实验：Optlab

一、实验目的

通过修改query.cpp文件，完成optlab，理解并实现各类程序优化的方法，如循环展开等 。

二、设计思路

根据数据和dataload.h文件里的loadTable函数可知，query.cpp的main函数会以|为分隔符读入lineorder.tbl的数据，并对每列的数据做加减乘操作，最后输出结果和所用时间，而我们的任务则是优化算数操作以减少所有时间。

方便起见，在要修改的部分先用一些名称较短的指针代替lineorder_table->info.table里的一些变量：

```
    int rows=lineorder_table_info.rows;
    int* l_q=lineorder_table_info.table -> lo_quantity;
    int* l_e=lineorder_table_info.table -> lo_extendedprice;
    double* l_d=lineorder_table_info.table -> lo_discount;
    double* l_t=lineorder_table_info.table -> lo_tax;
    int* l_o=lineorder_table_info.table -> lo_orderdate;

    for (int i = 0; i < rows; ++i) {
        quantity_sum = quantity_sum + l_q[i];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) *
                                   (1 + l_t[i] );

        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];
        }

        if (l_o[i] <= limit_orderdate) {
            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);
        }

        if (l_o[i] <= limit_orderdate) {
            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
    }
```

发现三个if语句的条件是一样的，故可以将它们合并：

```
    int rows=lineorder_table_info.rows;
    int* l_q=lineorder_table_info.table -> lo_quantity;
    int* l_e=lineorder_table_info.table -> lo_extendedprice;
    double* l_d=lineorder_table_info.table -> lo_discount;
    double* l_t=lineorder_table_info.table -> lo_tax;
```

```
    int* l_o=lineorder_table_info.table -> lo_orderdate;

    for (int i = 0; i < rows; ++i) {
        quantity_sum = quantity_sum + l_q[i];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) *
                                   (1 + l_t[i] );

        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
    }
```

根据所学知识和cachelab的经验，可能是同一次循环中的后一个变量b[i]覆盖了前一个变量a[i]导致下一次循环时要重新加载a[i+1]而导致冲突不命中使运行时间增加，故可用类似矩阵乘法的方式对元素进行分块处理，即一次处理多个相同类型的变量。

因为一开始不明确cache的大小，故分别取单次分块2、3、4的大小做了多次尝试最后得出较好的一种分块方式。代码如下：

二个一组：

```
    int rows=lineorder_table_info.rows;
    int* l_q=lineorder_table_info.table -> lo_quantity;
    int* l_e=lineorder_table_info.table -> lo_extendedprice;
    double* l_d=lineorder_table_info.table -> lo_discount;
    double* l_t=lineorder_table_info.table -> lo_tax;
    int* l_o=lineorder_table_info.table -> lo_orderdate;
    int i=0;
    for (i; i < rows-2; i+=2) {
        quantity_sum = quantity_sum + l_q[i] + l_q[i+1];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]) +
l_e[i+1] * (1 - l_d[i+1]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) * (1 + l_t[i] )+l_e[i+1] * (1 - l_d[i+1]) * (1 + l_t[i+1] );

        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
        if (l_o[i+1] <= limit_orderdate) {
```

```
            quantity_sum_with_condition = quantity_sum_with_condition +
l_q[i+1];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i+1] * (1 - l_d[i+1]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i+1] * (1 - l_d[i+1]) * (1 + l_t[i+1] );
        }
    }
    for (i; i < rows; i++) {
        quantity_sum = quantity_sum + l_q[i];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) *
                                    (1 + l_t[i] );

        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
    }
```

三个一组:

```
    int rows=lineorder_table_info.rows;
    int* l_q=lineorder_table_info.table -> lo_quantity;
    int* l_e=lineorder_table_info.table -> lo_extendedprice;
    double* l_d=lineorder_table_info.table -> lo_discount;
    double* l_t=lineorder_table_info.table -> lo_tax;
    int* l_o=lineorder_table_info.table -> lo_orderdate;
    int i=0;
    for (i; i < rows-3; i+=3) {
        quantity_sum = quantity_sum + l_q[i] + l_q[i+1] + l_q[i+2];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]) +
l_e[i+1] * (1 - l_d[i+1]) + l_e[i+2] * (1 - l_d[i+2]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) * (1 + l_t[i] )+l_e[i+1] * (1 - l_d[i+1]) * (1 + l_t[i+1] ) + l_e[i+2] *
(1 - l_d[i+2]) * (1 + l_t[i+2] );

        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);
```

```
            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
        if (l_o[i+1] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition +
l_q[i+1];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i+1] * (1 - l_d[i+1]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i+1] * (1 - l_d[i+1]) * (1 + l_t[i+1] );
        }
                if (l_o[i+2] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition +
l_q[i+2];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i+2] * (1 - l_d[i+2]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i+2] * (1 - l_d[i+2]) * (1 + l_t[i+2] );
        }
    }
    for (i; i < rows; i++) {
        quantity_sum = quantity_sum + l_q[i];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) *
                                    (1 + l_t[i] );

        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
    }
```

四个一组：

```
    int rows=lineorder_table_info.rows;
    int* l_q=lineorder_table_info.table -> lo_quantity;
    int* l_e=lineorder_table_info.table -> lo_extendedprice;
    double* l_d=lineorder_table_info.table -> lo_discount;
    double* l_t=lineorder_table_info.table -> lo_tax;
    int* l_o=lineorder_table_info.table -> lo_orderdate;
```

```
    int i=0;
    for (i; i < rows-4; i+=4) {
        quantity_sum = quantity_sum + l_q[i] + l_q[i+1] + l_q[i+2] + l_q[i+3];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]) +
l_e[i+1] * (1 - l_d[i+1]) + l_e[i+2] * (1 - l_d[i+2]) + l_e[i+3] * (1 -
l_d[i+3]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) * (1 + l_t[i] )+l_e[i+1] * (1 - l_d[i+1]) * (1 + l_t[i+1] )
                        + l_e[i+2] * (1 - l_d[i+2]) * (1 + l_t[i+2] ) +
l_e[i+3] * (1 - l_d[i+3]) * (1 + l_t[i+3] );

        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
        if (l_o[i+1] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition +
l_q[i+1];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i+1] * (1 - l_d[i+1]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i+1] * (1 - l_d[i+1]) * (1 + l_t[i+1] );
        }
                if (l_o[i+2] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition +
l_q[i+2];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i+2] * (1 - l_d[i+2]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i+2] * (1 - l_d[i+2]) * (1 + l_t[i+2] );
        }
                if (l_o[i+3] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition +
l_q[i+3];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i+3] * (1 - l_d[i+3]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i+3] * (1 - l_d[i+3]) * (1 + l_t[i+3] );
        }
```

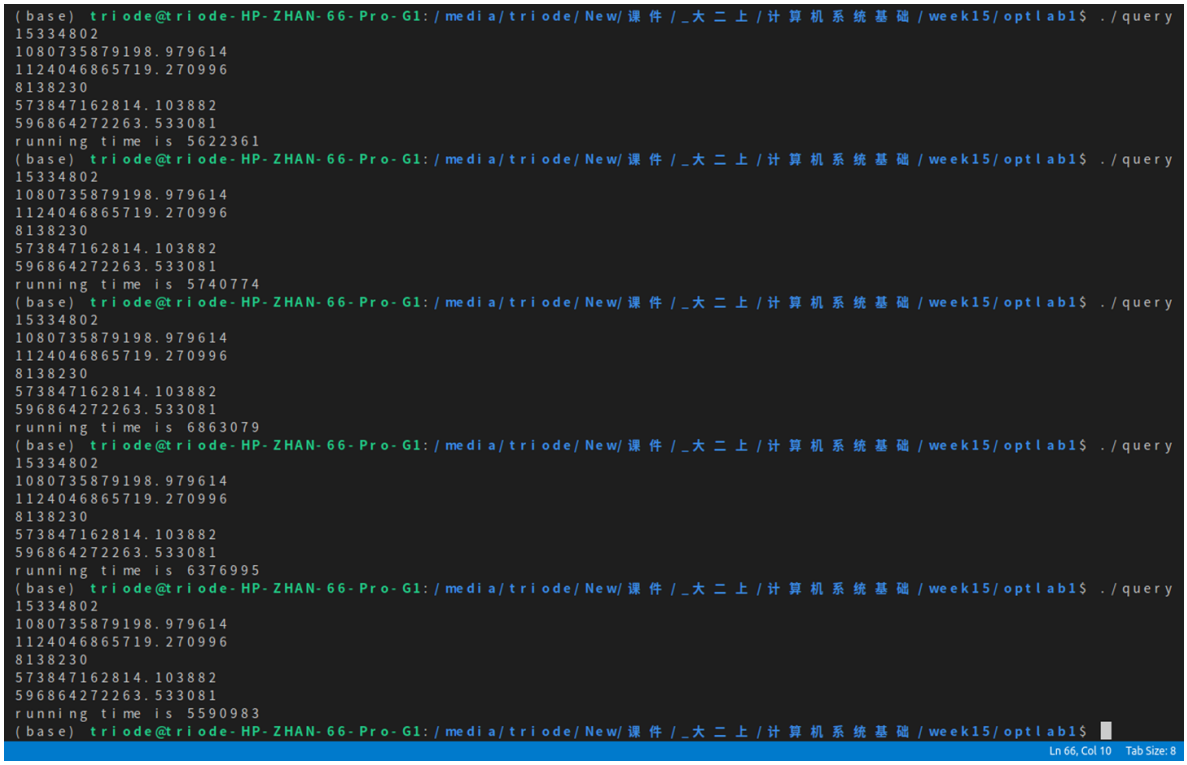
```
    }
    for (i; i < rows; i++) {
        quantity_sum = quantity_sum + l_q[i];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) *
                                   (1 + l_t[i] );

        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
    }
```

三、实验截图及分析

分别在本机和服务器上对每种方式进行了5次实验，下为实验的截图（如果嫌太多可直接跳到后方图表（ctrl+单击）处）。

本机：

原函数：



+2一组

```
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query2
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4431418
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query2
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4479241
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query2
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4095182
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query2
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4388394
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query2
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4074799
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$
Ln 86, Col 28  Spaces: 8  UT
```

+3一组

```
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query3
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 5043049
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query3
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4700550
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query3
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 5384951
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query3
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4790708
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query3
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 5148383
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$
Ln 103, Col 9  Spaces: 8  UT
```

+4一组

```
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query4
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4660822
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query4
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4612210
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query4
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4570999
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query4
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4784125
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$ ./query4
15334802
1080735879198.979614
1124046865719.270996
8138230
573847162814.103882
596864272263.533081
running time is 4632615
(base) triode@triode-HP-ZHAN-66-Pro-G1:/media/triode/New/课件/_大二上/计算机系统基础/week15/optlab1$
```

Ln 86, Col 28 Spaces: 8 UT

服务器

原函数

```
2018202196@VM-0-46-ubuntu:~/optlab$ ./query
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 52103486
2018202196@VM-0-46-ubuntu:~/optlab$ ./query
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 180539250
2018202196@VM-0-46-ubuntu:~/optlab$ ./query
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 95762356
2018202196@VM-0-46-ubuntu:~/optlab$ ./query
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 68775692
2018202196@VM-0-46-ubuntu:~/optlab$ ./query
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 87349966
2018202196@VM-0-46-ubuntu:~/optlab$
```

+2一组

```
2018202196@VM-0-46-ubuntu:~/optlab$ ./query2
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 80470236
2018202196@VM-0-46-ubuntu:~/optlab$ ./query2
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 125673580
2018202196@VM-0-46-ubuntu:~/optlab$ ./query2
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 67055672
2018202196@VM-0-46-ubuntu:~/optlab$ ./query2
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 124800670
2018202196@VM-0-46-ubuntu:~/optlab$ ./query2
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 66195818
2018202196@VM-0-46-ubuntu:~/optlab$
```

+3一组

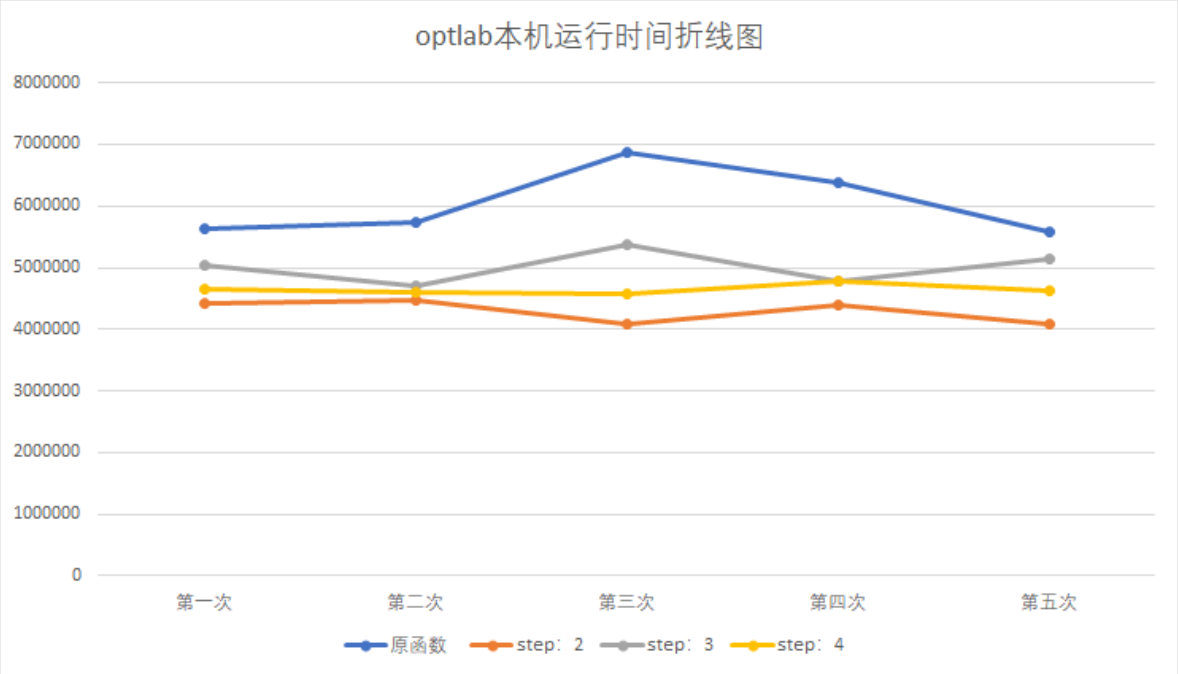
```
2018202196@VM-0-46-ubuntu:~/optlab$ ./query3
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 125835222
2018202196@VM-0-46-ubuntu:~/optlab$ ./query3
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 86953652
2018202196@VM-0-46-ubuntu:~/optlab$ ./query3
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 107107054
2018202196@VM-0-46-ubuntu:~/optlab$ ./query3
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 48642734
2018202196@VM-0-46-ubuntu:~/optlab$ ./query3
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 124699570
2018202196@VM-0-46-ubuntu:~/optlab$
```

+4一组

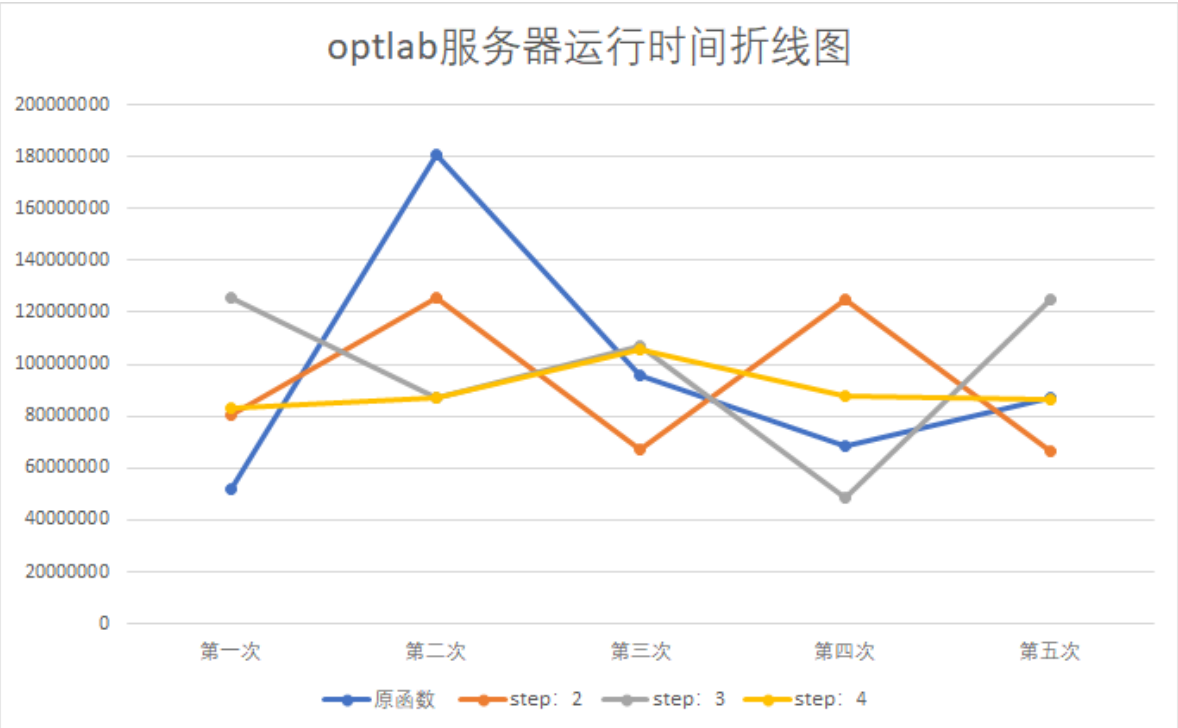
```
2018202196@VM-0-46-ubuntu:~/optlab$ ./query4
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 83172626
2018202196@VM-0-46-ubuntu:~/optlab$ ./query4
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 86771024
2018202196@VM-0-46-ubuntu:~/optlab$ ./query4
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 105979622
2018202196@VM-0-46-ubuntu:~/optlab$ ./query4
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 87695522
2018202196@VM-0-46-ubuntu:~/optlab$ ./query4
153078795
11482644073919.994141
11942201064623.660156
81150336
6087293950606.120117
6330745210686.512695
running time is 86184288
2018202196@VM-0-46-ubuntu:~/optlab$
```

图表

本机	第一次	第二次	第三次	第四次	第五次	平均值
原函数	5622361	5740774	6863079	6376995	5590983	6038838
step：2	4431418	4479241	4095182	4388394	4074799	4293807
step：3	5043049	4700550	5384951	4790708	5148383	5013528
step：4	4660822	4612210	4570999	4784125	4632615	4652154



服务器	第一次	第二次	第三次	第四次	第五次	平均值
原函数	52103486	180539250	95762356	68775692	87349966	96906150
step：2	80470236	125673580	67055672	124800670	66195818	92839195
step：3	125835222	86953652	107107054	48642734	124699570	98647646
step：4	83172626	86771024	105979622	87695522	86184288	89960616



通过图表可以看出，当2个分块和4个分块时，在本机上的效率较高，但2个分块的策略在服务器上运行时间波动较大，而4个分块的则较为稳定。因此得出结论：四个分块的运行时间较短。

四、完整代码

```
#include <cstdio>
#include "dataload.h"
```

```
#define limit_orderdate 19950630

const char lineorder_name[] = "lineorder.tbl";

static __inline__ uint64_t curtick() {
    uint64_t tick;
    unsigned long lo,hi;
    __asm__ __volatile__ ("rdtsc":"=a"(lo),"=d"(hi));
    tick = (uint64_t) hi << 32 | lo;
    return tick;
}

static __inline__ void startTimer(uint64_t *t) {
    (*t) = curtick();
}

static __inline__ void stopTimer(uint64_t *t) {
    (*t) = curtick() - *t;
}


int main() {
    table_info lineorder_table_info;
    FILE * lineorder_file;

    //load lineorder table from file
    lineorder_file = fopen(lineorder_name,"r");
    loadTable(lineorder_file, &lineorder_table_info);

    unsigned int quantity_sum = 0;
    double discount_total_price = 0;
    double tax_discount_total_price = 0;
    unsigned int quantity_sum_with_condition = 0;
    double discount_total_price_with_condition = 0;
    double tax_discount_total_price_with_condition = 0;

    uint64_t beg;
    startTimer(&beg);

    //you should editor the following the part to accelerate the calculation
    /*--------------------------------*/


    int rows=lineorder_table_info.rows;
    int* l_q=lineorder_table_info.table -> lo_quantity;
    int* l_e=lineorder_table_info.table -> lo_extendedprice;
    double* l_d=lineorder_table_info.table -> lo_discount;
    double* l_t=lineorder_table_info.table -> lo_tax;
    int* l_o=lineorder_table_info.table -> lo_orderdate;
    int i=0;
    for (i; i < rows-4; i+=4) {
        quantity_sum = quantity_sum + l_q[i] + l_q[i+1] + l_q[i+2] + l_q[i+3];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]) +
l_e[i+1] * (1 - l_d[i+1]) + l_e[i+2] * (1 - l_d[i+2]) + l_e[i+3] * (1 -
l_d[i+3]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) * (1 + l_t[i] )+l_e[i+1] * (1 - l_d[i+1]) * (1 + l_t[i+1] )
```

```
                        + l_e[i+2] * (1 - l_d[i+2]) * (1 + l_t[i+2] ) +
l_e[i+3] * (1 - l_d[i+3]) * (1 + l_t[i+3] );

        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
        if (l_o[i+1] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition +
l_q[i+1];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i+1] * (1 - l_d[i+1]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i+1] * (1 - l_d[i+1]) * (1 + l_t[i+1] );
        }
                if (l_o[i+2] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition +
l_q[i+2];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i+2] * (1 - l_d[i+2]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i+2] * (1 - l_d[i+2]) * (1 + l_t[i+2] );
        }
                if (l_o[i+3] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition +
l_q[i+3];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i+3] * (1 - l_d[i+3]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i+3] * (1 - l_d[i+3]) * (1 + l_t[i+3] );
        }
    }
    for (i; i < rows; i++) {
        quantity_sum = quantity_sum + l_q[i];
        discount_total_price = discount_total_price + l_e[i] * (1 - l_d[i]);
        tax_discount_total_price = tax_discount_total_price + l_e[i] * (1 -
l_d[i]) *
                                    (1 + l_t[i] );
```

```
        if (l_o[i] <= limit_orderdate) {
            quantity_sum_with_condition = quantity_sum_with_condition + l_q[i];

            discount_total_price_with_condition =
discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]);

            tax_discount_total_price_with_condition =
tax_discount_total_price_with_condition
            + l_e[i] * (1 - l_d[i]) * (1 + l_t[i] );
        }
    }
    /*---------------------------------*/


    stopTimer(&beg);

    //output
    printf("%d\n",quantity_sum);
    printf("%0.6lf\n",discount_total_price);
    printf("%0.6lf\n",tax_discount_total_price);
    printf("%d\n",quantity_sum_with_condition);
    printf("%0.6lf\n",discount_total_price_with_condition);
    printf("%0.6lf\n",tax_discount_total_price_with_condition);
    printf("running time is %ld\n", (long)(beg));
}
```