

OpenMP

F.CS306 ПАРАЛЛЕЛ ПРОГРАММЧЛАЛ – Лекц 10

Г.ГАНБАТ

ganbatg@must.edu.mn

Хичээлийн агуулга

- ❖ OpenMP гэж юу вэ
- ❖ Parallel for directive
- ❖ Basic Parallel Reductions

OpenMP-ын тухай

- ❖ C, C++, Fortran зэрэг дээрх платформ хамааралгүй, дундын санах ойн параллел программчлалын API.
- ❖ OpenMP нь хэрхэн параллеллах тухай хөрвүүлэгчид заавар өгөх *pragmas* гэдэг компиляторын comment-like тусгай чиглүүлэгч хэрэглэн дараалсан кодыг өргөтгөдөг.
 - ❖ Хэрэв хөрвүүлэгч нь *pragmas*-г дэмждэггүй бол алгасаад явна.
 - ❖ Хөрвүүлэгч нь OpenMP API-ийг дэмжих эсэхийг хөрвүүлэгчийн *-fopenmp* флагаар заана.
- ❖ Thread-ийн тоо нь анхандаа үйлдлийн системийн логик CPU-ны тоотой адил байдаг. *Team* гэнэ.
 - ❖ env var: **OMP_NUM_THREADS**
 - ❖ кодон дотроос: *set_num_threads()*

OpenMP Hello world!

- ❖ Хөрвүүлэх:
 - ❖ g++ -O2 -std=c++14 -fopenmp hello_world.cpp -o hello_world
- ❖ Ажиллуулах: Энд thread-ийн тоог зааж өгсөн байна.
 - ❖ OMP_NUM_THREADS=2 ./hello_world
- ❖ Yp дүн:

```
Hello world!
Hello world!
```

```
#include <iostream>

int main()
{
    // run the statement after the pragma in all threads
    #pragma omp parallel
    std::cout << "Hello world!" << std::endl;
}
```

❖ Хөрвүүлэх:

❖ g++ -O2 -std=c++14 -fopenmp hello_world.cpp -o hello_world

❖ Ажиллуулах: ./hello_world

❖ Yp дүн:

```
Hello world from thread 1 of 4
Hello world from thread 0 of 4
Hello world from thread 2 of 4
Hello world from thread 3 of 4
```

```
#include <iostream>
#include <omp.h>

int main()
{
    // run the block after the pragma in four threads
    #pragma omp parallel num_threads(4)
    {
        int i = omp_get_thread_num();
        int n = omp_get_num_threads();
        std::cout << "Hello world from thread " <<
            i << " of " << n << std::endl;
    }
}
```

THE parallel for DIRECTIVE

- ❖ *Work-sharing* зориулсан хамгийн түгээмэл аргуудын нэг нь давталтын параллелчлал юм.
- ❖ #pragma omp for нь заавал давталтын өмнө байна. Хязгаарлалтууд:
 1. Давталтын хувьсагч: *int* (v2.0 +), *uint* (v3.0 +).
 2. Давталтын хяналтын параметерүүд бүх thread-д ижил байна
 3. *goto, break* хориглогдсон.
- ❖ Thread-үүдийг #pragma хэсэг эхлэхэд хувилах ба блок дуусахад *join()* хийгдэнэ.
- ❖ OpenMP нь дараалсан давталтыг яг зөв параллел код болгоно гэсэн үг биш! RACE CONDITION-д анхаарах чухал

❖ Товчилсн хэлбэр

```
#pragma omp parallel for
for (...) {
    ...
}
```

❖ Дэлгэрэнгүй хэлбэр

```
#pragma omp parallel
{
    #pragma omp for
    for (...) {
        ...
    }
}
```

Суурь синхрончлол

```
#pragma omp parallel for  
for (...) { ... }
```

```
#pragma omp parallel for  
for (...) { ... }
```

#pragma omp parallel хэрэглэсэн үед давталтууд дараалж параллел ажиллана.

#pragma omp parallel бүр эхлэхдээ thread хувилалт хийж дуусахдаа нэгтгэдэг. #pragma omp for нь далд синхрончлол дэмждэг. Жны: Сул thread-үүд давталтын их биеийн төгсгөлд ажиллагааг хорьдог.

```
#pragma omp parallel  
{  
    #pragma omp for nowait  
    for (...) { ... }  
  
    #pragma omp for  
    for (...) { ... }  
}
```

```
#pragma omp parallel  
{  
    #pragma omp for  
    for (...) { ... }  
  
    #pragma omp for  
    for (...) { ... }  
}
```

#pragma omp nowait хэрэглэвэл давталтууд илүү бие даасан байдлаар, бие биенээсээ хамаarahгүйгээр ажиллана

Хувьсагч далдлалт

1. *Зарлах:* і хувьсагчийн давталтаас өмнө глобал зарлана.
 - ❖ Уралдааны нөхцлөөс сэргийлж OpenMP чиглүүлэгчийг `#pragma omp parallel private(i)` гэж бичнэ. Глобал і хувьсагчаас тусдаа thread бүр өөрийн утга тодорхойгүй і-ийн хуулбартай болно.
2. *Утга оноох:* Анхын утгыг глобал хувьсагчаас нь дамжуулах бол `#pragma omp parallel firstprivate(i)` гэсэн чиглүүлэгч хэрэглэнэ.

3. *Хандах:* Thread
доторх утгыг
авах бол глобал
массив руу
утгыг хуулна.

```
const int num = omp_get_max_threads();
int *aux = new int[num];
int i = 1;    // we pass this via copy by value
#pragma omp parallel firstprivate(i) num_threads(num)
{
    // get the thread identifier j
    const int j = omp_get_thread_num();
    i += j; // any arbitrary function f(i, j)
    aux[j] = i; // write i back to global scope
}

delete [] aux; // aux stores the values[1, 2, 3, ...]
```

Хувьсагч далдалт/хуваалцалт

- ❖ Анхандаа хувьсагчид далдлагдаагүй буюу дундын байна. Гэсэн ч кодон дотроо ялгаж тодруулах үүднээс `shared` гэж тодорхойлох хэрэгтэй.
- ❖ Олон хувьсагч далдлах/хуваалцах бол `private(i, j)` гэх мэт.
- ❖ Хаягийг далдлахад массив хэрэглэх шаардлагагүй. Мөн `private(data_ptr)` анхны утгыг өгөхгүй учир сегментчлэлийн алдаа гарна. `firstprivate` хэрэглэх.
- ❖ Thread түвшний массив хэрэгтэй бол параллел хэсэгт зай авч болон чөлөөлж хэрэгжүүлж болно.
- ❖ *pragmas*-ыг C++11-ын *promise*, *future* –тай болон Boost сангийн өгөгдлийн lock-free бүтцүүдтэй хослуулан хэрэглэх чөлөөтэй.

Энгийн параллел reduction

- ❖ Дундын аккумулятор хувьсагчийг параллел контекст дотор зөв өөрчлөх. Жны: Массивын элементүүдийн параллел нийлбэр. Дараах зүйлсийг авч үзнэ.
 - ❖ Хоёртын *reduction тар-ын ерөнхий* ашиглалт
 - ❖ Программ ажиллах нийт хугацаанд үзүүлэх параллелчлалын нөлөөллийн шинжилгээ
- ❖ Гараар бичсэн тооны түгээмэл MNIST өгөгдөл дээр суралцсан one-nearest-neighbor (1NN) ангилагчийн параллел чадвархийн шинжилгээ.
 - ❖ Машин сургалтын хамгийн нийтлэг ангилагч
 - ❖ Үндсэн асуудал нь хоёр объектийн ижил төстэй байдлын ялгааг хэмжих.

1NN ангилагч

- ❖ Жишээ: 65000 ширхэг 28×28 хэмжээтэй 0 – 9 цифрийн гараар бичсэн хар цагаан зураг өгөгдсөн. $N=55000$ -ыг сургалтанд, $m=10000$ -ыг тестэнд ашиглана.
 - ❖ Цифр бүрийг $d = 728$ ширхэг утгатай векторт хөрвүүнэ.
 - ❖ Сургалтын өгөгдлийг $n \times d$ хэмжээтэй $D_{jk}^{train} = x_{train}^{(j)} [k]$ матрицад хадгална. Энд j нь n зургийн дугаар, k нь зургийн d цэгийн дугаар
 - ❖ Тестийн өгөгдлийг $m \times d$ хэмжээтэй $D_{ik}^{test} = x_{test}^{(i)} [k]$ матрицад хадгална. Энд i нь m зургийн дугаар, k нь зургийн d цэгийн дугаар

Хэрэгжүүлэлт

- ❖ (i,j) дугаарлалтын хувьд давхар давталт гүйцэтгэнэ.
 - ❖ Гадаад - “coarse-grained”
 - ❖ Дотоод - “fine-grained.”
- ❖ (i, j) хос бүрийн хувьд бие даасан байдлаар тооцоолно. Цаашлаад үр дүнгийн утга нь к-ийн утга бүрийн нийлбэр байдлаар тооцоологдоно.
 - ❖ (i,j) дугаарлалтын олонлог дээр параллелчлана.
 - ❖ к дугаар бүрт sum-reduction-ыг зэрэгцээгээр гүйцэтгэнэ.

Баярлалаа.