

C++14 MULTITHREADING алгоритмууд болон OPENMP алгоритмууд

Зорилго: Лекцийн хичээл дээр үзсэн ойлголтуудаа бататгана.

- ▼ Multithreading C++ дээр Atomic, Mutex болон OpenMP дээр парааллел reduction хэрэглэж сурах дадлагажих

1. **COMPARE-AND-SWAP** хэрэглэн **MAX-REDUCTION** гүйцэтгэнэ. Дараа нь (A) **Mutex** болон (Б) **Atomic** ашиглан хэсэгчилсэн үр дүнг нэгтгэнэ.

```
#include <iostream>
#include <cstdint>
#include <vector>
#include <thread>
#include <atomic>
#include <mutex>
#include "../include/hpc_helpers.hpp"

// Atomic Parallel-reduction using compare-and-swap

int CAS(int *ptr,int oldvalue,int newvalue)
{
    int temp = *ptr;
    if(*ptr == oldvalue)
        *ptr = newvalue;
    return temp;
}

int main( ) {

    std::mutex mutex;
    std::vector<std::thread> threads;
    const uint64_t num_threads = 10;
    const uint64_t num_iters = 100'000'000;
    // 1 thread = 10,000,000

    // WARNING: this closure produces incorrect results
    auto mutex_max =
        [&] (volatile uint64_t* max,
            const auto& id) -> void {

            for (uint64_t i = id; i < num_iters; i += num_threads){
                std::lock_guard<std::mutex> lock_guard(mutex);
                if(i > *max){
                    *max = i;
                }
            }
        }
```

```

    }
}

};

auto atomic_max =
    [&] (volatile std::atomic<uint64_t>* max_value,
        const auto& id) -> void {

        for (uint64_t i = id; i < num_iters; i += num_threads) {
            auto previous = max_value->load();
            while (previous < i && !max_value->compare_exchange_weak(previous,
                i)) {
            }
        }
    };

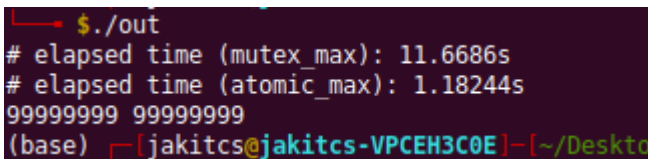
// mutex
TIMERSTART(mutex_max)
uint64_t m_max = 0;
threads.clear();
for (uint64_t id = 0; id < num_threads; id++) // 10
    threads.emplace_back(mutex_max, &m_max, id);
for (auto& thread : threads)
    thread.join();
TIMERSTOP(mutex_max)

// ATOMIC
TIMERSTART(atomic_max)
std::atomic<uint64_t> a_max(0);
threads.clear();
for (uint64_t id = 0; id < num_threads; id++)
    threads.emplace_back(atomic_max, &a_max, id);
for (auto& thread : threads)
    thread.join();
TIMERSTOP(atomic_max)

std::cout << m_max << " " << a_max << std::endl;
}

```

Үрдүн:



```

$ ./out
# elapsed time (mutex_max): 11.6686s
# elapsed time (atomic_max): 1.18244s
99999999 99999999
(base) [jakitcs@jakitcs-VPCEH3C0E] - [~/Desktop

```

Дүгнэлт: CAS(compare-and-swap) аргыг ашигласан multithread дээр atomic::compare_exchange_weak нь адилхан юм. Энэ нь дурын элементийн, санах ойн байршлыг өөрчлөх боломжийг олгодог бөгөөд ингэснээр олон процессортой мөргөлдөхөөс сэргийлдэг Параллел алгоритмд үр дүнтэй юм.

CAS:

1. хувьсагчийн бодит утгыг хувьсагчийн хүлээгдэж буй утгатай харьцуулна.
2. хэрэв бодит утга нь хүлээгдэж буй утгатай тохирч байвал
3. хувьсагчийн бодит утгыг шилжүүлсэн шинэ утгаар солино.

2. N урттай float массив дахь хамгийн их элементийг тооцоолох үр дүнтэй параллел програм бич. Openmp parallel reduction

```
#include <iostream>
#include <cstdint>
#include <cstdlib>
#include <ctime>
#include <limits>
#include <omp.h>

#include "../include/hpc_helpers.hpp"

int main() {

    const uint64_t num_threads = 10;
    const uint64_t n = 100'000'000;
    omp_set_num_threads(num_threads);

    TIMERSTART(alloc)
    float *A = (float*)malloc(n*sizeof(float));
    TIMERSTOP(alloc)

    float max_val = 0;

    TIMERSTART(init)
    #pragma omp parallel for
    for(uint64_t i = 0; i < n; i++)
        A[i] = float(5 + rand() % (150 + 1 - 5)) / 100;
    // 0 - ooc 1 хүртэл random float numbers
    TIMERSTOP(init)

    TIMERSTART(max)
    #pragma omp parallel for reduction(max:max_val)
    for(uint64_t i = 0; i < n; i++)
        max_val = max_val > A[i] ? max_val : A[i];
    // max = reduction operator
    // max_val = reduction variable
    TIMERSTOP(max)

    // for(uint64_t j = 0; j <10; j++){
    //     std::cout<< A[j]<<std::endl;
    // }

    std::cout<<"max val: "<< max_val <<std::endl;
}
```

Үр дүн:

```
# elapsed time (alloc): 1.8297e-05s
# elapsed time (init): 18.2659s
# elapsed time (max): 0.053353s
max val: 1.5
(base) [jakitcs@jakitcs-VPCEH3C0E]~/Desktop/p
```