posloupnosti. Testování aplikace proběhlo na školním serveru merlin. Výsledek experimentování je zobrazen na následujícím grafu:

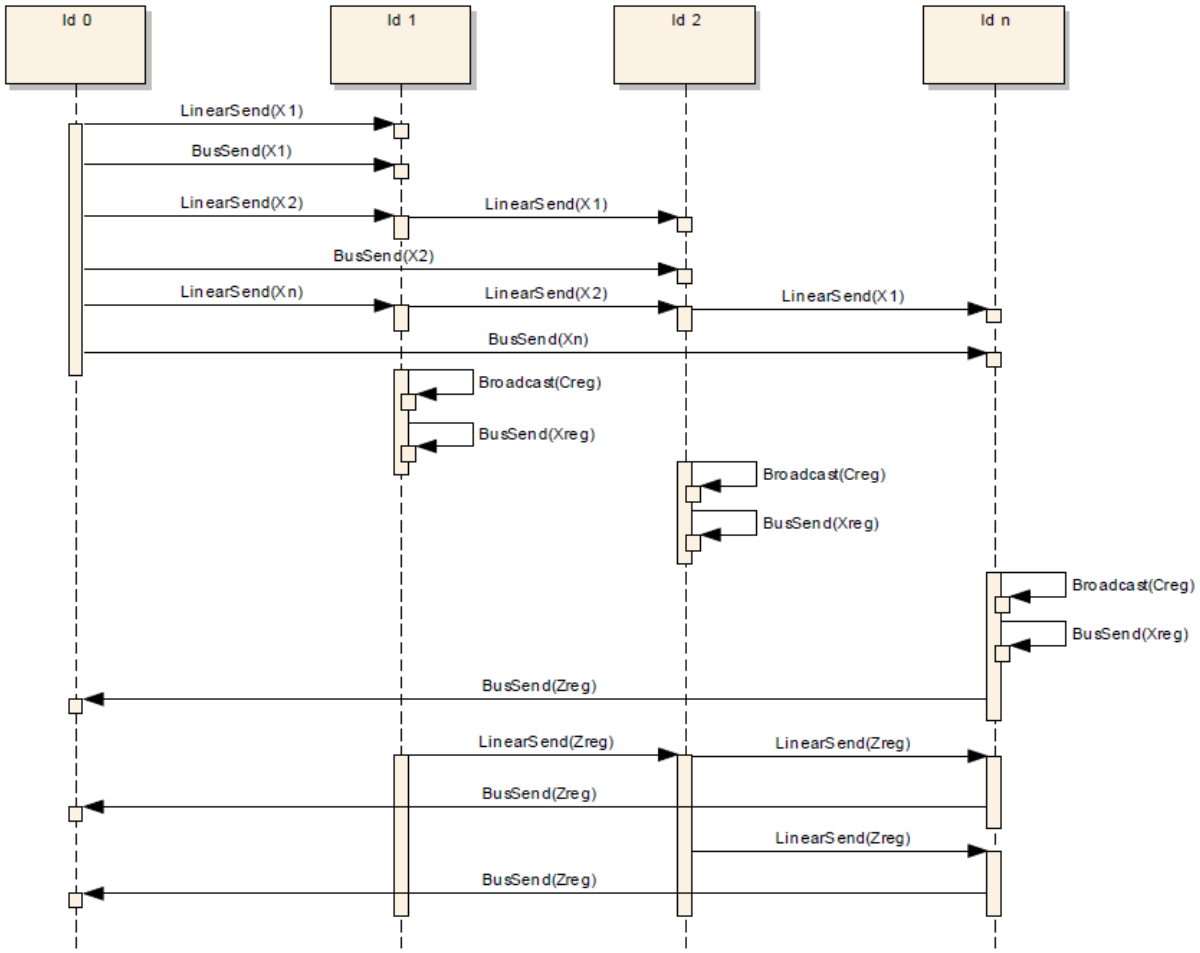


Obrázek 1: Doba běhu algoritmu v závislosti na počtu vstupních prvků

Závěr

Cílem projektu bylo analyzovat a implementovat paralelní řadící algoritmus Enumeration sort na lineárním poli n procesorů. Dle grafu (Obr. 1) je patrné, že čas potřebný k seřazení posloupnosti s přibývajícími prvky roste lineárně, a proto experimenty potvrdily teoretickou složitost. Komunikační protokol pro n procesů je přiložen v příloze.

Přílohy



Příloha 1: Sekvenční diagram znázorňující komunikační protokol