

COL 380 - Assignment 1

cs1190405

January 2023

1 Question 1

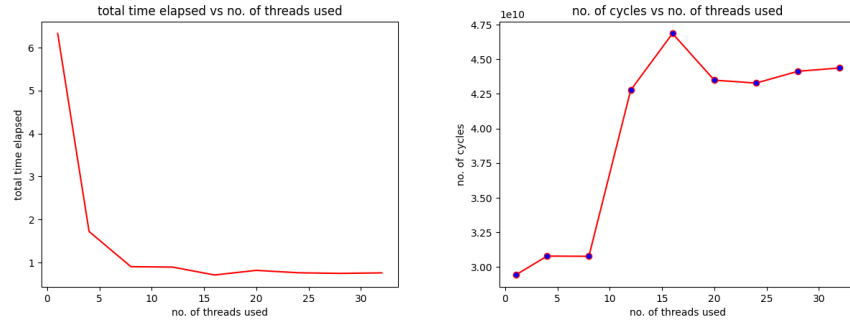
For this assignment, I connected to the css2 server and used perf for analysis.

2 Question 2

1. ***perf stat*** - The perf stat command instruments and summarizes key CPU counters (PMCs).

The *perf stat* command was run for *./classify rfile dfile 1009072 <no. of threads used >*. The no. of repetitions was not specified and the default value of 6 was used. The following observations was observed:

No. of Threads used	total time elapsed	no. of cycles
1	6.302815603	29,47,63,99,326
4	1.685299279	30,27,23,58,594
8	0.947380728	31,65,52,09,295
12	0.900934003	42,98,15,42,122
16	0.701129524	46,64,64,44,306
20	0.815482245	43,11,37,69,491
24	0.751055407	43,19,75,75,453
28	0.740205913	43,85,35,84,892
32	0.747876676	44,29,33,37,255



(a) Graph of time elapsed vs threads used (b) Graph of no. of cycles vs threads used

2. *perf record*

- (a) ***perf record*** - The *perf record* command was run to create the *perf.data* file which was used for analysis.
- (b) ***perf report*** - The *perf report* command was used to inspect the file generated in the previous step and then the file was annotated for hotspot analysis.
These commands were found in the during the hotspot analysis of classify event.:
 - i. 37.68| ↓ *fg93*
 - ii. 9.37|*cmp(%r11,%rax,8),%edx*
 - iii. 14.81 - ↑ *jge40*
18.50 - 93 : *lea0x1(%rax),%r13*
 - iv. 14.56 - ↑ *jmp8a*
- (c) The instructions that takes the maximum time is:
fg 93 (37.68% of time)
- (d) It maps to the the bool within functions defined in the classify.h
bool within(int val) const // Return if val is within this range
return(lo *leq* val val *leq* hi);
- (e) The make file was changed slightly to allow the perf report to show the source code along with the assembly code. This was done using the - ***g*** flag during compiling.

0.28	40:	cmp	0x4(%r11,%rax,8),%edx
27.44		↓ je	41
0.31		shl	\$0x6,%rax
0.04		add	%rbp,%rax
0.07	4e:	mov	%r1d,%x4(%r12)
0.31		mov	(%rax),%rdx
0.14		cmp	%r9d,0x8(%rax)
		↓ jbe	b9
		lea	(%rdx,%r11,1),%rax
		add	%ebx,%ecx
2.03		mov	(%rax),%edx
0.02		add	\$0x1,%edx
0.05		mov	%edx,(%rax)
		mov	%ecx,%eax
		cmp	%ecx,(%r8)
		↓ jbe	b0
	70:	cltq	
0.02		lea	(%r10,%rax,8),%r12
		mov	0x8(%r11),%eax
0.02		mov	(%r12),%edx
		test	%eax,%eax
		↓ jle	a0
0.03		mov	(%r11),%r11
		lea	-0x1(%rax),%r14d
0.00		xor	%eax,%eax
0.01	8a:	mov	%eax,%r13d
9.37		cmp	(%r11,%rax,8),%edx
14.01		↓ jge	40
18.50	93:	lea	0x1(%rax),%r13
1.13		cmp	%rax,%r14
		↓ je	a8
14.56		mov	%r13,%rax
		↓ je	80

Figure 2: Hotspots found

3. Hotspot Analysis

- Perf report was run with the *-g* flag in the makefile to show the source code with the assembly code and the output was saved as perf_2_1.data
-

```

0.35 40: cmp     0x4(%r11,%rax,8),%edx
37.54 41: jg      93
0.40    shl     $0x6,%rax
0.07    add     %rbp,%rax
_Z8classifyR4DataRK6Rangesj._omp_fn.0():
0.04 4e: mov     %r13d,0x4(%r12)
// and store the interval id in value. D is changed.
counts[v].increase(tid); // Found one key in interval v
0.36    mov     (%rax),%rdx
_ZN7Counter8increaseEj():
assert(id < numcount);
0.18    cmp     %r9d,0x8(%rax)
jbe     b9
_counts[id]++;
0.00    lea     (%rdx,%rdi,1),%rax
_Z8classifyR4DataRK6Rangesj._omp_fn.0():

for(int i=0; i<data->len(); i++) { // Threads together share-loop through all of data
    add     %r10,%r10
    _ZN7Counter8increaseEj():
    add     $0x1,%r10
    mov     %r10,%r10
    _ZN8classifyR4DataRK6Rangesj._omp_fn.0():
    mov     %r10,%r10
    cmp     %r10,%r10
    jbe     b9
    int v = 0; data[i].value = 0; range(0,data[i].key); // For each data, find the interval of data's key,
0.02 7f: lea     (%r10,%rax,8),%r12
    _ZN8Ranges8getlb():
    if(r12 < 0) {
        for(int i=0; i<num; i++) // Look through all intervals
            if(ranges[i].start<=val)
                return i;
    } else {
        for(int i=0; i<num; i++) // Look through all intervals
            if(ranges[i].end>val)
                return i;
    }
0.05    mov     %r12,%r10
    _ZN8classifyR4DataRK6Rangesj._omp_fn.0():
    mov     %r10,%r10
    _ZN8Ranges8getlb():
    test    %r10,%r10
    jbe     b9
    if(ranges[r].within(val))
        mov     (%r10),%r10
        lea     (%r10,%rax,8),%r10
        mov     %r10,%r10
0.02    mov     %r10,%r10
    _ZN8Ranges8getlb():
    return(lo <= val && val <= hi);
0.04    jbe     b9
0.07    _ZN8Ranges8getlb():
    for(int i=0; i<num; i++) // Look through all intervals
        if(ranges[i].start<=val)
            return i;
0.11    mov     %r10,%r10
    mov     %r10,%r10
0.00    mov     %r10,%r10
    mov     %r10,%r10
    mov     %r10,%r10
    return r;
    return B4RANGE; // Did not find any range
    mov     %r10,%r10

```

Figure 3: 3.2 - Top Hotspot

- (c) The above assembly code refers to the boolean *within* function in the *classify.h* file.
- (d) perf record was run to record branch instructions, branch misses, cache misses, page faults, cpu cycles. The following command was used to record the events and for analysis: *perf record -e branch-instructions,branch-misses,cache-misses,page-faults,cpu-cycles make run*

4. Memory Profiling

- (a) the perf mem record was used to record the memory resources used by different functions and the data was stored in perf_3.data
- (b)