CEN310 Paralel Programlama

Hafta-12 (Gerçek Dünya Uygulamaları I)

Bahar Dönemi, 2024-2025

Genel Bakış

Konular

1. Bilimsel Hesaplama Uygulamaları
2. Veri İşleme Uygulamaları
3. Performans Optimizasyonu
4. Örnek Çalışmalar

Hedefler

- Paralel programlamayı gerçek problemlere uygulama
- Bilimsel hesaplamaları optimize etme
- Büyük veri kümelerini verimli işleme
- Gerçek dünya performansını analiz etme

1. Bilimsel Hesaplama Uygulamaları

N-Cisim Simülasyonu

```
__global__ void kuvvet_hesapla(float4* konum, float4* hiz, float4* kuvvetler, int n) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    if (idx < n) {
        float4 benim_konum = konum[idx];
        float4 kuvvet = make_float4(0.0f, 0.0f, 0.0f, 0.0f);

        for(int j = 0; j < n; j++) {
            if(j != idx) {
                float4 diger_konum = konum[j];
                float3 r = make_float3(
                    diger_konum.x - benim_konum.x,
                    diger_konum.y - benim_konum.y,
                    diger_konum.z - benim_konum.z
                );
                float uzaklik = sqrtf(r.x*r.x + r.y*r.y + r.z*r.z);
                float f = (G * benim_konum.w * diger_konum.w) / (uzaklik * uzaklik);
                kuvvet.x += f * r.x/uzaklik;
                kuvvet.y += f * r.y/uzaklik;
                kuvvet.z += f * r.z/uzaklik;
            }
        }
        kuvvetler[idx] = kuvvet;
    }
}
```

2. Veri İşleme Uygulamaları

Görüntü İşleme

```
__global__ void gaussian_bulanik(
    unsigned char* girdi,
    unsigned char* cikti,
    int genislik,
    int yukseklik,
    float* cekirdek,
    int cekirdek_boyutu
) {
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;

    if(x < genislik && y < yukseklik) {
        float toplam = 0.0f;
        int c_yaricap = cekirdek_boyutu / 2;

        for(int ky = -c_yaricap; ky <= c_yaricap; ky++) {
            for(int kx = -c_yaricap; kx <= c_yaricap; kx++) {
                int px = min(max(x + kx, 0), genislik - 1);
                int py = min(max(y + ky, 0), yukseklik - 1);
                float cekirdek_deger = cekirdek[(ky+c_yaricap)*cekirdek_boyutu + (kx+c_yaricap)];
                toplam += girdi[py*genislik + px] * cekirdek_deger;
            }
        }

        cikti[y*genislik + x] = (unsigned char)toplam;
    }
}
```

3. Performans Optimizasyonu

Bellek Erişim Optimizasyonu

```
// Matris transpozunu optimize et
__global__ void matris_transpoz(float* girdi, float* cikti, int genislik, int yukseklik) {
    __shared__ float karo[BLOK_BOYUTU][BLOK_BOYUTU+1]; // Bank çakışmalarını önle

    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;

    if(x < genislik && y < yukseklik) {
        // Paylaşımlı belleğe yükle
        karo[threadIdx.y][threadIdx.x] = girdi[y*genislik + x];
        __syncthreads();

        // Transpoz indislerini hesapla
        int yeni_x = blockIdx.y * blockDim.y + threadIdx.x;
        int yeni_y = blockIdx.x * blockDim.x + threadIdx.y;

        if(yeni_x < yukseklik && yeni_y < genislik) {
            cikti[yeni_y*yukseklik + yeni_x] = karo[threadIdx.x][threadIdx.y];
        }
    }
}
```

4. Örnek Çalışmalar

Monte Carlo Simülasyonu

```
__global__ void monte_carlo_pi(float* noktalar_x, float* noktalar_y, int* daire_icinde, int n) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    if(idx < n) {
        float x = noktalar_x[idx];
        float y = noktalar_y[idx];
        float uzaklik = x*x + y*y;

        if(uzaklik <= 1.0f) {
            atomicAdd(daire_icinde, 1);
        }
    }
}

int main() {
    int n = 1000000;
    float *h_x, *h_y, *d_x, *d_y;
    int *h_icinde, *d_icinde;

    // Bellek ayır ve başlat
    // ... (bellek ayırma kodu)

    // Rastgele noktalar üret
    for(int i = 0; i < n; i++) {
        h_x[i] = (float)rand()/RAND_MAX;
        h_y[i] = (float)rand()/RAND_MAX;
    }

    // Veriyi cihaza kopyala ve çekirdeği çalıştır
    // ... (CUDA bellek işlemleri ve çekirdek başlatma)

    // Pi'yi hesapla
    float pi = 4.0f * (*h_icinde) / (float)n;
    printf("Hesaplanan Pi: %f\n", pi);

    // Temizlik
    // ... (bellek temizleme kodu)

    return 0;
}
```

Laboratuvar Alıştırması

Görevler

1. N-cisim simülasyonu uygulama
2. Görüntü işleme çekirdeğini optimize etme
3. Monte Carlo simülasyonu geliştirme
4. CPU versiyonları ile performans karşılaştırması

Performans Analizi

- Çalışma süresi
- Bellek bant genişliği
- GPU kullanımı
- Ölçekleme davranışı

Kaynaklar

Dokümantasyon

- CUDA Örnek Uygulamaları
- Bilimsel Hesaplama Kütüphaneleri
- Performans Analiz Araçları

Araçlar

- NVIDIA Görsel Profilleyici
- Paralel Hesaplama Araç Seti
- Performans Kütüphaneleri

Sorular ve Tartışma


RECEP TAYYIP
ERDOĞAN
ÜNİVERSİTESİ