Kraków 16.01.2019

Sprawozdanie z seminarium Intel TBB

Autor: Patryk Chodur
Informatyka Stosowana
Semestr III
Nr Indeksu: 292487
Prowadzący: dr Bartosz Mindur
Data seminarium: 3.01.2019

Przebieg ćwiczenia:

Ćwiczenie polegało na zrobieniu 3 zadań związanych z seminarium. Rozwiązanie polegało na napisaniu w całości plików nagłówkowych o rozszerzeniu .h tak, aby kolejne zadania kompilowały się, a wynik ich działania był analogoczny do przedstawionego w plikach main.cpp

Zadania:

Pierwsze zadanie

polegało na przećwiczeniu używania klasy tbb::blocked_range oraz realizacji wielowątkowości za pomocą podstawowych algorytmów wbudowanych w bibliotekę:
tbb::parallel_for - wpisywanie do tablicy 2 wymiarowej kolejnych wartości
tbb::parallel_reduce - zsumowanie wszystkich elementów tablicy
tbb::parallel_pipeline - stworzenie zestawu filtrów, które podnosiły do kwadratu elementy wczytane z tablicy, a następnie wpisywały je w odpowiedniej kolejności z powrotem do tablicy

Zalety:

Zadanie pokazywało najbardziej podstawowe i najprostsze algorytmy znajdujące się w bibliotece, dzięki którym łatwo można zacząć programowanie wielowątkowe.

Wady:

Zadanie mogło się wydawać trochę naciągane i trywialne, choć dzięki temu można się było skupić na funkcjonalności algorytmów

Drugie zadanie

polegało na przećwiczeniu implementacji wielowątkowości za pomocą klasy tbb::task_group, używania atomic expressions, mutexów oraz miało zachęcić do korzystania z wbudowanych w bibliotekę kontenerów. Zadanie polegało na stworzeniu klas producenta i konsumenta używających tbb::concurrent_queue (korzystanie ze zwykłej kolejki std::queue w tym zadaniu wiązało się z oczywistymi problemami wywołanymi przez wielowątkowość).

Zalety:

Zadanie na znanym już schemacie uczyło trochę bardziej zaawansowanych, aczkolwiek niezwykle przydatnych elementów biblioteki.

Wady:

Zadanie wymuszało użycie wzajemnego wykluczania w celu poprawnego wyświetlenia przeprowadzanych operacji.

Trzecie zadanie

polegało na stworzeniu funktora używającego algorytmu quicksort oraz wielowątkowości. Klasę należało zbudować w zgodzie z mechanizmem tworzenia zadań za pomocą klasy tbb::task tak, aby wykonywał się rekurencyjnie alokując kolejne zadania, dzieląc obiekt klasy tbb::blocked_range, który wyznacza nam zakres działania konkretnego wywołania. W przypadku, gdy obiekt był niepodzielny (metoda is_divisible()) należało użyć dowolnego prostego algorytmu iteracyjnego o złożoności O(n^2).

Zalety:

Na użytecznym przykładzie uczyło mechanizmu alokowania zadań oraz pokazywało praktyczne zastosowanie klasy blocked_range.

Wady:

Możliwe, że poziom trudności był większy, ze względu na konieczność zadbania o prawidłowe zachowanie na granicach zakresów. Ponadto użycie tbb::concurrent_vector byłoby lepsze niż std::vector (pomimo iż wątki pracowały na różnych fragmentach vectora, current_vector ma zaimpementowaną optymalizację użycia pamięci podręcznej procesora).

Uwagi:

Instalacja biblioteki mogła stanowić drobny problem. W seminarium wiele ważnych elementów zostało pominiętych, jako że aby opowiedzieć o całej bibliotece prawdopodobnie dałoby się zrobić osobny przedmiot, choć z perspektywy czasu nie jestem pewien, czy nie powinienem był rozszerzyć tematu alokacji w bibliotece, choć głównym założeniem seminarium było zachęcenie ludzi do używania biblioteki. Ponadto w zadaniach mógł się pojawić temat wyjątków.