sum_opt.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>
#include<limits.h>

//#define MAX_NUM 100000000
//#define MAX_NUM 10000
#define MAX_NUM 10000000
//#define MAX_NUM 1000000000
//#define MAX_NUM INT_MAX
//#define MAX_NUM 100000

struct vec {
    int len;
    int *data;
};
typedef struct vec vec_t;

// int get_vec_element(vec_t* v, int idx, int *val)
// {
//  // assert(v); 조건이 아닌 값이 들어감, 불필요한 함수 제거

//  // if(idx >= v->len)
//  //  return 0;
//  //이미 반복문에서 인덱스가 길이보다 작게 설정 함

//  *val = v->data[idx];
//  return 1;
// }

int* get_vec_start(vec_t* v) {
    return v->data;
} //벡터의 시작 메모리를 반환하고 사용자가 element에 접근하도록 변경

int vec_length(vec_t* v)
{
    // assert(v);
    return v->len;
}

void combine(vec_t* v, int *dest)
{
    // assert(dest);

    // for(int i = 0; i < vec_length(v); i++){
    //지속적인 함수 호출대신 값으로 받아놓기
```

sum_opt.c

```c
    int v_len = vec_length(v);

    int *data = get_vec_start(v); //벡터의 시작 메모리를 반환하고 사용자가 element에 접근하도록 변경

    // *dest = 0; 포인터 연산 제거
    int temp = 0;

    for(int i = 0; i < v_len; i+=2){
        // get_vec_element(v, i, &val);

        // *dest = *dest + val; 포인터 연산 제거
        // temp = temp + data[i]; //지역 변수 활용 연산으로 포인터 연산 대체
        temp = temp + (data[i]+data[i+1]); //+루프 언롤링까지
    }
    *dest = temp;
}

void init(vec_t* v)
{
    // assert(v);
    v->len = MAX_NUM;
    v->data = (int*) malloc(sizeof(int) * MAX_NUM);

    // init
    for(int i = 0; i < MAX_NUM; i++)
        v->data[i] = i;
}

//assert: 에러 검출용 조건 확인 함수

int main()
{
    printf("This is the naive version .. \n");

    vec_t info;
    int result;
    // init
    init(&info);

    // combine
    combine(&info, &result);
    //printf("combined val = %d\n", result);

    return 0;
}
```

gprof 툴을 이용한 프로파일링 sum

user@LAPTOP-1HKVLFNK: /m

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 33.69      0.05     0.05 10000001     0.00     0.00  vec_length
 26.96      0.09     0.04 10000000     0.00     0.00  get_vec_element
 20.22      0.12     0.03        1    30.32   121.30  combine
 20.22      0.15     0.03        1    30.32    30.32  init

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
"sum.gprof" 161L, 6740C                                1,1           Top
```

gprof 툴을 이용한 프로파일링 sum_opt

user@LAPTOP-1HKVLFNK: /m

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 81.05      0.04     0.04        1    40.52    40.52  init
 20.26      0.05     0.01        1    10.13    10.13  combine
  0.00      0.05     0.00        1     0.00     0.00  get_vec_start
  0.00      0.05     0.00        1     0.00     0.00  vec_length

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
"sum_opt.gprof" 161L, 6725C                                  1,1           Top
```

perf 툴을 이용한 프로파일링 sum

```
user@LAPTOP-1HKVLFNK: /mi

# Children      Self  Command  Shared Object      Symbol
# ........  ........  .......  .................  ...................................
#
    99.16%     0.07%  sum      libc-2.31.so       [.] __libc_start_main
            |
            ---__libc_start_main
               |
                --99.09%--main
                          |
                          |--85.29%--combine
                          |          |
                          |          |--23.15%--_mcount
                          |          |
                          |          |--12.76%--0x7f10602fdb74
                          |          |
                          |          |--11.79%--get_vec_element
                          |          |
                          |          |--4.95%--0x7f10602fdbc0
                          |          |
                          |          |--3.70%--vec_length
                          |          |
                          |          |--2.72%--0x7f10602fdbb0
                          |          |
                          |          |--2.51%--0x7f10602fdcb5
                          |          |
                          |          |--1.46%--0x7f10602fdc2a
                          |          |
                          |          |--1.46%--0x7f10602fdbb3
                          |          |
                          |          |--1.12%--0x7f10602fdb8e
                          |          |
                          |          |--1.12%--0x7f10602fdb88
                          |          |
                          |           --0.91%--0x7f10602fdb64
                          |
                          |--11.92%--init
                          |          |
                          |           --4.46%--asm_exc_page_fault
                          |                     exc_page_fault
                          |                     do_user_addr_fault
                          |                     |
                          |                      --4.39%--handle_mm_fault
                          |                                |
                          |                                 --4.04%--do_huge_pmd_anonymous_page
                          |                                           |
                          |                                           |--3.07%--clear_huge_page
                          |                                           |          clear_subpage
                          |                                           |          clear_page_erms
                          |                                           |
                          |                                           --0.98%--__alloc_pages_nodemask
                          |                                                     get_page_from_freelist
                          |
                          |--1.32%--get_vec_element
                          |
                           --0.56%--vec_length
:
```

perf 툴을 이용한 프로파일링 sum_opt

```
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/opt$ sudo perf record -g ./sum_opt
This is the naive version ..
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.025 MB perf.data (246 samples) ]
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/opt$ sudo perf report -f
# To display the perf.data header info, please use --header/--header-only options.
#
#
# Total Lost Samples: 0
#
# Samples: 246  of event 'cpu-clock:pppH'
# Event count (approx.): 61500000
#
# Children      Self  Command  Shared Object      Symbol
# ........  ........  .......  .................  ...................................
#
    94.72%     0.00%  sum_opt  libc-2.31.so       [.] __libc_start_main
            |
            ---__libc_start_main
               main
               |
               |--66.67%--init
               |          |
               |           --23.58%--asm_exc_page_fault
               |                     exc_page_fault
               |                     do_user_addr_fault
               |                     handle_mm_fault
               |                     |
               |                      --22.76%--do_huge_pmd_anonymous_page
               |                                |
               |                                |--17.48%--clear_huge_page
               |                                |          clear_subpage
               |                                |          clear_page_erms
               |                                |
               |                                 --5.28%--__alloc_pages_nodemask
               |                                           get_page_from_freelist
               |                                           |
               |                                            --1.63%--prep_new_page
               |                                                      prep_compound_page
               |
                --28.05%--combine
```

uftrace 를 사용한 프로파일링 sum, sum_opt

user@LAPTOP-1HKVLFNK: ~/s

```
user@LAPTOP-1HKVLFNK:~/spt$ uftrace record ./sum
This is the naive version ..
user@LAPTOP-1HKVLFNK:~/spt$ uftrace report
  Total time   Self time       Calls  Function
  ==========  ==========  ==========  ====================
    4.166  s    3.908 us           1  main
    4.126  s    2.615  s           1  combine
  772.173 ms  772.157 ms    10000000  get_vec_element
  739.629 ms  739.629 ms    10000001  vec_length
   39.398 ms   39.387 ms           1  init
   67.118 us   67.118 us           1  puts
   52.049 us   52.049 us           3  linux:schedule
   11.041 us   11.041 us           1  malloc
    0.802 us    0.802 us           1  __monstartup
    0.361 us    0.361 us           1  __cxa_atexit
user@LAPTOP-1HKVLFNK:~/spt$ uftrace record ./sum_opt
This is the naive version ..
user@LAPTOP-1HKVLFNK:~/spt$ uftrace report
  Total time   Self time       Calls  Function
  ==========  ==========  ==========  ====================
   59.801 ms    7.094 us           1  main
   43.581 ms   43.573 ms           1  init
   16.168 ms   16.168 ms           1  combine
   45.116 us   45.116 us           1  puts
    7.905 us    7.905 us           1  malloc
    0.511 us    0.511 us           1  __monstartup
    0.310 us    0.310 us           1  __cxa_atexit
    0.071 us    0.071 us           1  get_vec_start
    0.071 us    0.071 us           1  vec_length
user@LAPTOP-1HKVLFNK:~/spt$
```

분석 내용

기존에 get_vec_element와 vec_length가 10000000번 이상 호출됐지만
최적화 후 get_vec_elemen를 대체한 get_vec_start와 vec_length가 1번만 호출되면서 combine의 실행 시간이 줄어든 것을 확인할 수 있다.

combine에서 포인터를 사용한 직접 연산과 루프 언롤링을 통해 최적화를 진행했는데 함수 호출을 줄이는 것과 더불어서 combine의 실행 시간을 줄인 것을 확인할 수 있다.

또한 오류 검출을 위한 assert는 trash code기 때문에 삭제했는데 init은 비슷한(이후로 여러 번 돌려봤지만 오차 범위 내) 것을 보면 assert는 실행 시간에 많은 영향을 끼치지 않는다는 것도 확인할 수 있었다.