Advanced Programming - Supplementary Examples

C++ Parallel Programming with OpenMP

This document provides additional exercises and examples related to OpenMP and parallel programming in C++. Please ensure that you compile these examples with the OpenMP flag enabled (e.g., `g++ -fopenmp -std=c++17 example.cpp`).

Examples

Simple Parallel Region with OpenMP

This example is an alternative implementation for the first OpenMP example in the supplementary material handout.

```cpp
#include <omp.h>
#include <iostream>
using namespace std;

int main() {
    #pragma omp parallel
    {
        int thread_id = omp_get_thread_num();
        cout << "Hello from task: " << thread_id << endl;
    }
    return 0;
}
```

Dynamic Scheduling in OpenMP

This example demonstrates the use of dynamic scheduling in OpenMP. It allows threads to dynamically pick up chunks of iterations from a loop as they complete, which is useful when the loop iterations take different amounts of time to complete.

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

void process_task(int i) {
    printf("Processing task %d on thread %d\n", i, omp_get_thread_num());
}

int main() {
    int num_tasks = 20;

    // Parallel region with dynamic scheduling
    #pragma omp parallel for schedule(dynamic, 2) num_threads(4)
    for (int i = 0; i < num_tasks; ++i) {
        process_task(i);
    }

    return 0;
}
```

Exercise: Modify the above example to experiment with different chunk sizes (e.g., 1, 4, etc.) and observe how it impacts load balancing across the threads.

Parallel Matrix Multiplication

This example shows how to parallelise matrix multiplication using OpenMP. This is a more computationally intensive task where we can gain significant speedup through parallelisation.

```
#include <omp.h>
#include <iostream>
using namespace std;

const int N = 3;

void matrixMultiply(int a[N][N], int b[N][N], int c[N][N]) {
    #pragma omp parallel for collapse(2)
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            c[i][j] = 0;
            for (int k = 0; k < N; ++k) {
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
}

int main() {
    int a[N][N] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int b[N][N] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
    int c[N][N];

    matrixMultiply(a, b, c);

    cout << "Result of matrix multiplication:" << endl;
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            cout << c[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Exercise: Measure the execution time of both the parallelised and the serial version of matrix multiplication. Experiment with different matrix sizes and number of threads to analyse the speedup.

Nested Parallelism

This example showcases nested parallelism, where parallel regions are created within another parallel region. OpenMP supports nested parallelism, but you need to enable it explicitly.

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

```

```
void inner_parallel() {
    #pragma omp parallel num_threads(2)
    {
        printf("Inner thread %d, outer thread %d\n", omp_get_thread_num(),
        ↪  omp_get_ancestor_thread_num(1));
    }
}

int main() {
    omp_set_nested(1); // Enable nested parallelism

    #pragma omp parallel num_threads(3)
    {
        printf("Outer thread %d\n", omp_get_thread_num());
        inner_parallel();
    }

    return 0;
}
```

Exercise: Modify the example to create three levels of nested parallelism. Analyze how the thread hierarchy is handled in nested parallelism.

OpenMP Tasking

This example introduces the concept of tasking in OpenMP, where independent units of work (tasks) are created. Tasking is useful when the work to be done is irregular and cannot be evenly divided between threads.

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

void independent_task(int task_num) {
    printf("Executing task %d on thread %d\n", task_num, omp_get_thread_num());
}

int main() {
    #pragma omp parallel
    {
        #pragma omp single
        {
            for (int i = 0; i < 10; ++i) {
                #pragma omp task
                independent_task(i);
            }
        }
    }

    return 0;
}
```

Exercise: Modify the example to wait for all tasks to finish using `#pragma omp taskwait`. How does the execution change when tasks are not synchronised?