PRL Paralelní a distribuované algoritmy
FIT VUT Brno

Pipeline merge sort

8. dubna 2015

Jan Bednařík (xbedna45)

1 Úvod

V tomto dokumentu je popsána problematika řazení prvků pomocí algoritmu `Pipeline merge sort` na multiprocesorovém systému. Práce vznikla v rámci projektu v kurzu Paralelní a distribuované algoritmy (PRL), kdy bylo úkolem algoritmus implementovat v jazyce `C++` s použitím knihovny `OpenMPI` a experimentálně ověřit jeho předpokládanou časovou složitost.

2 Rozbor algoritmu

Algoritmus staví na lineární topologii procesorů s tím, že vstupní data postupně proudí od procesoru s nižším pořadovým číslem k procesoru s vyšším pořadovým číslem ve formě *zřetězené linky*. Každý procesor P_i (kromě procesoru P_0) vytváří seřazenou posloupnost délky 2^i ze dvou příchozích seřazených posloupností délky $2^{(i-1)}$, disponuje tak dvěma frontami délky $2^{(i-1)}$. Nejprve plní příchozími daty první frontu, následně začne plnit druhou frontu a současně obě fronty řadit a posílat na výstup příslušné hodnoty. Procesor P_0 pouze čte vstupní data a posílá je procesoru P_1, tedy nepotřebuje žádnou vstupní frontu.

Časová složitost Procesor S_i řadí dvě seřazené posloupnosti délky $d = 2^{(i-1)}$ a začne pracovat tehdy, jakmile od procesoru S_{i-1} přijme první prvek (tedy jakmile procesor S_{i-1} zapíše první prvek do své druhé fronty), což trvá $2d = 2*2^{i-1} = 2^i$ kroků. Vycházíme-li z předpokladu, že algoritmus běží na $P = log_2(N)+1$ procesorech (kde N odpovídá počtu řazených prvků), lze odvodit počet kroků k nutných k zahájení řazení na posledním procesoru s pořadovým číslem $i = P - 1$.

$$k = 2^i = 2^{P-1} = 2^{log_2(N)} = N \tag{1}$$

Časová složitost algoritmu je tedy *lineární*, $O(N)$.

Počet procesorů Počet kroků pro spuštění posledního procesoru, jenž má na výstupu seřazenou posloupnost, je 2^{P-1}, tedy lineární časové složitosti je dosaženo tehdy, je-li počet procesorů dán binárním logaritmem počtu vstupních prvků. K lineárnímu poli procesorů je přidán jeden další procesor, který čte vstupní posloupnost, tedy počet procesorů $P = log_2 N + 1$

Paměťová složitost Procesor i disponuje dvěma frontami délky $2^{(i-1)}$, tedy vyžaduje $2 * 2^{(i-1)} = 2^i$ paměťových jednotek. V případě posledního procesoru se jedná o celkem N jednotek,

v případě předposledního procesoru se jedná o $\frac{1}{2}N$ jednotek atd. Celkový počet paměťových jednotek dostaneme součtem této geometrické posloupnosti:

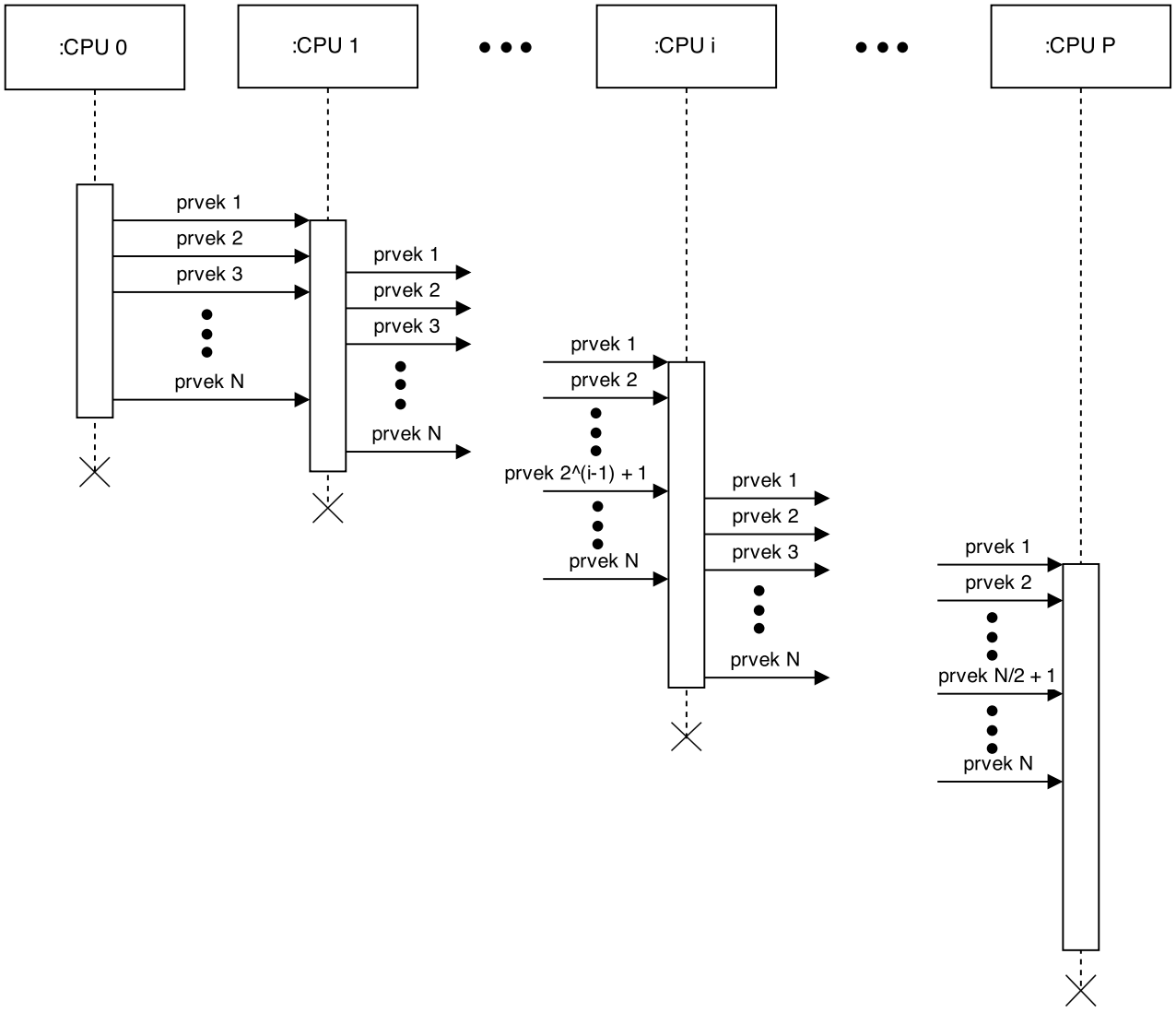
$$m = \frac{N}{1 - \frac{1}{2}} = 2N \tag{2}$$

Paměťová složitost je tedy *lineární*, $O(N)$.

Cena Cena algoritmu je dána jako násobek počtu procesorů a časové složitosti, tedy $C(N) = N * log_2 N$.

3 Implementace

Program je implementován v `C++` s použitím knihovny `OpenMPI`, jež staví na zasílání zpráv mezi procesory. Komunikační protokol použitý pro implementaci algoritmu *Pipeline merge sort* je znázorněný ve schématu 1.



Obrázek 1: Komunikační protokol procesorů vyjádřený sekvenčním diagramem pro obecný počet N vstupních prvků a P procesorů.

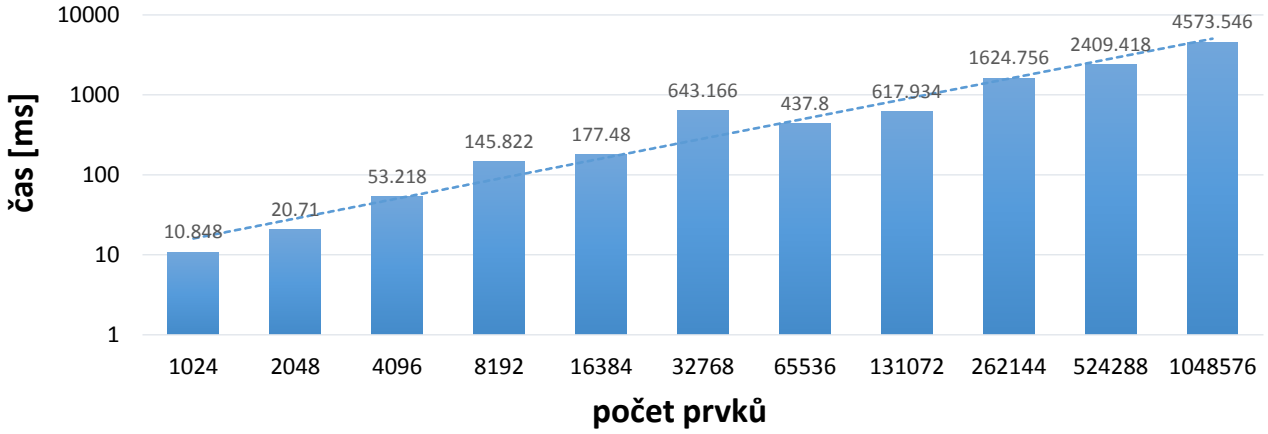
Za účelem přehledné implementace byla oproti teoretickému odvození využita paměťová složitost dvojnásobná. Každému procesoru jsou namísto dvou front přiřazeny dvě dvojice front. Dokud procesor provádí řazení nad jednou dvojicí front, zapisuje se do druhé dvojice, a obráceně.

4 Experimenty

Běh algoritmu byl otestován na referenčním serveru `merlin` vybaveným 6jádrovým CPU (12 vlákny). Testovány byly posloupnosti délky $N = 2^i, i \in \mathbb{N}, 10 \leq i \leq 20$, kde počet použitých procesorů $P = log_2 N + 1$. Čas běhu staví na knihovní funkci `MPI::Wtime()` pro přesné měření času. Měření provádí pouze první procesor, kdy spouští časovač po načtení všech vstupních dat a před zasláním prvního prvku druhému procesoru. Jakmile poslední procesor zpracuje poslední hodnotu, zašle zprávu o dokončení řazení prvnímu procesoru, kterýžto po jejím přijetí zastaví časovač. Doba běhu tak není zatížena vstupně výstupními operacemi.

Testovací hardware musí fyzické procesory ve většině případů emulovat a testy byly navíc prováděny za běžného provozu serveru. Pro dosažení co nejvěrnějších výsledků tak byl pro každou délku posloupnosti N test proveden desetkrát, z výsledků byly odstraněny dvě nejnižší a tři nejvyšší hodnoty a zbylých pět hodnot bylo průměrováno.

S použitím logaritmického měřítka časové osy a s ohledem na teoretickou časovou složitost $O(N)$ je očekáván lineární průběh závislosti doby běhu programu na počtu vstupních hodnot. Jak je patrné z grafu 2, přes nepřesnosti způsobené nerovnoměrným vytížením testovacího stroje je závislost doby běhu na čase vskutku lineární.



Obrázek 2: Závislost doby běhu programu na počtu vstupních prvků.

5 Závěr

V rámci projektu byl implementován algoritmus *Pipeline merge sort*, byla odvozena jeho teoretická časová a paměťová složitost i cena a provedeny experimenty nad reálnými daty. Testy potvrdily lineární časovou složitost algoritmu, přesto byly zaznamenány nezanedbatelné odchylky. Ty jsou způsobené především proto, že testovací stroj nedisponuje dostatečným počtem procesorů (jader procesoru), tedy jednotlivé procesory požadované algoritmem jsou nahrazeny procesy sdílejícími výpočetní prostředky a neběží tak zcela paralelně. Testování bylo navíc prováděno za běžného provozu testovacího stroje, tudíž jsou výsledky ovlivněny aktuálními výpočetními požadavky ostatních uživatelů.