项目结构如下：

```
.
├── libparallel-for
├── omp-mat-mul
├── omp-schedule
├── README.pdf
└── report.pdf
```

- omp-mat-mul 存放 OpenMP 矩阵乘法的代码
- omp-schedule 存放对比静态和动态调度区别的代码
- libparallel-for 存放实现 parallel_for 函数的代码

编译运行方法：

- omp-mat-mul
  - 编译：

    ```
    cd omp-mat-mul
    mkdir build && mkdir build
    make
    ```

  - 运行：

    ```
    make test
    ```

    执行后，线程数从 1 到 8，矩阵规模从 512 到 2048，进行测试并统计运行时间，数据会输出到 `asset` 子目录下，文件命名为 `time_<thread>`，其中 `<thread>` 指的是线程数，文件内容是在 512 ～ 2048 规模的矩阵下，朴素矩阵乘法和 OpenMP 矩阵乘法的耗时，每行两个。

    ```
    make plot
    ```

    执行后，会统计数据并画图，以矢量图格式保存在 `asset` 下，文件命名为 `Performance_<thread>.svg`。
- omp-schedule
  - 编译：

    ```
    cd omp-schedule
    mkdir build && mkdir bin
    make
    ```

  - 运行：

    ```
    make test
    ```

    同上。

    ```
    make plot
    ```

    同上。
- libparallel-for
  - 编译：

```
cd libparallel-for
mkdir build && mkdir lib
make lib
```

执行后，会编译制作名为 `libparallel-for.so` 的共享库，存放在 `lib` 子目录下。

```
make test
```

继续执行该条命令，会将 `test.cpp` 编译并与 `libparallel-for.so` 链接，生成可执行文件 test。

- 运行：

```
make run
```

默认矩阵规模为 1024，线程数为 8，对我们写的 parallel_for 函数进行测试，并与 gemm 进行性能对比。