Пухов Максим Станиславович (ИС-242)

Отчёт

Лабораторная работа 5. Параллелизм задач

Описание работы программы

Функция разбиения для быстрой сортировки

```
void partition(int *v, int *i, int *j, int low, int high)
{
    *i = low;
    *j = high;
    int pivot = v[(low + high) / 2];
    do
    {
        while (v[*i] < pivot)
            (*i)++;
        while (v[*j] > pivot)
            (*j)--;
        if (*i <= *j)
        {
            swap(&(v[*i]), &(v[*j]));
            (*i)++;
            (*j)--;
        }
    } while (*i <= *j);
}
```

- **`partition`**: Выполняет разбиение массива на две части относительно опорного элемента (pivot).

Параллельная и последовательная быстрая сортировка

```
void quicksort_tasks(int *v, int low, int high)
{
    int i, j;
    partition(v, &i, &j, low, high);
    if (high - low < THRESHOLD || (j - low < THRESHOLD || high - i < THRESHOLD))
    {
        if (low < j)
            quicksort_tasks(v, low, j);
        if (i < high)
            quicksort_tasks(v, i, high);
    }
    else
    {
#pragma omp task untied // Открепить задачу от потока (задачу может выполнять любой поток)
        {
            quicksort_tasks(v, low, j);
        }
        quicksort_tasks(v, i, high);
    }
}

void quicksort(int *v, int low, int high)
{
    int i, j;
    // print_arr(v);
    partition(v, &i, &j, low, high);
    if (low < j)
        quicksort(v, low, j);
    if (i < high)
        quicksort(v, i, high);
}
```

- **`quicksort_tasks`**: Параллельная версия быстрой сортировки с использованием OpenMP задач.
- **`quicksort`**: Последовательная версия быстрой сортировки.

Инициализация и печать массива

```
void init(int **arr)
{
    for (int i = 0; i < N; i++)
        (*arr)[i] = rand() % 100;
}

void print_arr(int *arr)
{
    for (int i = 0; i < N; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

- **`init`**: Заполняет массив случайными числами от 0 до 99.
- **`print_arr`**: Печатает массив (не используется в основной программе, оставлено для отладки).

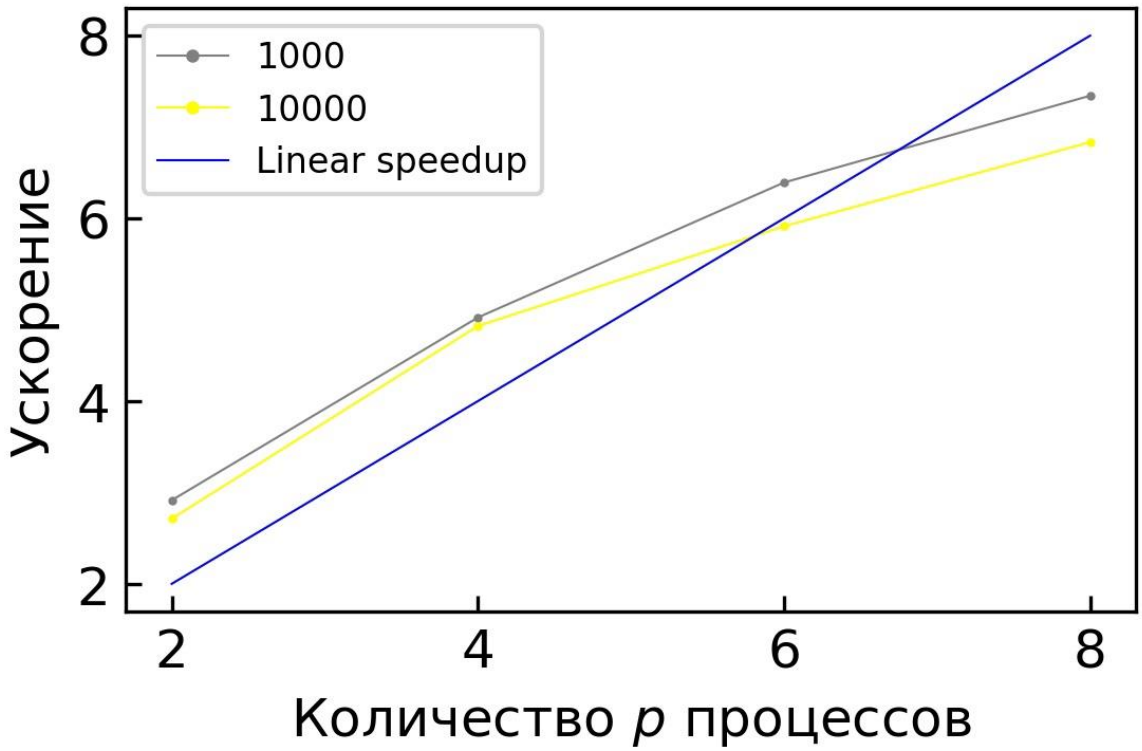
Основная функция

```
int main()
{
    int *arr = malloc(sizeof(int) * N);
    init(&arr);
    // print_arr(arr);
    double t = wtime();
    quicksort(arr, 0, N - 1);
    t = wtime() - t;
    printf("%lf - время последовательной программы\n", t);
// quicksort(arr, 0, N - 1);
    for (int i = 2 ;i < 8; i+=2)
    {
        double time = wtime();
    #pragma omp parallel num_threads(i)
        {
        #pragma omp single
            quicksort_tasks(arr, 0, N - 1);
        }
    time = wtime() - time;
    printf("время работы паралел прог - %lf, потоков - %d  speedup:%lf\n", time,i, t/time);
    }
    return 0;
}
```

- **`main`**: Основная функция программы, где:
- Выделяется память для массива.
- Массив инициализируется случайными числами.
- Выполняется последовательная быстрая сортировка и измеряется время выполнения.
- Выполняется параллельная быстрая сортировка с различным количеством потоков (от 2 до 6) и измеряется время выполнения и ускорение.

Ключевой момент заключается в использовании OpenMP для параллельного выполнения быстрой сортировки с задачами (`#pragma omp task`). Это позволяет эффективно использовать многопоточность для больших массивов, уменьшая время выполнения.

График



Характеристики процессора

Процессор:	AMD A4-7300 APU with Radeon HD Graphics	3.80 GHz

Базовая скорость:	3,80 ГГц
Сокетов:	1
Ядра:	1
Логических процессоров:	2
Виртуализация:	Включено