# Temperaturna plošča

Rok Grmek, Matej Klemen

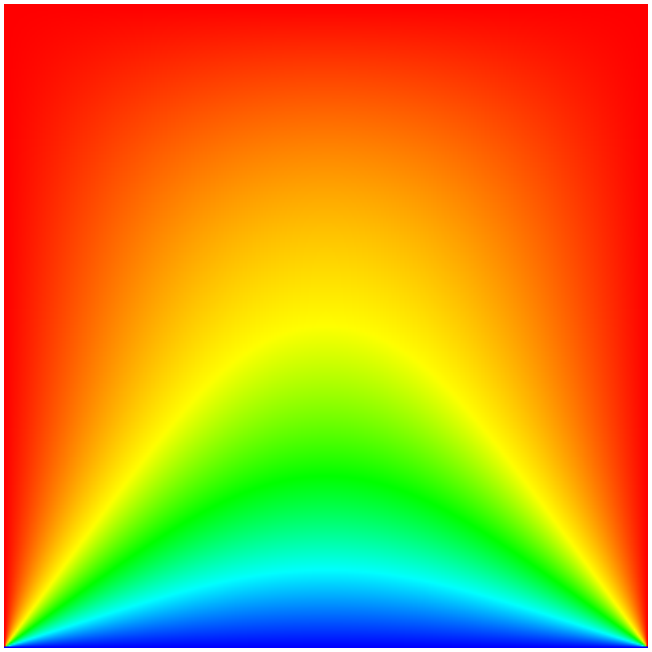
3. november 2017

# Kazalo

# Uvod

V okviru predmeta Porazdeljeni sistemi rešujeva problem temperaturne plošče. Na tem problemu bova tekom semestra spoznavala različne pristope za paralelno programiranje.

Problem temperaturne plošče je predstavljen s ploščo, ki je na treh straneh (zgoraj, levo in desno) segreta na 100 °C, na spodnji stranici pa ohlajena na 0 °C. Zanima nas, kako se v takem primeru toplota porazdeli po plošči (primer porazdelitve temperature je viden na sliki 1).



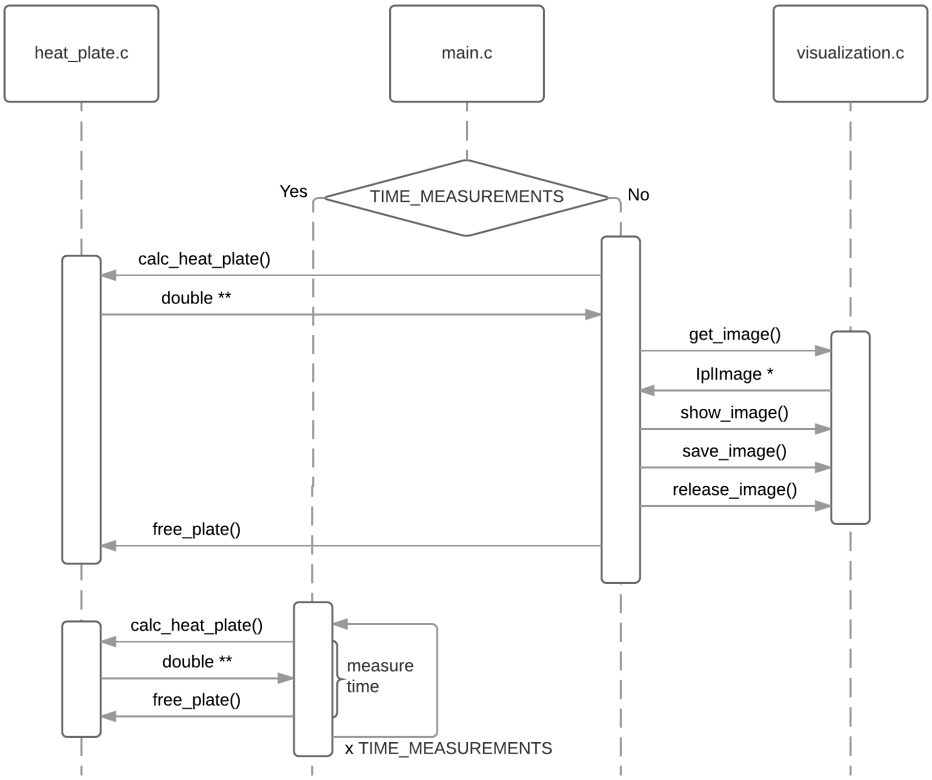
Slika 1: Primer temperaturne plošče, kjer rdeča barva prikazuje najvišjo, temno modra barva pa najnižjo temperaturo.

Iskala bova stacionarno rešitev enačbe $\nabla^2 W = f(x, y)$, pomagala pa si bova z metodo končnih diferenc. Ploščo bova najprej razdelila na mrežo točk in posameznim točkam določila začetne vrednosti. Nato bova v vsaki točki izračunala novo temperaturo na podlagi sosednjih točk. Postopek računanja novih točk se bo ponavljal, dokler rešitev ne skonvergira.

# 1 Serijski algoritem

## 1.1 Opis implementacije

Program ima dva načina delovanja. Če je definirana vrednost *TIME_MEASUREMENTS*, se bo program za podane argumente večkrat izvedel in izračunal nekaj uporabnih statistik. Ta način je namenjen predvsem opazovanju časa izvajanja programa. Sicer pa se bo program izvedel 1-krat in na koncu vizualiziral porazdelitev temperature na plošči. V nadaljevanju je opisan slednji način izvajanja, celotno delovanje programa pa je predstavljeno tudi z diagramom zaporedja, vidnim na sliki 2.




Slika 2: Diagram zaporedja, ki prikazuje delovanje programa.

Algoritem sprejme 3 argumente: višino plošče, širino plošče in mejno razliko med dvema iteracijama (v nadaljevanju označeno z $\varepsilon$), na podlagi katere se zaključi računanje. Plošči, ki jo določata vnešena višina in širina, algoritem na vseh štirih stranicah doda pas širine 1, ki služi za nastavljanje robnega pogoja temperature. Novi dimenziji sta torej:

$$višina := višina + 2 \tag{1}$$

$$širina := širina + 2 \tag{2}$$

Na začetku se alocirata dve tabeli dimenzij $višina \times širina$. Ena predstavlja trenutno stanje plošče, druga pa stanje plošče v prejšnji iteraciji. Alokaciji sledi inicializacija plošče - trem

stranicam (levi, desni, zgornji) algoritem nastavi temperaturo na 100 °C, eni (spodnji) pa na 0 °C. Vsem ostalim celicam plošče se zaporedno (*od zgoraj navzdol, od leve proti desni*) dodeli povprečje leve sosednje, zgornje sosednje, povsem desne ter povsem spodnje celice plošče. Eno izmed plošč algoritem izbere kot ploščo trenutnega, drugo pa kot ploščo prejšnjega stanja.

Nato sledi glavna zanka. V vsaki iteraciji gre algoritem skozi "dinamične" celice plošče (celice, ki niso del katerega izmed robov plošče) in za vsako tako celico c[i][j] v $i$-ti vrstici in $j$-tem stolpcu po naslednji formuli izračuna novo temperaturo:

$$c[i][j] := \frac{c'[i-1][j] + c'[i][j-1] + c'[i][j+1] + c'[i+1][j]}{4}, \tag{3}$$

kjer $c'[i][j]$ predstavlja temperaturo celice v $i$-ti vrstici in $j$-tem stolpcu v prejšnji iteraciji. Ob vsakem izračunu nove temperature algoritem izračuna še absolutno razliko med trenutno (novo) in prejšnjo temperaturo ter jo v primeru, da je to v trenutni iteraciji največja izračunana absolutna razlika temperatur, shrani. Ob koncu iteracije algoritem zamenja vlogi plošč (tista, ki je do sedaj predstavljala prejšnje stanje plošče, bo v naslednji iteraciji vsebovala novo stanje plošče in obratno), pred prehodom v naslednjo iteracijo pa preveri, če je temperaturna razlika v tej iteraciji že manjša od $\varepsilon$ (v takem primeru se računanje zaključi).

Na koncu algoritem izpiše število izvedenih iteracij in zažene vizualizacijo končnega stanja temperaturne plošče. O vizualizaciji temperaturne plošče pa je več napisano v naslednjem poglavju.

## 1.2 Uporabljene knjižnice

Za vizualizacijo temperaturne plošče uporabljava knjižnico *OpenCV*. Najprej se pripravi prazna slika (*cvCreateImage*) z dimenzijami naše plošče, nato pa se za vsako točko na plošči pretvori temperaturo v 3 8-bitne kanale (rdeča, zelena, modra) in vrednosti prepiše na sliko. Če slika z največjo stranico presega *MAX_SIZE*, potem se jo še pomanjša (*cvResize*). Pripravljeno sliko se prikaže (*cvShowImage*) v oknu (*cvNamedWindow*, *cvMoveWindow*, *cvResizeWindow*) in shrani na disk (*cvSaveImage*). Na koncu se le še sprosti zaseden pomnilnik (*cvReleaseImage*). Primer vizualizacije je viden na sliki 1.

## 1.3 Rezultati

Program je bil testiran na sistemu, katerega specifikacije so navedene v tabeli 1. Da bi k izmerjenemu času čim manj pripomogli stroški režije operacijskega sistema, je bil sistem med testiranjem minimalno obremenjen z drugimi procesi.

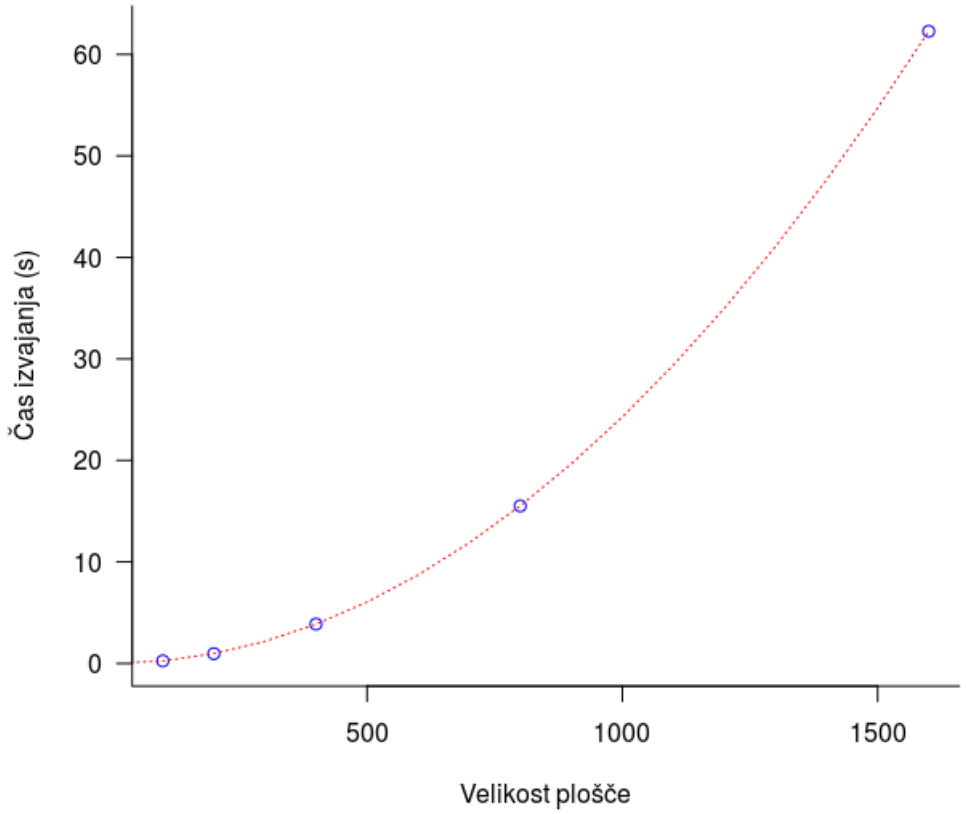
Tabela 1: Specifikacije testnega sistema.

| | |
|---|---|
| Procesor: | Intel Core i5-4210U |
| Frekvenca procesorja: | 1.70GHz |
| Število jeder: | 2 |
| Maksimalno število niti: | 4 |
| Velikost predpomnilnika: | 3MB |
| Velikost in tip glavnega pomnilnika: | 16GB DDR3 |
| Grafična kartica: | NVIDIA GeForce 820M 2GB DDR3 |
| Operacijski sistem: | Ubuntu 16.04 |

Pri testiranju sva se omejila na fiksno vrednost $\varepsilon = 0,01$ in na plošče kvadratne oblike, spreminjala pa sva zgolj velikost stranice. Za vsako izbrano velikost sva algoritem 100-krat zagnala in vsakič izmerila čas izvajanja. Iz meritev sva nato izračunala povprečni čas izvajanja in standardno napako (ta predstavlja razpršenost meritev okrog povprečnega časa). Rezultati so navedeni tabelarično v tabeli 2 in z grafom, prikazanim na sliki 3.

Tabela 2: Povprečni čas izvajanja programa in standardna napaka v odvisnosti od velikosti stranice.

| Velikost stranice [št. točk] | Povprečni čas izvajanja [s] | Standardna napaka [s] |
|---|---|---|
| 100 | 0,250 | 0,000 |
| 200 | 0,965 | 0,001 |
| 400 | 3,877 | 0,003 |
| 800 | 15,516 | 0,006 |
| 1600 | 62,272 | 0,062 |




Slika 3: Graf, ki prikazuje povprečni čas izvajanja programa v odvisnosti od velikosti stranice.

TODO: DODAJ ŠE KAKŠEN STAVEK K POVZETKU REZULTATOV (podatki za fit polinoma: $y = 2.441 \cdot 10^{-5} \cdot x^2 - 1.576 \cdot 10^{-4} \cdot x + 2.324 \cdot 10^{-2}$)

Iz rezultatov je vidno, da je povprečni čas izvajanja kvadratno odvisen od velikosti plošče.

# 2 Paralelizacija s knjižnico "Pthreads"

## 2.1 Ideja paralelizacije

TODO: Shema delitve dela med niti.

TODO: dopolni spodnji opis.

Na začetku vsaka nit dobi svoj identifikator iz intervala $[0, \textit{število uporabljenih niti} - 1]$. Iz tega indeksa se s pomočjo spodaj navedenih enačb izračunata spodnja in zgornja meja računanja posamezne niti.

$$\textit{spodnja meja} := 1 + (int)\left(\frac{(double)index \cdot (h-2)}{NUM\_THREADS}\right) \tag{4}$$

$$\textit{zgornja meja} := 1 + (int)\left(\frac{(double)(index+1) \cdot (h-2)}{NUM\_THREADS}\right) \tag{5}$$

V zgornji enačbi *index* predstavlja identifikator niti, $h$ višino plošče, $NUM\_THREADS$ pa število niti, ki jih program uporablja. $(h-2)$ predstavlja število vrstic, katerih temperatura ni fiksno nastavljena in jo v vsaki iteraciji ponovno računamo, ki ga želimo razdeliti. Obema mejama prištejemo 1, saj je temperatura prve vrstice fiksno nastavljena na 100 °C in zato želimo delo razdeliti od druge vrstice naprej.
Delo se med niti porazdeli čim bolj enakomerno, v najboljšem primeru dobijo vse niti za izračunati enako število vrstic, v najslabšem primeru pa dobi nit z največjim identifikatorjem ($\textit{število uporabljenih niti} - 1$) eno vrstico več kot preostale niti.

**Manjka še kšna poved o tem, da grejo pol niti računat**

Ko niti poračunajo svoj del plošče, morajo počakati na preostale niti, saj je nadaljnje izvajanje programa odvisno od izračunov vseh niti v trenutni iteraciji. To je doseženo s postavitvijo t.i. prepreke (angl. *barrier*), ki nitim ne pusti opravljati nadaljnjega dela, dokler vse niti ne pridejo do mesta, kjer je postavljena.

Sledi postopek sinhronizacije največje izračunane absolutne razlike temperature v trenutni iteraciji. Vsaka nit je ta podatek izračunala na svojem delu plošče, v tem delu pa niti pregledajo, če so vsi izračunani lokalni maksimumi pod mejo $\varepsilon$, ki predstavlja konvergenco. Opis komunikacije, ki poteka med nitmi (lahko shematsko).

## 2.2 Rezultati

Meritve časa izvajanja za različne velikosti problema (tukaj pomoje le za najboljše število niti). Omenit je treba, da so specifikacije enake (referenca na tabelo specifikacij) in da so bile meritve opravljene na enak način kot pri serijskem algoritmu.

Tabela in graf za meritve.

Pohitritev (časi za nek primer plošče v odvisnosti od števila niti). Ali dosežete pohitritev? Kdaj je pohitritev največja in zakaj? Je to pri največjem številu niti? Ali pohitritev limitira k nekemu številu?

Učinkovitost (verjetno na enakih podatkih kot pohitritev). Komentar na spreminjanje učinkovitosti.

Tabela in graf za meritve.