3. projekt pro předmět PRL 2012
Paralelní celulární automat (Game of life)

Mikulka Jiří
<xmikul39@stud.fit.vutbr.cz>

25. dubna 2012

1 Zadání projektu

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

1.1 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.

1.2 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.

1.3 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

Celulární automat (CA) je dynamický systém a matematický model, který modeluje živý systém. Mezi jeho vlastnosti patří – (1) pracuje v diskrétním prostoru a čase (fyzikální veličiny nabývají diskrétních hodnot z konečné množiny), (2) je tvořen buňkami, (3) (v případě hry *Game of life*) jsou buňky uspořádány do dvourozměrné mřížky, (4) každá buňka může nabývat právě jednoho ze dvou možných stavů (1 – živá buňka, 0 – mrtvá buňka), (5) hodnoty stavů buněk se určují podle lokální přechodové funkce, která je totožná pro všechny buňky systému, (6) každá buňka nese informaci o sobě samé, ale také i o svém okolí, (7) (v případě hry *Game of life*) je uvažováno Mooreovské okolí buňky (tedy 8 sousedů).

Hlavními rysy celulárního automatu ve hře *Game of life* spočívají ve třech vlastnostech – (1) *paralelismus* (výpočty ve všech buňkách probíhají zároveň, nikoli sériově), (2) *lokalita* (nový stav závisí na aktuálním stavu buňky a stavu okolních buněk) a (3) *homogenita* (všechny buňky používají stejnou lokální přechodovou funkci).

Hra *Game of life* je výjimečná tím, že ji nehraje žádný hráč; jediná interakce s uživatelem je počáteční nastavení celulárního automatu (resp. hrací plochy). Pro tuto hru platí stejné vlastnosti a rysy, jež byly uvedeny výše pro obecné celulární automaty. Oficiální pravidla této hry jsou následující:

- pro živou buňku (tj. stav buňky je 1)
  - pokud má buňka ve svém okolí méně než 2 živé buňky, pak umírá (tj. mění stav na 0) na osamělost
  - pokud má buňka ve svém okolí více než 3 živé buňky, pak umírá (tj. mění stav na 0) na „přemnožení“
  - pokud má buňka ve svém okolí 2 nebo 3 živé buňky, pak zůstává žít (tj. její stav zůstává 1)
- pro mrtvou buňku (tj. stav buňky je 0)
  - pokud má buňka ve svém okolí 3 živé buňky, pak ožívá (tj. mění stav na 1)

V implementaci jsem výše uvedená pravidla shrnul do této podmínky „živosti“ buňky:

- pokud má buňka ve svém okolí 3 živé buňky, nebo pokud má živá buňka ve svém okolí 2 živé buňky, pak buňka žije
- jinak buňka umírá

2.1 Implementace algoritmu

Hra *Game of life* byla implementována v jazyce `C++` s využitím knihovny `OpenMPI`. Vstupní soubor *lattice*, jak již bylo uvedeno v 1, obsahuje `rows` řádků stejné délky `columns`. Paralelismus je implementován tak, že každý procesor má na starosti právě jeden řádek ze vstupního souboru, celkem je tedy potřeba `rows` procesorů; žádný další řídící procesor není potřeba. Testovací skript `test.sh` nejprve spočítá počet řádků a sloupců v souboru `lattice` a poté s patřičnými parametry spustí samotnou aplikaci provádějící *Game of life*. Chování každého procesoru lze popsat následujícím algoritmem:

```
1) inicializace MPI
   - získání ID procesoru proc_id
   - získání celkového počtu procesorů procs_count
2) načtení parametrů spuštění
   - počet řádků rows
   - počet sloupců columns
   - počet iterací (tj. kroků hry Game of life) iterations
3) načtení proc_id-tého řádku ze souboru lattice
   - tato operace je provedena otevřením souboru lattice v binárním módu
     a přesunem čtecí hlavy na pozici proc_id * (columns + 1) a načtením columns znaků
   - tuto operaci provádějí všechny procesory paralelně
     a každý přečte právě jeden řádek a to svůj
4) v každé iteraci (počet je dán parametrem iterations) každý procesor provádí
   - vynuluje pomocné buffery (vlastní, kopie vlastního, nad, pod)
   - procesor se sudým ID
     - je nutné ošetřit první a poslední řádek
     - odešle svůj řádek procesorům "nad" a "pod"
     - přijme řádky procesorů "nad" a "pod"
   - procesor s lichým ID
     - je nutné ošetřit první a poslední řádek
     - přijme řádky procesorů "nad" a "pod"
     - odešle svůj řádek procesorům "nad" a "pod"
   - pro každý sloupec (tj. columns iterací) se provádí
     - spočítá se počet živých buněk v okolí (nutné zde pracovat s kopií řádku!)
     - na základě pravidel hry Game of life je v bufferu změněn stav buňky
5) procesor vytiskne svůj řádek po iterations iteracích
6) procesor se ukončí
```

2.2 Složitost implementace algoritmu

Všechny procesory provádějí výše uvedený algoritmus paralelně, včetně čtení ze souboru. Z toho se také odvíjí celková časová (i paměťová) složitost paralelního algoritmu. Celková časová složitost (bude uvedena jen horní asymptotická složitost) se tedy skládá z inicializace programu, otevření souboru, přesunu čtecí hlavy v souboru a načtení daného úseku souboru a poté samotná hra *Game of life*. Do celkové složitosti nejsou započítány rutinní operace, jako je inicializace prostředí, neboť předpokládáme, že jsou prováděny v konstantním čase $O(1)$.

Uvažujme, že soubor `lattice` má n řádků délky l. Uvažujme dále, že budeme sledovat i iterací hry.

- každý procesor p_j z $p_0, p_1, \ldots, p_{n-1}$ otevře soubor `lattice`, přesune čtecí hlavu na pozici $j \cdot (l+1)$ a načte l znaků (tedy jeden řádek) v čase $O(j \cdot (l+1) + l) = O(j \cdot l + j + l) = O(l \cdot (j+1) + j) = O(l)$
- v i iteracích si procesor nejprve se svými sousedy vymění řádky (tj. $O(2 \cdot l + 2 \cdot l)$) a poté pro každou buňku ve svém řádku spočítá počet živých buněk v okolí a na základě tohoto počtu změní její stav (tj. $O(l \cdot (8+1))$), celkově to bude v čase $O(i \cdot (2 \cdot l + 2 \cdot l + l \cdot (8+1))) = O(13 \cdot i \cdot l) = O(i \cdot l)$
- a poté procesor vytiskne svůj řádek v čase $O(l)$

Celková časová složitost algoritmu je pak dána součtem těchto složitostí, tedy $O(l + i \cdot l + l) = O(k \cdot l) = O(l)$. Z výpočtu složitosti algoritmu vyplývá, že má *lineární složitost* vzhledem k délce řádků (resp. počtu sloupců). Cena algoritmu je pak dána vztahem $c(n) = p(n) \cdot t(n) = n \cdot O(l) = O(n \cdot l)$, což je *optimální*.

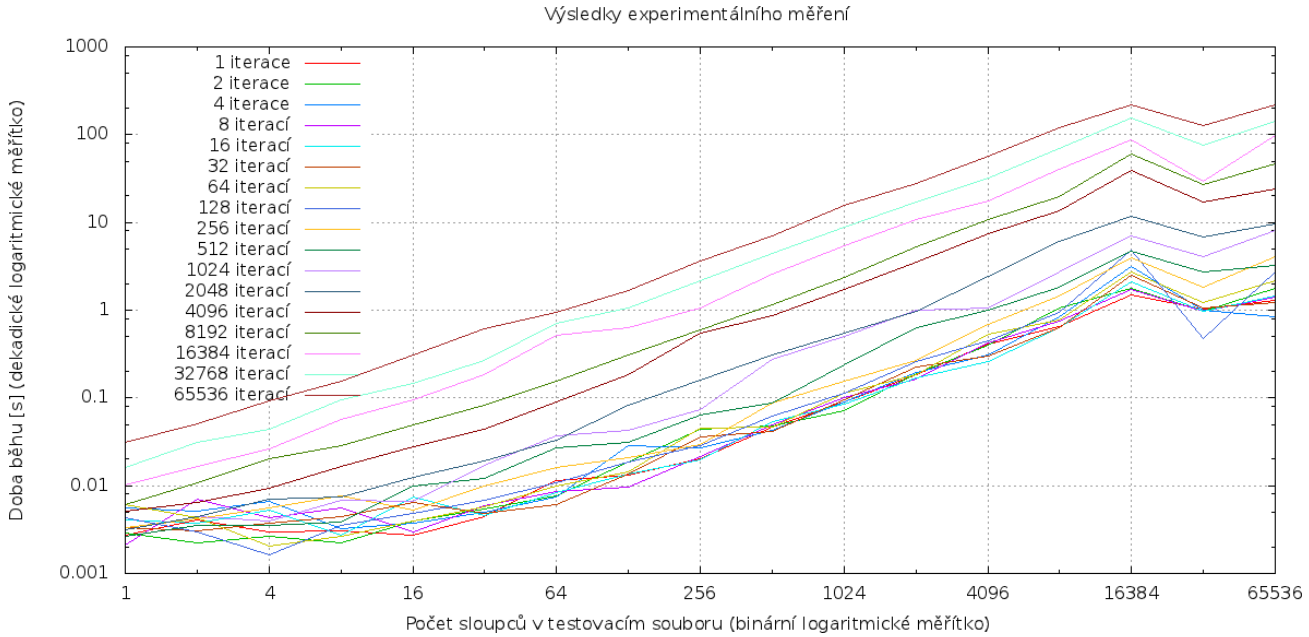
2.3 Experimentálně získané výsledky

Při testování této paralelní implementace hry *Game of life* byla snaha co nejméně zatěžovat systém množstvím vytvořených vláken (tedy jednotlivých procesorů v MPI), proto všechny testovací soubory byly vytvářeny s pevným a relativně malým

počtem řádků (při testování byly použity soubory s právě 4 řádky). Testování se tedy zaměřovalo na různou délku řádků (od 2^0 do 2^{16}) a různý počet iterací (opět od 2^0 do 2^{16}). Doba činnosti jednoho procesoru byla měřena pomocí funkce `MPI_Wtime` – na začátku činnosti procesoru (tzn. ihned po inicializaci) byl zaznamenán aktuální čas a na konci (tzn. mezi výpisem a `MPI_Finalize`) byl opět zaznamenán aktuální čas; rozdíl těchto časů pak dal délku běhu procesoru. Přestože všechny procesory by měly teoreticky běžet stejnou dobu, není tomu tak kvůli přepínání kontextu a činnosti operačního systému. Proto bylo před oběma čteními aktuálního času volána funkce `MPI_Barrier`, která způsobila, že v daném místě se všechny běžící procesory „počkaly“, a tudíž jejich doba běhu byla přibližně stejná. Celková doba běhu programu byla dobou nejdéle běžícího procesoru. Aby byly eliminovány různé vlivy plánování operačního systému či zatížení fyzického procesoru, bylo měření nad každým souborem `lattice` prováděno 4× a z výstupních hodnot byl vypočten aritmetický průměr. Výsledky experimentování jsou shrnuty v následující tabulce a grafu. V tabulce řádky představují různé počty iterací (či kroku hry), sloupce pak představují délku řádků.

Počet iterací \ Počet sloupců na řádku	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536
1	0,0027	0,0029	0,0056	0,0021	0,0040	0,0033	0,0061	0,0043	0,0033	0,0026	0,0032	0,0032	0,0051	0,0060	0,0101	0,0162	0,0310
2	0,0041	0,0022	0,0051	0,0071	0,0037	0,0030	0,0043	0,0030	0,0042	0,0035	0,0044	0,0044	0,0064	0,0108	0,0165	0,0316	0,0506
4	0,0030	0,0026	0,0066	0,0043	0,0052	0,0037	0,0020	0,0016	0,0056	0,0035	0,0040	0,0070	0,0094	0,0202	0,0263	0,0444	0,0918
8	0,0030	0,0022	0,0032	0,0056	0,0027	0,0044	0,0026	0,0035	0,0076	0,0039	0,0068	0,0075	0,0165	0,0282	0,0568	0,0948	0,1566
16	0,0027	0,0040	0,0037	0,0029	0,0074	0,0065	0,0040	0,0049	0,0052	0,0099	0,0065	0,0123	0,0282	0,0487	0,0940	0,1480	0,3072
32	0,0044	0,0055	0,0050	0,0059	0,0045	0,0048	0,0057	0,0069	0,0100	0,0120	0,0170	0,0193	0,0442	0,0825	0,1822	0,2682	0,6162
64	0,0114	0,0077	0,0074	0,0087	0,0084	0,0061	0,0098	0,0107	0,0161	0,0267	0,0369	0,0331	0,0891	0,1531	0,5113	0,7051	0,9515
128	0,0132	0,0186	0,0290	0,0095	0,0134	0,0136	0,0145	0,0184	0,0211	0,0315	0,0423	0,0835	0,1832	0,3101	0,6393	1,0463	1,6835
256	0,0204	0,0445	0,0269	0,0216	0,0198	0,0360	0,0446	0,0283	0,0295	0,0633	0,0730	0,1598	0,5410	0,6045	1,0654	2,1756	3,6348
512	0,0464	0,0475	0,0424	0,0492	0,0538	0,0410	0,0464	0,0619	0,0869	0,0867	0,2780	0,3056	0,8766	1,1598	2,6102	4,4338	7,1187
1024	0,0893	0,0709	0,0902	0,1003	0,0861	0,0942	0,1137	0,1116	0,1531	0,2394	0,5042	0,5513	1,7017	2,3423	5,3951	8,7347	15,5535
2048	0,1854	0,1909	0,1954	0,1642	0,1705	0,2263	0,1840	0,2570	0,2657	0,6263	1,0033	0,9625	3,5609	5,2860	10,7959	16,9541	28,0701
4096	0,4108	0,3967	0,3084	0,4291	0,2578	0,2979	0,5373	0,4514	0,6963	0,9867	1,0521	2,4354	7,5095	10,8936	17,5838	31,8234	57,6319
8192	0,6445	1,0494	0,8479	0,7448	0,6382	0,6391	0,7637	0,9327	1,4561	1,8275	2,7240	6,0653	13,7606	19,6353	40,5406	68,6991	120,5428
16384	1,4961	1,7953	3,1846	1,7004	2,0790	2,5193	2,7431	4,7972	3,9136	4,7323	7,0765	11,672	39,0492	60,8488	86,9548	157,1660	217,8838
32768	1,0198	0,9730	0,9882	0,9786	0,9597	1,0556	1,2319	0,4772	1,8467	2,7560	4,0904	6,7278	17,2133	26,7728	29,6465	76,3103	127,9073
65536	1,3077	1,7492	0,8378	1,4291	1,4541	1,2107	2,1385	2,7250	4,0998	3,2303	8,1772	9,7263	23,8404	45,9860	99,2535	143,2015	221,9182

Tabulka 1: Tabulka závislosti času na počtu sloupců na jednom řádku pro různý počet iterací.

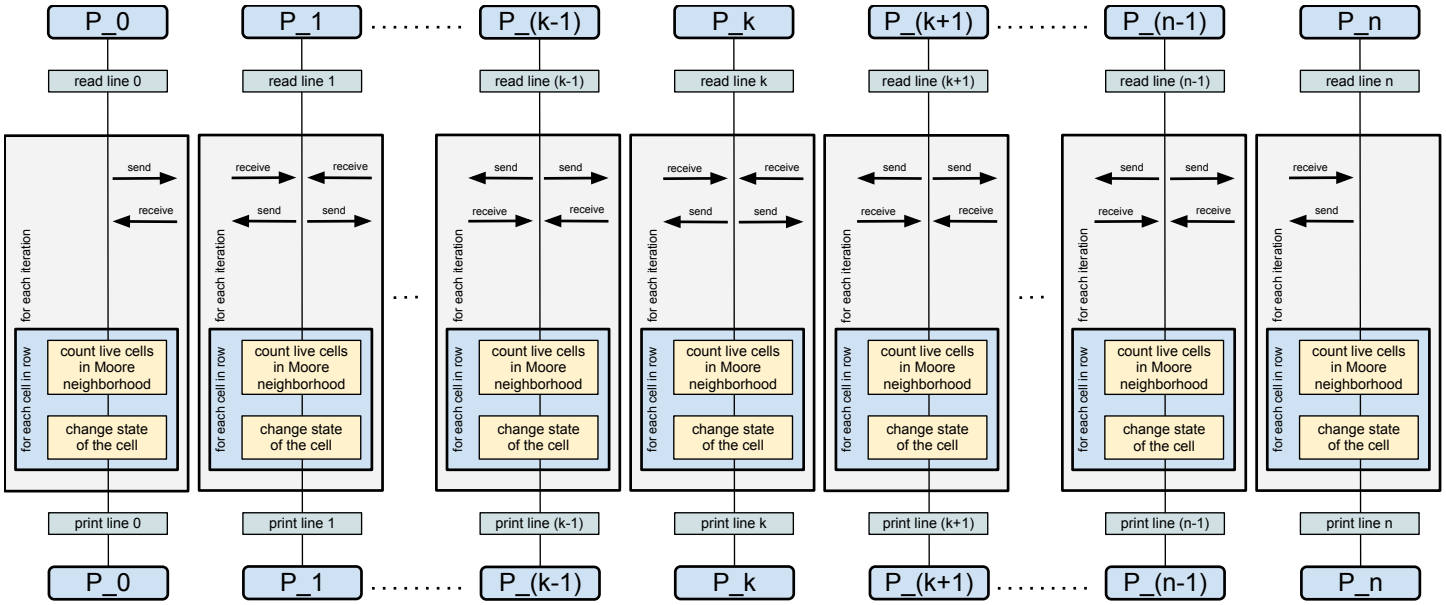


Obrázek 1: Graf závislosti času na počtu sloupců na jednom řádku pro různý počet iterací.

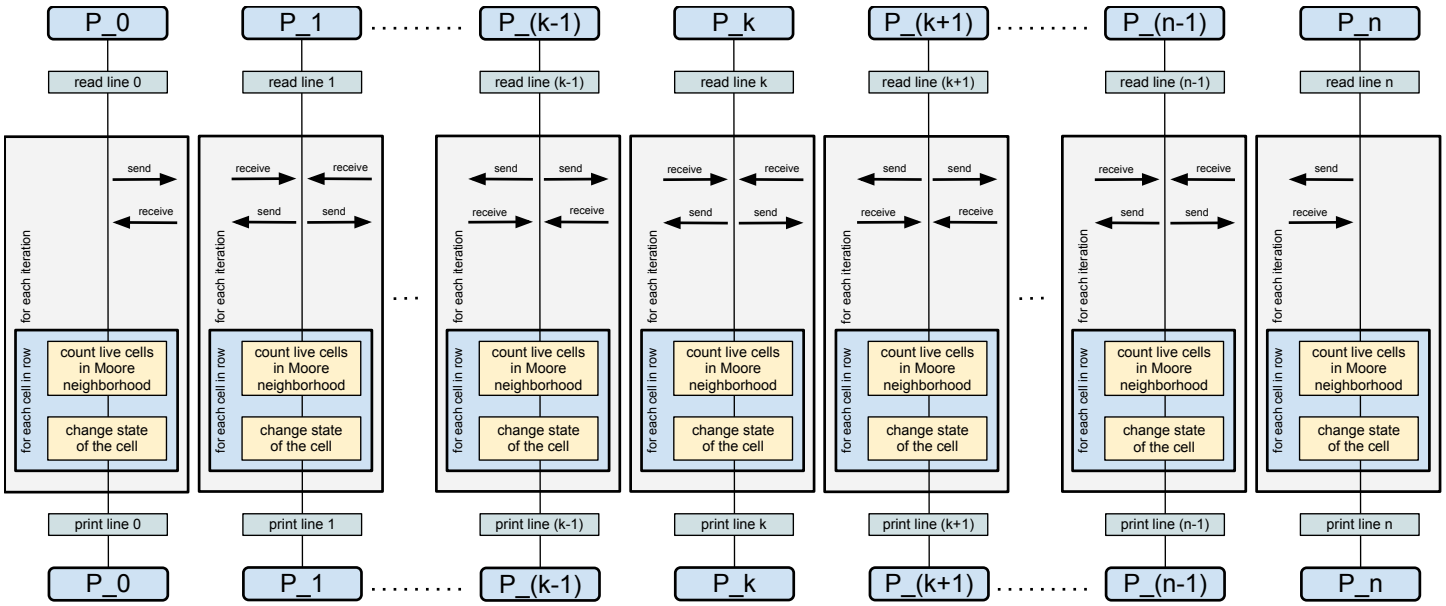
V tabulce a grafu výše jsou uvedeny průměrné z maximálních časů běhu programu pro 2^0–2^{10} iterací nad soubory s 2^0–2^{10} sloupci. Je zřejmé, že doba běhu programu závisí jak na počtu iterací (křivky v grafu mají neklesající tendenci), tak na počtu sloupců na řádku (čím více je sloupců na řádku, tím více roste čas provádění) – obojí jen potvrzuje v předchozí části určenou teoretickou složitost $O(l+i\cdot l+l) = O((i+2)\cdot l) \approx O(i\cdot l)$, kde i je počet iterací a l je počet sloupců na řádku.

3 Komunikační protokol

V následujícím diagramu je znázorněna vzájemná komunikace n procesorů, kde n je počet řádků vstupního souboru `lattice` (tedy každý procesor pracuje s právě jedním řádkem vstupního souboru). Dále je také znázorněno, které akce jsou prováděny v rámci jedné iterace (resp. kroku) hry – rozeslání vlastního řádku sousedům, přijetí řádků od sousedů a poté pro každou buňku řádku je vypočteno „živé okolí“ a na základě toho je změněn stav buňky.



Obrázek 2: Diagram komunikačního protokolu mezi procesory při paralelní implementaci hry *Game of life* (v případě, že soubor obsahuje sudý počet řádků – řádky jsou číslovány od 0, tzn. poslední řádek má lichý index).



Obrázek 3: Diagram komunikačního protokolu mezi procesory při paralelní implementaci hry *Game of life* (v případě, že soubor obsahuje lichý počet řádků– řádky jsou číslovány od 0, tzn. poslední řádek má sudý index).