CEN310 Paralel Programlama

Hafta-13 (Gerçek Dünya Uygulamaları II)

Bahar Dönemi, 2024-2025

Genel Bakış

Konular

1. İleri Paralel Desenler
2. N-Cisim Simülasyonları
3. Matris Hesaplamaları
4. Büyük Veri İşleme

Hedefler

- Karmaşık paralel desenleri uygulamak
- Bilimsel simülasyonları optimize etmek
- Büyük ölçekli matris işlemleri gerçekleştirmek
- Büyük veriyi verimli işlemek

1. İleri Paralel Desenler

Boru Hattı Deseni

```
template<typename T>
class ParalelBoruHatti {
private:
    std::vector<std::thread> asamalar;
    std::vector<std::queue<T>> kuyruklar;
    std::vector<std::mutex> muteksler;
    std::vector<std::condition_variable> kosul_degiskenleri;
    bool calisiyor;

public:
    ParalelBoruHatti(int asama_sayisi) {
        kuyruklar.resize(asama_sayisi - 1);
        muteksler.resize(asama_sayisi - 1);
        kosul_degiskenleri.resize(asama_sayisi - 1);
        calisiyor = true;
    }

    void asama_ekle(std::function<void(T&)> asama_fonk, int asama_id) {
        asamalar.emplace_back([this, asama_fonk, asama_id]() {
            while(calisiyor) {
                T veri;
                if(asama_id == 0) {
                    // İlk aşama: veri üret
                    veri = veri_uret();
                } else {
                    // Önceki aşamadan veri al
                    std::unique_lock<std::mutex> kilit(muteksler[asama_id-1]);
                    kosul_degiskenleri[asama_id-1].wait(kilit,
                        [this, asama_id]() {
                            return !kuyruklar[asama_id-1].empty() || !calisiyor;
                        });
                    if(!calisiyor) break;
                    veri = kuyruklar[asama_id-1].front();
                    kuyruklar[asama_id-1].pop();
                    kilit.unlock();
                    kosul_degiskenleri[asama_id-1].notify_one();
                }

                // Veriyi işle
                asama_fonk(veri);

                if(asama_id < asamalar.size() - 1) {
                    // Sonraki aşamaya geç
                    std::unique_lock<std::mutex> kilit(muteksler[asama_id]);
                    kuyruklar[asama_id].push(veri);
                    kilit.unlock();
                    kosul_degiskenleri[asama_id].notify_one();
                }
            }
        });
    }

    void baslat() {
        for(auto& asama : asamalar) {
            asama.join();
        }
    }

    void durdur() {
        calisiyor = false;
        for(auto& kd : kosul_degiskenleri) {
            kd.notify_all();
        }
    }
};
```

2. N-Cisim Simülasyonları

Barnes-Hut Algoritması

```
struct Sekizli_Agac {
    struct Dugum {
        vec3 merkez;
        float boyut;
        float kutle;
        vec3 kutle_merkezi;
        std::vector<Dugum*> cocuklar;
    };

    Dugum* kok;
    float teta;

    __device__ void kuvvet_hesapla(vec3& konum, vec3& kuvvet, Dugum* dugum) {
        vec3 fark = dugum->kutle_merkezi - konum;
        float uzaklik = length(fark);

        if(dugum->boyut / uzaklik < teta || dugum->cocuklar.empty()) {
            // Yaklaşık hesaplama kullan
            float f = G * dugum->kutle / (uzaklik * uzaklik * uzaklik);
            kuvvet += fark * f;
        } else {
            // Çocuklara özyinele
            for(auto cocuk : dugum->cocuklar) {
                if(cocuk != nullptr) {
                    kuvvet_hesapla(konum, kuvvet, cocuk);
                }
            }
        }
    }

    __global__ void cisimleri_guncelle(vec3* konum, vec3* hiz, vec3* ivme,
                                       float dt, int n) {
        int idx = blockIdx.x * blockDim.x + threadIdx.x;
        if(idx < n) {
            vec3 kuvvet(0.0f);
            kuvvet_hesapla(konum[idx], kuvvet, kok);
            ivme[idx] = kuvvet;
            hiz[idx] += ivme[idx] * dt;
            konum[idx] += hiz[idx] * dt;
        }
    }
};
```

3. Matris Hesaplamaları

Paralel Matris Ayrıştırma

```
__global__ void lu_ayristirma(float* A, int n, int k) {
    int satir = blockIdx.y * blockDim.y + threadIdx.y;
    int sutun = blockIdx.x * blockDim.x + threadIdx.x;

    if(satir > k && satir < n && sutun > k && sutun < n) {
        A[satir * n + sutun] -= A[satir * n + k] * A[k * n + sutun] / A[k * n + k];
    }
}

void paralel_lu(float* A, int n) {
    dim3 blok(16, 16);
    dim3 izgara((n + blok.x - 1) / blok.x,
                (n + blok.y - 1) / blok.y);

    for(int k = 0; k < n-1; k++) {
        lu_ayristirma<<<izgara, blok>>>(A, n, k);
        cudaDeviceSynchronize();
    }
}
```

4. Büyük Veri İşleme

Paralel Veri Analizi

```
template<typename T>
class ParalelVeriIsleyici {
private:
    std::vector<T> veri;
    int is_parcacigi_sayisi;

public:
    ParalelVeriIsleyici(const std::vector<T>& girdi, int is_parcaciklari)
        : veri(girdi), is_parcacigi_sayisi(is_parcaciklari) {}

    template<typename Fonk>
    std::vector<T> haritalama(Fonk f) {
        std::vector<T> sonuc(veri.size());
        #pragma omp parallel for num_threads(is_parcacigi_sayisi)
        for(size_t i = 0; i < veri.size(); i++) {
            sonuc[i] = f(veri[i]);
        }
        return sonuc;
    }

    template<typename Fonk>
    T indirgeme(Fonk f, T baslangic) {
        T sonuc = baslangic;
        #pragma omp parallel num_threads(is_parcacigi_sayisi)
        {
            T yerel_toplam = baslangic;
            #pragma omp for nowait
            for(size_t i = 0; i < veri.size(); i++) {
                yerel_toplam = f(yerel_toplam, veri[i]);
            }
            #pragma omp critical
            {
                sonuc = f(sonuc, yerel_toplam);
            }
        }
        return sonuc;
    }
};
```

Laboratuvar Alıştırması

Görevler

1. Barnes-Hut simülasyonu uygulama
2. Paralel LU ayrıştırma geliştirme
3. Büyük veri işleme boru hattı oluşturma
4. Performans özelliklerini analiz etme

Performans Analizi

- Algoritma karmaşıklığı
- Bellek erişim desenleri
- Yük dengeleme
- Ölçeklenebilirlik testi

Kaynaklar

Dokümantasyon

- İleri CUDA Programlama Kılavuzu
- Paralel Algoritmalar Referansı
- Bilimsel Hesaplama Kütüphaneleri

Araçlar

- Performans Profilleyiciler
- Hata Ayıklama Araçları
- Analiz Çerçeveleri

Sorular ve Tartışma