week-17

JX-Ma

2024/1/12

1 本周工作

本周的工作是分别使用 linux 命令 time 和 valgrind 工具测量了无优化，index-hoist,output-element-hoist,RB,unroll,simd, 这 6 种算法的内存使用量。

2 实验环境

- 系统: Ubantu 22.01
- gcc version : 9.5.0
- 优化选项: -O3
- cpu:AMD Ryzen 7 6800H 3.20GHz
- inputTensor: 10,4,232,232
- filterTensor: 96,4,12,12
- outputTensor: 10,96,56,56
- stride : 4

3 实验结果

3.1 none-opt

```
void directConvolutuion(FTensor1D& input, FTensor1D& fiter ,FTensor1D& output,size_t s){
    float* inptr = input.getDataPtr();
    float* fitr = fiter.getDataPtr();
    float* outptr = output.getDataPtr();
    for (size_t i = 0; i < output.batch; ++i){
        for (size_t j = 0; j < output.channel; ++j) {
            for (size_t m = 0; m < output.height; ++m){
                for (size_t n = 0; n < output.width; ++n){
                    for (size_t r = 0; r < input.channel; ++r){
                        for (size_t u = 0; u < fiter.height; ++u){
                            for (size_t v = 0; v < fiter.width; ++v) {
                                outptr[i*output.channel*output.width*output.height + j*output.width*output.height + m*output.width + n] +=
                               inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v]*
                               fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];
                            }
                        }
                    }
                }
            }
         }
    }
}
```

使用 time

```
    Command being timed: "./main"
    User time (seconds): 1.71
    System time (seconds): 0.01
    Percent of CPU this job got: 99%
    Elapsed (wall clock) time (h:mm:ss or m:ss): 0:01.72
    Average shared text size (kbytes): 0
    Average unshared data size (kbytes): 0
    Average stack size (kbytes): 0
    Average total size (kbytes): 0
    Maximum resident set size (kbytes): 23296
    Average resident set size (kbytes): 0
    Major (requiring I/O) page faults: 0
    Minor (reclaiming a frame) page faults: 8176
    Voluntary context switches: 1
    Involuntary context switches: 2
    Swaps: 0
    File system inputs: 0
    File system outputs: 0
    Socket messages sent: 0
    Socket messages received: 0
    Signals delivered: 0
    Page size (bytes): 4096
    Exit status: 0
```

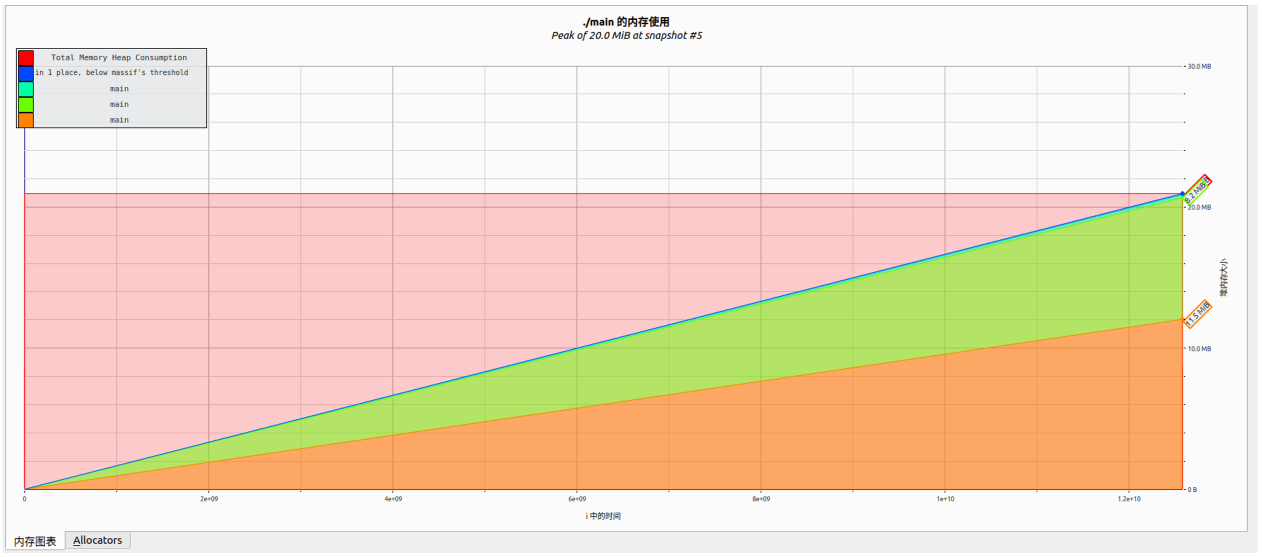

图 1: memory-directConvolutuion

3.2 index-hoist

```
void directConvolution_index_hoist( FTensor1D& input, FTensor1D& fiter ,FTensor1D& output,size_t s){
    float *inptr, *fitr,*outptr;
    for (size_t i = 0; i < output.batch; ++i){
        size_t input_i = i * input.channel * input.height * input.width;
        size_t output_i = i * output.channel * output.height * output.width;
        for (size_t j = 0; j < output.channel; ++j) {
            size_t output_j = j * output.height * output.width;
            size_t filter_j = j * fiter.channel * fiter.height * fiter.width;
            for (size_t m = 0; m < output.height; ++m){
                size_t output_m = m * output.width;
                size_t ms = m * s * input.width;
                for (size_t n = 0; n < output.width; ++n){
                    outptr = output.getDataPtr() + output_i + output_j + output_m + n;
                    size_t ns = n * s;
                    for (size_t r = 0; r < input.channel; ++r){
                        size_t input_r = r * input.height * input.width;
                        size_t fiter_r = r * fiter.height * fiter.width;
                        for (size_t u = 0; u < fiter.height; ++u){
                            inptr = input.getDataPtr() + input_i + input_r + ms + u *input.width+ ns;
                            fitr = fiter.getDataPtr() + filter_j + fiter_r + u * fiter.width;
                            for (size_t v = 0; v < fiter.width; ++v) {
                                *outptr+= *(inptr + v) * *(fitr + v);
                            }
                        }
                    }
                }
            }
        }
    }
}
```

使用 time

```
index-hoist
Command being timed: "./main"
User time (seconds): 1.76
System time (seconds): 0.02
Percent of CPU this job got: 100%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:01.78
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 23296
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 8176
Voluntary context switches: 1
Involuntary context switches: 7
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

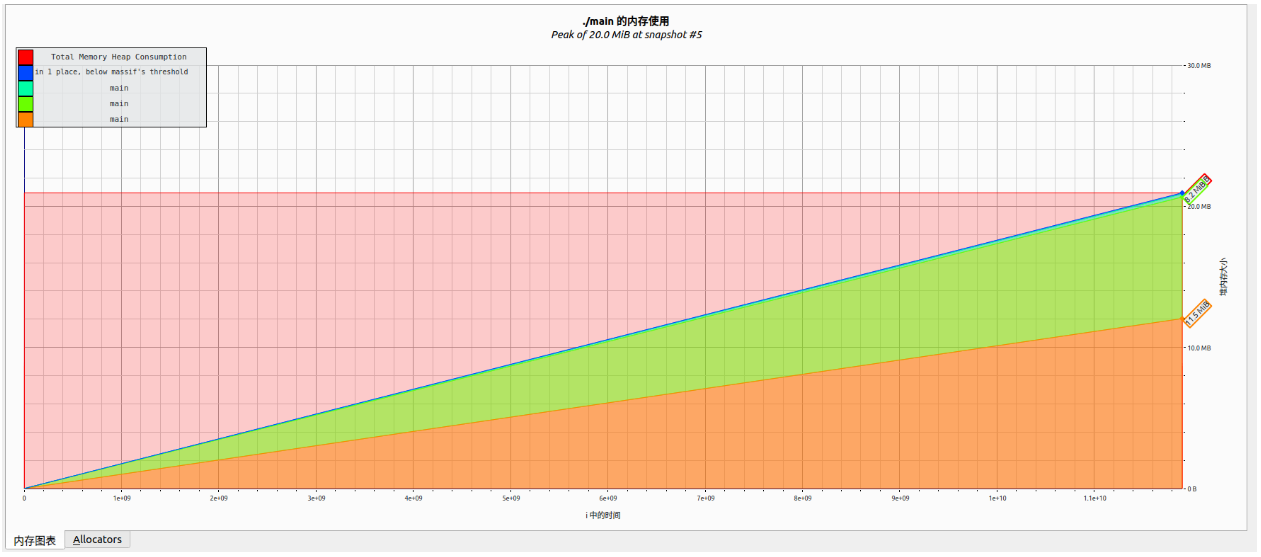

图 2: memory-index-hoist

3.3 hoist-output-element

```
void directConvolution_hoist_c(FTensor1D& input, FTensor1D& fiter,FTensor1D& output,size_t s){
    float* inptr = input.getDataPtr();
    float* fitr = fiter.getDataPtr();
    float* outptr = output.getDataPtr();
    for (size_t i = 0; i < output.batch; ++i){
        for (size_t j = 0; j < output.channel; ++j) {
            for (size_t m = 0; m < output.height; ++m){
                for (size_t n = 0; n < output.width; ++n){
                    float t = 0.0;
                    for (size_t r = 0; r < input.channel; ++r){
                        for (size_t u = 0; u < fiter.height; ++u){
                            for (size_t v = 0; v < fiter.width; ++v) {
                                t +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];
                            }
                        }
                    }
                    outptr[i*output.channel*output.width*output.height + j*output.width*output.height + m*output.width + n] =t;
                }
            }
        }
    }
}
```

使用 time

```
 hoist c element
Command being timed: "./main"
User time (seconds): 1.21
System time (seconds): 0.00
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:01.22
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 23296
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 5236
Voluntary context switches: 1
Involuntary context switches: 2
```

```
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

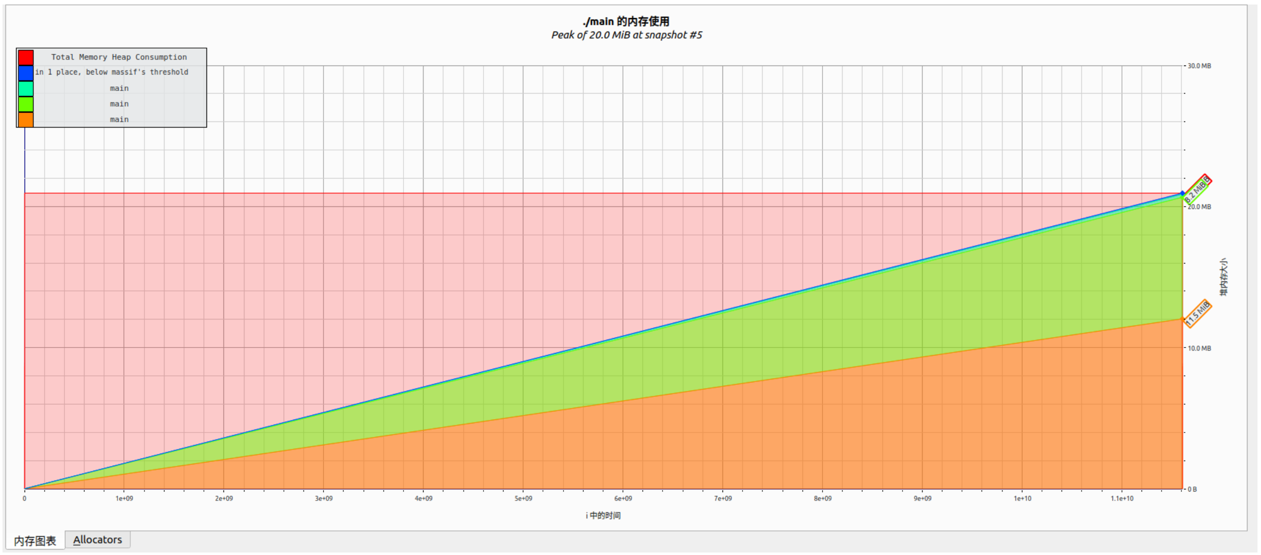

图 3: memory-output-element-hoist

3.4 RB-output-width

```
void directConvolution_RB_width( FTensor1D& input, FTensor1D& fiter ,FTensor1D& output,size_t s){
float* inptr = input.getDataPtr();
float* fitr = fiter.getDataPtr();
float* outptr = output.getDataPtr();
for (size_t i = 0; i < output.batch; ++i){
    for (size_t j = 0; j < output.channel; ++j) {
                for (size_t m = 0; m < output.height; ++m){
            for (size_t n = 0; n < output.width; n+=4){
                for (size_t r = 0; r < input.channel; ++r){
                    for (size_t u = 0; u < fiter.height; ++u){
                        for (size_t v = 0; v < fiter.width; ++v) {
                            outptr[i*output.channel*output.width*output.height + j*output.width*output.height + m*output.width + n] +=
                            inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v]*
                            fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];

                            outptr[i*output.channel*output.width*output.height + j*output.width*output.height + m*output.width + n + 1] +=
                            inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + (n+1)*s+v]*
                            fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];

                            outptr[i*output.channel*output.width*output.height + j*output.width*output.height + m*output.width + n + 2] +=
                            inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + (n+2)*s+v]*
                            fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];

                            outptr[i*output.channel*output.width*output.height + j*output.width*output.height + m*output.width + n + 3] +=
                            inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + (n + 3)*s+v]*
                            fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];
                                                 }
                    }
                }
            }
        }
        }
}
}
```

使用 time

```
Command being timed: "./main"
User time (seconds): 1.24
System time (seconds): 0.01
Percent of CPU this job got: 100%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:01.26
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 23296
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 8177
Voluntary context switches: 1
Involuntary context switches: 4
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

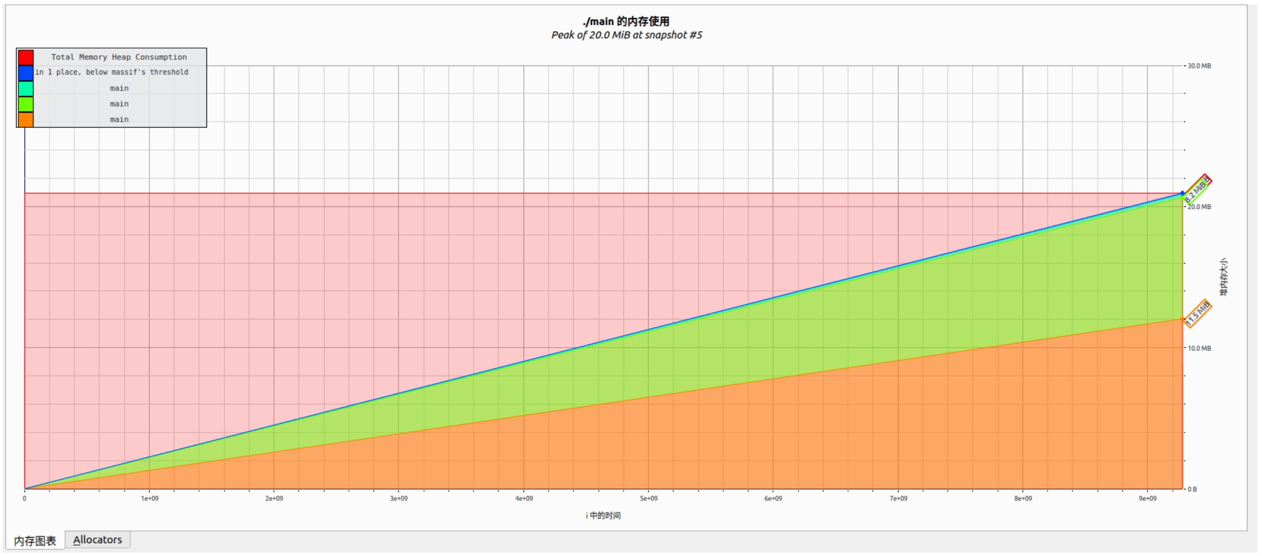

图 4: memory-RB-width

3.5 RB-output-height

```
void directConvolution_RB_height( FTensor1D& input, FTensor1D& fiter ,FTensor1D& output,size_t s){
    float* inptr = input.getDataPtr();
    float* fitr = fiter.getDataPtr();
    float* outptr = output.getDataPtr();
    for (size_t i = 0; i < output.batch; ++i){
        for (size_t j = 0; j < output.channel; ++j) {
            for (size_t m= 0; m< output.height; m+=4){
                for (size_t n = 0; n < output.width; ++n){
                    for (size_t r = 0; r < input.channel; ++r){
                        for (size_t u = 0; u < fiter.height; ++u){
                            for (size_t v = 0; v < fiter.width; ++v) {
                                outptr[i*output.channel*output.width*output.height + j*output.width*output.height +m*output.width + n] +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];

                                outptr[i*output.channel*output.width*output.height + j*output.width*output.height + (m+1)*output.width + n] +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + ((m+1)*s+u)*input.width + n*s+v]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];

                                outptr[i*output.channel*output.width*output.height + j*output.width*output.height + (m+2)*output.width + n] +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + ((m+2)*s+u)*input.width + n*s+v]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];

                                outptr[i*output.channel*output.width*output.height + j*output.width*output.height + (m+3)*output.width + n] +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + ((m+3)*s+u)*input.width + n*s+v]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];
                            }
                        }
                    }
                }
            }
        }
    }
}
```

使用 time

```
Command being timed: "./main"
User time (seconds): 1.25
System time (seconds): 0.00
Percent of CPU this job got: 100%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:01.25
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 23424
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 8176
Voluntary context switches: 1
Involuntary context switches: 2
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

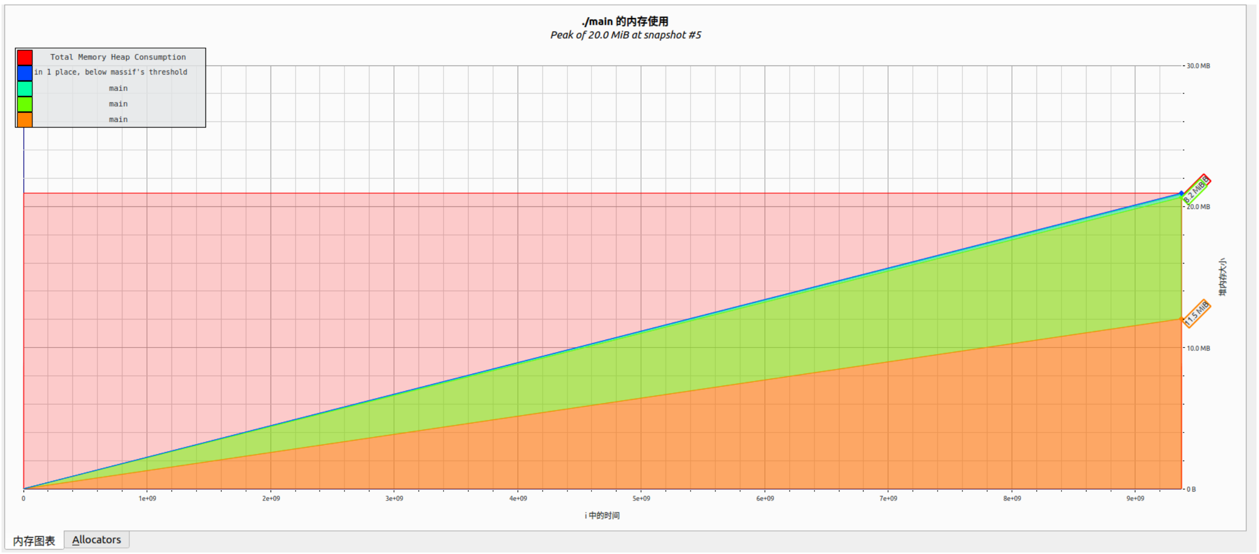

图 5: memory-RB-height

3.6 RB-output-channel

```
void directConvolution__RB__channel( FTensor1D& input, FTensor1D& fiter ,FTensor1D& output,size__t s){
    float* inptr = input.getDataPtr();
    float* fitr = fiter.getDataPtr();
    float* outptr = output.getDataPtr();
    for (size__t i = 0; i < output.batch; ++i){
        for (size__t j = 0; j < output.channel; j+=4) {
            for (size__t m= 0; m< output.height; ++m){
                for (size__t n = 0; n < output.width; ++n){
                    for (size__t r = 0; r < input.channel; ++r){
                        for (size__t u = 0; u < fiter.height; ++u){
                            for (size__t v = 0; v < fiter.width; ++v) {
                                outptr[i*output.channel*output.width*output.height + j*output.width*output.height + m*output.width + n] +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];

                                outptr[i*output.channel*output.width*output.height + (j+1)*output.width*output.height + m*output.width + n] +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v]*
                                fitr[(j+1)*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];

                                outptr[i*output.channel*output.width*output.height + (j+2)*output.width*output.height + m*output.width + n] +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v]*
                                fitr[(j+2)*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];

                                outptr[i*output.channel*output.width*output.height + (j+3)*output.width*output.height + m*output.width + n] +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v]*
                                fitr[(j+3)*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v];
                            }
                        }
                    }
                }
            }
        }
    }
}
```

使用 time

```
Command being timed: "./main"
User time (seconds): 1.25
System time (seconds): 0.00
Percent of CPU this job got: 100%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:01.25
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 23296
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 8177
Voluntary context switches: 1
Involuntary context switches: 8
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

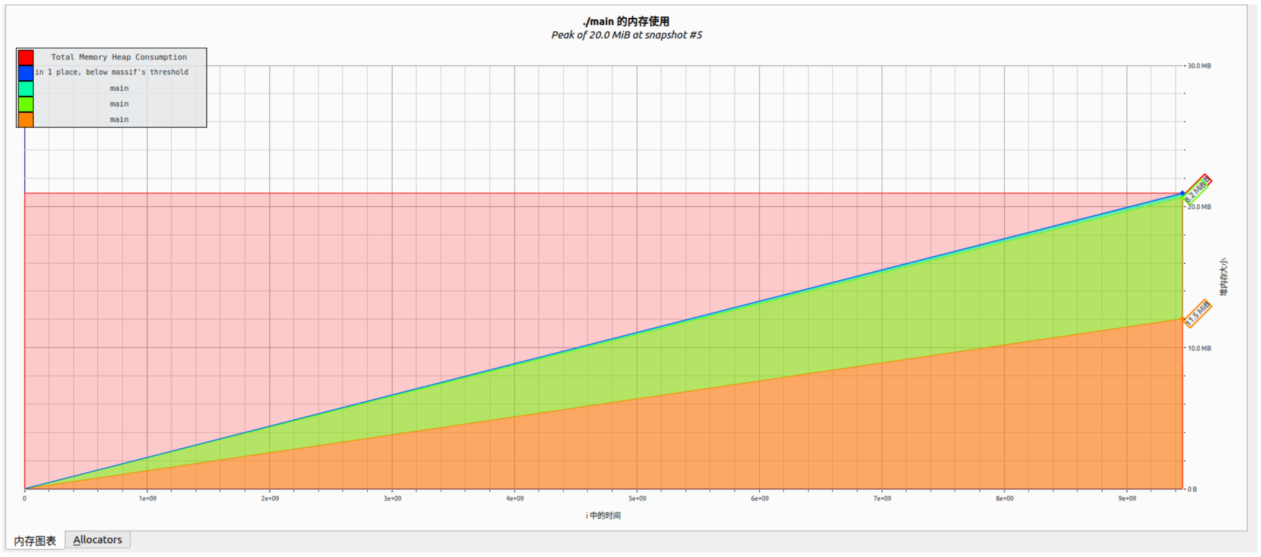

图 6: memory-RB-channel

3.7 unroll-b-width

```
void directConvolution_unroll_width( FTensor1D& input, FTensor1D& fiter ,FTensor1D& output,size_t s){
    float* inptr = input.getDataPtr();
    float* fitr = fiter.getDataPtr();
    float* outptr = output.getDataPtr();
    for (size_t i = 0; i < output.batch; ++i){
        for (size_t j = 0; j < output.channel; ++j) {
            for (size_t m= 0; m< output.height; ++m){
                for (size_t n = 0; n < output.width; ++n){
                    for (size_t r = 0; r < input.channel; ++r){
                        for (size_t u = 0; u < fiter.height; ++u){
                            for (size_t v = 0; v < fiter.width; v+=4) {
                                outptr[i*output.channel*output.width*output.height + j*output.width*output.height +m*output.width + n] +=
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v] +
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v + 1]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v + 1] +
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v + 2]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v + 2] +
                                inptr[i*input.channel*input.width*input.height + r*input.width*input.height + (m*s+u)*input.width + n*s+v+3]*
                                fitr[j*fiter.channel*fiter.width*fiter.height + r*fiter.width*fiter.height + u*fiter.width + v+3];
                            }
                        }
                    }
                }
            }
        }
    }
}
```

使用 time

```
Command being timed: "./main"
User time (seconds): 0.60
System time (seconds): 0.02
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.63
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 23424
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
```

```
Minor (reclaiming a frame) page faults: 8177
Voluntary context switches: 1
Involuntary context switches: 2
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

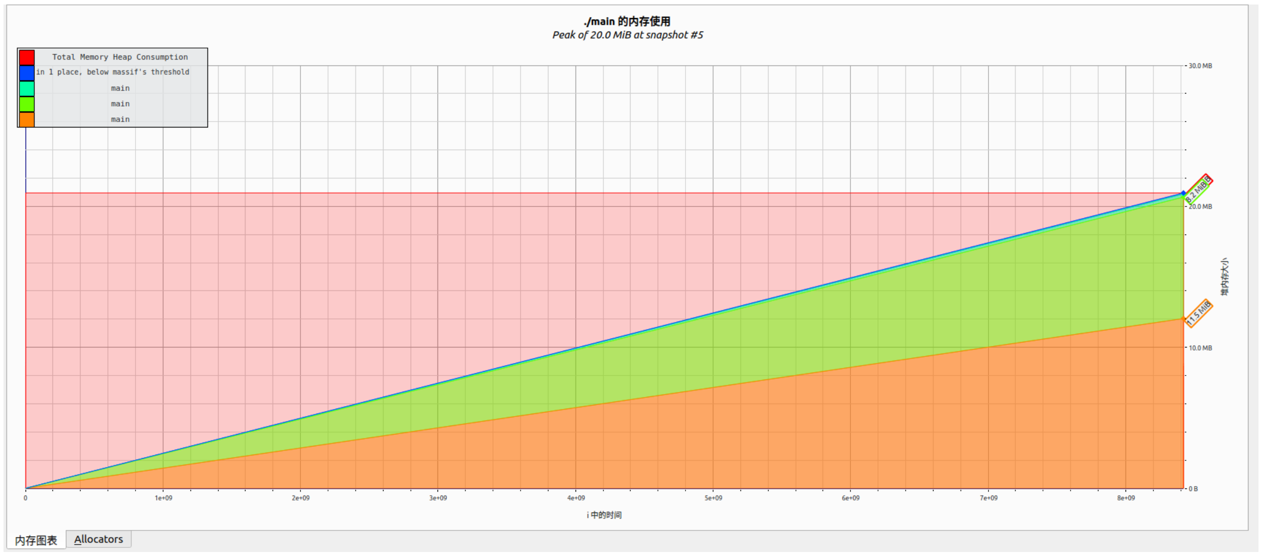

图 7: memory-unroll-b-width

3.8 simd

```
void directConvolution_simd( FTensor1D& input, FTensor1D& fiter ,FTensor1D& output,size_t s){
    float *inptr, *fitr,*outptr;
    for (size_t i = 0; i < output.batch; ++i){
        size_t input_i = i * input.channel * input.height * input.width;
        size_t output_i = i * output.channel * output.height * output.width;
        for (size_t j = 0; j < output.channel; ++j) {
            size_t output_j = j * output.height * output.width;
            size_t filter_j = j * fiter.channel * fiter.height * fiter.width;
            for (size_t m = 0; m < output.height; ++m){
                size_t output_m = m * output.width;
                size_t ms = m * s * input.width;
                for (size_t n = 0; n < output.width; n+=4){
                    __m128 c0_3 = _mm_setzero_ps();
                    outptr = output.getDataPtr() + output_i + output_j + output_m + n;
                    size_t ns = n * s;
                    for (size_t r = 0; r < input.channel; ++r){
                        size_t input_r = r * input.height * input.width;
                        size_t fiter_r = r * fiter.height * fiter.width;
                        for (size_t u = 0; u < fiter.height; ++u){
                            inptr = input.getDataPtr() + input_i + input_r + ms + u *input.width+ ns;
                            fitr = fiter.getDataPtr() + filter_j + fiter_r + u * fiter.width;
                            for (size_t v = 0; v < fiter.width; v+=4) {
                                __m128 a0_3,a1_4,a2_5,a3_6,b0,b1,b2,b3;
                                float data[4] = {*(inptr + v),*(inptr + s + v),*(inptr + 2 * s + v),*(inptr + 3 * s + v)};
                                float data1[4] = {*(inptr + v + 1),*(inptr + s + v + 1),*(inptr + 2 * s + v + 1),*(inptr + 3 * s + v + 1)};
                                float data2[4] = {*(inptr + v + 2),*(inptr + s + v + 2),*(inptr + 2 * s + v + 2),*(inptr + 3 * s + v + 2)};
                                float data3[4] = {*(inptr + v + 3),*(inptr + s + v + 3),*(inptr + 2 * s + v + 3),*(inptr + 3 * s + v + 3)};
                                a0_3 = _mm_loadu_ps(data);
                                a1_4 = _mm_load_ps(data1);
                                a2_5 = _mm_load_ps(data2);
                                a3_6 = _mm_load_ps(data3);
                                b0 = _mm_set1_ps(*(fitr+v));
                                b1 = _mm_set1_ps(*(fitr+v+1));
                                b2 = _mm_set1_ps(*(fitr+v+2));
                                b3 = _mm_set1_ps(*(fitr+v+3));
                                c0_3 = _mm_fmadd_ps(a0_3,b0,c0_3);
                                c0_3 = _mm_fmadd_ps(a1_4,b1,c0_3);
                                c0_3 = _mm_fmadd_ps(a2_5,b2,c0_3);
                                c0_3 = _mm_fmadd_ps(a3_6,b3,c0_3);


                                //c0_3 = _mm_fmadd_ps(a3_6,b3,_mm_fmadd_ps(a2_5,b2,_mm_fmadd_ps(a1_4,b1,)));
                            }
                        }
                    }
                    *outptr = c0_3[0];*(outptr+1) = c0_3[1];
                    *(outptr+2) = c0_3[2];*(outptr+3) = c0_3[3];
                }
            }
        }
    }
}
```

使用 time

```
Command being timed: ”./main”
User time (seconds): 0.38
System time (seconds): 0.00
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.39
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 23296
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 5236
Voluntary context switches: 1
Involuntary context switches: 1
Swaps: 0
File system inputs: 0
File system outputs: 0
```

```
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

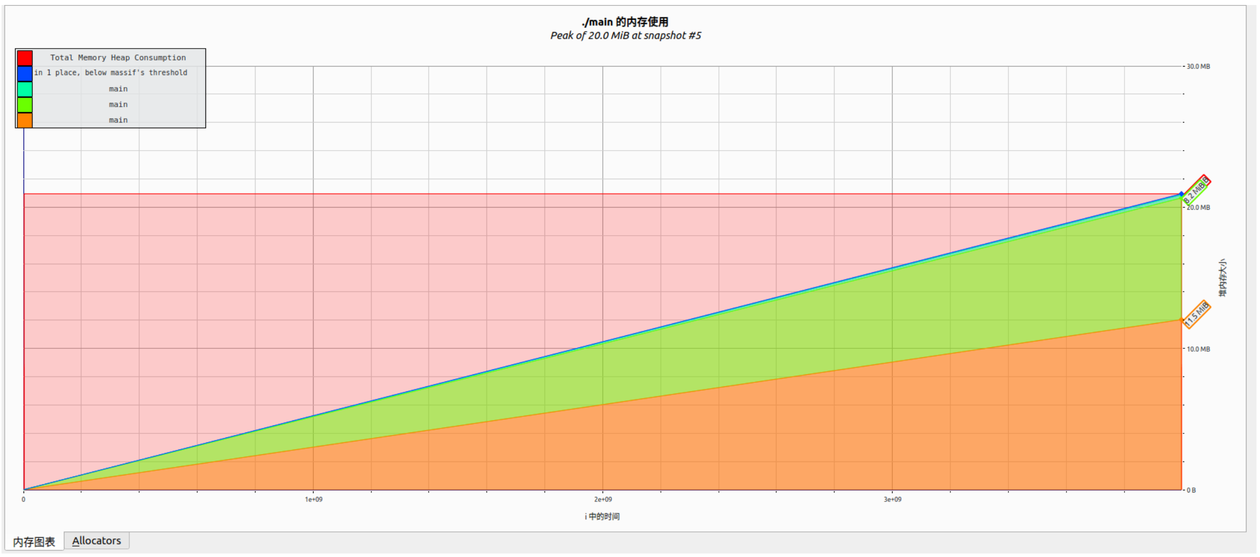

图 8: memory-simd

4 结果分析

使用 valgrind 测出来的是堆内存空间使用量可以出看基本没什么变化，使用 time 指令测出来不同种优化之间，从无优化到索引优化 Involuntary context switches 由 2 变为 7，Involuntary context switches 表示在执行程序时发生的非自愿上下文切换次数。
当使用了 hoist 优化后 Minor (reclaiming a frame) page faults 由 8176 变为了 5236，Minor (reclaiming a frame) page faults 表示发生了多少次页面错误，也就是在主存中读取数据失败，需要去从磁盘中读取。
RB 优化和无优化效果差不多，之前实验测过，如果只是单纯的分块，不在分块基础上叠加 hoist 或者 simd，那么实际上的 gflops 并不会增大，反而还会下降。
unroll 的话 Maximum resident set size (kbytes）变大了从 23296 变为了 23424，Maximum resident set size (kbytes）表示程序在执行期间所占用的物理内存的最大值.
simd: 因为不可能单独的使用 simd, 使用 simd 肯定建立在 hoist+RB 情况下，我在测试的时候还加上了 unrollb，表现的效果为页面错误次数减少，Involuntary context switches 变为 1.