Izvještaj o optimizaciji algoritma konvolucije za obradu slika

1. Opis problema

Konvolucija predstavlja fundamentalnu matematičku operaciju u digitalnoj obradi slika koja se koristi za implementaciju različitih filtara i efekata. Problem koji se razmatra u ovom radu je optimizacija algoritma konvolucije za potrebe efikasne obrade slikovnih podataka u BMP formatu.

Konvolucija u kontekstu obrade slika podrazumijeva kombinovanje dvije funkcije: originalne slike i "kernela" (matrice težinskih koeficijenata). Rezultat ove operacije je nova slika koja predstavlja modifikovanu verziju originala. Matematički, ova operacija se može predstaviti kao:

gdje je:

* f(x,y) originalna slika
* h(i,j) kernel matrica
* g(x,y) rezultujuća slika
* a,b dimenzije kernela

Konvolucija predstavlja računski zahtjevnu operaciju, naročito za velike slike i kernele, te je od posebnog interesa pronaći optimalne načine njene implementacije.

2. Opis osnovnog algoritma

Implementacija algoritma konvolucije je realizovana u programskom jeziku C++. Osnovni algoritam obuhvata sljedeće korake:

1. Učitavanje ulazne BMP slike i razdvajanje na RGB kanale
2. Primjena konvolucije na svaki kanal pojedinačno
3. Kombinovanje obrađenih kanala
4. Zapisivanje rezultujuće slike u BMP format

Centralni dio algoritma je funkcija convolveChannel() koja implementira operaciju konvolucije za jedan kanal slike:

void convolveChannel(const vector<uint8\_t>& input, int inWidth, int inHeight, const vector<float>& kernel, int kernelRows, int kernelCols, vector<uint8\_t>& output)

3. Opis optimizovanih varijanti algoritma

U cilju poboljšanja performansi, implementirane su sljedeće optimizacije:

3.1 Optimizacije na nivou kompajlera

Upotrijebljene su različite razine optimizacije kompajlera, od -O0 (bez optimizacije) do -O3 (agresivna optimizacija):

* -O0: Bez optimizacije, najbrže kompajliranje
* -O1: Osnovna optimizacija koja ne zahtijeva značajno više vremena za kompajliranje
* -O2: Umjereno agresivna optimizacija koja obuhvata sve optimizacije iz -O1 i dodatne
* -O3: Najagresivnija optimizacija koja uključuje sve optimizacije iz -O2 sa dodatnim transformacijama koda koje mogu značajno povećati brzinu izvršavanja

3.2 Paralelizacija korišćenjem OpenMP

Implementirano je paralelno izvršavanje najintenzivnijeg dijela algoritma korišćenjem OpenMP direktive #pragma omp parallel for. Ovo omogućava raspodjelu iteracija vanjske petlje na više dostupnih niti procesora. Testirane su konfiguracije sa 1, 2, 4 i 8 niti.

3.3 SIMD optimizacije

Kompajliranje sa -march=native -mavx2 omogućava upotrebu AVX2 (Advanced Vector Extensions 2) instrukcijskog seta koji podržava SIMD (Single Instruction, Multiple Data) operacije. Ove instrukcije omogućavaju paralelno izvršavanje iste operacije nad više podataka istovremeno.

3.4 Memorijske optimizacije

Implementirane su sljedeće optimizacije pristupa memoriji:

* Vektorizacija podataka korišćenjem std::vector za efikasno upravljanje memorijom
* Kopiranje podataka između kanala slike koristeći std::copy umjesto ručnog kopiranja
* Izbjegavanje nepotrebnih alokacija memorije tokom obrade
* Reorganizacija petlji za prolazak kroz kernel, radi boljih performaski keš memorije

4. Metodologija mjerenja

4.1 Mjerenje performansi

Za mjerenje performansi algoritma korišćena je sljedeća metodologija:

1. Izvršavanje 100 zagrijavajućih iteracija (warmup)
2. Mjerenje vremena izvršavanja za 100 iteracija algoritma
3. Izračunavanje prosjeka, varijanse i standardne devijacije

Vrijeme izvršavanja je mjereno u mikrosekundama korišćenjem std::chrono::high\_resolution\_clock.

4.2 Testni podaci

Za testiranje su korištene BMP slike različitih dimenzija:

* 10×10 piksela
* 50×50 piksela
* 500×500 piksela
* 1000×1000 piksela
* 2000×2000 piksela

Za konvoluciju je korišten kernel dimenzija 3×3 sa sljedećim vrijednostima:

[0.11 0.11 0.11]

[0.11 0.11 0.11]

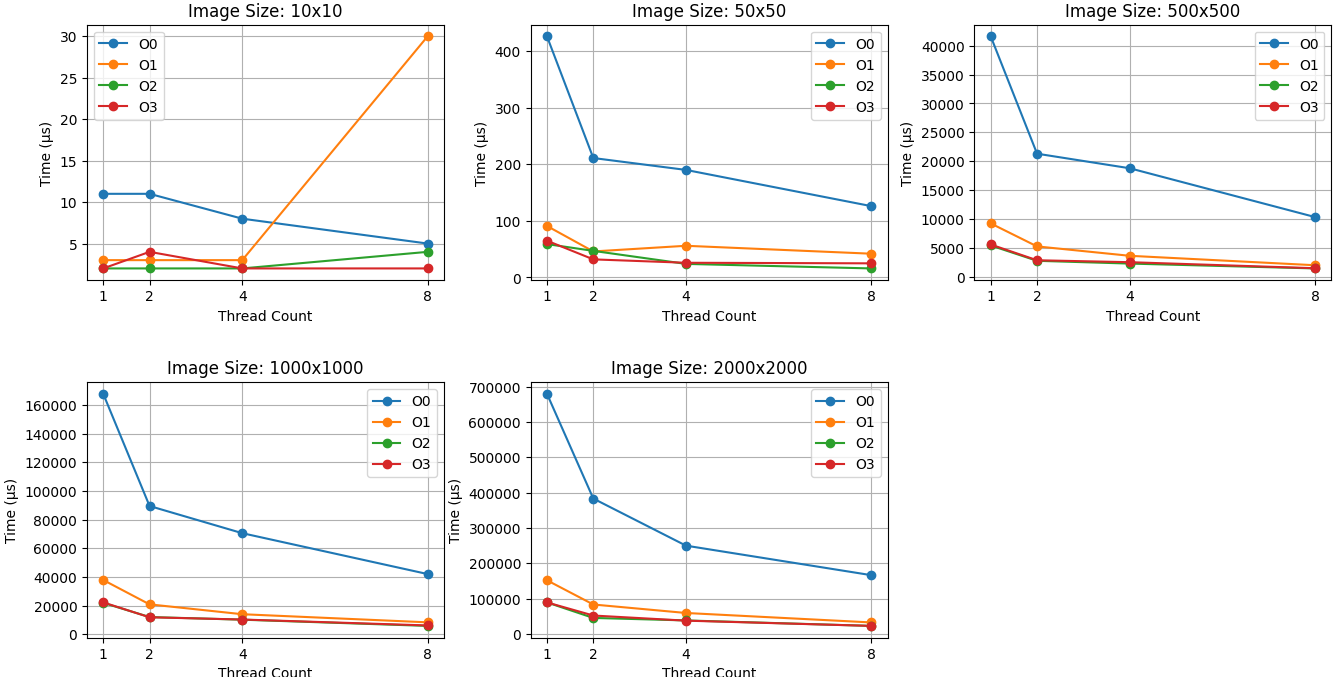
[0.11 0.11 0.11]

Ovaj kernel efektivno implementira filter za zamagljivanje (blur filter).

4.3 Okruženje za testiranje

Benchmark testovi su organizovani pomoću bash skripte run\_benchmarks.sh koja automatizuje proces kompajliranja programa sa različitim nivoima optimizacije, izvršavanje sa različitim brojem niti i nad različitim ulaznim podacima. Rezultati se zapisuju u CSV format za dalju analizu.

5. Rezultati mjerenja



6. Zaključak

Na osnovu provedenih eksperimenata i analize rezultata, možemo izvesti sljedeće zaključke:

1. **Nivo optimizacije kompajlera** ima značajan utjecaj na performanse algoritma. Najveće poboljšanje se postiže prelaskom sa -O0 na -O1, dok su dalja poboljšanja manje izražena.
2. **Paralelizacija** putem OpenMP-a pruža skoro linearno ubrzanje do broja fizičkih jezgara procesora. Povećanje broja niti sa 1 na 4 rezultira približno trostrukim ubrzanjem, što ukazuje na dobru paralelizaciju algoritma.
3. **Skaliranje sa veličinom slike** je kvadratno, što je očekivano s obzirom na prirodu problema.
4. **Kombinacija optimizacija** (-O3, AVX2 instrukcije i paralelizacija) daje najbolje rezultate, posebno za veće slike gdje je razlika u performansama najizraženija.

Rezultati pokazuju da je moguće značajno ubrzati algoritam konvolucije kroz kombinaciju različitih tehnika optimizacije, što je od posebnog značaja za aplikacije koje zahtijevaju obradu slika u realnom vremenu.

Informacije o uredjaju:

* Laptop – Acer Aspire A515-45
* Procesor – AMD Ryzen 3 5300U with Radeon Graphics 2.60 GHz
* RAM – 8.00GB
* OS – Ubuntu 24.04 (WSL)
* Kompajler – g++ 13.2.0