workshop优化

fork

优化

fork

```shell
git clone https://github.com/xingfeng2510/workshop.git
cd workshop
make

./workshop
```

```
ubuntu@VM-0-11-ubuntu:~/workshop$ sudo time ./workshop
Matrix multiply iteration 1: cost 35.525 seconds
Matrix multiply iteration 2: cost 36.618 seconds
Matrix multiply iteration 3: cost 33.435 seconds
Matrix multiply iteration 4: cost 33.503 seconds
Matrix multiply iteration 5: cost 33.881 seconds
Matrix multiply iteration 6: cost 34.330 seconds
Matrix multiply iteration 7: cost 34.003 seconds
Command terminated by signal 9
535.10user 2.40system 4:34.00elapsed 196%CPU (0avgtext+0avgdata 263356maxresident)k
0inputs+0outputs (0major+360406minor)pagefaults 0swaps
ubuntu@VM-0-11-ubuntu:~/workshop$
```

原始程序在第7次迭代后因内存超限被 `signal 9(SIGKILL)` 终止。平均在34.4s

优化

1、 `main.c` 中 `Multiplyonce` 函数调用了 `ParallelMultiply` ，后者通过多线程执行 `mul` 函数。

矩阵乘法本身是三重循环，是计算密集型。

2、

- 循环顺序：原代码中循环顺序是 `i-j-k`，即外层循环是行 `i`，中层是列 `j`，内层是累加 `k`。这种顺序可能导致对内存的非连续访问，降低缓存效率。
- 额外操作：每次计算完 `c[i][i]` 后调用 `vector_append`，但这个向量未被使用。
- 未初始化内存：`c` 数组在计算前未清零。

mul.c — Shell

```
void mul(int msize, int tidx, int numt, Vector *vec, TYPE a[][NUM], TYPE b[][NUM], TYPE c[][NUM], TYPE t[][NUM]) {
    int i, j, k;
    for (i = tidx; i < msize; i += numt) {
        // 初始化c[i][j]为0
        for (j = 0; j < msize; j++) {
            c[i][j] = 0.0;
        }
        // 调整循环顺序为i-k-j
        for (k = 0; k < msize; k++) {
            // 缓存a[i][k]减少重复访问
            TYPE a_ik = a[i][k];
            for (j = 0; j < msize; j++) {
                c[i][j] += a_ik * b[k][j];
            }
        }
    }
}
```

3、在 `thrmode1.c` 的线程参数中，每个线程通过 `vector_create()` 创建了 `Vector` 对象，但未调用 `vector_destroy()` 释放内存。注：

thrmodel.c — Shell

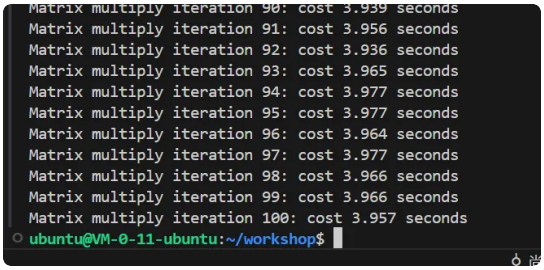
```
    // destroy vector here 33line
    vector_destroy(par->vec);  //34 line
```

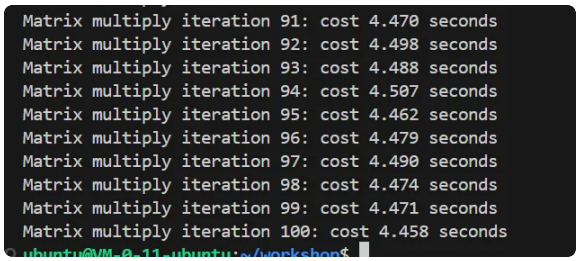
4、`-03` 会开启 `-02` 所包含的所有优化，提高程序的运行性能。编译时间的增加不再考虑范围内。

Makefile — Shell

```
# CFLAGS  = -g -O2 -fno-asm
CFLAGS  = -g -O3 -fno-asm
```

-O3的优化结果在3.96 s 左右，-O2的优化结果在4.47 s 左右。





（a） O3

（b）O2

优化倍数分别是：34.4/3.96≈8.67

34.4/4.47≈7.69

附CPU利用率 `htop`