

Profiling

farhan

18 October 2016

What I have done so far

- ▶ Make profiling process using gperftools profiling tool
- ▶ Make some change on Rayan's code to make it more optimized.
- ▶ Compile the code with optimization option (example: -O3/-msse2) to make the program more optimized
- ▶ Reduce cpu time used by original Rayan's code

Rayan's Original pprof result

Total: 3847 samples					
2488	64.7%	64.7%	3671	95.4%	NW::nw_align
599	15.6%	80.2%	599	15.6%	NW::max
583	15.2%	95.4%	588	15.3%	std::basic_string::operator[]
35	0.9%	96.3%	3846	100.0%	main
33	0.9%	97.2%	33	0.9%	_init
13	0.3%	97.5%	36	0.9%	Input::complementInput
12	0.3%	97.8%	12	0.3%	std::swap
11	0.3%	98.1%	11	0.3%	NW::dpm_init
11	0.3%	98.4%	11	0.3%	std::basic_string::operator+=
8	0.2%	98.6%	10	0.3%	__gnu_cxx::operator<
7	0.2%	98.8%	15	0.4%	NW::verifyPercentage
5	0.1%	98.9%	5	0.1%	__gnu_cxx::__normal_iterator::operator++
5	0.1%	99.0%	5	0.1%	std::basic_string::length
4	0.1%	99.1%	4	0.1%	__gnu_cxx::__normal_iterator::operator--
4	0.1%	99.2%	17	0.4%	std::_iter_swap::_iter_swap
4	0.1%	99.4%	21	0.5%	std::iter_swap
3	0.1%	99.4%	3	0.1%	malloc_set_state
2	0.1%	99.5%	2	0.1%	__gnu_cxx::__normal_iterator::base
2	0.1%	99.5%	2	0.1%	_nss_hosts_lookup
2	0.1%	99.6%	2	0.1%	_read
2	0.1%	99.6%	42	1.1%	std::_reverse
2	0.1%	99.7%	2	0.1%	std::basic_fstream::close
1	0.0%	99.7%	1	0.0%	__gnu_cxx::__normal_iterator::operator*
1	0.0%	99.7%	4	0.1%	free
1	0.0%	99.8%	2	0.1%	malloc
1	0.0%	99.8%	1	0.0%	malloc_trim
1	0.0%	99.8%	1	0.0%	memchr
1	0.0%	99.8%	1	0.0%	std::_once_callable
1	0.0%	99.9%	6	0.2%	std::basic_string::M_mutate
1	0.0%	99.9%	1	0.0%	std::basic_string::Rep::M_clone
1	0.0%	99.9%	3	0.1%	std::basic_string::Rep::S_create
1	0.0%	99.9%	3	0.1%	std::basic_string::append
1	0.0%	100.0%	8	0.2%	std::getline@7a5e0
1	0.0%	100.0%	1	0.0%	std::getline@98800
0	0.0%	100.0%	1	0.0%	0x0000000a00000009
0	0.0%	100.0%	31	0.8%	0x00007ffcbb072f2f
0	0.0%	100.0%	2	0.1%	0x00007ffcbb072f3f
0	0.0%	100.0%	3714	96.5%	NW::nw
0	0.0%	100.0%	3846	100.0%	_libc_start_main
0	0.0%	100.0%	3846	100.0%	_start

Figure 1: we can see *nw_align* function use most cpu time which is 3671 and *max* function is the second function that use most cpu time which is

Rayan's Original code in nw_align

```
.      . 127:
12      12 128:         for( i = 1; i <= L2; i++ )
.      . 129:         {
247     247 130:             for( j = 1; j <= L1; j++ )
.      . 131:             {
65      239 132:                 nuc = seq_1[ j-1 ] ;
.      . 133:
277     277 134:                 switch( nuc )
.      . 135:                 {
74      74 136:                     case 'A': x = 0 ; break ;
71      71 137:                     case 'C': x = 1 ; break ;
101     101 138:                     case 'G': x = 2 ; break ;
13      13 139:                     case 'T': x = 3 ;
.      . 140:                 }
.      . 141:
117     469 142:                 nuc = seq_2[ i-1 ] ;
.      . 143:
229     229 144:                 switch( nuc )
.      . 145:                 {
9       9 146:                     case 'A': y = 0 ; break ;
11      11 147:                     case 'C': y = 1 ; break ;
13      13 148:                     case 'G': y = 2 ; break ;
7       7 149:                     case 'T': y = 3 ;
.      . 150:                 }
.      . 151:
```

Figure 2: using switch for this function use most cpu time.

Changed code in nw_align function

```
. . 110:
. . 119:     const int  a = 2;  /* Match */
. . 120:     const int  b = -1; /* Mismatch */
. . 121:
. . 122: //         const int  s[ 4 ][ 4 ] = { { a, b, b, b },      /* Substitution matrix */
. . 123: //                                     { b, a, b, b },
. . 124: //                                     { b, b, a, b },
. . 125: //                                     { b, b, b, a } } ;
. . 126:
. . 127:     int  L1 = seq_1.length();
. . 128:     int  L2 = seq_2.length();
. . 129:
. 9 130:     strncpy(nuc, seq_1.c_str(), sizeof(nuc));
. 7 131:     strncpy(nuc2, seq_2.c_str(), sizeof(nuc2));
. . 132:
. . 133:
2 2 134:     for( i = 0; i < L2; i++ )
. . 135:     {
6 6 136:         for( j = 0; j < L1; j++ )
. . 137:         {
59 59 138:             if (nuc[j]!='N' || nuc[i]!='N')
. . 139:                 if (nuc[j]==nuc2[i]){
64 64 140:                     checkMatch=a;
. . 141:                 }
. . 142:                 else if(nuc[j]!=nuc2[i]){
. . 143:                     checkMatch=b;
. . 144:                 }
. . 145:                 else {
. . 146:                     checkMatch=0;
. . 147:                 }
. . 148:             }
```

Figure 3: The code changed by using if else statement

Profiling's result

```

Total: 653 samples
408 62.5% 62.5%      593 90.8% Nw::nw_align
149 22.8% 85.3%      149 22.8% Nw::max (inline)
17 2.6% 87.9%        17 2.6% __nss_hosts_lookup
12 1.8% 89.7%        651 99.7% main
7 1.1% 90.8%         11 1.7% Input::complementInput
6 0.9% 91.7%         12 1.8% std::string::_M_leak (inline)
5 0.8% 92.5%         5 0.8% Nw::dpm_init (inline)
5 0.8% 93.3%         5 0.8% Nw::verifyPercentage (inline)
5 0.8% 94.0%         5 0.8% __read
3 0.5% 94.5%         3 0.5% malloc_set_state
3 0.5% 94.9%         7 1.1% std::basic_string::_M_mutate
3 0.5% 95.4%         3 0.5% std::basic_string::basic_string
3 0.5% 95.9%         3 0.5% std::char_traits::assign (inline)
3 0.5% 96.3%         3 0.5% std::string::_M_data (inline)
3 0.5% 96.8%        17 2.6% std::string::operator[] (inline)
2 0.3% 97.1%         2 0.3% __gnu_cxx::__normal_iterator::operator++ (inline)
2 0.3% 97.4%         2 0.3% __init
2 0.3% 97.7%         5 0.8% free
2 0.3% 98.0%         4 0.6% malloc
2 0.3% 98.3%         2 0.3% malloc_trim
2 0.3% 98.6%         2 0.3% std::basic_string::_M_leak
2 0.3% 98.9%         2 0.3% std::getline@98800
1 0.2% 99.1%         1 0.2% __gnu_cxx::__normal_iterator::operator-- (inline)
1 0.2% 99.2%         5 0.8% __reverse (inline)
1 0.2% 99.4%         1 0.2% memchr
1 0.2% 99.5%         6 0.9% std::basic_string::_M_leak_hard
1 0.2% 99.7%         2 0.3% std::basic_string::assign
1 0.2% 99.8%         6 0.9% std::string::push_back (inline)
1 0.2% 100.0%        1 0.2% swap (inline)
0 0.0% 100.0%        2 0.3% 0x00000000000e4200f
0 0.0% 100.0%        603 92.3% Nw::nw
0 0.0% 100.0%        651 99.7% __libc_start_main
0 0.0% 100.0%        651 99.7% __start
0 0.0% 100.0%         1 0.2% iter_swap (inline)
0 0.0% 100.0%         4 0.6% operator new
0 0.0% 100.0%         5 0.8% reverse (inline)
0 0.0% 100.0%         5 0.8% std::_basic_file::xsgetn
0 0.0% 100.0%         5 0.8% std::basic_filebuf::underflow
0 0.0% 100.0%         1 0.2% std::basic_string::_M_replace_safe

```

Figure 4: The cpu time had been reduce for *nw_align* function and *max* function

Optimization by compile option

- ▶ Used -O3 (Optimization option to speed up the cpu time for the program that manage big data):
 - ▶ `g++ -O3 -g -lprofiler main.cpp Input.cpp Input.h CS.cpp CS.h NW.cpp NW.h`