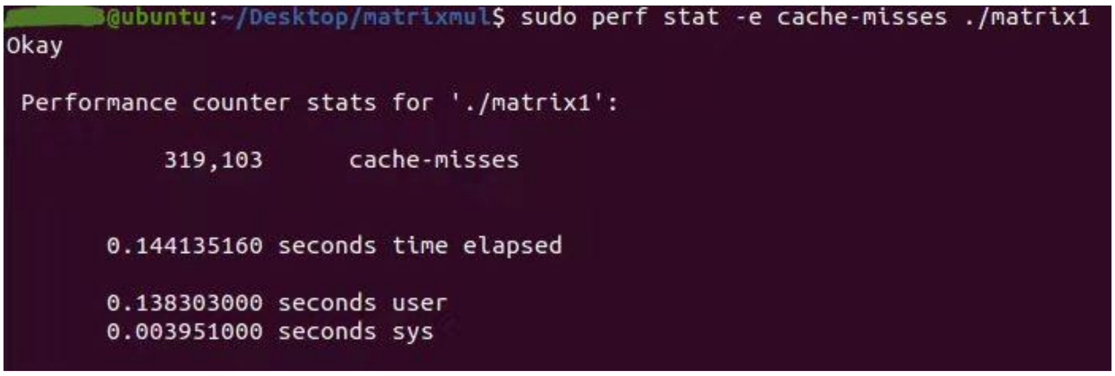
19335015 陈恩婷

说明：鉴于本人的电脑无法在 VMware 中开启虚拟化 CPU 的功能，以下是在同学的电脑上运行的结果，mem-loads 的结果有异常所以没有放出来。

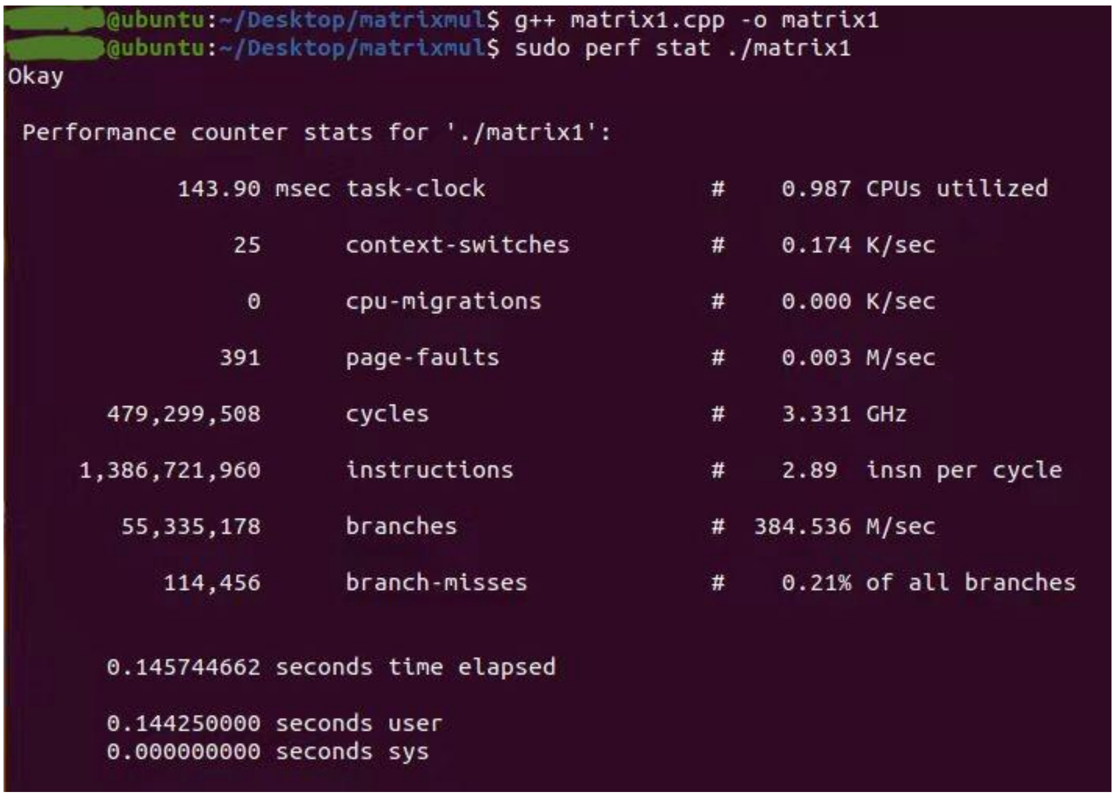
1. 普通算法计算矩阵的乘积

Cache-misses



```
[redacted]@ubuntu:~/Desktop/matrixmul$ sudo perf stat -e cache-misses ./matrix1
Okay

 Performance counter stats for './matrix1':

           319,103      cache-misses


       0.144135160 seconds time elapsed

       0.138303000 seconds user
       0.003951000 seconds sys
```

CPI

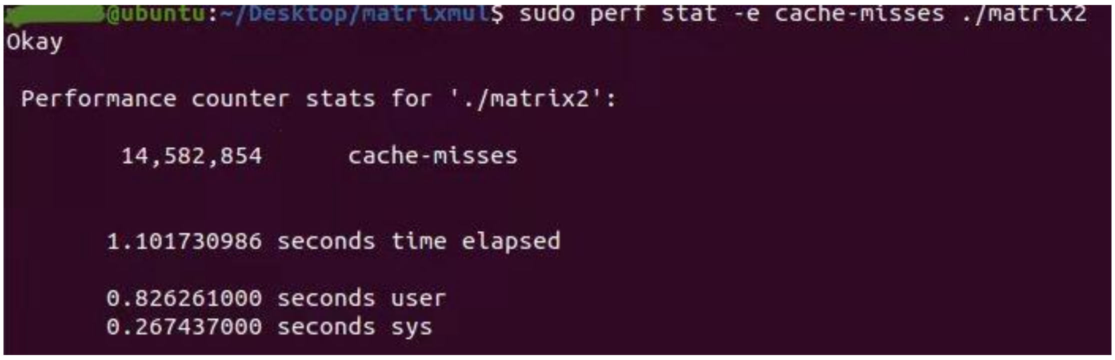


```
[redacted]@ubuntu:~/Desktop/matrixmul$ g++ matrix1.cpp -o matrix1
[redacted]@ubuntu:~/Desktop/matrixmul$ sudo perf stat ./matrix1
Okay

 Performance counter stats for './matrix1':

            143.90 msec task-clock                #    0.987 CPUs utilized
                25      context-switches          #    0.174 K/sec
                 0      cpu-migrations            #    0.000 K/sec
               391      page-faults               #    0.003 M/sec
       479,299,508      cycles                    #    3.331 GHz
     1,386,721,960      instructions              #    2.89  insn per cycle
        55,335,178      branches                  #  384.536 M/sec
           114,456      branch-misses             #    0.21% of all branches

       0.145744662 seconds time elapsed

       0.144250000 seconds user
       0.000000000 seconds sys
```

计算得出 CPI = cycles/instructions = 0.35

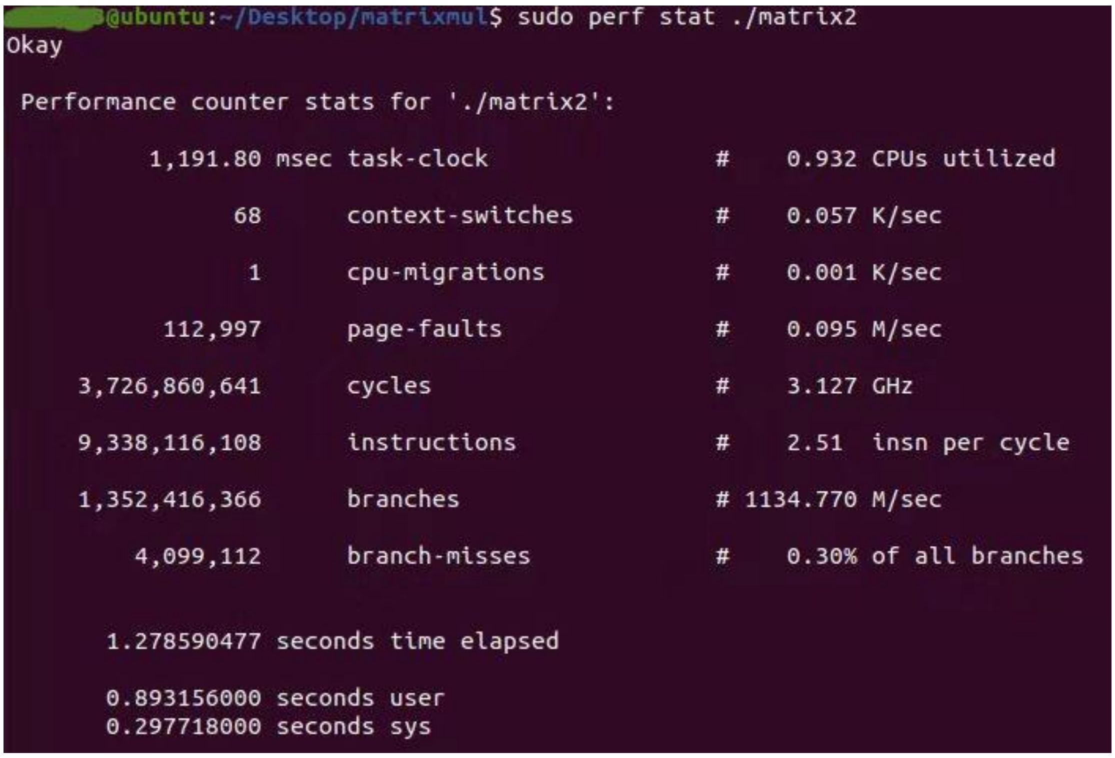
2. 分治算法

Cache-misses



```
[redacted]@ubuntu:~/Desktop/matrixmul$ sudo perf stat -e cache-misses ./matrix2
Okay

 Performance counter stats for './matrix2':

        14,582,854      cache-misses


       1.101730986 seconds time elapsed

       0.826261000 seconds user
       0.267437000 seconds sys
```

CPI



```
[redacted]@ubuntu:~/Desktop/matrixmul$ sudo perf stat ./matrix2
Okay

 Performance counter stats for './matrix2':

          1,191.80 msec task-clock                #    0.932 CPUs utilized

                68      context-switches          #    0.057 K/sec

                 1      cpu-migrations            #    0.001 K/sec

           112,997      page-faults               #    0.095 M/sec

     3,726,860,641      cycles                    #    3.127 GHz

     9,338,116,108      instructions              #    2.51  insn per cycle

     1,352,416,366      branches                  # 1134.770 M/sec

         4,099,112      branch-misses             #    0.30% of all branches


       1.278590477 seconds time elapsed

       0.893156000 seconds user
       0.297718000 seconds sys
```

计算得出 CPI = cycles/instructions = 0.40

可见分治算法的 Cache-misses 明显多于普通算法，cycle 和 instructions 也要多得多。这可能是因为分治算法需要申请比较多的内存，频繁的内存操作导致的运行时间增加。