Paralelní a distribuované algoritmy – dokumentace
Paralelní celulární automat

Bc. Jaroslav Sendler, xsendl00
xsendl00@stud.fit.vutbr.cz

15. dubna 2012

Dokumentace k 2.projektu do předmětu Paralelní a distribuované algoritmy (PRL). Obsahuje popis zadání, rozbor a analýzu Paralelního celulárního automatu. V závěru dokumentu se nachází komunikační protokol mezi „procesory“ (způsob zasílání zpráv). Pro vizualizaci je využit sekvenční diagram.

1 Zadání

Pomocí knihovny Open MPI implementujte celulární automat, který bude využívat paralelního prostředí pro urychlení výpočtu. Celulární automat bude implementovat pravidla hry *Game of life*.

Vstup: Soubor „lattice“ reprezentující mřížku automatu a obsahující binární číslice 0,1, kde 0 znamená mrtvou a 1 znamená živou buňku. Číslice 0 a 1 budou uspořádány do obdélníkové matice, kde každý řádek bude zpracováván právě jedním procesorem (navíc můžete použít jeden řídící/synchronizační procesor). Zároveň můžete počítat s tím, že všechny řádky jsou stejně dlouhé. Následuje příklad, jak vypadá soubor *lattice*. Příkladem budiž:

```
00000000
00111000
01110000
00000000
```

Výstup: Na standardní výstup vypište stav matice po požadovaném počtu kroků a to tak, že každému řádku bude předcházet id procesoru a dvojtečka. Tento formát je zvolen z toho důvodu, že procesory budou hodnoty vypisovat v náhodném pořadí (pro seřazení výstupu použijte utilitu *sort* ve spouštěcím skriptu, nezapomeňte vyřešit dvoumístná id procesorů) Přesný formát výstupu je opět nutno dodržet kvůli strojové kontrole výstupu. Za nedodržení budou strhávány body. Příklad výstupu po 3 krocích:

```
0:00010000
1:01001000
2:01001000
3:00100000
```

Postup: Vytvořte testovací skript se jménem *test* nebo *test.sh*. Skript přijímá právě jeden parametr a to počet kroků. Skript spočte počet řádků (aby bylo jasné, kolik je třeba procesorů), přeloží a spustí program s parametrem *pocet_kroku*. Je vhodné spočíst i počet sloupců a předat ho programu kvůli načítání souboru (každý procesor si pak může načíst vlastní část souboru hodnot - vlastní řádek). Po načtení (ideálně paralelně - každý procesor svůj řádek) hodnot je proveden zadaný počet iterací podle předaného parametru a nakonec jsou na standardní výstup vypsány řádky jednotlivých procesorů. Vzhledem k tomu, že použijete utilitu *sort*, řádky budou seřazeny správně, jak mají být v matici.

2 Rozbor a analýza algoritmu

Algoritmus pracuje s celulárním automate, který implementuje pravidla hry Game of life. Hra pracuje s matici o rozměrech N x M, kde N je počet řádků a M počet sloupců. Počet procesorů, které se podílejí na zpracování hry je určen na základě proměnné N ($N >= 1$). Tedy každý řádek je zpracováván jiným CPU.

Základní body algoritmu: •

- Každý procesor uchovává jeden řádek hracího pole.
- Všechny procesory pracují zároveň.
- Každý procesor posílá svůj řádek vedlejšímu procesoru.
- Každý procesor přijímá řádek od vedlejšího procesoru.
- Všechny procesory startují i končí po stejném kroku.
- Celý algoritmus skončí po K krocích.

Rozbor algoritmu

Každý procesor obsahuje dvě fronty s maximální délkou M, kde M je délka hracího pole ($M >= 1$). První fronta obsahuje aktuálně zpracovávaný řádek a druhá slouží pro ukládání již vypočtených nových hodnot jednotlivých buněk. Pro příjem řádku od spodního/vrchního souseda slouží další jedna/dvě pole.

Implementace pravidel hry je rozděleno do několika následujících částí:

- Prní procesor (1) a poslední procesor (N)
  - pošle řádek *[konstantní čas]*
  - přijme řádek *[konstantní čas]*
  - zpracování první buňky (3-okolí) *[proměnný čas]*
  - zpracování poslední buňky (3-okolí) *[proměnný čas]*
  - zpracování zbylých buněk (5-okolí)
    - loop(M-2) zpracuj buňku *[proměnný čas]*
- Zbylé procesory (2-(N-1)-tý)
  - pošle řádek (2x) *[konstantní čas]*
  - přijme řádek (2x) *[konstantní čas]*
  - zpracování první buňky (5-okolí) *[proměnný čas]*
  - zpracování poslední buňky (5-okolí) *[proměnný čas]*
  - zpracování zbylých buněk (8-okolí)
    - loop(M-2) zpracuj buňku *[proměnný čas]*
- uložení nového řádku *[konstantní čas]*

Rozbor celé aplikace je možné vyjádřit pomocí následující odrážek:

- inicializace *[konstantní čas]*
- načtení dat *[proměnný čas]*
- pravidla hry *[proměnný čas]*
- tisk jednotlivých řádků po K-krocích *[konstantní čas]*
- úklid *[konstantní čas]*

Částí označené *[proměnný čas]* jsou závislé na vstupních hodnotách, naopak části *[konstantní čas]* nikoliv. Čas a cena algoritmu závisí na následujících proměnných:

- počet kroků
- počet sloupců hracího pole
- počet řádků hracího pole (ovlivňuje cenu)

3 Složitost algoritmu

Prostorová složitost se odvyjí od velikosti vstupní matice. Uvažujeme-li matici NxM (N řádky, M sloupce) pak $p(n) = N$, jelikož každý řádek je načten a zpracováván právě jedním procesorem. Časová složitost výpočtu je ovlivněna jak vstupní maticí, přesněji počtem sloupců M a počtem průchodů S, neboli počtem kroků hry. Tedy časovou složitost lze vyjádřit jako součin počtu sloupců a počet kroků ve hře.

časová složitost: $\mathbf{t}(n) = \mathbf{O}(S * M)$, což je třída složitosti $\mathbf{O}(n)$

cena: $t(n).p(n)$, tedy $\mathbf{c}(n) = \mathbf{O}(N * S * M)$

4 Naměřené hodnoty

V tabulce 1 je zobrazena závislost mezi počtem vstupních prvků a časem potřebným k jejich seřazení. Výsledky byly zjištěny při experimentování s posloupnostmi různých délek. Pro každou hodnotu bylo provedeno 10 měření a následně udělán průměr. Měření probíhalo na školním serveru Merlin pomocí příkazu *time* při vypnutých výpisech.Do výsledných hodnot není započítáno generování vstupních prvků příkazem *dd*.

počet prvků	2	4	8	16	32	64	128	256	512
čas[s]	0,066	0,105	0,145	0,171	0,176	0,186	0,205	0,324	0,386

počet prvků	1024	2048	4096	8192	16384	32768
čas[s]	0,441	0,772	1,176	2,379	4,715	4,921

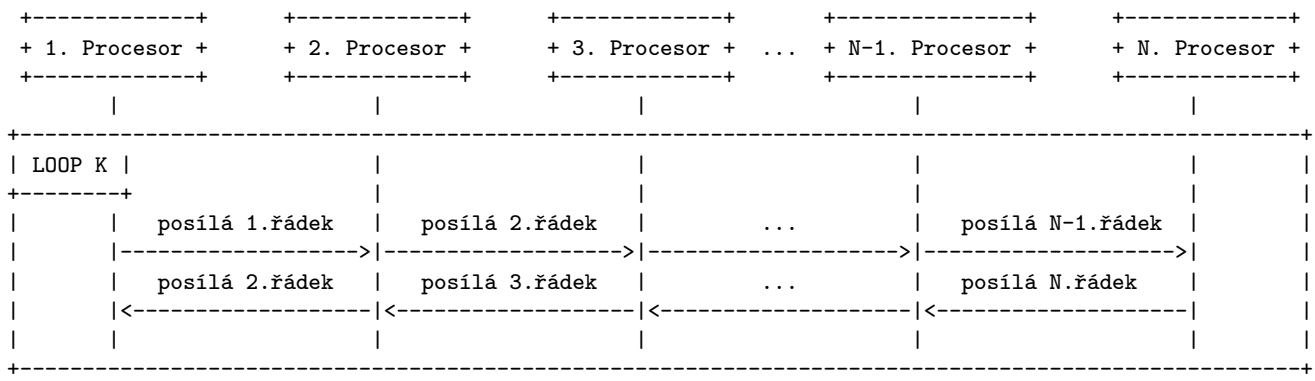
Tabulka 1: Přehled naměřených časů v závislosti na počtu prvků.

Na obrázku 1 je graficky znázorněn vztah mezi počtem prvků a časem potřebným k jejich seřazení.

5 Komunikační protokol

Na obrázku 1 zobrazeném níže je pomocí sekvenčního diagramu znázorněna komunikace mezi jednotlivými procesory. Celkový počet zasílaných zpráv se odvyjí od počtu kroků, které se v algoritmu provedou. To je znázorněné pomocí smyčky *LOOP K*, kde K je počet kroků. Procesory od 2 až po N-1 zasílají 2 zprávy a také 2 zprávy přijímají. První a poslední procesor, jelikož jsou krajní zasílají i přijímají pouze zprávu jednu.

V označení „posílá X. řádek“ X reprezentuje číslo řádku ze zdrojového soubor (jak řádky tak i procesory jsou číslovány od 1).



Obrázek 1: Příklad komunikace N procesorů v paralelním celulárním automatu.