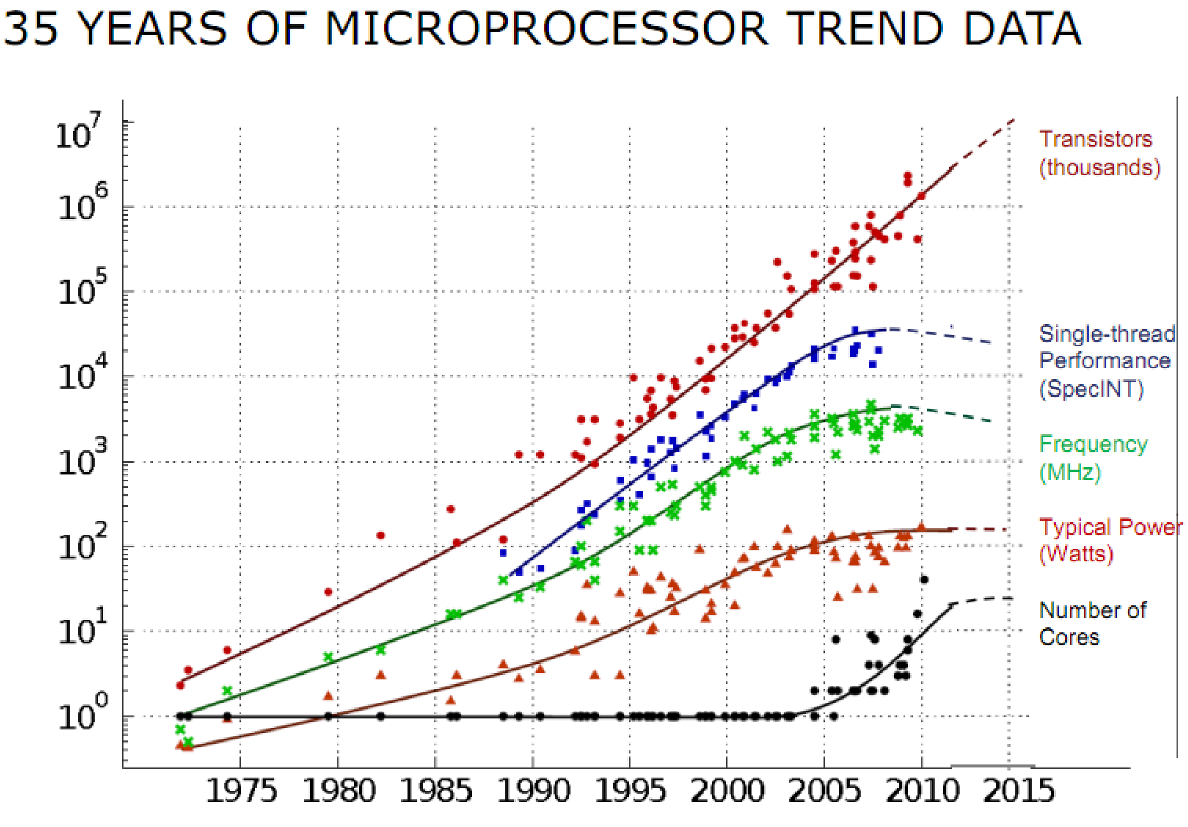
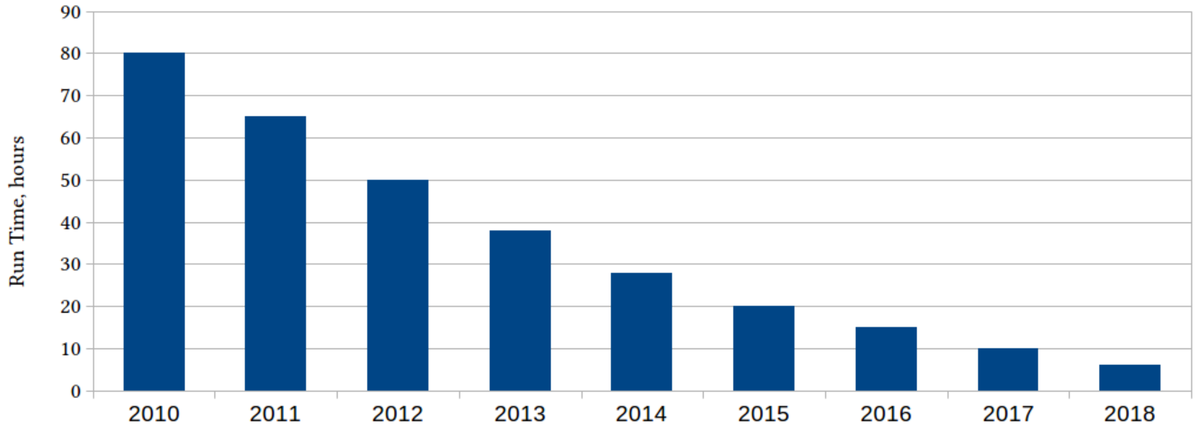
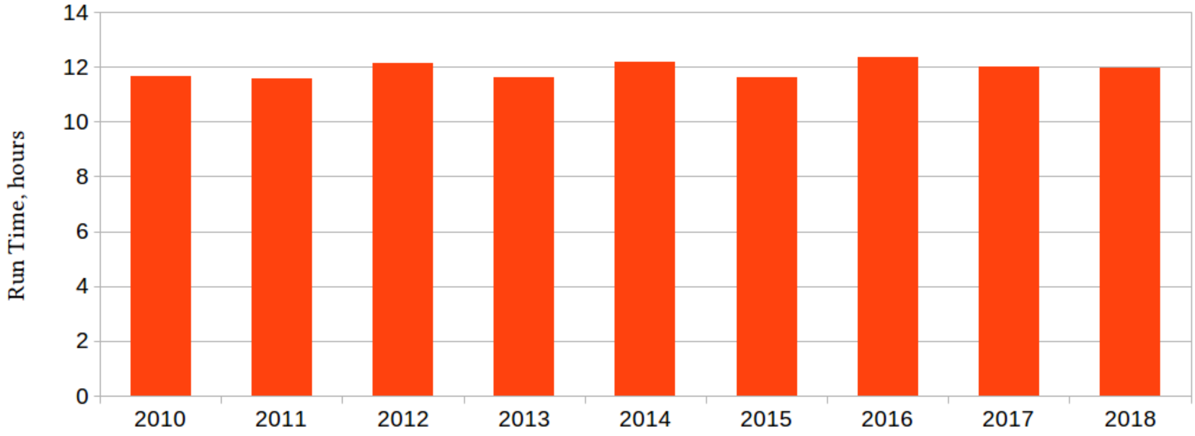
Chapter 1: Introduction to Performance and Concurrency

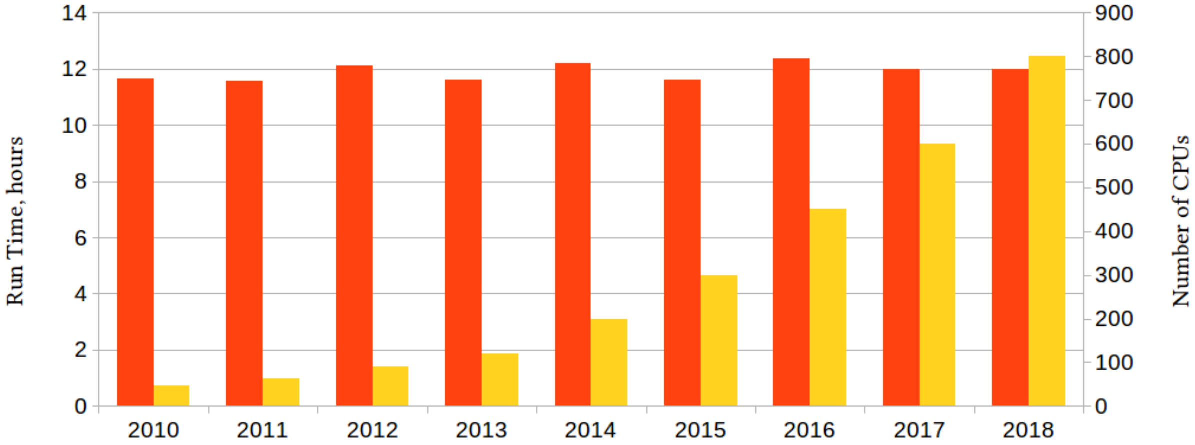


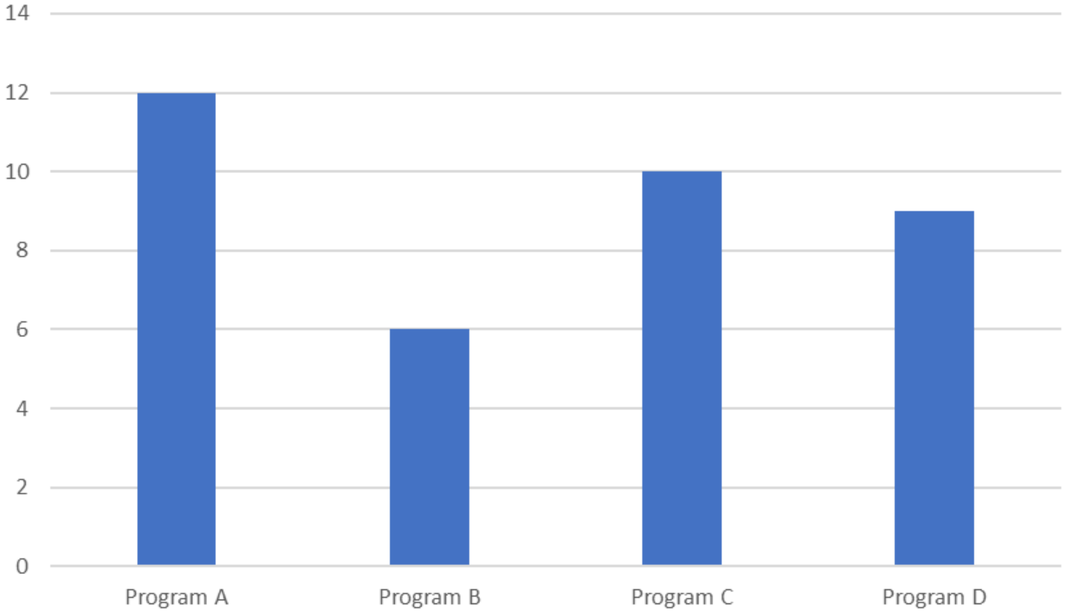
35 YEARS OF MICROPROCESSOR TREND DATA

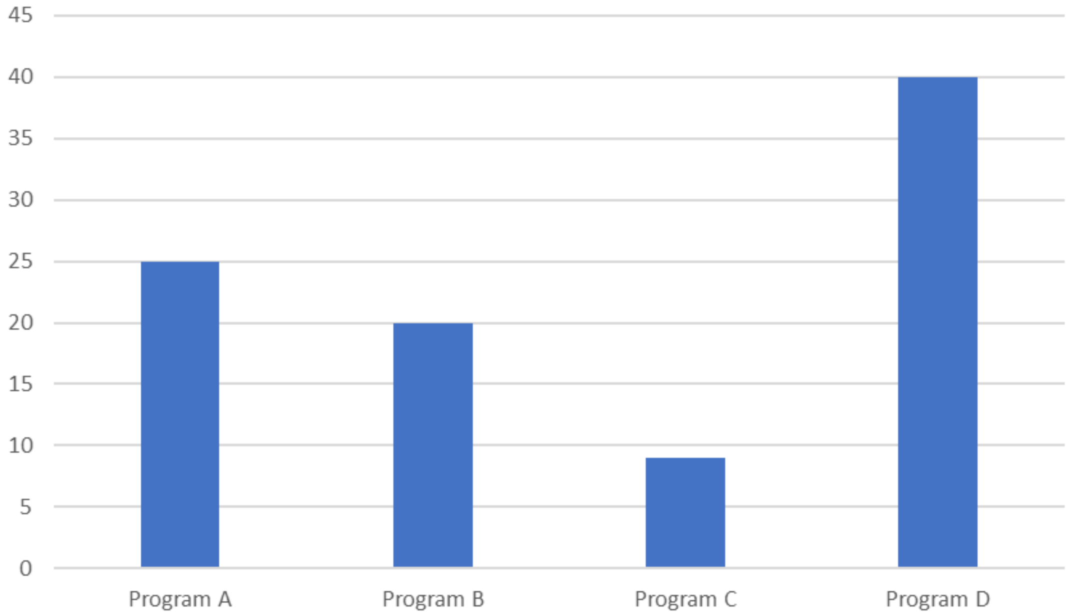
Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore



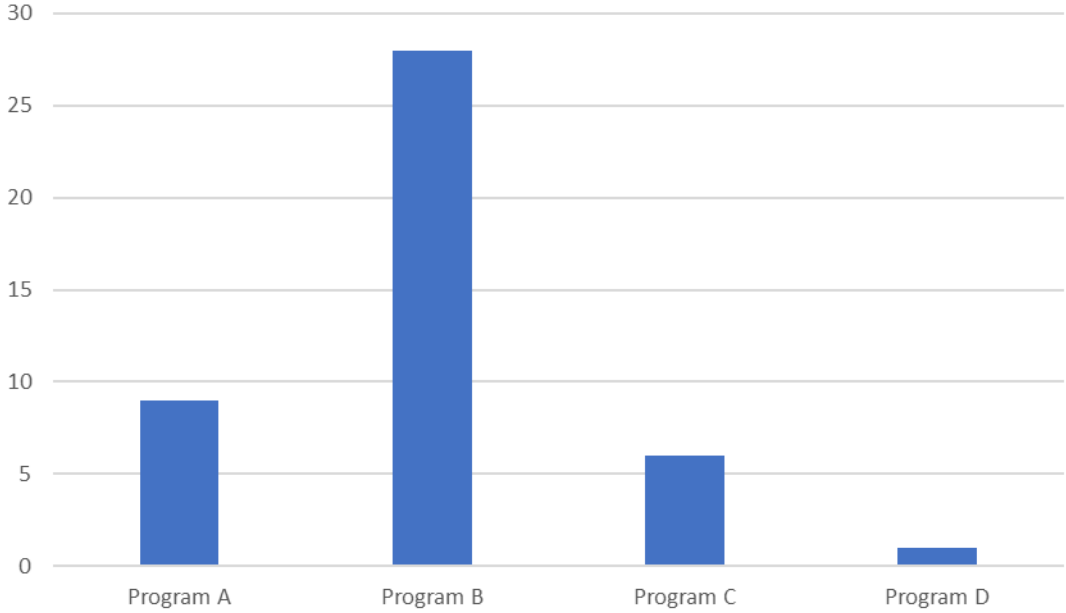

Run Time, hours
14
12
10
8
6
4
2
0
2010
2011
2012
2013
2014
2015
2016
2017
2018


Run Time, hours
14
12
10
8
6
4
2
0
900
800
700
600
500
400
300
200
100
0
Number of CPUs
2010
2011
2012
2013
2014
2015
2016
2017
2018


14
12
10
8
6
4
2
0
Program A
Program B
Program C
Program D

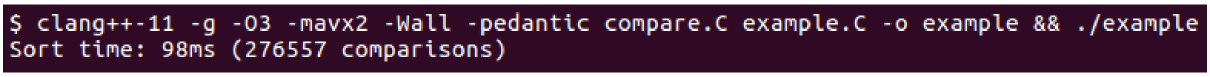


45
40
35
30
25
20
15
10
5
0
Program A
Program B
Program C
Program D

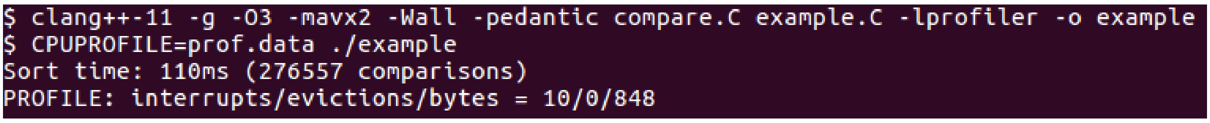


30
25
20
15
10
5
0
Program A
Program B
Program C
Program D

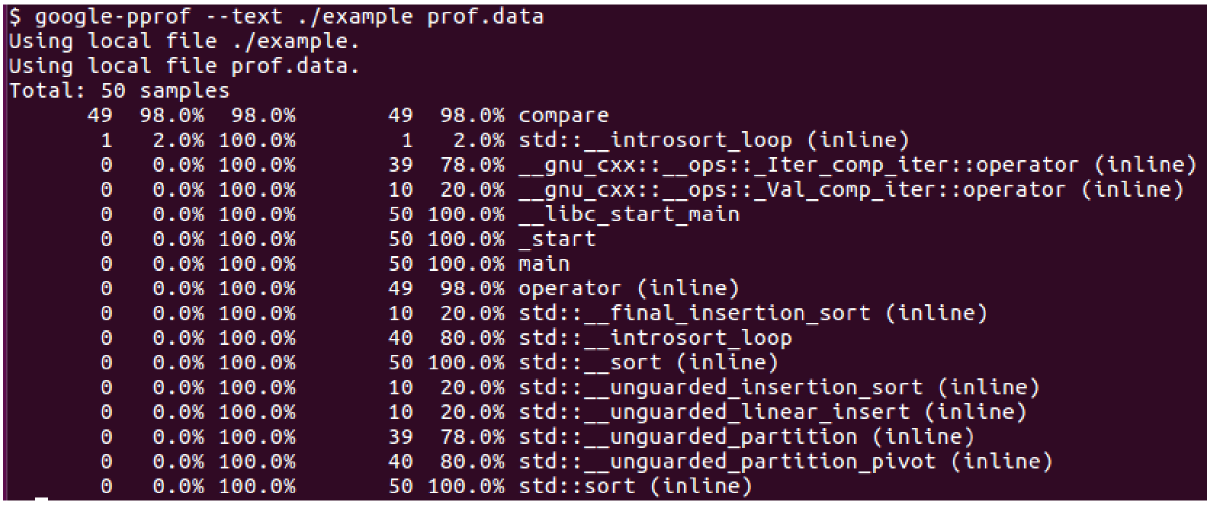
Chapter 2: Performance Measurements



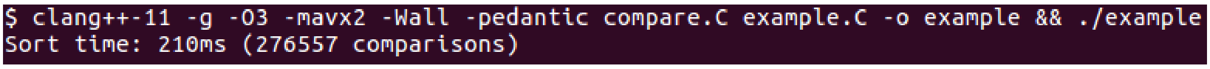
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic compare.C example.C -o example && ./example
Sort time: 98ms (276557 comparisons)
```



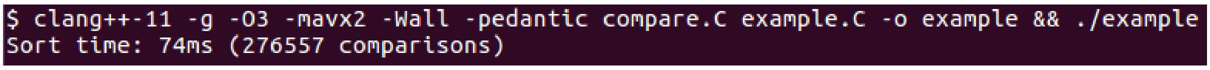
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic compare.C example.C -lprofiler -o example
$ CPUPROFILE=prof.data ./example
Sort time: 110ms (276557 comparisons)
PROFILE: interrupts/evictions/bytes = 10/0/848
```



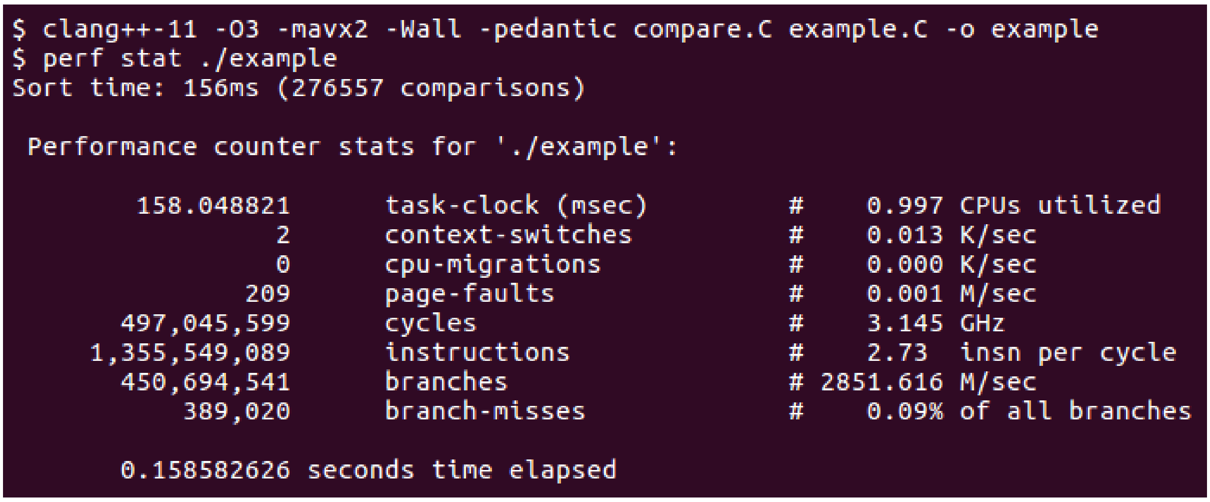
```
$ google-pprof --text ./example prof.data
Using local file ./example.
Using local file prof.data.
Total: 50 samples
      49  98.0%  98.0%       49  98.0% compare
       1   2.0% 100.0%        1   2.0% std::__introsort_loop (inline)
       0   0.0% 100.0%       39  78.0% __gnu_cxx::__ops::_Iter_comp_iter::operator (inline)
       0   0.0% 100.0%       10  20.0% __gnu_cxx::__ops::_Val_comp_iter::operator (inline)
       0   0.0% 100.0%       50 100.0% __libc_start_main
       0   0.0% 100.0%       50 100.0% _start
       0   0.0% 100.0%       50 100.0% main
       0   0.0% 100.0%       49  98.0% operator (inline)
       0   0.0% 100.0%       10  20.0% std::__final_insertion_sort (inline)
       0   0.0% 100.0%       40  80.0% std::__introsort_loop
       0   0.0% 100.0%       50 100.0% std::__sort (inline)
       0   0.0% 100.0%       10  20.0% std::__unguarded_insertion_sort (inline)
       0   0.0% 100.0%       10  20.0% std::__unguarded_linear_insert (inline)
       0   0.0% 100.0%       39  78.0% std::__unguarded_partition (inline)
       0   0.0% 100.0%       40  80.0% std::__unguarded_partition_pivot (inline)
       0   0.0% 100.0%       50 100.0% std::sort (inline)
```



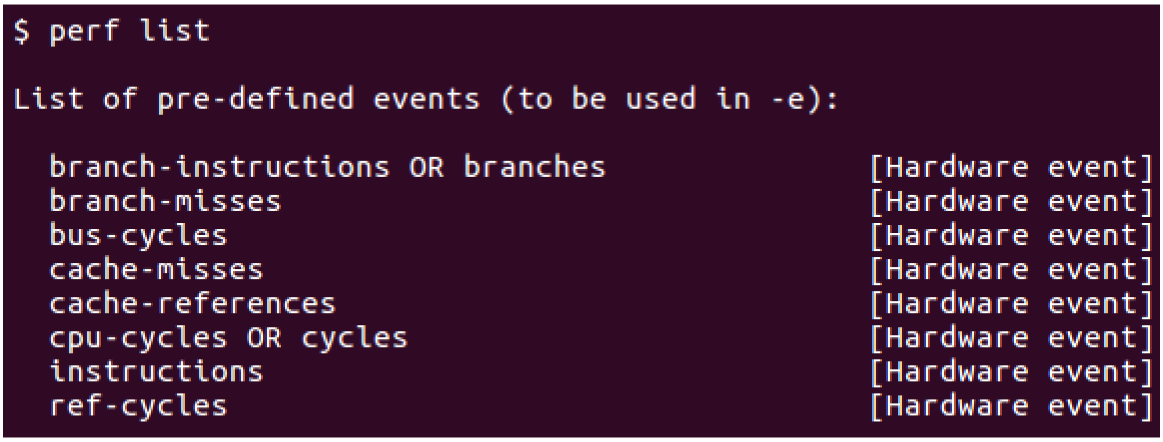
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic compare.C example.C -o example && ./example
Sort time: 210ms (276557 comparisons)
```



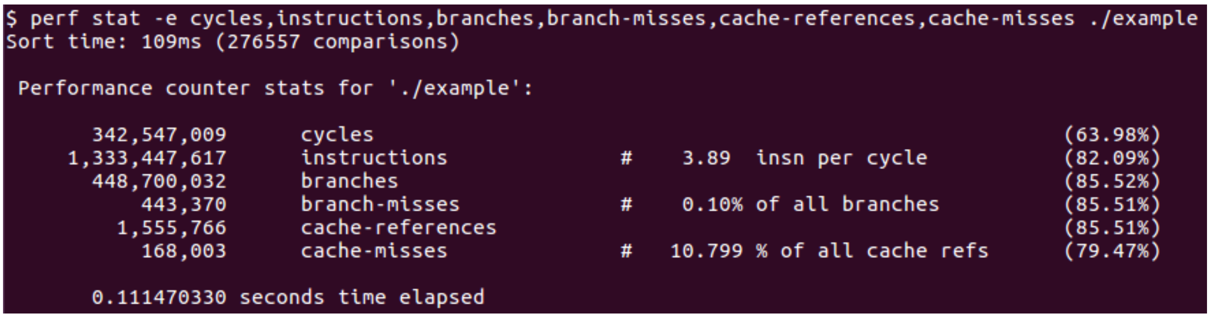
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic compare.C example.C -o example && ./example
Sort time: 74ms (276557 comparisons)
```



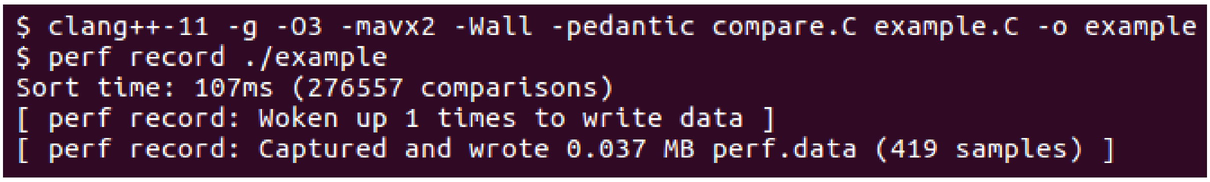
```
$ clang++-11 -O3 -mavx2 -Wall -pedantic compare.C example.C -o example
$ perf stat ./example
Sort time: 156ms (276557 comparisons)

 Performance counter stats for './example':

       158.048821      task-clock (msec)         #    0.997 CPUs utilized
                2      context-switches          #    0.013 K/sec
                0      cpu-migrations            #    0.000 K/sec
              209      page-faults               #    0.001 M/sec
      497,045,599      cycles                    #    3.145 GHz
    1,355,549,089      instructions              #    2.73  insn per cycle
      450,694,541      branches                  # 2851.616 M/sec
          389,020      branch-misses             #    0.09% of all branches

      0.158582626 seconds time elapsed
```



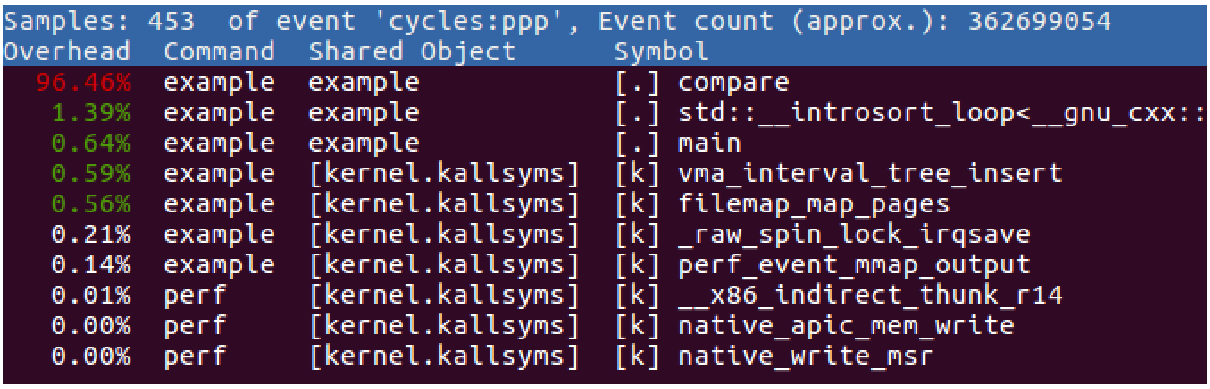
```
$ perf list

List of pre-defined events (to be used in -e):

  branch-instructions OR branches                    [Hardware event]
  branch-misses                                      [Hardware event]
  bus-cycles                                         [Hardware event]
  cache-misses                                       [Hardware event]
  cache-references                                   [Hardware event]
  cpu-cycles OR cycles                               [Hardware event]
  instructions                                       [Hardware event]
  ref-cycles                                         [Hardware event]
```



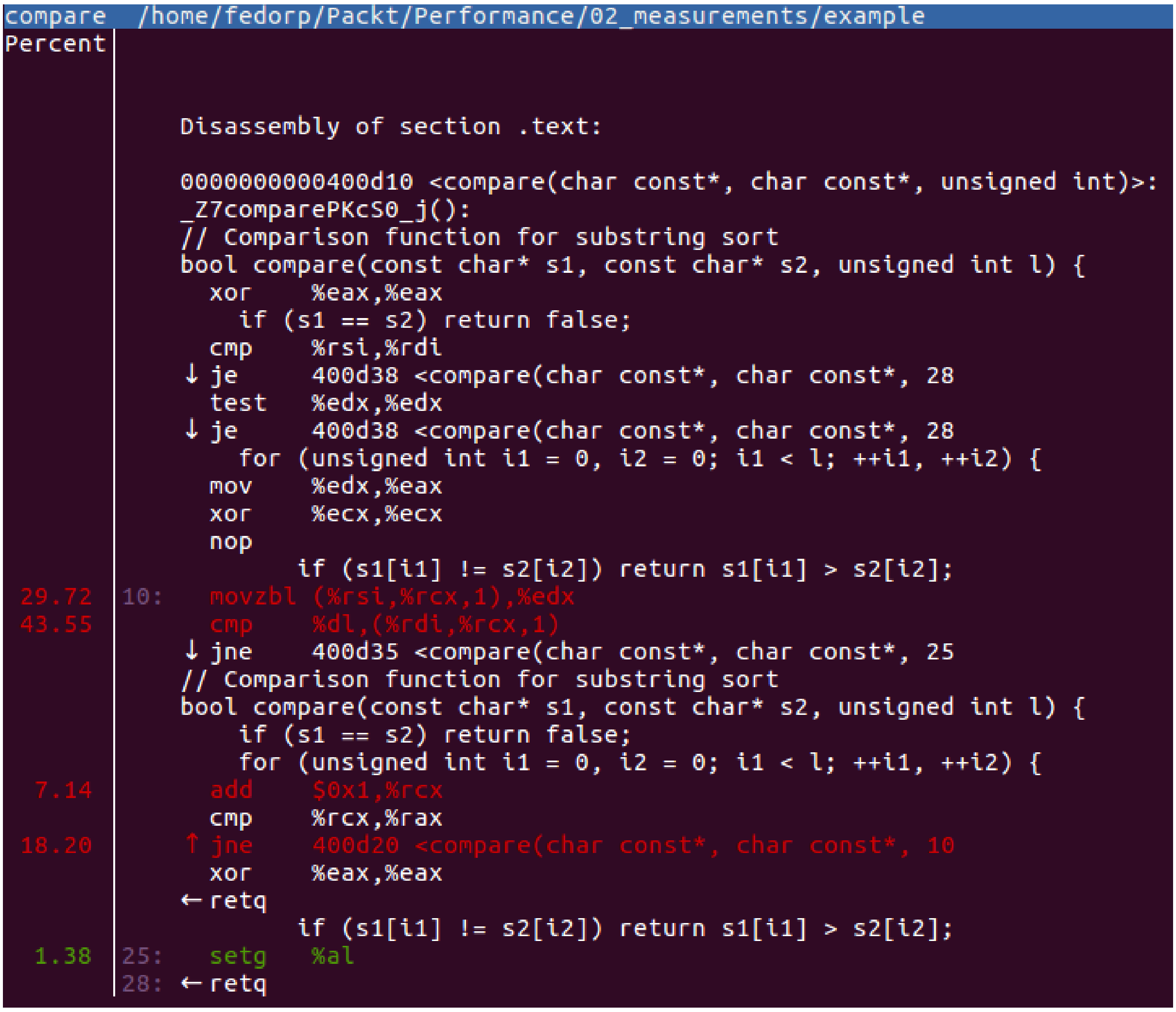
```
$ perf stat -e cycles,instructions,branches,branch-misses,cache-references,cache-misses ./example
Sort time: 109ms (276557 comparisons)

 Performance counter stats for './example':

       342,547,009      cycles                                                      (63.98%)
     1,333,447,617      instructions              #    3.89  insn per cycle         (82.09%)
       448,700,032      branches                                                    (85.52%)
           443,370      branch-misses             #    0.10% of all branches        (85.51%)
         1,555,766      cache-references                                            (85.51%)
           168,003      cache-misses              #   10.799 % of all cache refs    (79.47%)

       0.111470330 seconds time elapsed
```



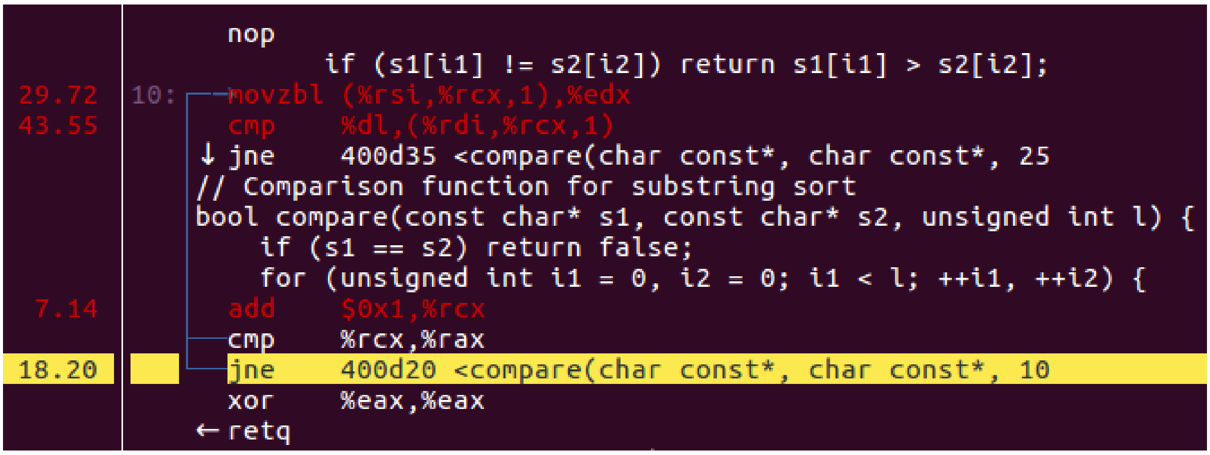
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic compare.C example.C -o example
$ perf record ./example
Sort time: 107ms (276557 comparisons)
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.037 MB perf.data (419 samples) ]
```



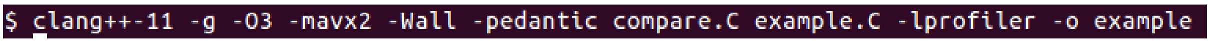
```
Samples: 453  of event 'cycles:ppp', Event count (approx.): 362699054
Overhead  Command  Shared Object      Symbol
  96.46%  example  example            [.] compare
   1.39%  example  example            [.] std::__introsort_loop<__gnu_cxx::
   0.64%  example  example            [.] main
   0.59%  example  [kernel.kallsyms]  [k] vma_interval_tree_insert
   0.56%  example  [kernel.kallsyms]  [k] filemap_map_pages
   0.21%  example  [kernel.kallsyms]  [k] _raw_spin_lock_irqsave
   0.14%  example  [kernel.kallsyms]  [k] perf_event_mmap_output
   0.01%  perf     [kernel.kallsyms]  [k] __x86_indirect_thunk_r14
   0.00%  perf     [kernel.kallsyms]  [k] native_apic_mem_write
   0.00%  perf     [kernel.kallsyms]  [k] native_write_msr
```



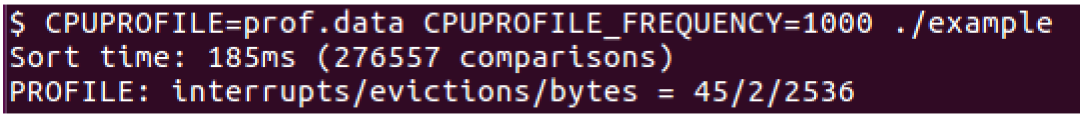
```
compare  /home/fedorp/Packt/Performance/02_measurements/example
Percent│
       │
       │
       │    Disassembly of section .text:
       │
       │    0000000000400d10 <compare(char const*, char const*, unsigned int)>:
       │    _Z7comparePKcS0_j():
       │    // Comparison function for substring sort
       │    bool compare(const char* s1, const char* s2, unsigned int l) {
       │      xor    %eax,%eax
       │        if (s1 == s2) return false;
       │      cmp    %rsi,%rdi
       │    ↓ je     400d38 <compare(char const*, char const*, 28
       │      test   %edx,%edx
       │    ↓ je     400d38 <compare(char const*, char const*, 28
       │        for (unsigned int i1 = 0, i2 = 0; i1 < l; ++i1, ++i2) {
       │      mov    %edx,%eax
       │      xor    %ecx,%ecx
       │      nop
       │            if (s1[i1] != s2[i2]) return s1[i1] > s2[i2];
 29.72 │10:   movzbl (%rsi,%rcx,1),%edx
 43.55 │      cmp    %dl,(%rdi,%rcx,1)
       │    ↓ jne    400d35 <compare(char const*, char const*, 25
       │    // Comparison function for substring sort
       │    bool compare(const char* s1, const char* s2, unsigned int l) {
       │        if (s1 == s2) return false;
       │        for (unsigned int i1 = 0, i2 = 0; i1 < l; ++i1, ++i2) {
  7.14 │      add    $0x1,%rcx
       │      cmp    %rcx,%rax
 18.20 │    ↑ jne    400d20 <compare(char const*, char const*, 10
       │      xor    %eax,%eax
       │    ← retq
       │            if (s1[i1] != s2[i2]) return s1[i1] > s2[i2];
  1.38 │25:   setg   %al
       │28: ← retq
```



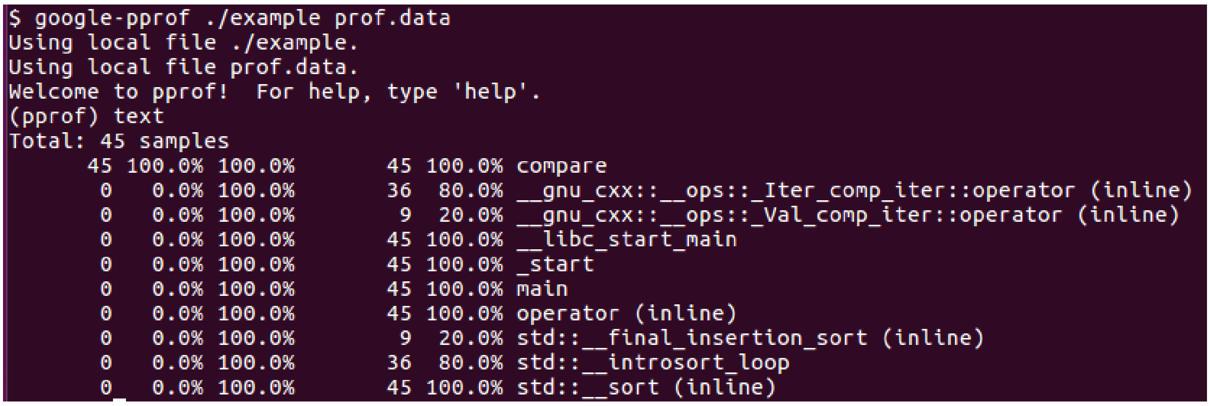
```
       │      nop
       │            if (s1[i1] != s2[i2]) return s1[i1] > s2[i2];
 29.72 │10: ┌─movzbl (%rsi,%rcx,1),%edx
 43.55 │    │ cmp    %dl,(%rdi,%rcx,1)
       │    │↓ jne    400d35 <compare(char const*, char const*, 25
       │    │// Comparison function for substring sort
       │    │bool compare(const char* s1, const char* s2, unsigned int l) {
       │    │    if (s1 == s2) return false;
       │    │    for (unsigned int i1 = 0, i2 = 0; i1 < l; ++i1, ++i2) {
  7.14 │    │ add    $0x1,%rcx
       │    ├─cmp    %rcx,%rax
 18.20 │    └─jne    400d20 <compare(char const*, char const*, 10
       │      xor    %eax,%eax
       │    ← retq
```



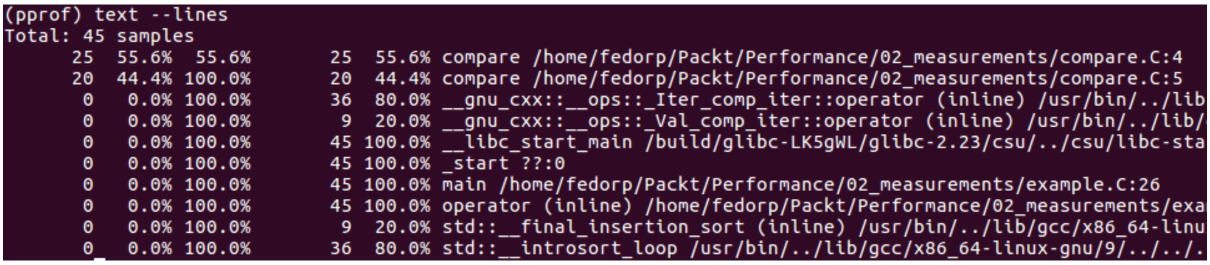
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic compare.C example.C -lprofiler -o example
```



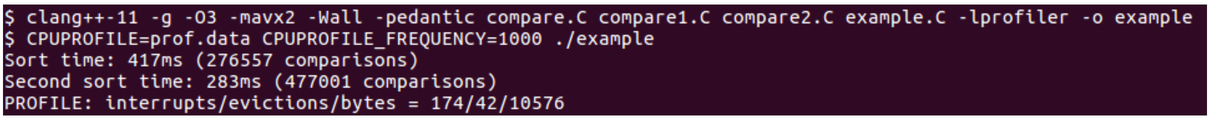
```
$ CPUPROFILE=prof.data CPUPROFILE_FREQUENCY=1000 ./example
Sort time: 185ms (276557 comparisons)
PROFILE: interrupts/evictions/bytes = 45/2/2536
```



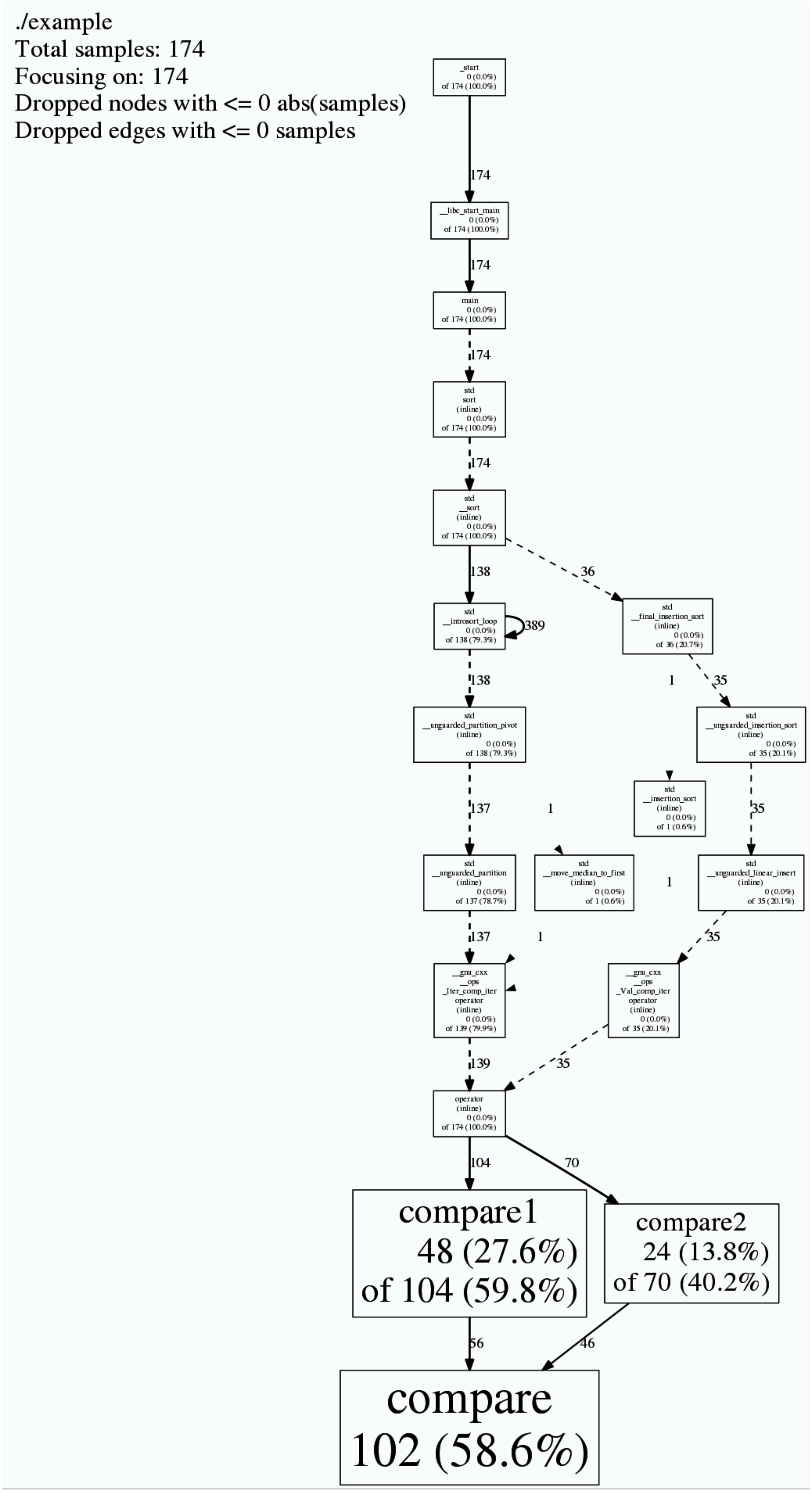
```
$ google-pprof ./example prof.data
Using local file ./example.
Using local file prof.data.
Welcome to pprof!  For help, type 'help'.
(pprof) text
Total: 45 samples
      45 100.0% 100.0%       45 100.0% compare
       0   0.0% 100.0%       36  80.0% __gnu_cxx::__ops::_Iter_comp_iter::operator (inline)
       0   0.0% 100.0%        9  20.0% __gnu_cxx::__ops::_Val_comp_iter::operator (inline)
       0   0.0% 100.0%       45 100.0% __libc_start_main
       0   0.0% 100.0%       45 100.0% _start
       0   0.0% 100.0%       45 100.0% main
       0   0.0% 100.0%       45 100.0% operator (inline)
       0   0.0% 100.0%        9  20.0% std::__final_insertion_sort (inline)
       0   0.0% 100.0%       36  80.0% std::__introsort_loop
       0   0.0% 100.0%       45 100.0% std::__sort (inline)
```

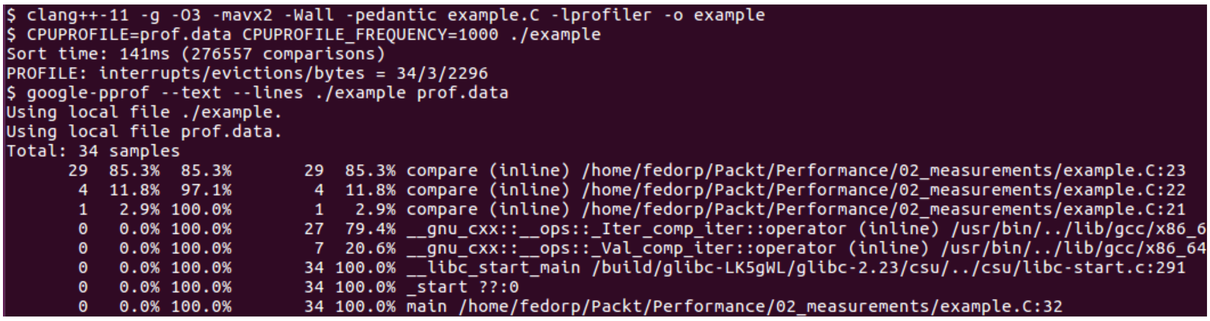


```
(pprof) text --lines
Total: 45 samples
      25  55.6%  55.6%       25  55.6% compare /home/fedorp/Packt/Performance/02_measurements/compare.C:4
      20  44.4% 100.0%       20  44.4% compare /home/fedorp/Packt/Performance/02_measurements/compare.C:5
       0   0.0% 100.0%       36  80.0% __gnu_cxx::__ops::_Iter_comp_iter::operator (inline) /usr/bin/../lib
       0   0.0% 100.0%        9  20.0% __gnu_cxx::__ops::_Val_comp_iter::operator (inline) /usr/bin/../lib/
       0   0.0% 100.0%       45 100.0% __libc_start_main /build/glibc-LK5gWL/glibc-2.23/csu/../csu/libc-sta
       0   0.0% 100.0%       45 100.0% _start ??:0
       0   0.0% 100.0%       45 100.0% main /home/fedorp/Packt/Performance/02_measurements/example.C:26
       0   0.0% 100.0%       45 100.0% operator (inline) /home/fedorp/Packt/Performance/02_measurements/exa
       0   0.0% 100.0%        9  20.0% std::__final_insertion_sort (inline) /usr/bin/../lib/gcc/x86_64-linu
       0   0.0% 100.0%       36  80.0% std::__introsort_loop /usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../.
```

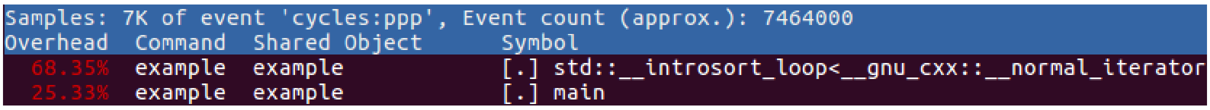


```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic compare.C compare1.C compare2.C example.C -lprofiler -o example
$ CPUPROFILE=prof.data CPUPROFILE_FREQUENCY=1000 ./example
Sort time: 417ms (276557 comparisons)
Second sort time: 283ms (477001 comparisons)
PROFILE: interrupts/evictions/bytes = 174/42/10576
```

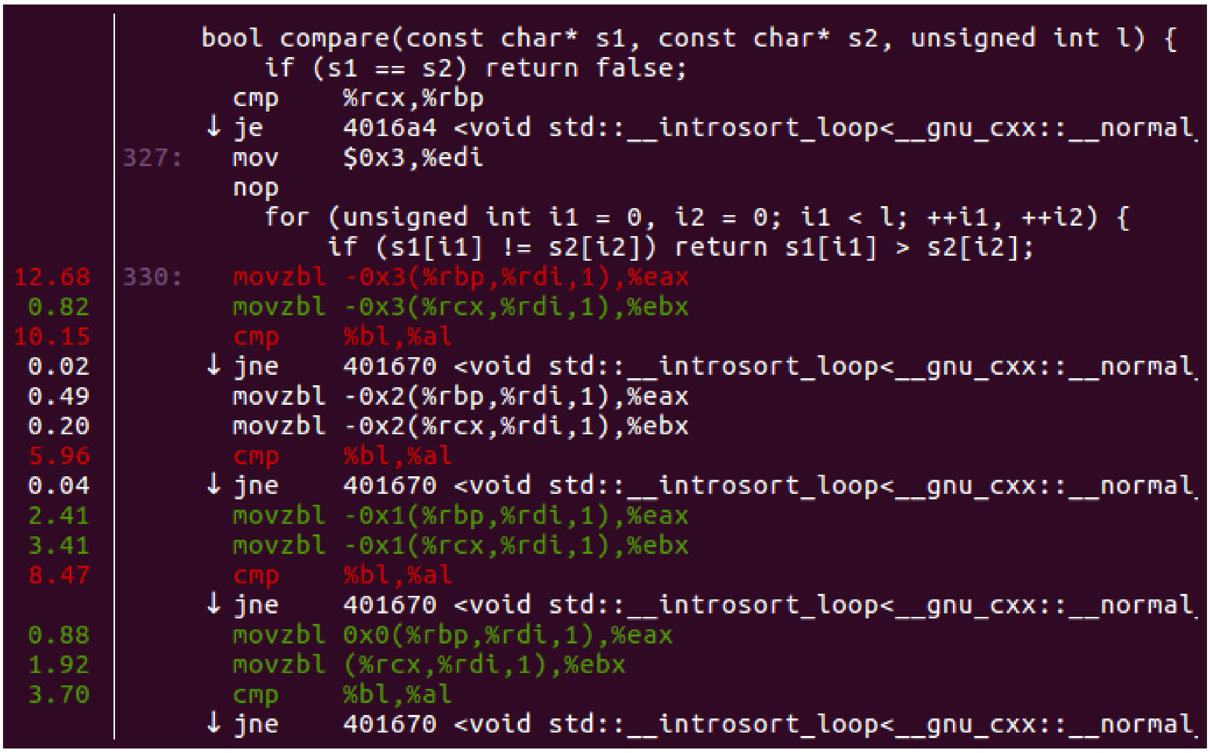

./example
Total samples: 174
Focusing on: 174
Dropped nodes with <= 0 abs(samples)
Dropped edges with <= 0 samples
_start 0 (0.0%) of 174 (100.0%)
174
__libc_start_main 0 (0.0%) of 174 (100.0%)
174
main 0 (0.0%) of 174 (100.0%)
174
std sort (inline) 0 (0.0%) of 174 (100.0%)
174
std __sort (inline) 0 (0.0%) of 174 (100.0%)
138
36
std __introsort_loop 0 (0.0%) of 138 (79.3%)
389
std __final_insertion_sort (inline) 0 (0.0%) of 36 (20.7%)
138
1
35
std __unguarded_partition_pivot (inline) 0 (0.0%) of 138 (79.3%)
std __unguarded_insertion_sort (inline) 0 (0.0%) of 35 (20.1%)
std __insertion_sort (inline) 0 (0.0%) of 1 (0.6%)
137
1
35
std __unguarded_partition (inline) 0 (0.0%) of 137 (78.7%)
std __move_median_to_first (inline) 0 (0.0%) of 1 (0.6%)
1
std __unguarded_linear_insert (inline) 0 (0.0%) of 35 (20.1%)
137
1
35
__gnu_cxx __ops _Iter_comp_iter operator (inline) 0 (0.0%) of 139 (79.9%)
__gnu_cxx __ops _Val_comp_iter operator (inline) 0 (0.0%) of 35 (20.1%)
139
35
operator (inline) 0 (0.0%) of 174 (100.0%)
104
70
compare1 48 (27.6%) of 104 (59.8%)
compare2 24 (13.8%) of 70 (40.2%)
56
46
compare 102 (58.6%)



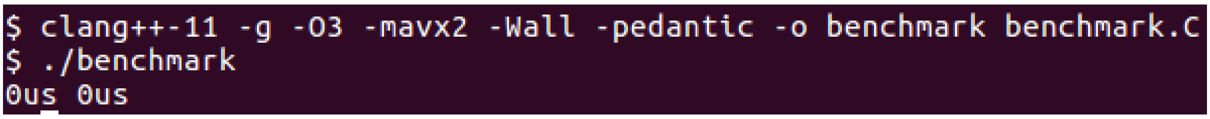
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic example.C -lprofiler -o example
$ CPUPROFILE=prof.data CPUPROFILE_FREQUENCY=1000 ./example
Sort time: 141ms (276557 comparisons)
PROFILE: interrupts/evictions/bytes = 34/3/2296
$ google-pprof --text --lines ./example prof.data
Using local file ./example.
Using local file prof.data.
Total: 34 samples
      29  85.3%  85.3%       29  85.3% compare (inline) /home/fedorp/Packt/Performance/02_measurements/example.C:23
       4  11.8%  97.1%        4  11.8% compare (inline) /home/fedorp/Packt/Performance/02_measurements/example.C:22
       1   2.9% 100.0%        1   2.9% compare (inline) /home/fedorp/Packt/Performance/02_measurements/example.C:21
       0   0.0% 100.0%       27  79.4% __gnu_cxx::__ops::_Iter_comp_iter::operator (inline) /usr/bin/../lib/gcc/x86_6
       0   0.0% 100.0%        7  20.6% __gnu_cxx::__ops::_Val_comp_iter::operator (inline) /usr/bin/../lib/gcc/x86_64
       0   0.0% 100.0%       34 100.0% __libc_start_main /build/glibc-LK5gWL/glibc-2.23/csu/../csu/libc-start.c:291
       0   0.0% 100.0%       34 100.0% _start ??:0
       0   0.0% 100.0%       34 100.0% main /home/fedorp/Packt/Performance/02_measurements/example.C:32
```



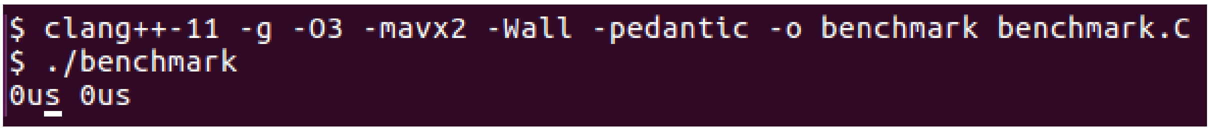
```
Samples: 7K of event 'cycles:ppp', Event count (approx.): 7464000
Overhead  Command  Shared Object      Symbol
  68.35%  example  example            [.] std::__introsort_loop<__gnu_cxx::__normal_iterator
  25.33%  example  example            [.] main
```



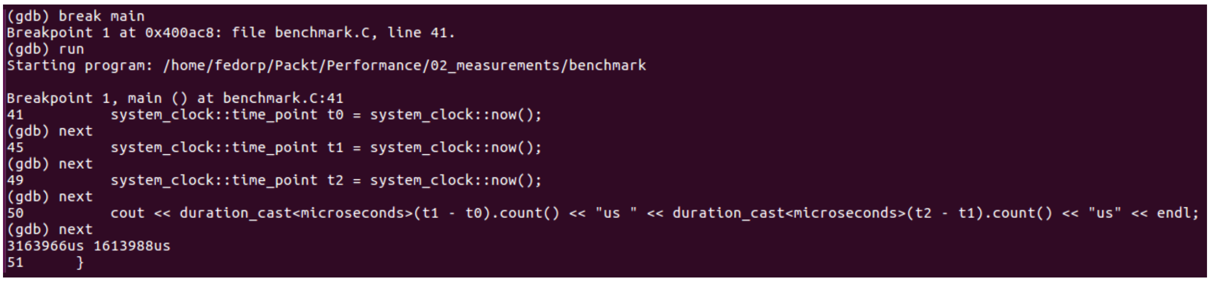
```
              │      bool compare(const char* s1, const char* s2, unsigned int l) {
              │        if (s1 == s2) return false;
              │        cmp    %rcx,%rbp
              │      ↓ je     4016a4 <void std::__introsort_loop<__gnu_cxx::__normal_
              │327:    mov    $0x3,%edi
              │        nop
              │        for (unsigned int i1 = 0, i2 = 0; i1 < l; ++i1, ++i2) {
              │          if (s1[i1] != s2[i2]) return s1[i1] > s2[i2];
  12.68       │330:    movzbl -0x3(%rbp,%rdi,1),%eax
   0.82       │        movzbl -0x3(%rcx,%rdi,1),%ebx
  10.15       │        cmp    %bl,%al
   0.02       │      ↓ jne    401670 <void std::__introsort_loop<__gnu_cxx::__normal_
   0.49       │        movzbl -0x2(%rbp,%rdi,1),%eax
   0.20       │        movzbl -0x2(%rcx,%rdi,1),%ebx
   5.96       │        cmp    %bl,%al
   0.04       │      ↓ jne    401670 <void std::__introsort_loop<__gnu_cxx::__normal_
   2.41       │        movzbl -0x1(%rbp,%rdi,1),%eax
   3.41       │        movzbl -0x1(%rcx,%rdi,1),%ebx
   8.47       │        cmp    %bl,%al
              │      ↓ jne    401670 <void std::__introsort_loop<__gnu_cxx::__normal_
   0.88       │        movzbl 0x0(%rbp,%rdi,1),%eax
   1.92       │        movzbl (%rcx,%rdi,1),%ebx
   3.70       │        cmp    %bl,%al
              │      ↓ jne    401670 <void std::__introsort_loop<__gnu_cxx::__normal_
```



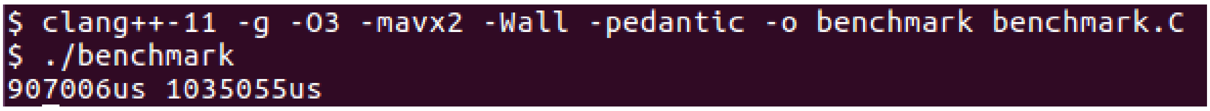
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic -o benchmark benchmark.C
$ ./benchmark
0us 0us
```



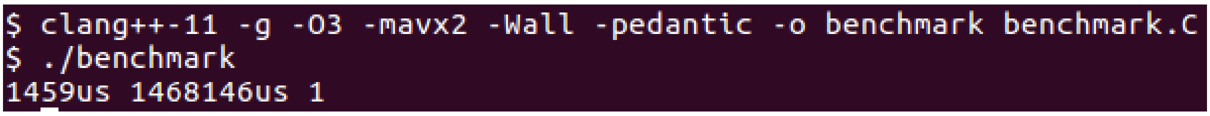
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic -o benchmark benchmark.C
$ ./benchmark
0us 0us
```



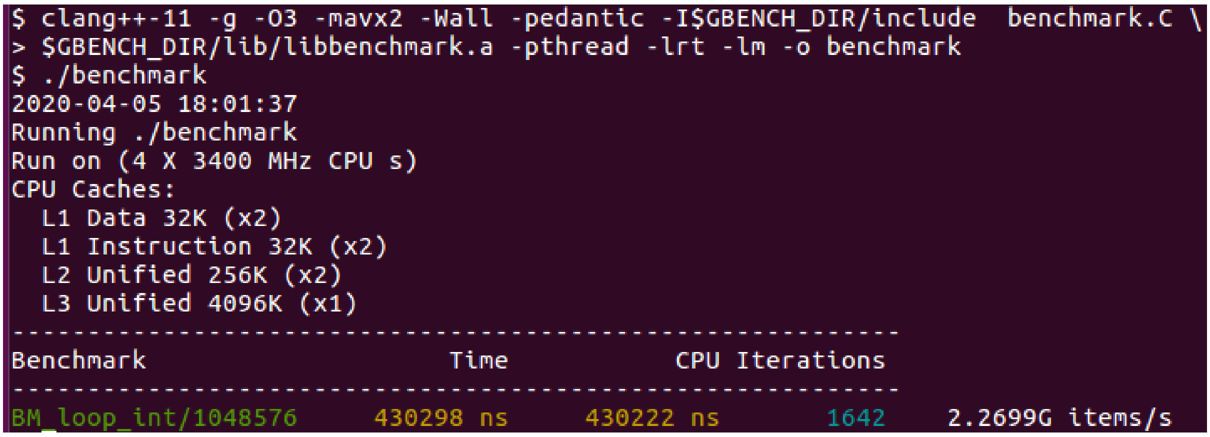
```
(gdb) break main
Breakpoint 1 at 0x400ac8: file benchmark.C, line 41.
(gdb) run
Starting program: /home/fedorp/Packt/Performance/02_measurements/benchmark

Breakpoint 1, main () at benchmark.C:41
41          system_clock::time_point t0 = system_clock::now();
(gdb) next
45          system_clock::time_point t1 = system_clock::now();
(gdb) next
49          system_clock::time_point t2 = system_clock::now();
(gdb) next
50          cout << duration_cast<microseconds>(t1 - t0).count() << "us " << duration_cast<microseconds>(t2 - t1).count() << "us" << endl;
(gdb) next
3163966us 1613988us
51      }
```



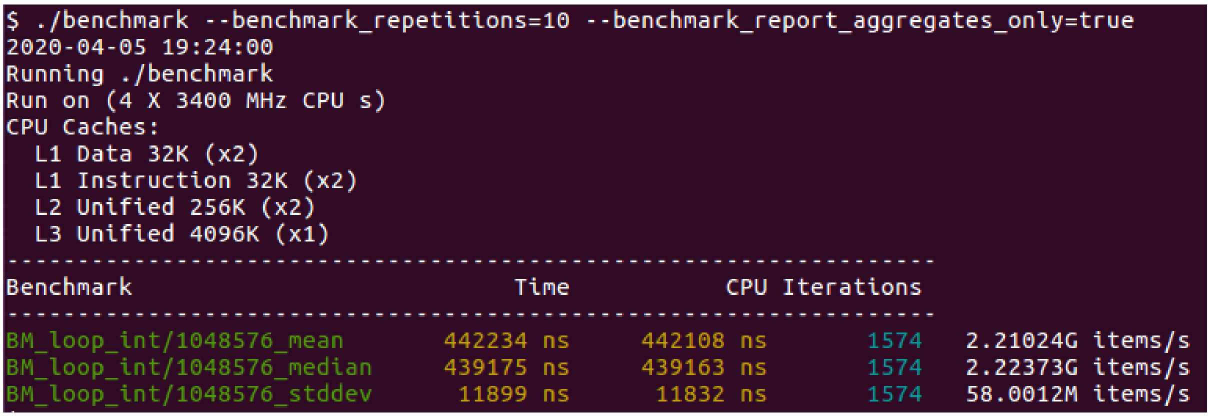
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic -o benchmark benchmark.C
$ ./benchmark
907006us 1035055us
```



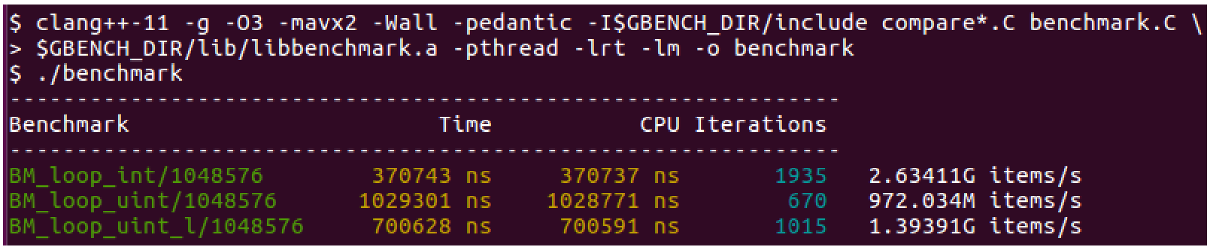
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic -o benchmark benchmark.C
$ ./benchmark
1459us 1468146us 1
```



```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic -I$GBENCH_DIR/include  benchmark.C \
> $GBENCH_DIR/lib/libbenchmark.a -pthread -lrt -lm -o benchmark
$ ./benchmark
2020-04-05 18:01:37
Running ./benchmark
Run on (4 X 3400 MHz CPU s)
CPU Caches:
  L1 Data 32K (x2)
  L1 Instruction 32K (x2)
  L2 Unified 256K (x2)
  L3 Unified 4096K (x1)
-----------------------------------------------------------
Benchmark                       Time             CPU Iterations
-----------------------------------------------------------
BM_loop_int/1048576      430298 ns      430222 ns         1642     2.2699G items/s
```

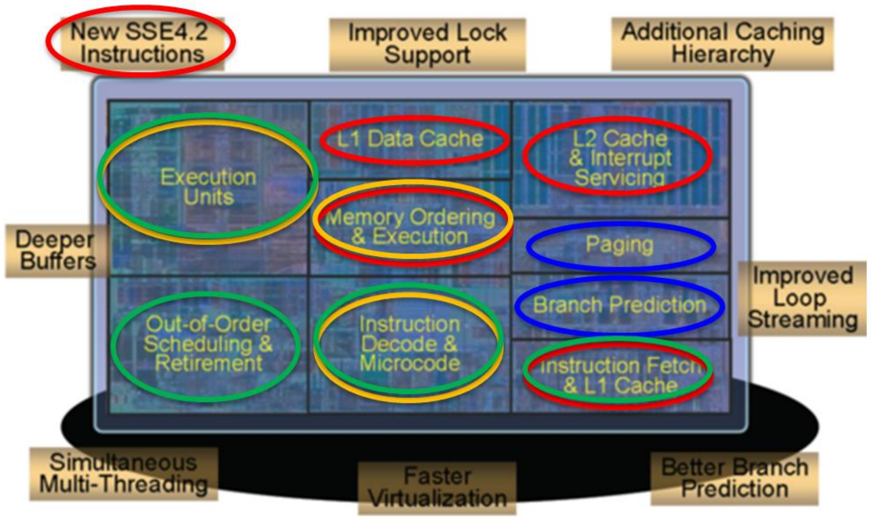


```
$ ./benchmark --benchmark_repetitions=10 --benchmark_report_aggregates_only=true
2020-04-05 19:24:00
Running ./benchmark
Run on (4 X 3400 MHz CPU s)
CPU Caches:
  L1 Data 32K (x2)
  L1 Instruction 32K (x2)
  L2 Unified 256K (x2)
  L3 Unified 4096K (x1)
------------------------------------------------------------------
Benchmark                              Time             CPU Iterations
------------------------------------------------------------------
BM_loop_int/1048576_mean        442234 ns      442108 ns         1574   2.21024G items/s
BM_loop_int/1048576_median      439175 ns      439163 ns         1574   2.22373G items/s
BM_loop_int/1048576_stddev       11899 ns       11832 ns         1574   58.0012M items/s
```



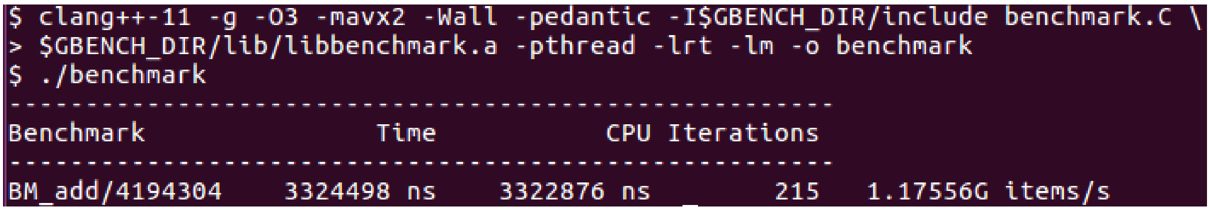
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic -I$GBENCH_DIR/include compare*.C benchmark.C \
> $GBENCH_DIR/lib/libbenchmark.a -pthread -lrt -lm -o benchmark
$ ./benchmark
--------------------------------------------------------------
Benchmark                        Time             CPU Iterations
--------------------------------------------------------------
BM_loop_int/1048576         370743 ns      370737 ns         1935   2.63411G items/s
BM_loop_uint/1048576       1029301 ns     1028771 ns          670   972.034M items/s
BM_loop_uint_l/1048576      700628 ns      700591 ns         1015   1.39391G items/s
```

Chapter 3: CPU Architecture, Resources, and Performance Implications

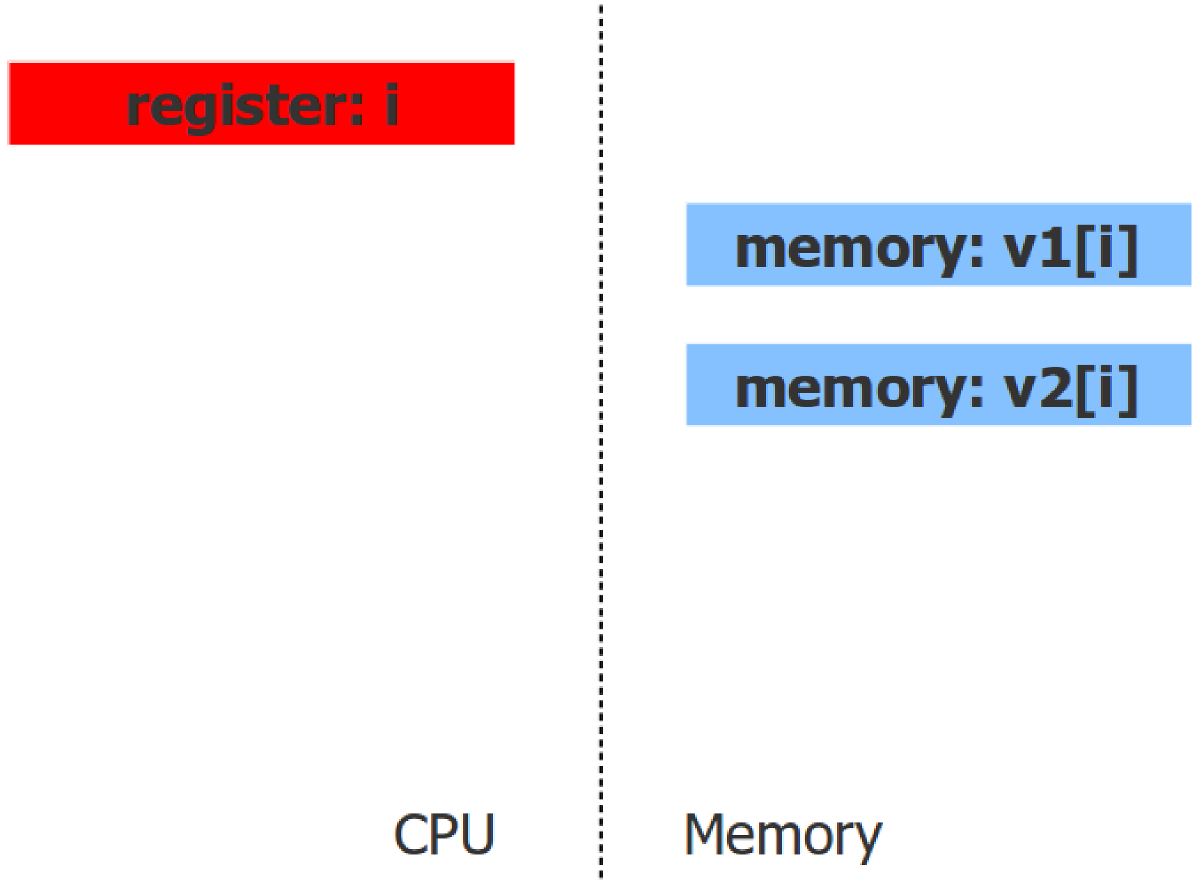


New SSE4.2 Instructions
Improved Lock Support
Additional Caching Hierarchy
Execution Units
L1 Data Cache
L2 Cache & Interrupt Servicing
Memory Ordering & Execution
Paging
Deeper Buffers
Improved Loop Streaming
Branch Prediction
Out-of-Order Scheduling & Retirement
Instruction Decode & Microcode
Instruction Fetch & L1 Cache
Simultaneous Multi-Threading
Faster Virtualization
Better Branch Prediction

- ➢ “Just works”
- ➢ Compiler takes care of it
- ➢ Needs care
- ➢ “Just works” but Good to know



```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic -I$GBENCH_DIR/include benchmark.C \
> $GBENCH_DIR/lib/libbenchmark.a -pthread -lrt -lm -o benchmark
$ ./benchmark
------------------------------------------------------
Benchmark                 Time             CPU Iterations
------------------------------------------------------
BM_add/4194304     3324498 ns     3322876 ns       215    1.17556G items/s
```



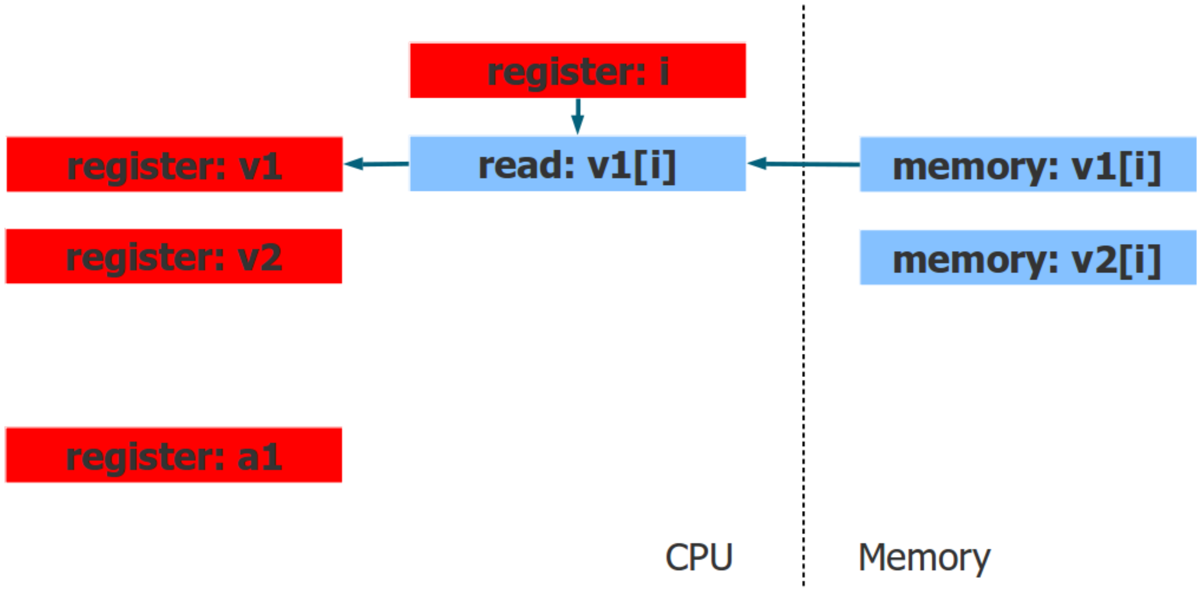
register: i

memory: v1[i]

memory: v2[i]

CPU

Memory



register: i

register: v1

read: v1[i]

memory: v1[i]

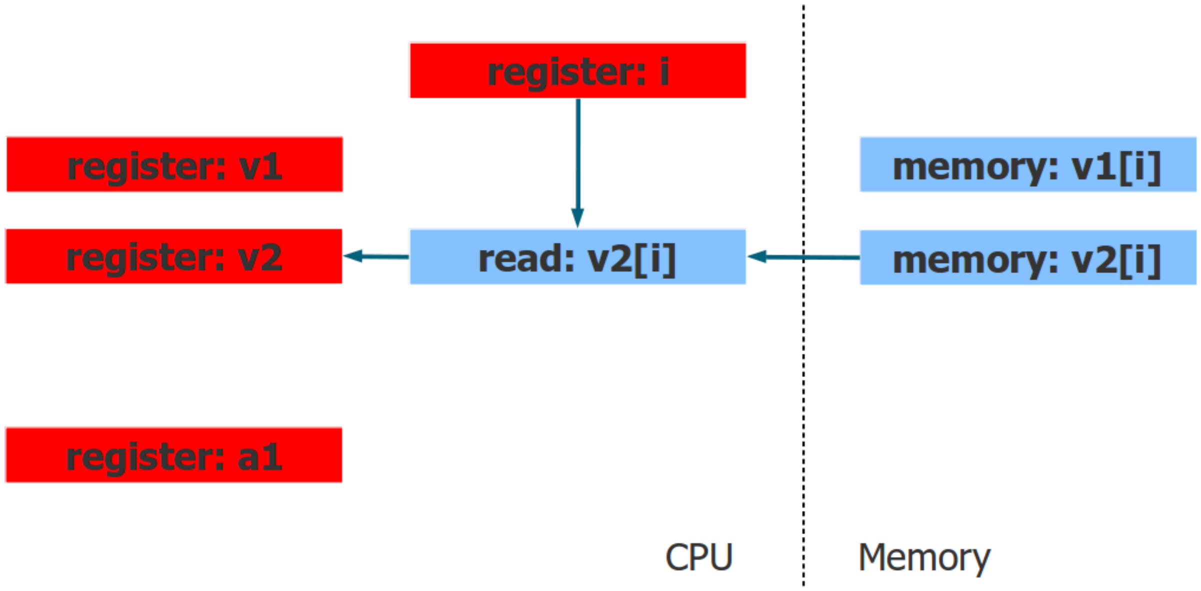
register: v2

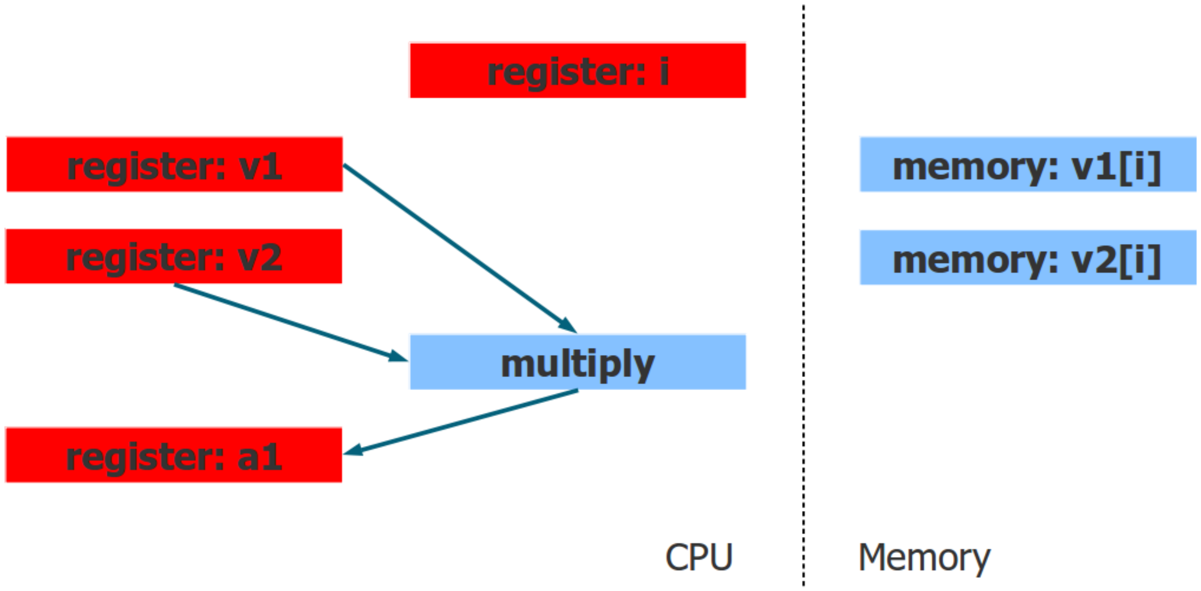
memory: v2[i]

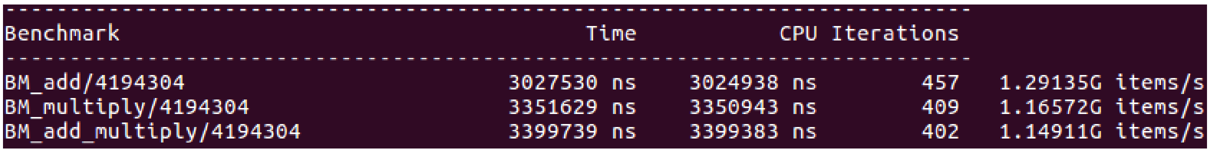
register: a1

CPU

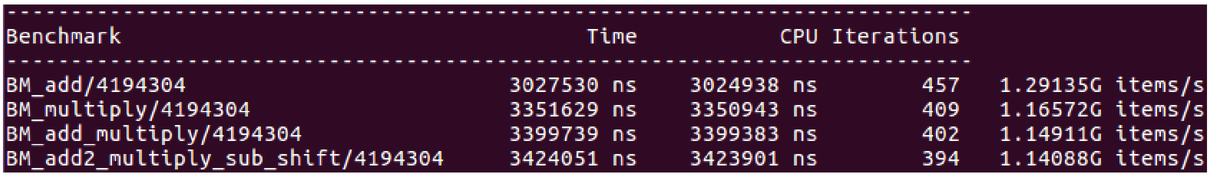
Memory



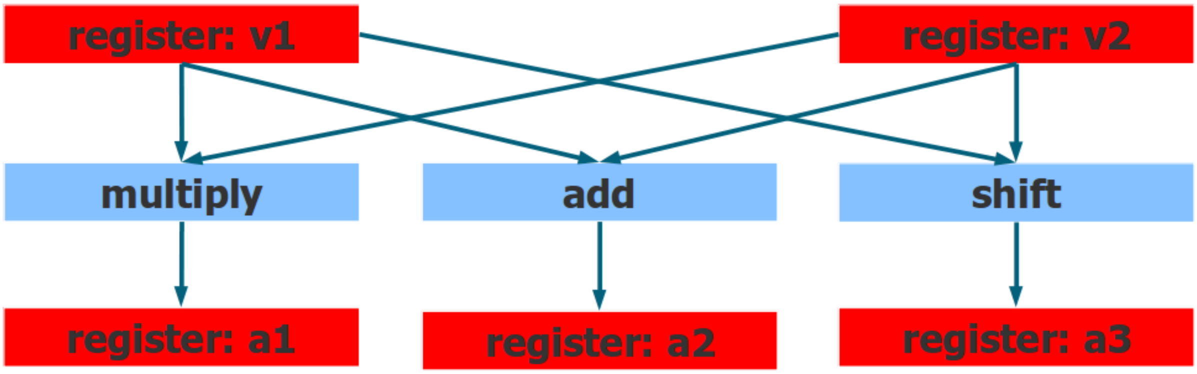


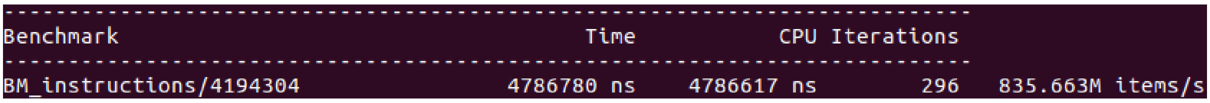


```
----------------------------------------------------------------------------
Benchmark                                   Time             CPU Iterations
----------------------------------------------------------------------------
BM_add/4194304                        3027530 ns      3024938 ns        457    1.29135G items/s
BM_multiply/4194304                   3351629 ns      3350943 ns        409    1.16572G items/s
BM_add_multiply/4194304               3399739 ns      3399383 ns        402    1.14911G items/s
```

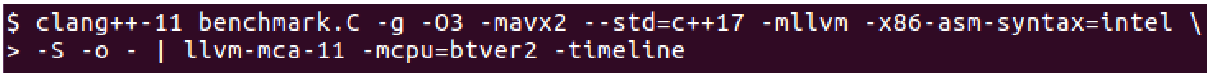


```
----------------------------------------------------------------------------
Benchmark                                   Time             CPU Iterations
----------------------------------------------------------------------------
BM_add/4194304                        3027530 ns      3024938 ns        457    1.29135G items/s
BM_multiply/4194304                   3351629 ns      3350943 ns        409    1.16572G items/s
BM_add_multiply/4194304               3399739 ns      3399383 ns        402    1.14911G items/s
BM_add2_multiply_sub_shift/4194304    3424051 ns      3423901 ns        394    1.14088G items/s
```

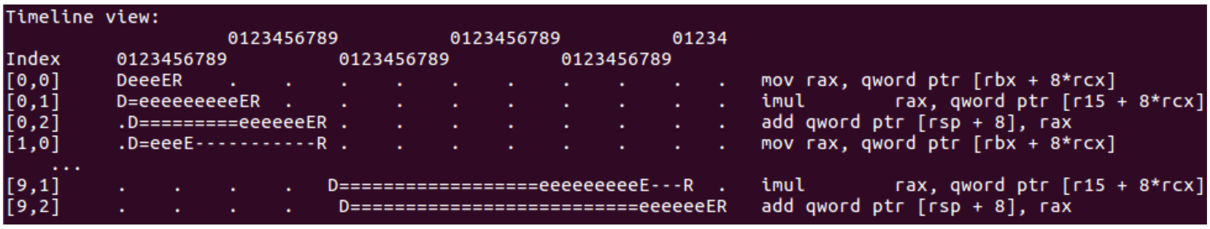




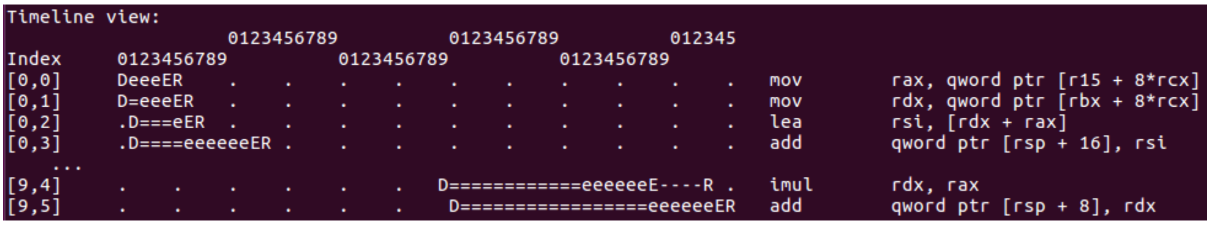
```
------------------------------------------------------------------
Benchmark                                Time             CPU Iterations
------------------------------------------------------------------
BM_instructions/4194304            4786780 ns      4786617 ns          296    835.663M items/s
```



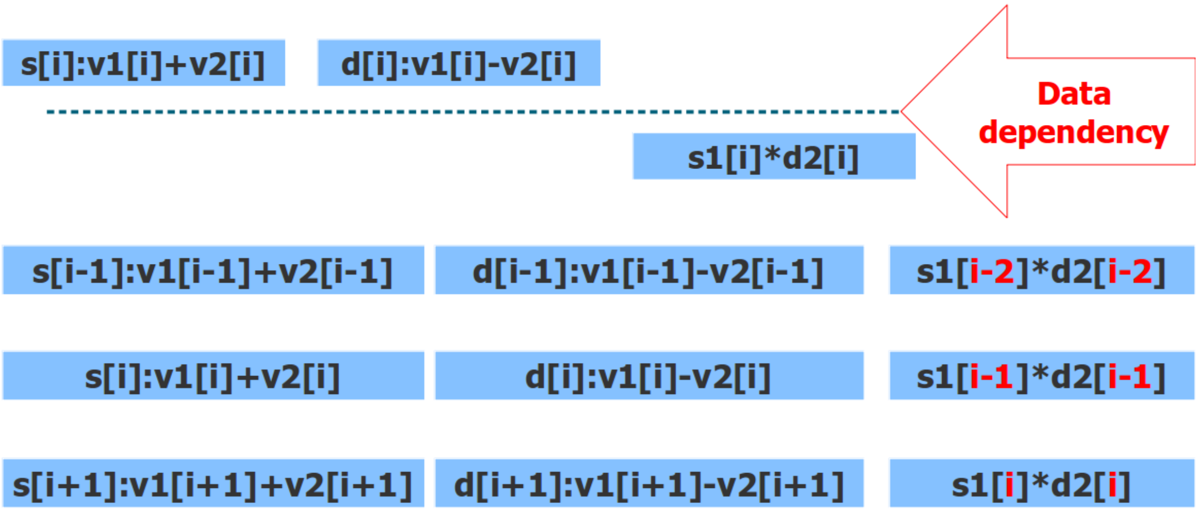
```
$ clang++-11 benchmark.C -g -O3 -mavx2 --std=c++17 -mllvm -x86-asm-syntax=intel \
> -S -o - | llvm-mca-11 -mcpu=btver2 -timeline
```

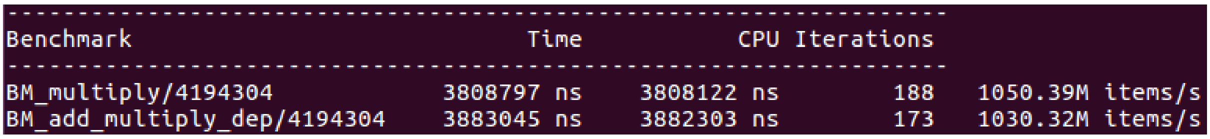


```
Timeline view:
                    0123456789          0123456789          01234
Index     0123456789          0123456789          0123456789
[0,0]     DeeeER    .    .    .    .    .    .    .    .    .   .   mov rax, qword ptr [rbx + 8*rcx]
[0,1]     D=eeeeeeeeeER  .    .    .    .    .    .    .    .   .   imul        rax, qword ptr [r15 + 8*rcx]
[0,2]     .D=========eeeeeeER .    .    .    .    .    .    .   .   add qword ptr [rsp + 8], rax
[1,0]     .D=eeeE-----------R .    .    .    .    .    .    .   .   mov rax, qword ptr [rbx + 8*rcx]
    ...
[9,1]     .    .    .    .   D==================eeeeeeeeeE---R  .   imul        rax, qword ptr [r15 + 8*rcx]
[9,2]     .    .    .    .    D==========================eeeeeeER   add qword ptr [rsp + 8], rax
```

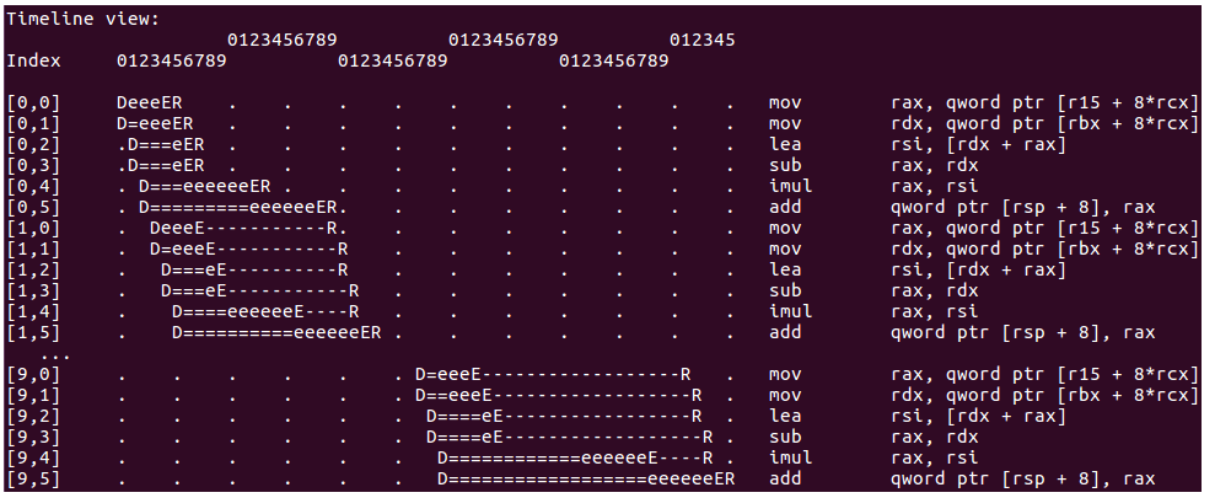


```
Timeline view:
                    0123456789          0123456789          012345
Index     0123456789          0123456789          0123456789
[0,0]     DeeeER    .    .    .    .    .    .    .    .    .    .   mov        rax, qword ptr [r15 + 8*rcx]
[0,1]     D=eeeER   .    .    .    .    .    .    .    .    .    .   mov        rdx, qword ptr [rbx + 8*rcx]
[0,2]     .D===eER  .    .    .    .    .    .    .    .    .    .   lea        rsi, [rdx + rax]
[0,3]     .D====eeeeeeER .    .    .    .    .    .    .    .    .   add        qword ptr [rsp + 16], rsi
    ...
[9,4]     .    .    .    .    .    .   D============eeeeeeE----R .   imul       rdx, rax
[9,5]     .    .    .    .    .    .    D=================eeeeeeER   add        qword ptr [rsp + 8], rdx
```

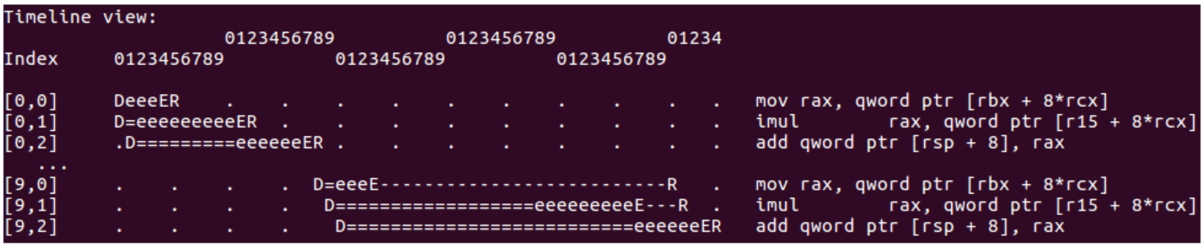




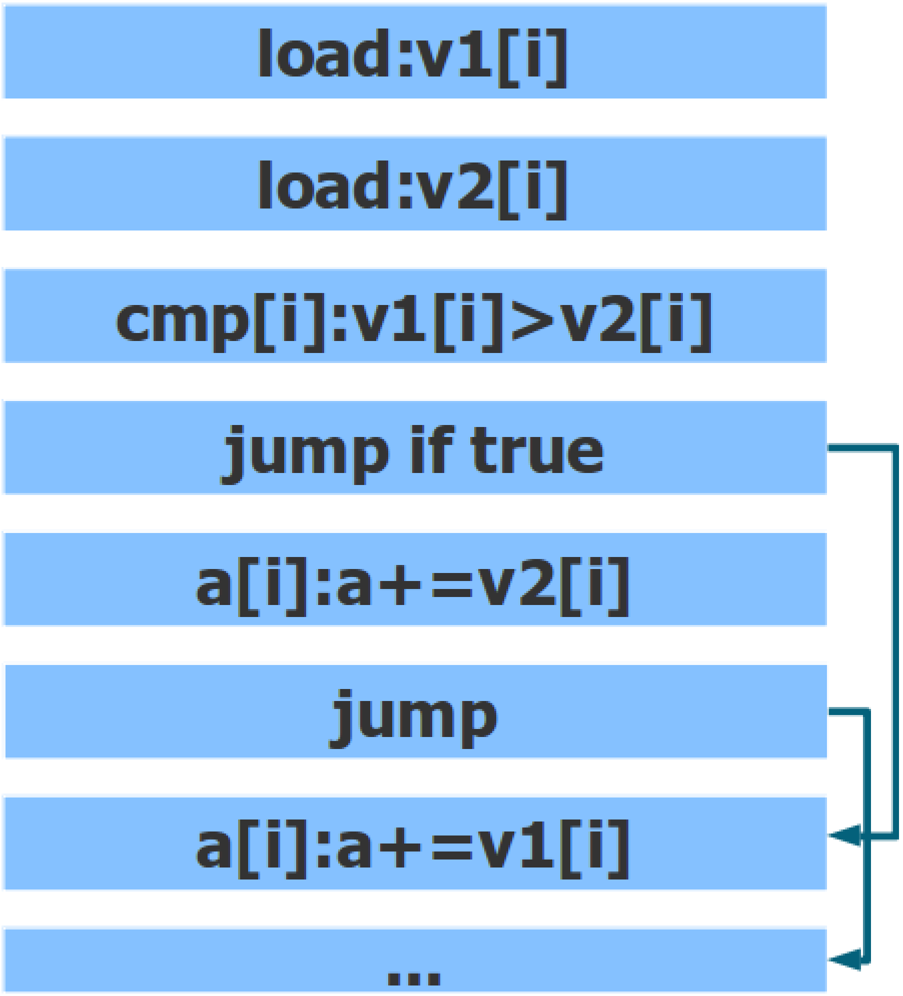
```
-----------------------------------------------------------------
Benchmark                             Time             CPU Iterations
-----------------------------------------------------------------
BM_multiply/4194304             3808797 ns      3808122 ns          188    1050.39M items/s
BM_add_multiply_dep/4194304     3883045 ns      3882303 ns          173    1030.32M items/s
```

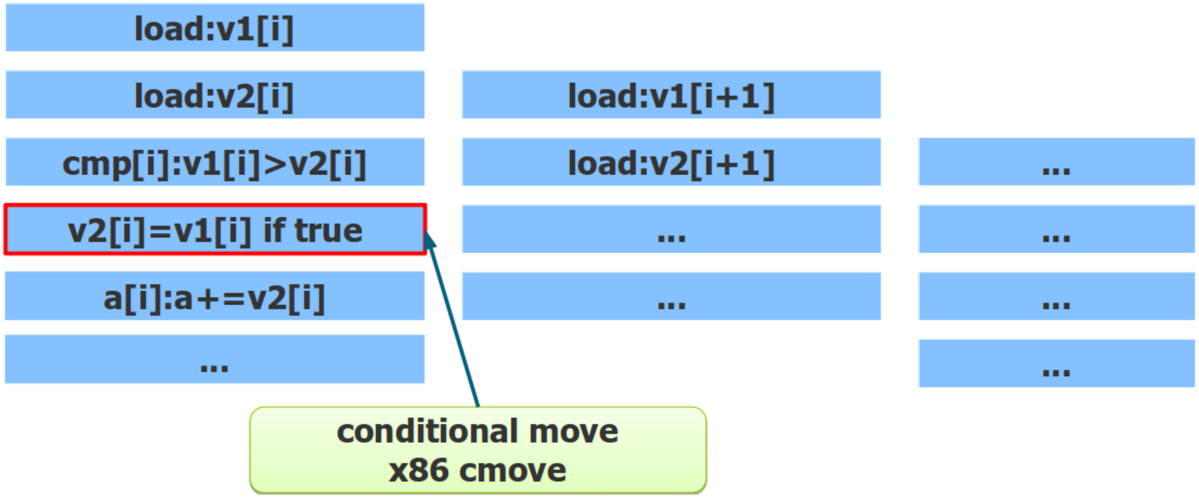


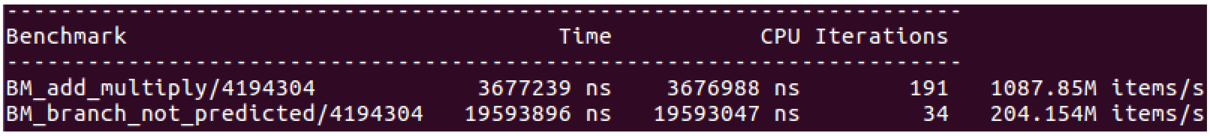
```
Timeline view:
                    0123456789          0123456789          012345
Index     0123456789          0123456789          0123456789

[0,0]     DeeeER    .    .    .    .    .    .    .    .    .    .   mov        rax, qword ptr [r15 + 8*rcx]
[0,1]     D=eeeER   .    .    .    .    .    .    .    .    .    .   mov        rdx, qword ptr [rbx + 8*rcx]
[0,2]     .D===eER  .    .    .    .    .    .    .    .    .    .   lea        rsi, [rdx + rax]
[0,3]     .D===eER  .    .    .    .    .    .    .    .    .    .   sub        rax, rdx
[0,4]     . D===eeeeeeER .    .    .    .    .    .    .    .    .   imul       rax, rsi
[0,5]     . D=========eeeeeeER.    .    .    .    .    .    .    .   add        qword ptr [rsp + 8], rax
[1,0]     .  DeeeE-----------R.    .    .    .    .    .    .    .   mov        rax, qword ptr [r15 + 8*rcx]
[1,1]     .  D=eeeE-----------R    .    .    .    .    .    .    .   mov        rdx, qword ptr [rbx + 8*rcx]
[1,2]     .   D===eE----------R    .    .    .    .    .    .    .   lea        rsi, [rdx + rax]
[1,3]     .   D===eE-----------R   .    .    .    .    .    .    .   sub        rax, rdx
[1,4]     .    D====eeeeeeE----R   .    .    .    .    .    .    .   imul       rax, rsi
[1,5]     .    D==========eeeeeeER .    .    .    .    .    .    .   add        qword ptr [rsp + 8], rax
   ...
[9,0]     .    .    .    .    .    . D=eeeE------------------R   .   mov        rax, qword ptr [r15 + 8*rcx]
[9,1]     .    .    .    .    .    . D==eeeE------------------R  .   mov        rdx, qword ptr [rbx + 8*rcx]
[9,2]     .    .    .    .    .    .  D====eE-----------------R  .   lea        rsi, [rdx + rax]
[9,3]     .    .    .    .    .    .  D====eE------------------R .   sub        rax, rdx
[9,4]     .    .    .    .    .    .   D============eeeeeeE----R .   imul       rax, rsi
[9,5]     .    .    .    .    .    .   D==================eeeeeeER   add        qword ptr [rsp + 8], rax
```

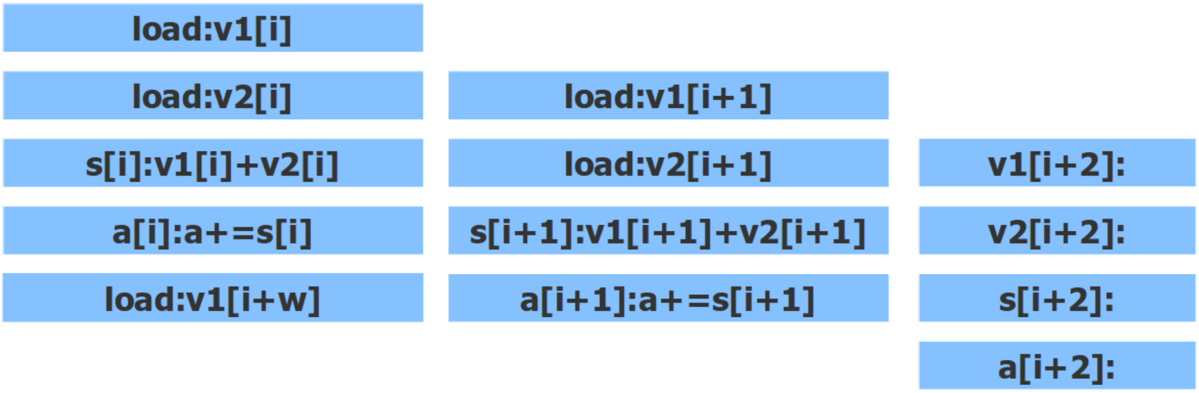


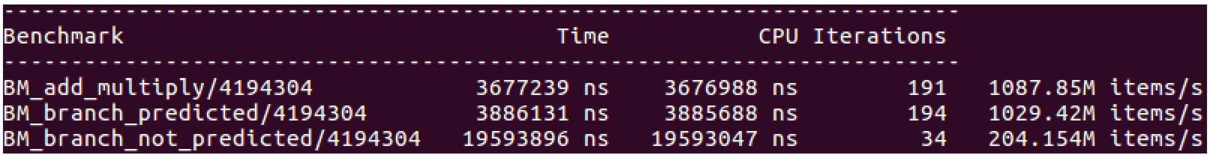
```
Timeline view:
                    0123456789          0123456789          01234
Index     0123456789          0123456789          0123456789

[0,0]     DeeeER    .    .    .    .    .    .    .    .    .   .   mov rax, qword ptr [rbx + 8*rcx]
[0,1]     D=eeeeeeeeeER  .    .    .    .    .    .    .    .   .   imul        rax, qword ptr [r15 + 8*rcx]
[0,2]     .D=========eeeeeeER .    .    .    .    .    .    .   .   add qword ptr [rsp + 8], rax
   ...
[9,0]     .    .    .    .  D=eeeE--------------------------R   .   mov rax, qword ptr [rbx + 8*rcx]
[9,1]     .    .    .    .   D==================eeeeeeeeeE---R  .   imul        rax, qword ptr [r15 + 8*rcx]
[9,2]     .    .    .    .    D==========================eeeeeeER   add qword ptr [rsp + 8], rax
```

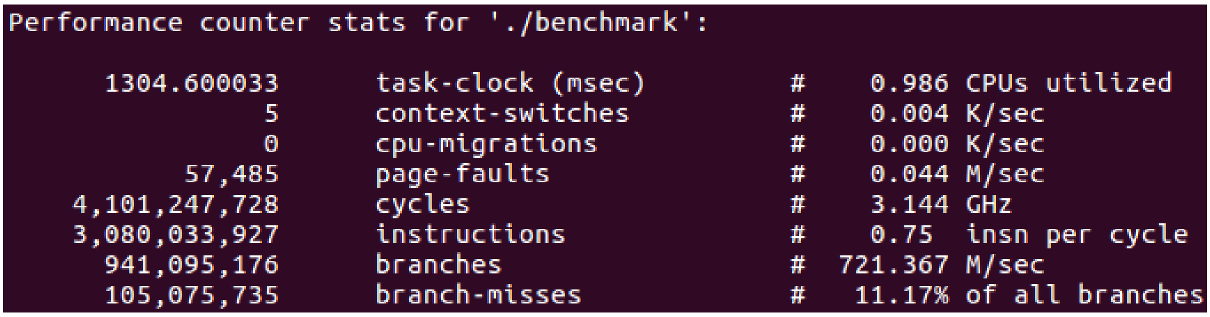


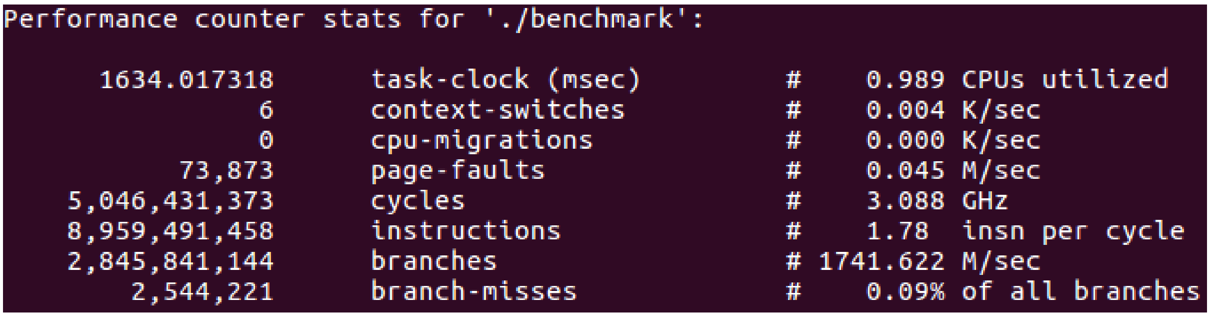



```
---------------------------------------------------------------------------
Benchmark                                     Time             CPU Iterations
---------------------------------------------------------------------------
BM_add_multiply/4194304                 3677239 ns      3676988 ns        191    1087.85M items/s
BM_branch_not_predicted/4194304        19593896 ns     19593047 ns         34    204.154M items/s
```

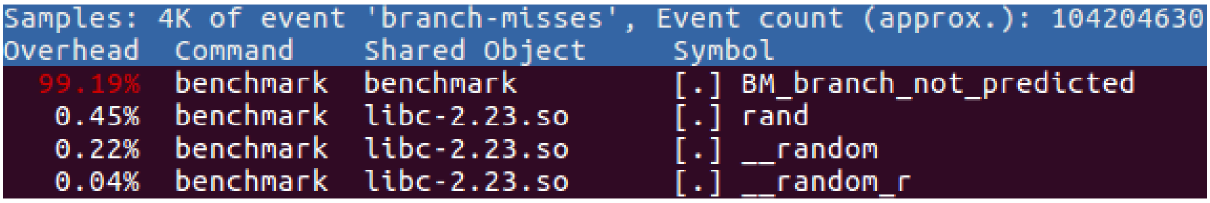



```
---------------------------------------------------------------------------
Benchmark                                     Time             CPU Iterations
---------------------------------------------------------------------------
BM_add_multiply/4194304                 3677239 ns      3676988 ns        191    1087.85M items/s
BM_branch_predicted/4194304             3886131 ns      3885688 ns        194    1029.42M items/s
BM_branch_not_predicted/4194304        19593896 ns     19593047 ns         34    204.154M items/s
```

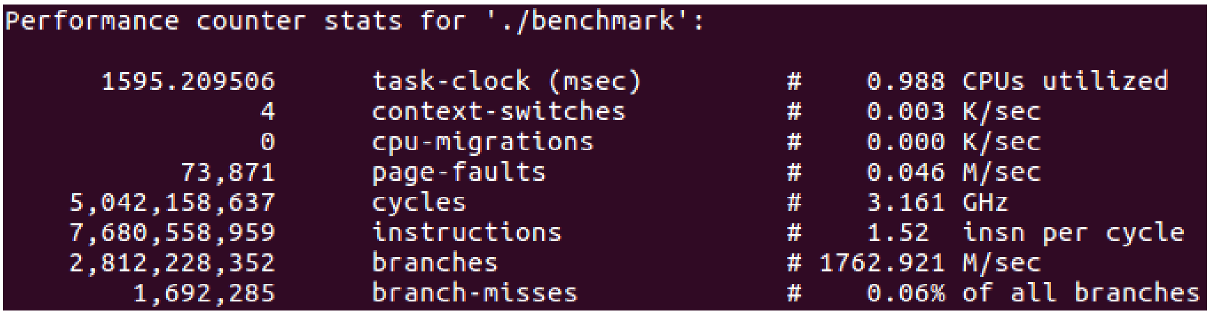

```
Performance counter stats for './benchmark':

       1304.600033      task-clock (msec)         #    0.986 CPUs utilized
                 5      context-switches          #    0.004 K/sec
                 0      cpu-migrations            #    0.000 K/sec
            57,485      page-faults               #    0.044 M/sec
     4,101,247,728      cycles                    #    3.144 GHz
     3,080,033,927      instructions              #    0.75  insn per cycle
       941,095,176      branches                  #  721.367 M/sec
       105,075,735      branch-misses             #   11.17% of all branches
```



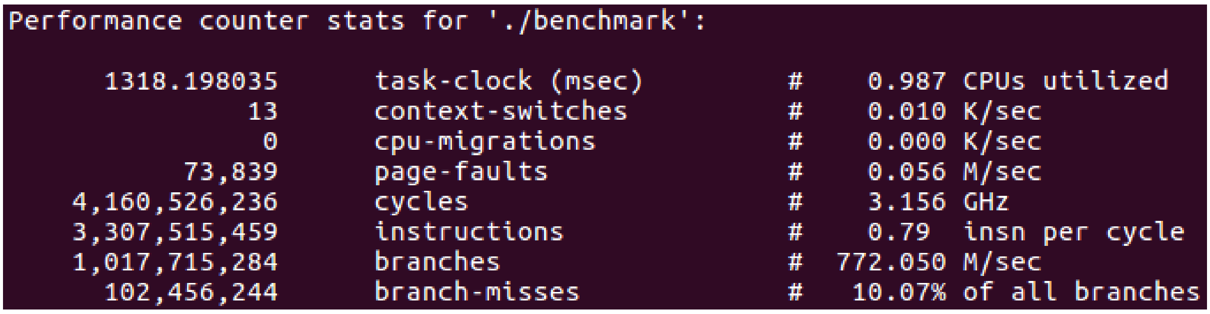
```
Performance counter stats for './benchmark':

      1634.017318      task-clock (msec)         #    0.989 CPUs utilized
                6      context-switches          #    0.004 K/sec
                0      cpu-migrations            #    0.000 K/sec
           73,873      page-faults               #    0.045 M/sec
    5,046,431,373      cycles                    #    3.088 GHz
    8,959,491,458      instructions              #    1.78  insn per cycle
    2,845,841,144      branches                  # 1741.622 M/sec
        2,544,221      branch-misses             #    0.09% of all branches
```



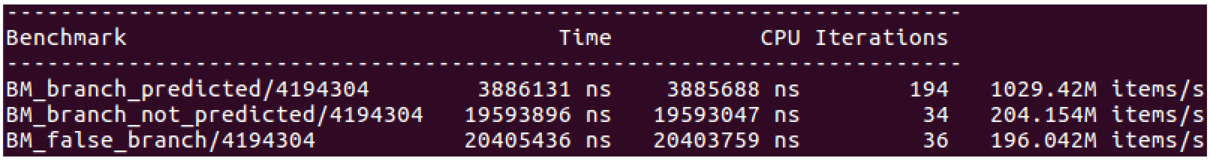
```
Samples: 4K of event 'branch-misses', Event count (approx.): 104204630
Overhead  Command    Shared Object      Symbol
  99.19%  benchmark  benchmark          [.] BM_branch_not_predicted
   0.45%  benchmark  libc-2.23.so       [.] rand
   0.22%  benchmark  libc-2.23.so       [.] __random
   0.04%  benchmark  libc-2.23.so       [.] __random_r
```



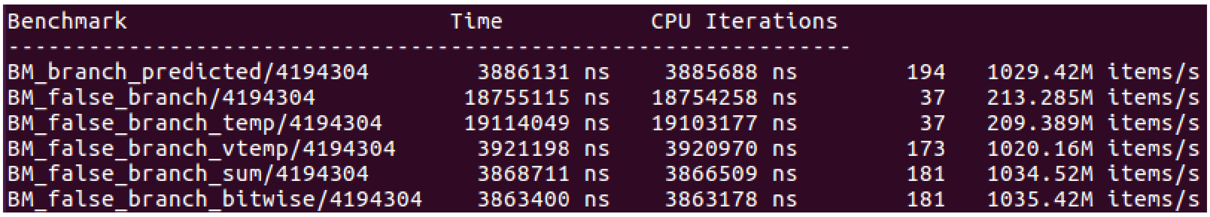
```
Performance counter stats for './benchmark':

      1595.209506      task-clock (msec)         #    0.988 CPUs utilized
                4      context-switches          #    0.003 K/sec
                0      cpu-migrations            #    0.000 K/sec
           73,871      page-faults               #    0.046 M/sec
    5,042,158,637      cycles                    #    3.161 GHz
    7,680,558,959      instructions              #    1.52  insn per cycle
    2,812,228,352      branches                  # 1762.921 M/sec
        1,692,285      branch-misses             #    0.06% of all branches
```



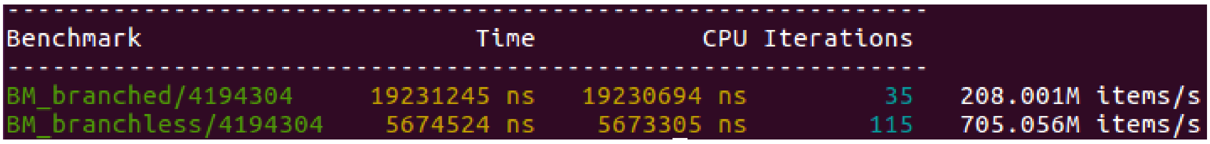
```
Performance counter stats for './benchmark':

      1318.198035      task-clock (msec)         #    0.987 CPUs utilized
               13      context-switches          #    0.010 K/sec
                0      cpu-migrations            #    0.000 K/sec
           73,839      page-faults               #    0.056 M/sec
    4,160,526,236      cycles                    #    3.156 GHz
    3,307,515,459      instructions              #    0.79  insn per cycle
    1,017,715,284      branches                  #  772.050 M/sec
      102,456,244      branch-misses             #   10.07% of all branches
```



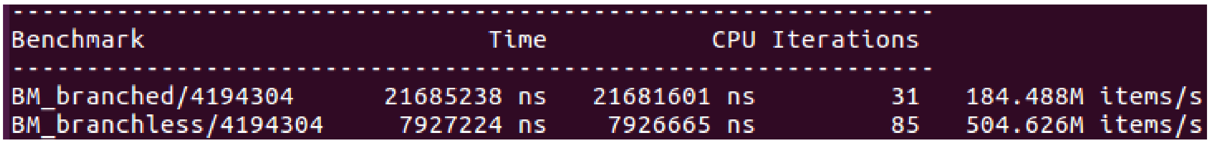
```
-----------------------------------------------------------------------
Benchmark                                    Time           CPU Iterations
-----------------------------------------------------------------------
BM_branch_predicted/4194304         3886131 ns    3885688 ns        194   1029.42M items/s
BM_branch_not_predicted/4194304    19593896 ns   19593047 ns         34   204.154M items/s
BM_false_branch/4194304            20405436 ns   20403759 ns         36   196.042M items/s
```



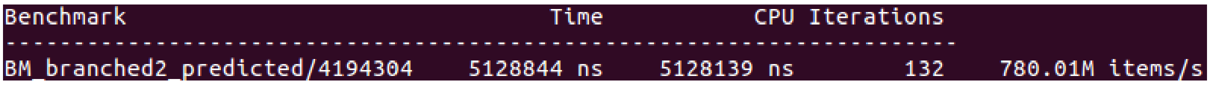
```
Benchmark                                    Time           CPU Iterations
-----------------------------------------------------------------------
BM_branch_predicted/4194304         3886131 ns    3885688 ns        194   1029.42M items/s
BM_false_branch/4194304            18755115 ns   18754258 ns         37   213.285M items/s
BM_false_branch_temp/4194304       19114049 ns   19103177 ns         37   209.389M items/s
BM_false_branch_vtemp/4194304       3921198 ns    3920970 ns        173   1020.16M items/s
BM_false_branch_sum/4194304         3868711 ns    3866509 ns        181   1034.52M items/s
BM_false_branch_bitwise/4194304     3863400 ns    3863178 ns        181   1035.42M items/s
```



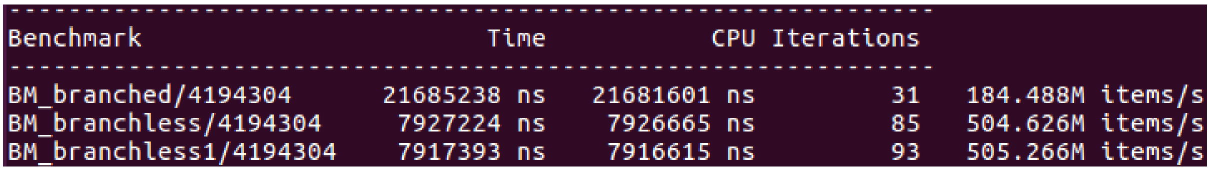
```
--------------------------------------------------------------
Benchmark                     Time           CPU Iterations
--------------------------------------------------------------
BM_branched/4194304    19231245 ns   19230694 ns         35   208.001M items/s
BM_branchless/4194304   5674524 ns    5673305 ns        115   705.056M items/s
```



```
--------------------------------------------------------------
Benchmark                     Time           CPU Iterations
--------------------------------------------------------------
BM_branched/4194304    21685238 ns   21681601 ns         31   184.488M items/s
BM_branchless/4194304   7927224 ns    7926665 ns         85   504.626M items/s
```

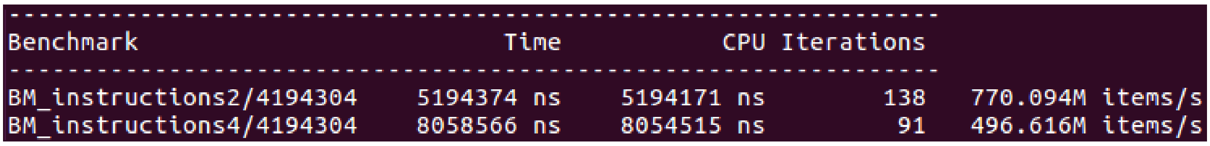


```
Benchmark                               Time           CPU Iterations
----------------------------------------------------------------------
BM_branched2_predicted/4194304    5128844 ns    5128139 ns        132    780.01M items/s
```

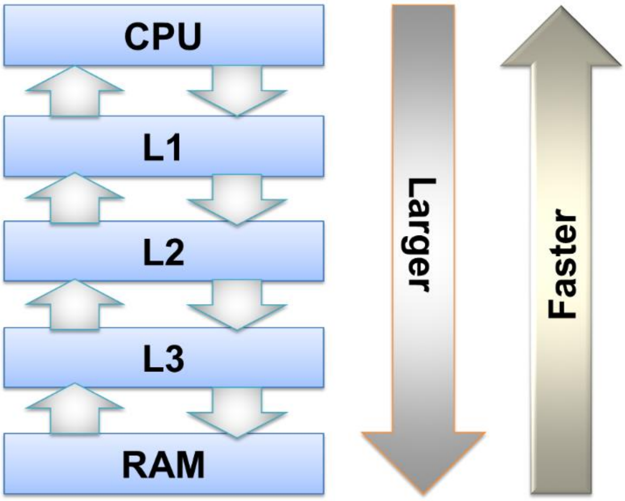


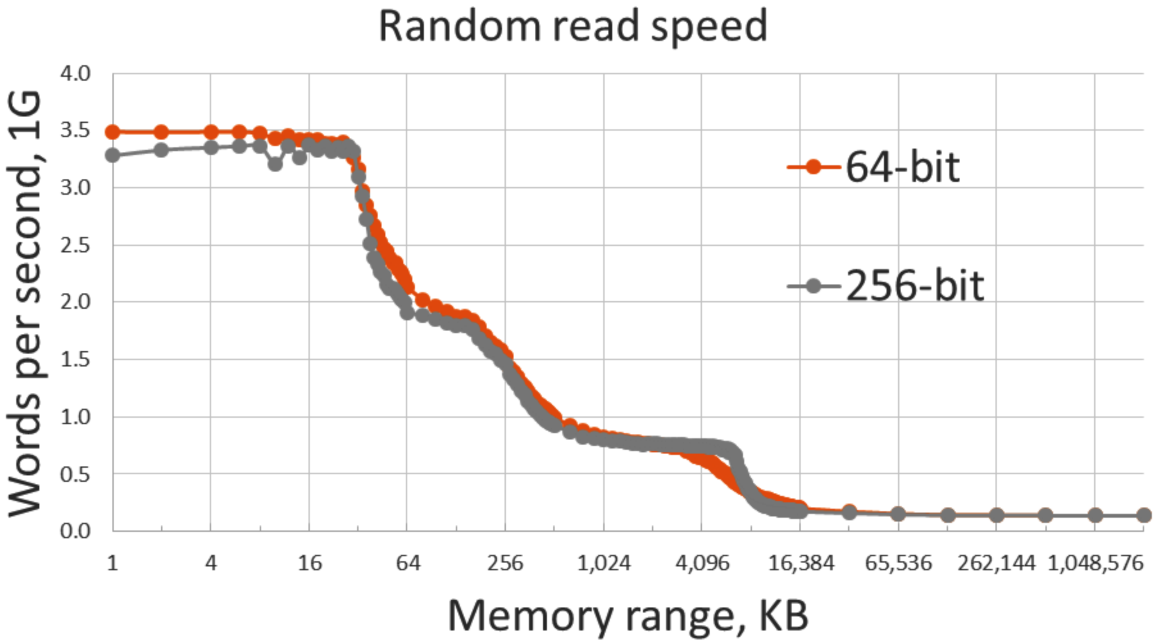
```
--------------------------------------------------------------
Benchmark                       Time           CPU Iterations
--------------------------------------------------------------
BM_branched/4194304      21685238 ns   21681601 ns         31   184.488M items/s
BM_branchless/4194304     7927224 ns    7926665 ns         85   504.626M items/s
BM_branchless1/4194304    7917393 ns    7916615 ns         93   505.266M items/s
```

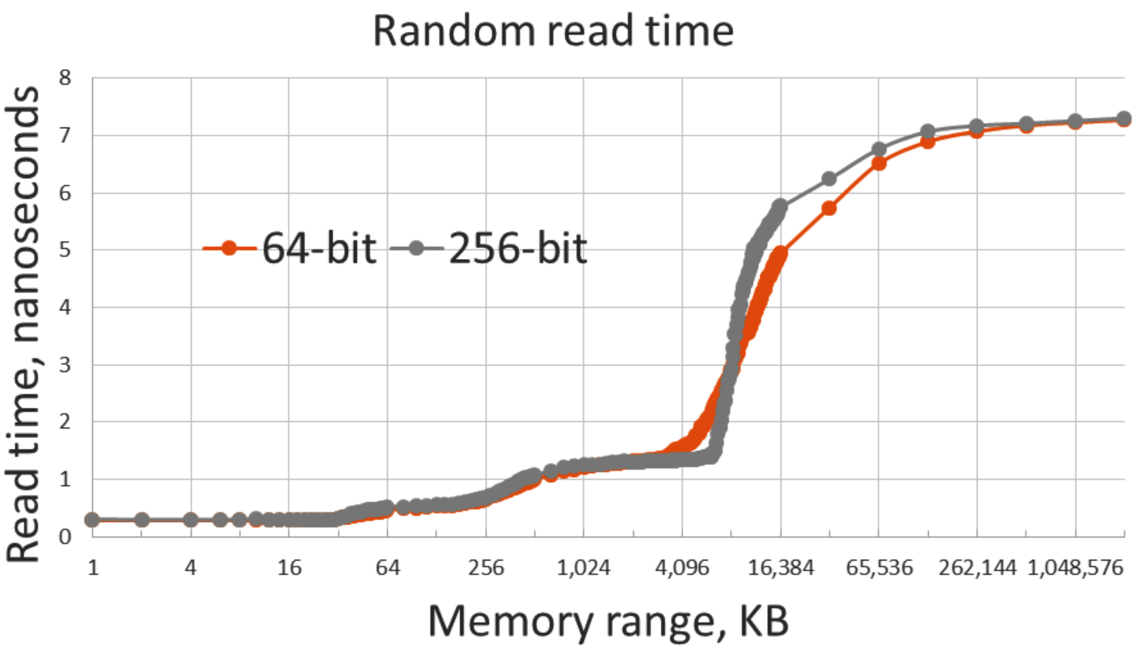
Chapter 4: Memory Architecture and Performance

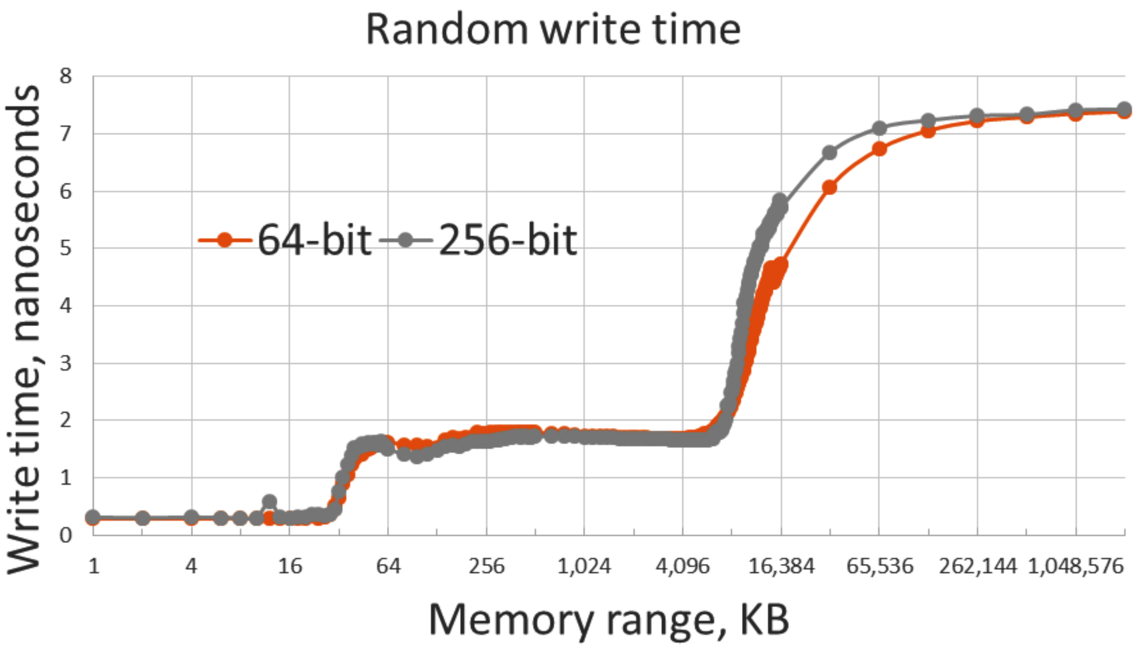


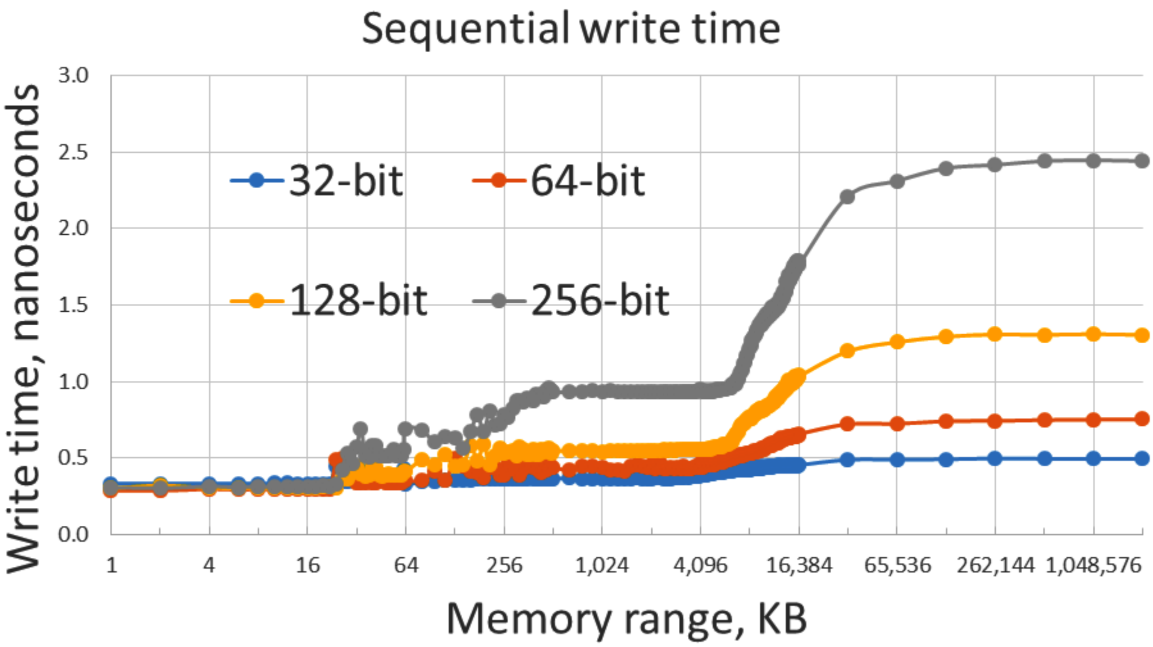
```
--------------------------------------------------------------
Benchmark                           Time           CPU Iterations
--------------------------------------------------------------
BM_instructions2/4194304    5194374 ns    5194171 ns        138    770.094M items/s
BM_instructions4/4194304    8058566 ns    8054515 ns         91    496.616M items/s
```

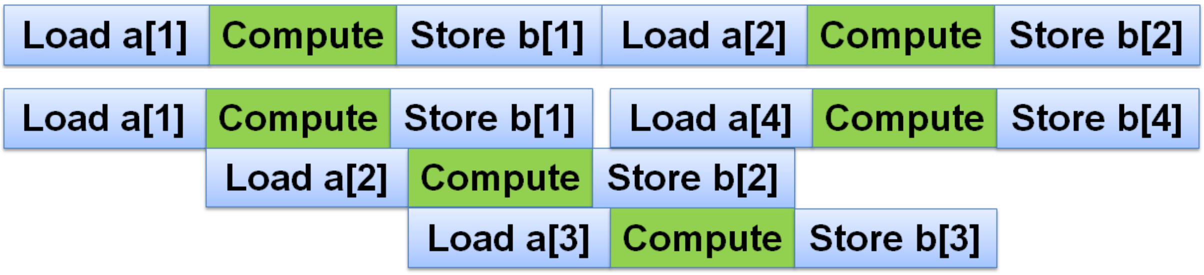





Random read time
Read time, nanoseconds
0
1
2
3
4
5
6
7
8
64-bit
256-bit
1
4
16
64
256
1,024
4,096
16,384
65,536
262,144
1,048,576
Memory range, KB


Random write time
Write time, nanoseconds
0
1
2
3
4
5
6
7
8
64-bit
256-bit
1
4
16
64
256
1,024
4,096
16,384
65,536
262,144
1,048,576
Memory range, KB



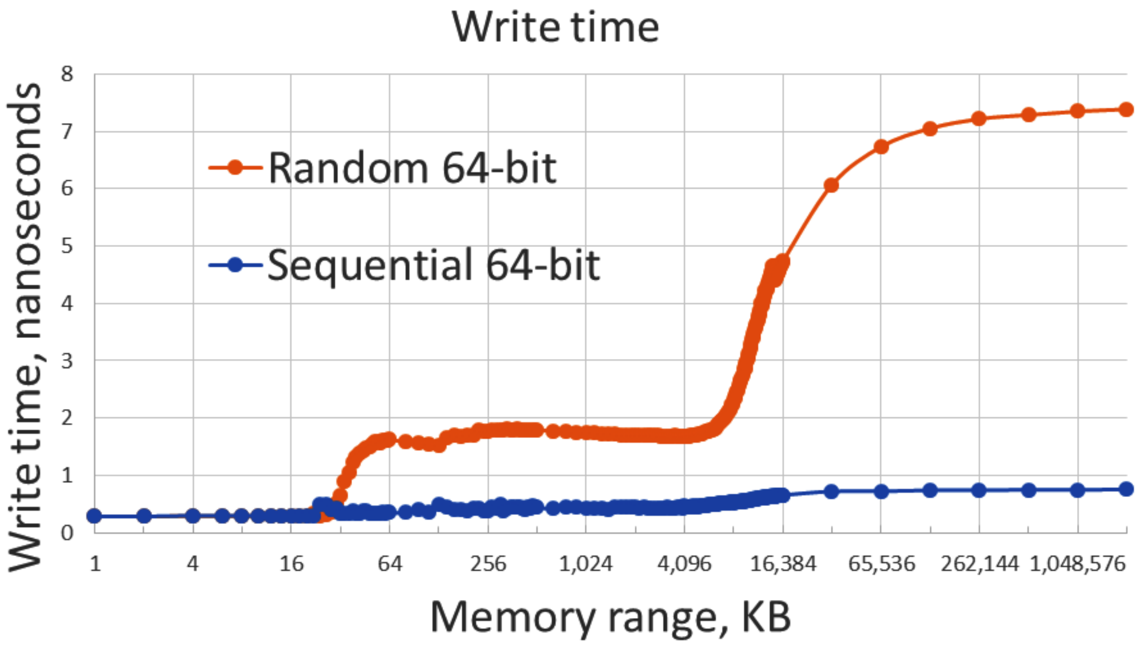


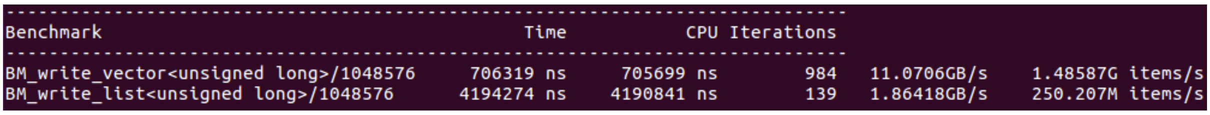
Load a[1] | Compute | Store b[1] | Load a[2] | Compute | Store b[2]

Load a[1] | Compute | Store b[1] | Load a[4] | Compute | Store b[4]

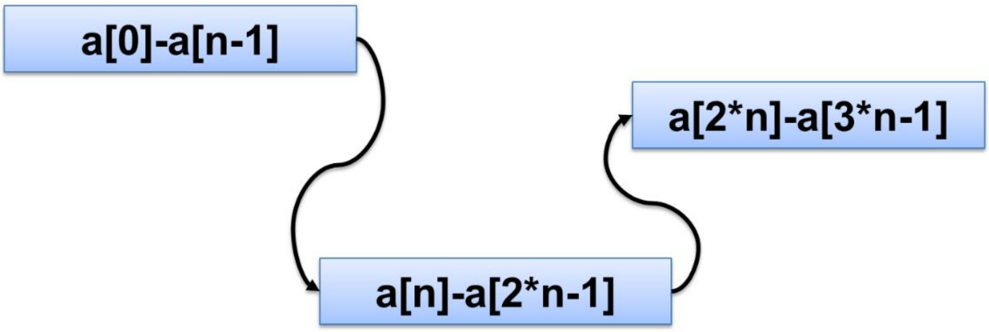
Load a[2] | Compute | Store b[2]

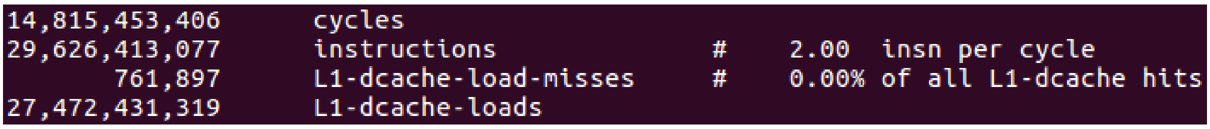
Load a[3] | Compute | Store b[3]



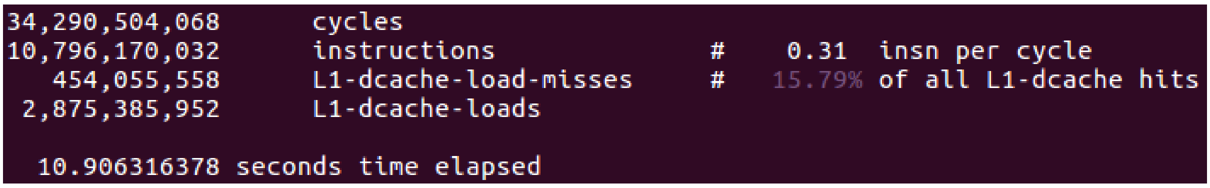


```
------------------------------------------------------------------------------
Benchmark                                       Time           CPU Iterations
------------------------------------------------------------------------------
BM_write_vector<unsigned long>/1048576     706319 ns     705699 ns        984   11.0706GB/s    1.48587G items/s
BM_write_list<unsigned long>/1048576      4194274 ns    4190841 ns        139   1.86418GB/s    250.207M items/s
```

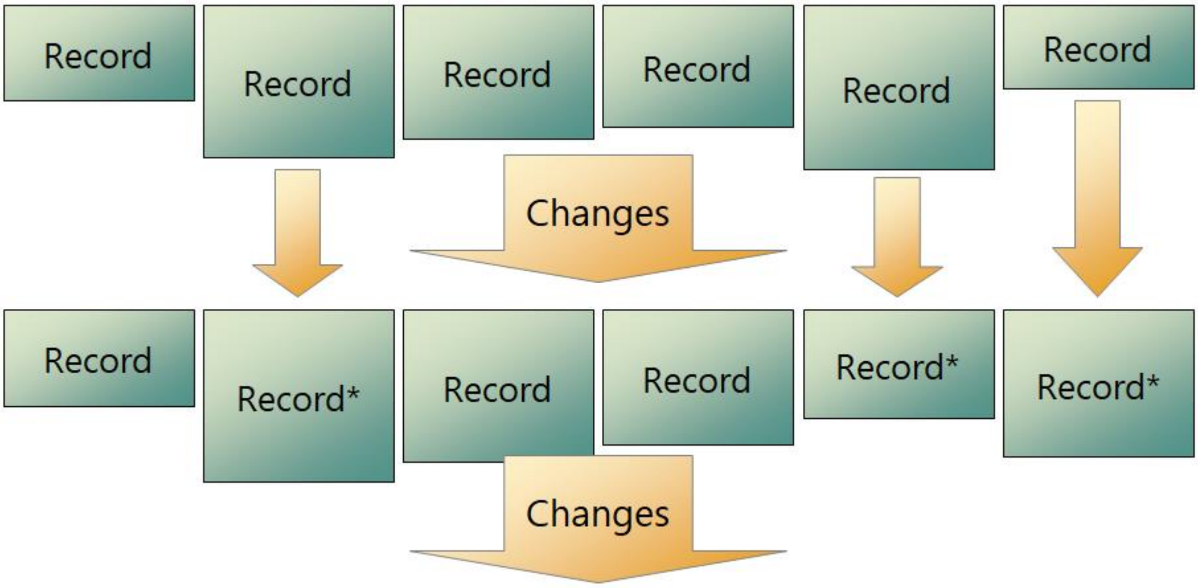


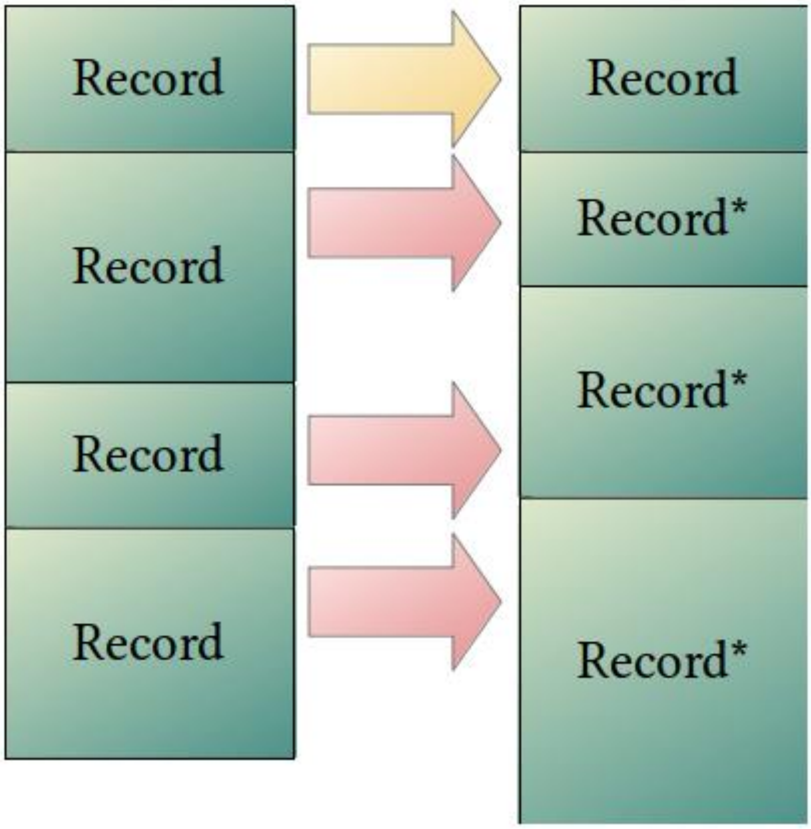


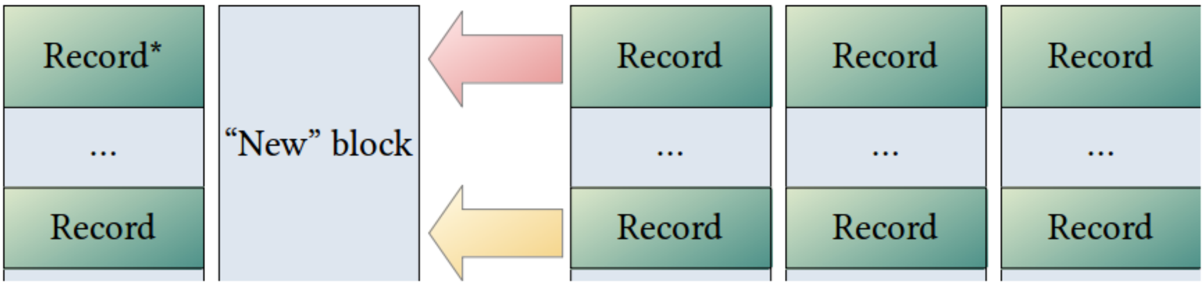
```
14,815,453,406      cycles
29,626,413,077      instructions              #    2.00  insn per cycle
       761,897      L1-dcache-load-misses     #    0.00% of all L1-dcache hits
27,472,431,319      L1-dcache-loads
```

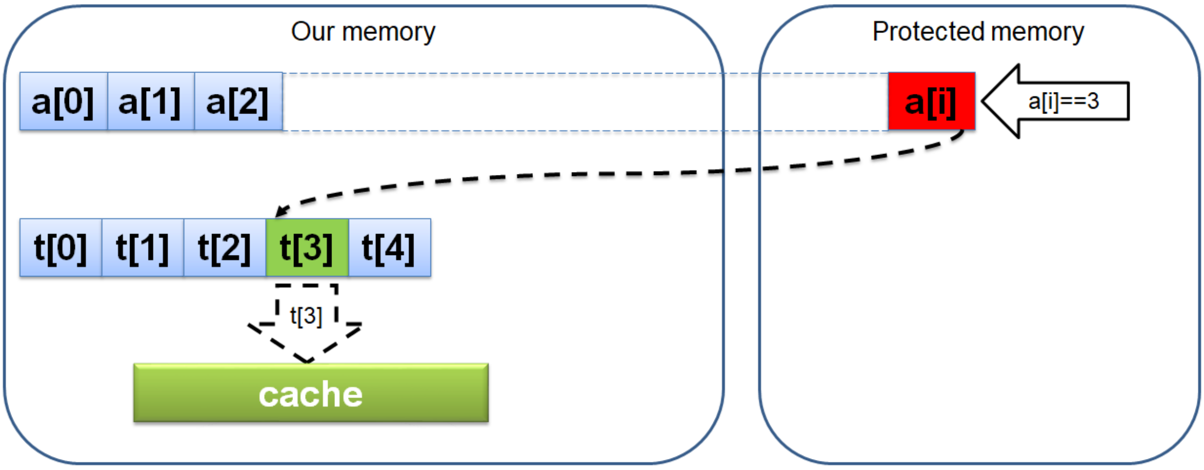


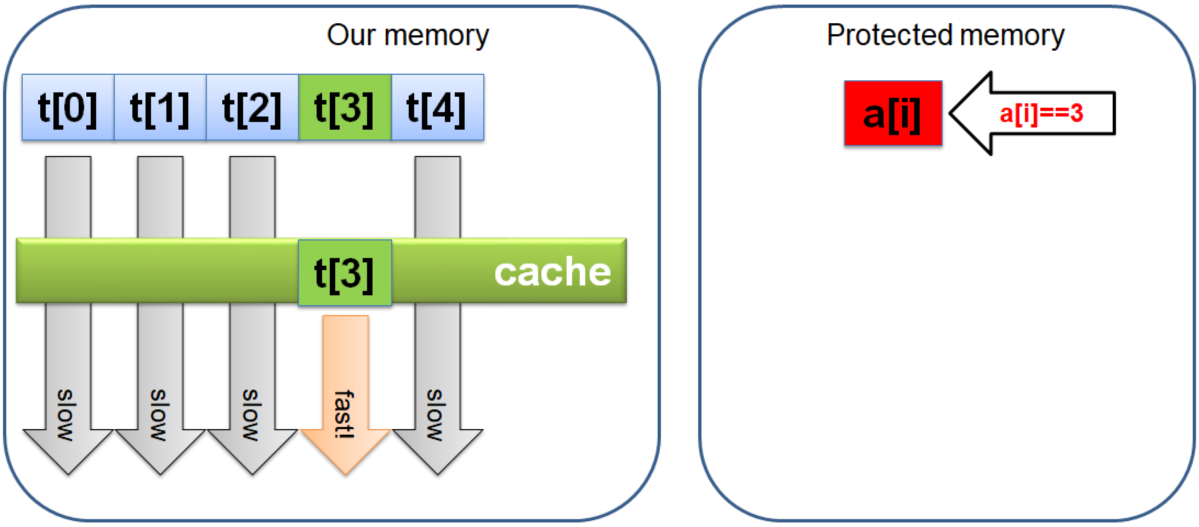
```
34,290,504,068      cycles
10,796,170,032      instructions              #    0.31  insn per cycle
   454,055,558      L1-dcache-load-misses     #   15.79% of all L1-dcache hits
 2,875,385,952      L1-dcache-loads

  10.906316378 seconds time elapsed
```



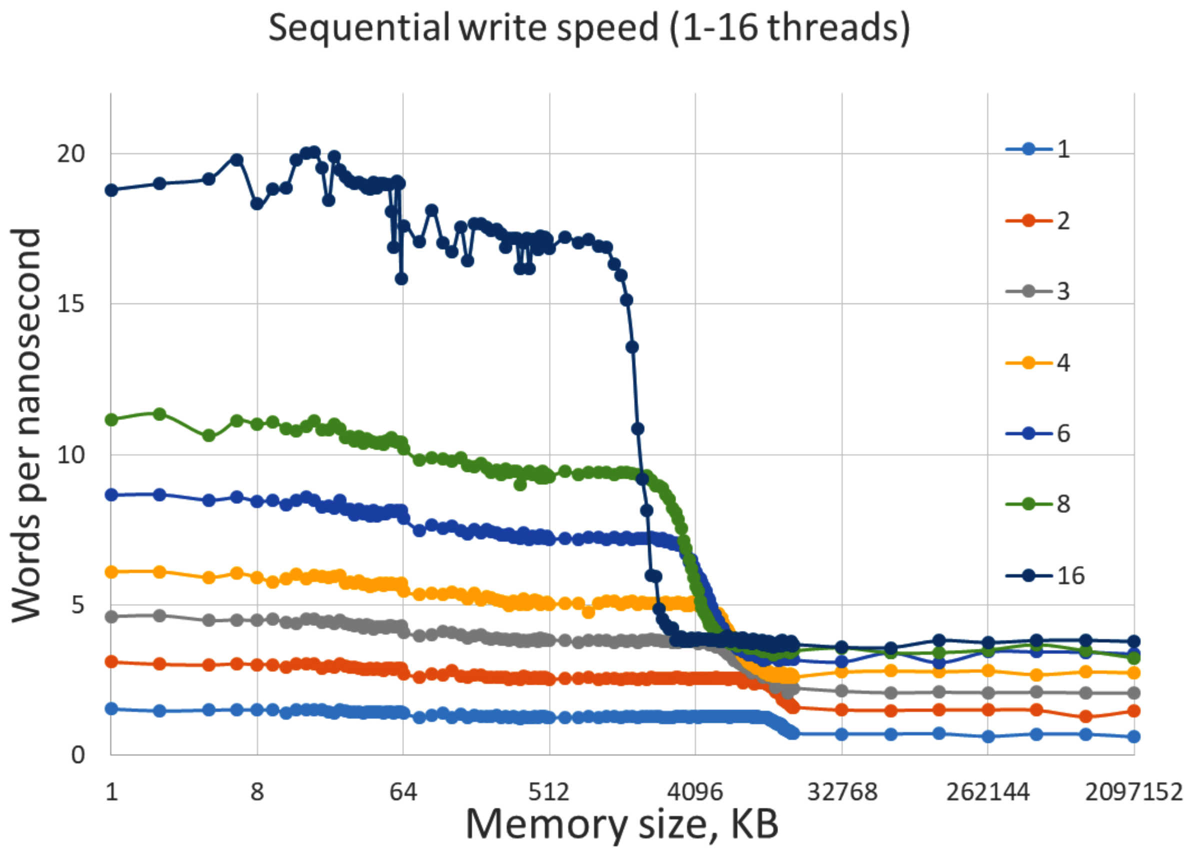

Record
Record
Record
Record
Record
Record*
Record*
Record*

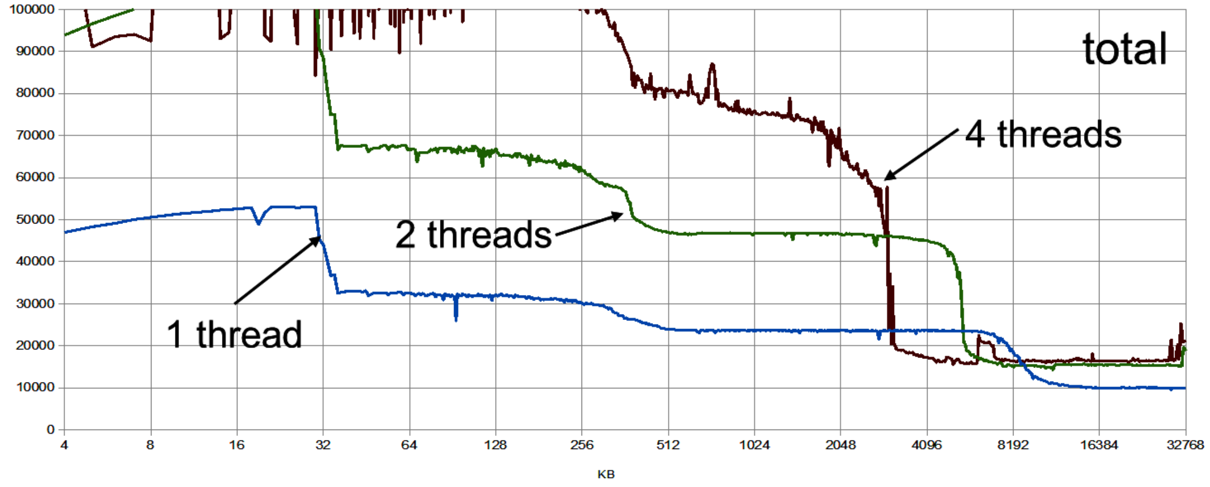

Record*
...
Record
"New" block
Record
...
Record
Record
...
Record
Record
...
Record

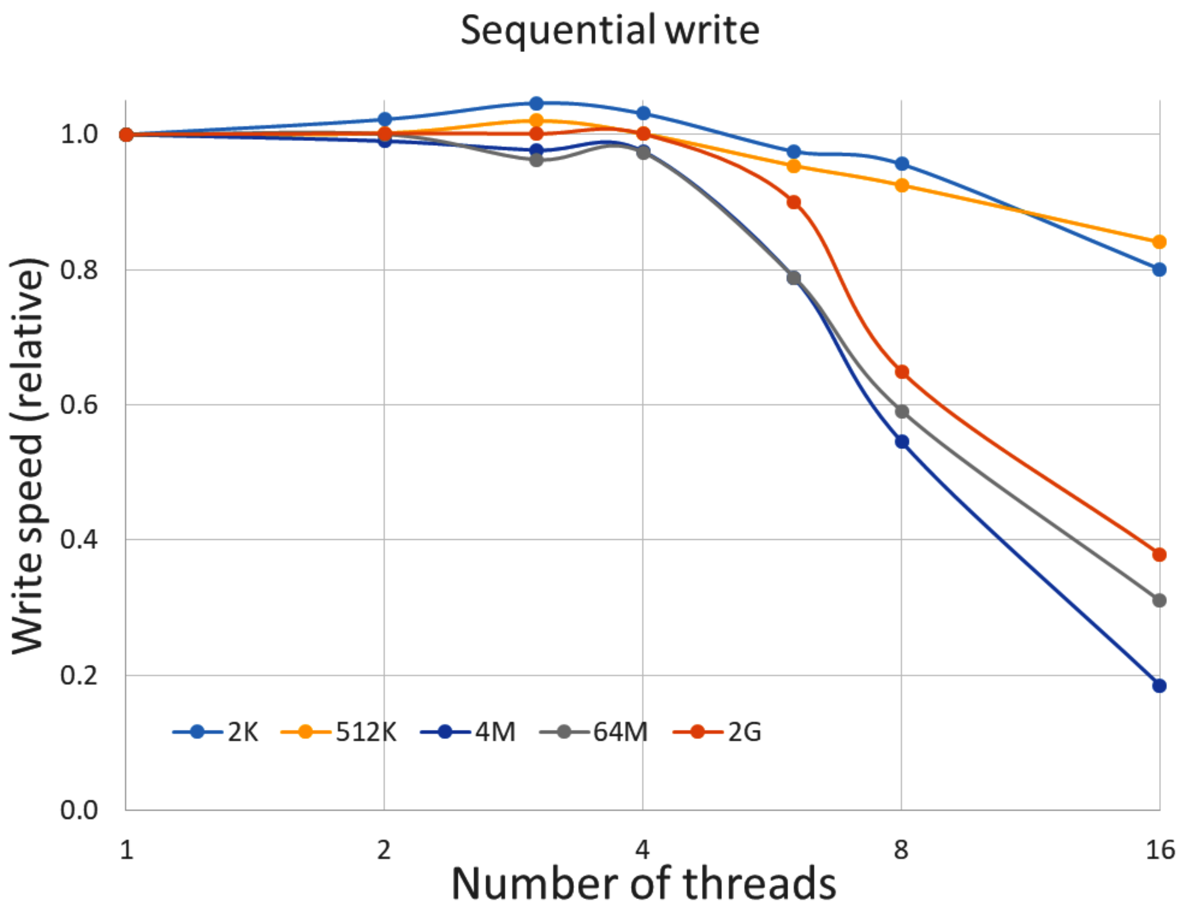

Our memory
Protected memory
a[0]
a[1]
a[2]
a[i]
a[i]==3
t[0]
t[1]
t[2]
t[3]
t[4]
t[3]
cache

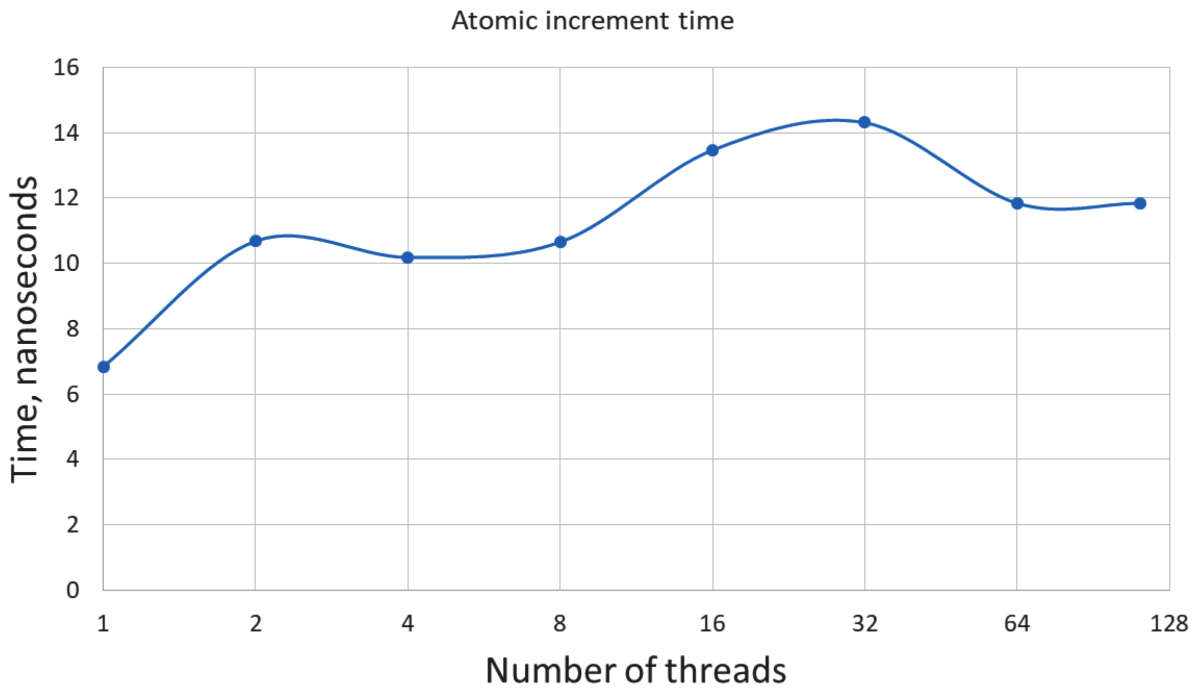

Our memory
Protected memory
t[0]
t[1]
t[2]
t[3]
t[4]
a[i]
a[i]==3
t[3]
cache
slow
slow
slow
fast!
slow

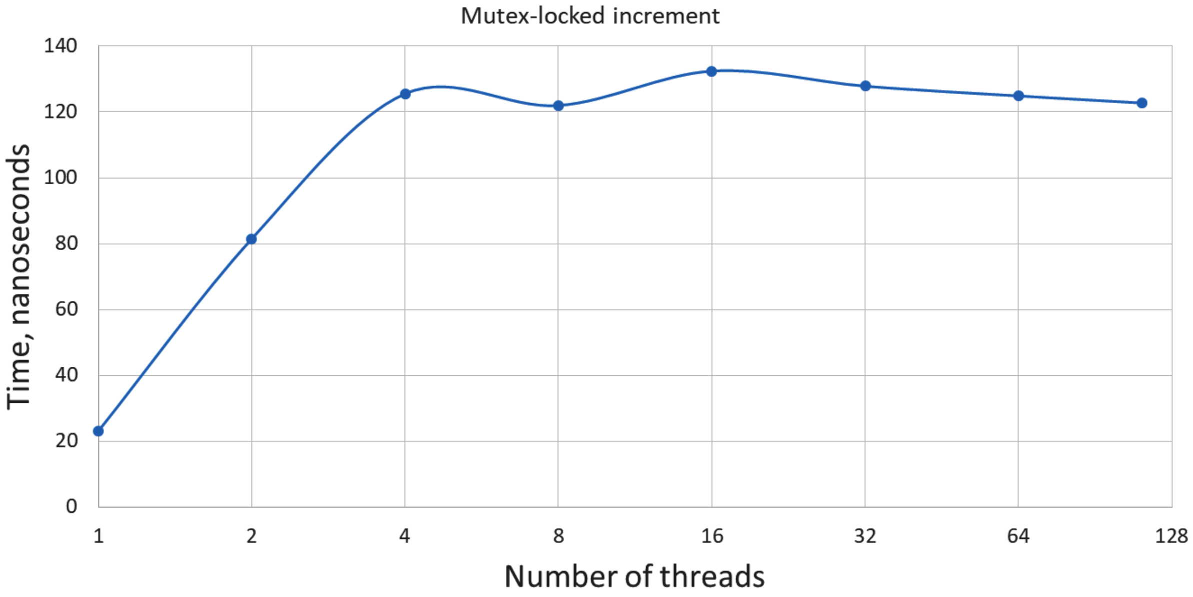
Chapter 5: Threads, Memory, and Concurrency

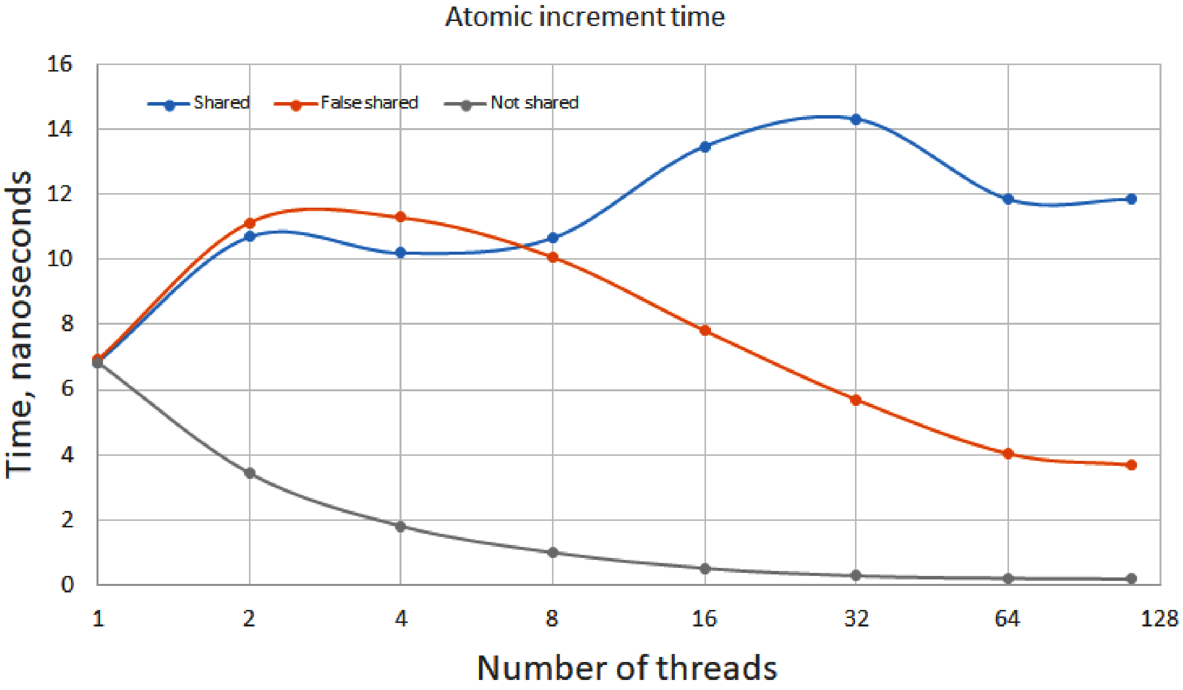


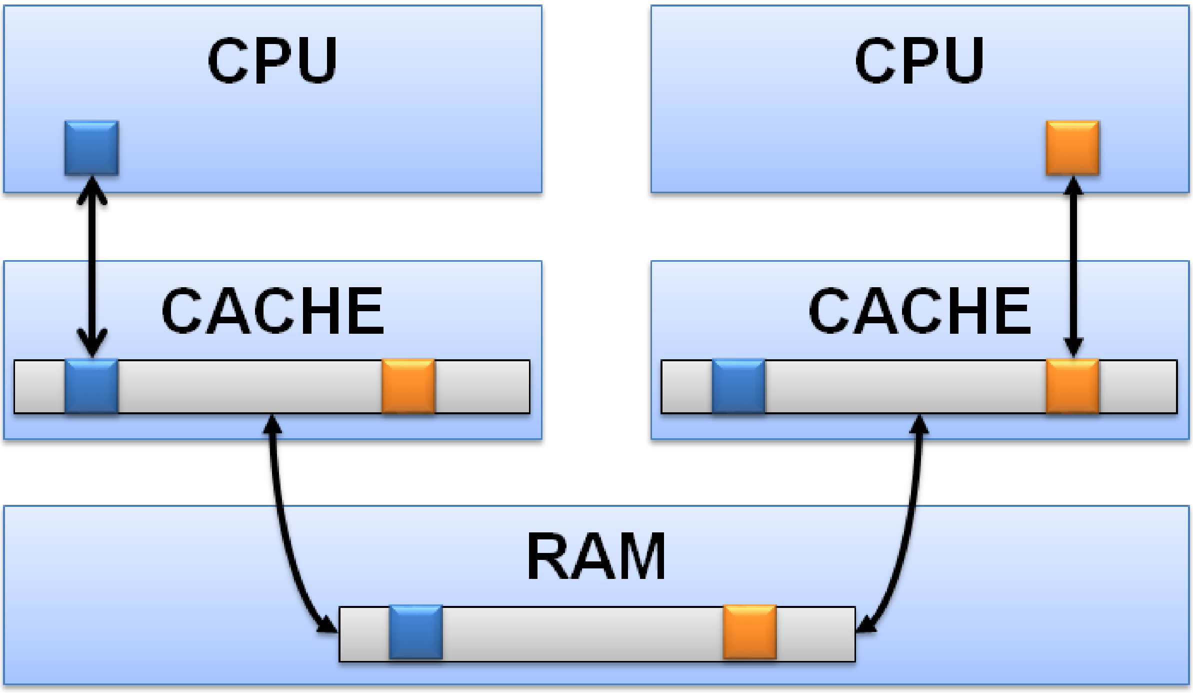


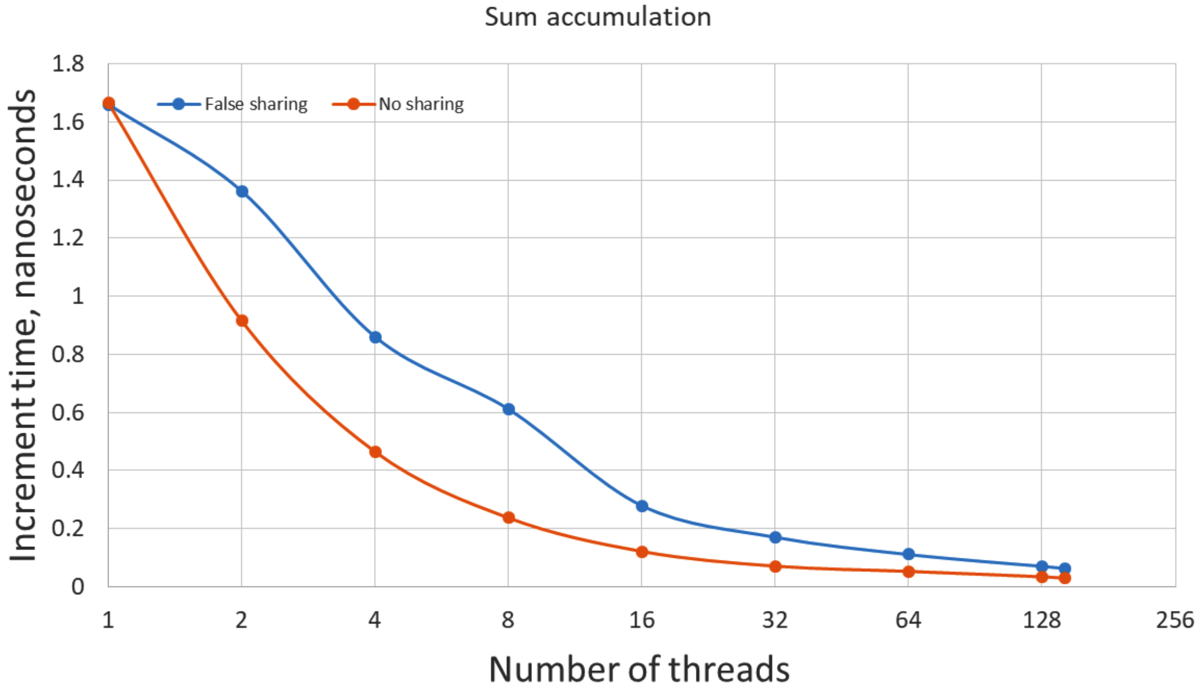

Sequential write
Write speed (relative)
1.0
0.8
0.6
0.4
0.2
0.0
2K
512K
4M
64M
2G
1
2
4
8
16
Number of threads

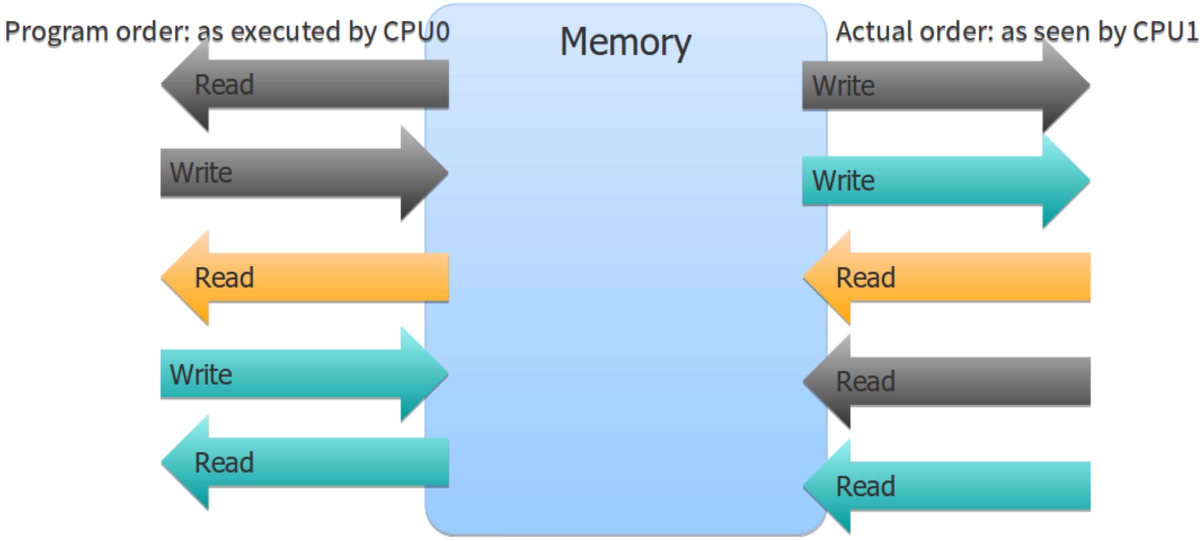

Atomic increment time
Time, nanoseconds
16
14
12
10
8
6
4
2
0
1
2
4
8
16
32
64
128
Number of threads

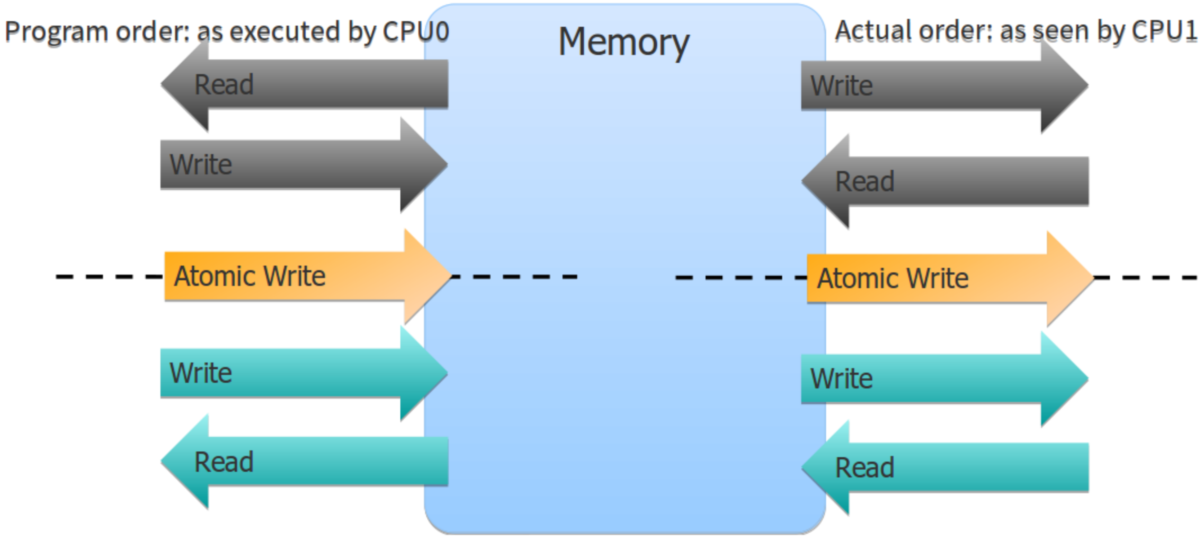

Mutex-locked increment
Time, nanoseconds
0
20
40
60
80
100
120
140
1
2
4
8
16
32
64
128
Number of threads

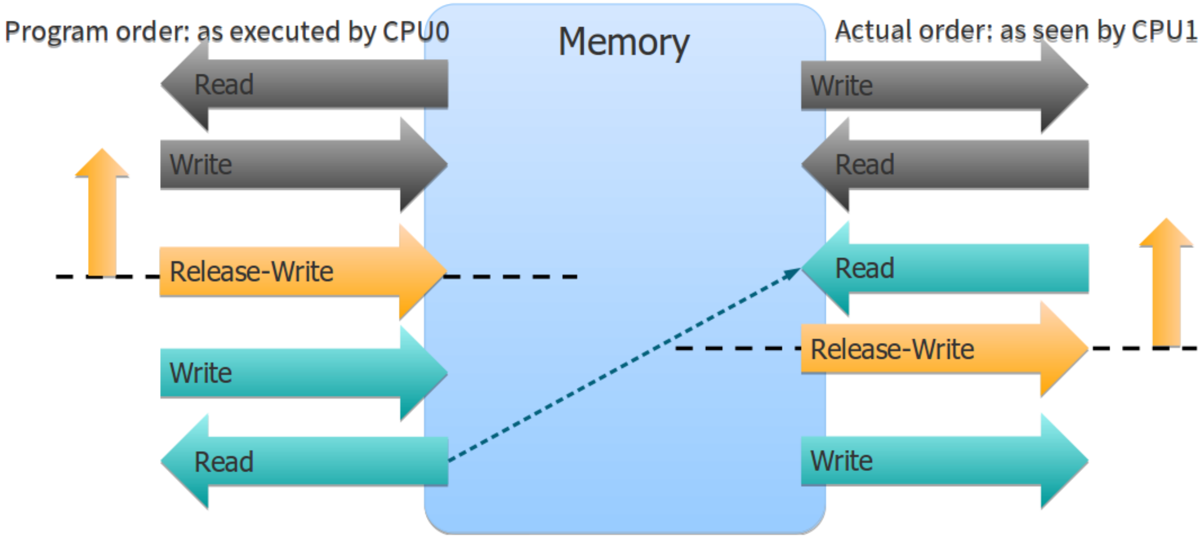

Atomic increment time
Shared
False shared
Not shared
Time, nanoseconds
0
2
4
6
8
10
12
14
16
1
2
4
8
16
32
64
128
Number of threads

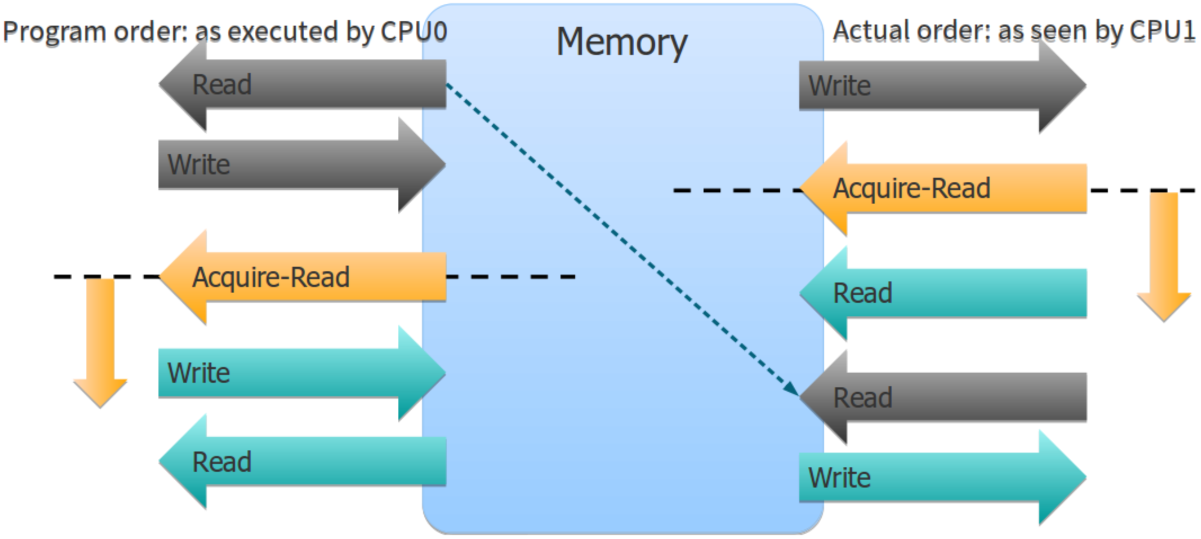

CPU
CPU
CACHE
CACHE
RAM

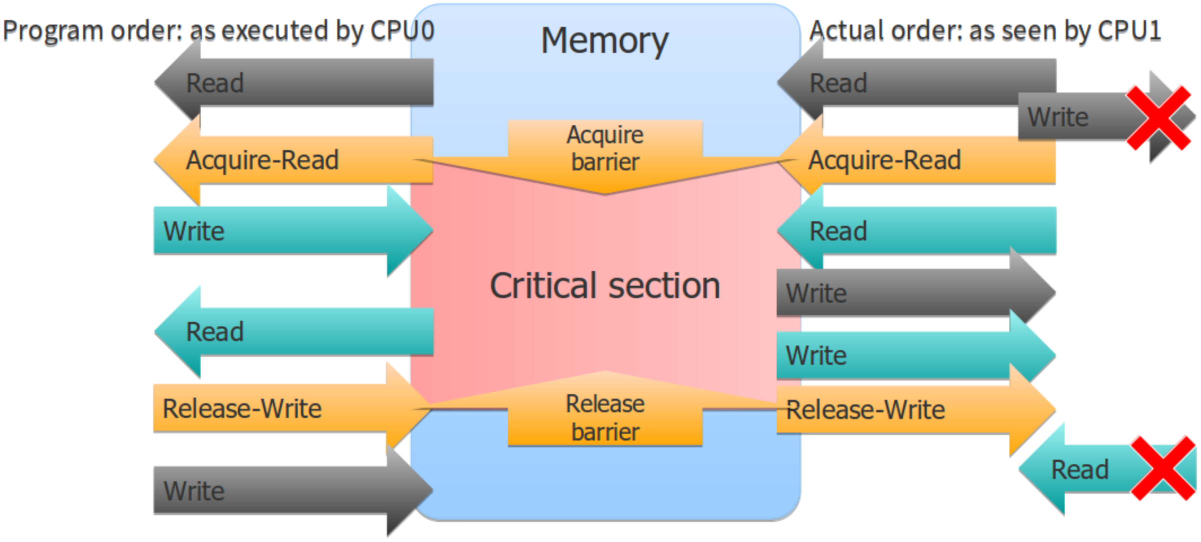

Sum accumulation
False sharing
No sharing
Increment time, nanoseconds
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
4
8
16
32
64
128
256
Number of threads

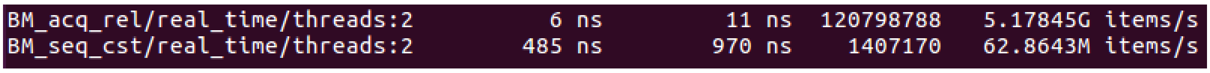

Program order: as executed by CPU0
Memory
Actual order: as seen by CPU1
Read
Write
Read
Write
Read
Write
Write
Read
Read
Read


Program order: as executed by CPU0
Memory
Actual order: as seen by CPU1
Read
Write
Atomic Write
Write
Read
Write
Read
Atomic Write
Write
Read


Program order: as executed by CPU0
Memory
Actual order: as seen by CPU1
Read
Write
Release-Write
Write
Read
Write
Read
Read
Release-Write
Write

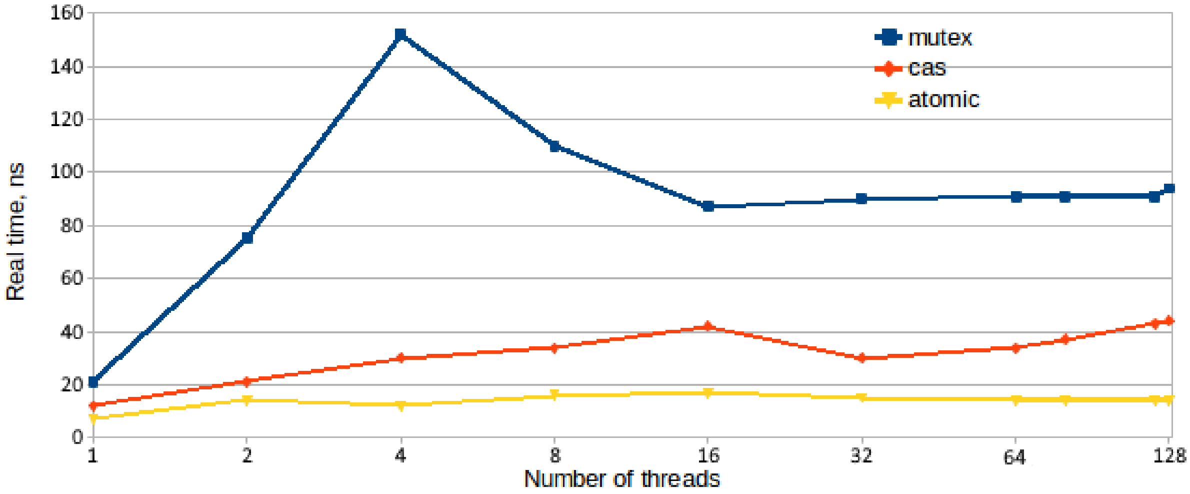


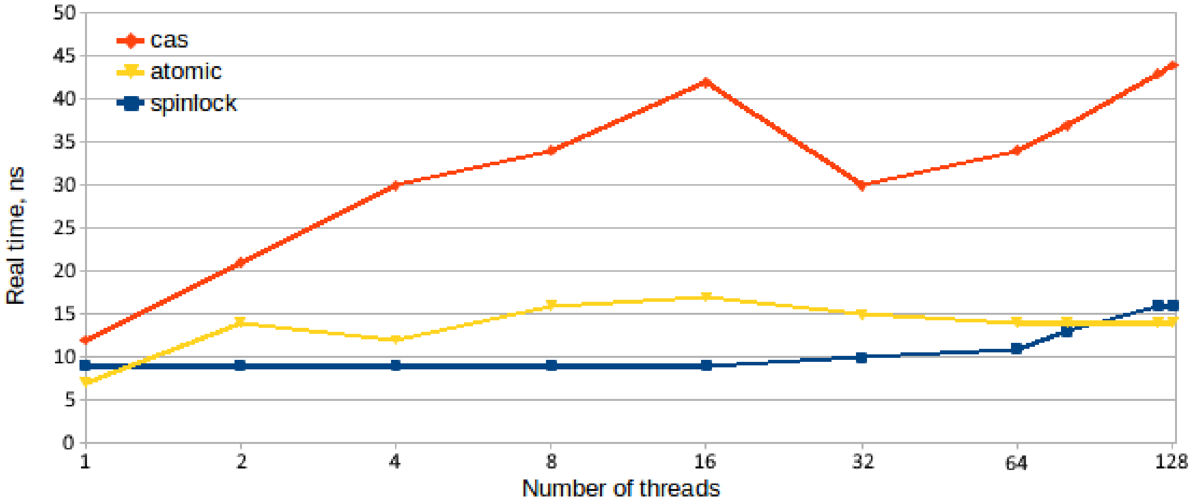


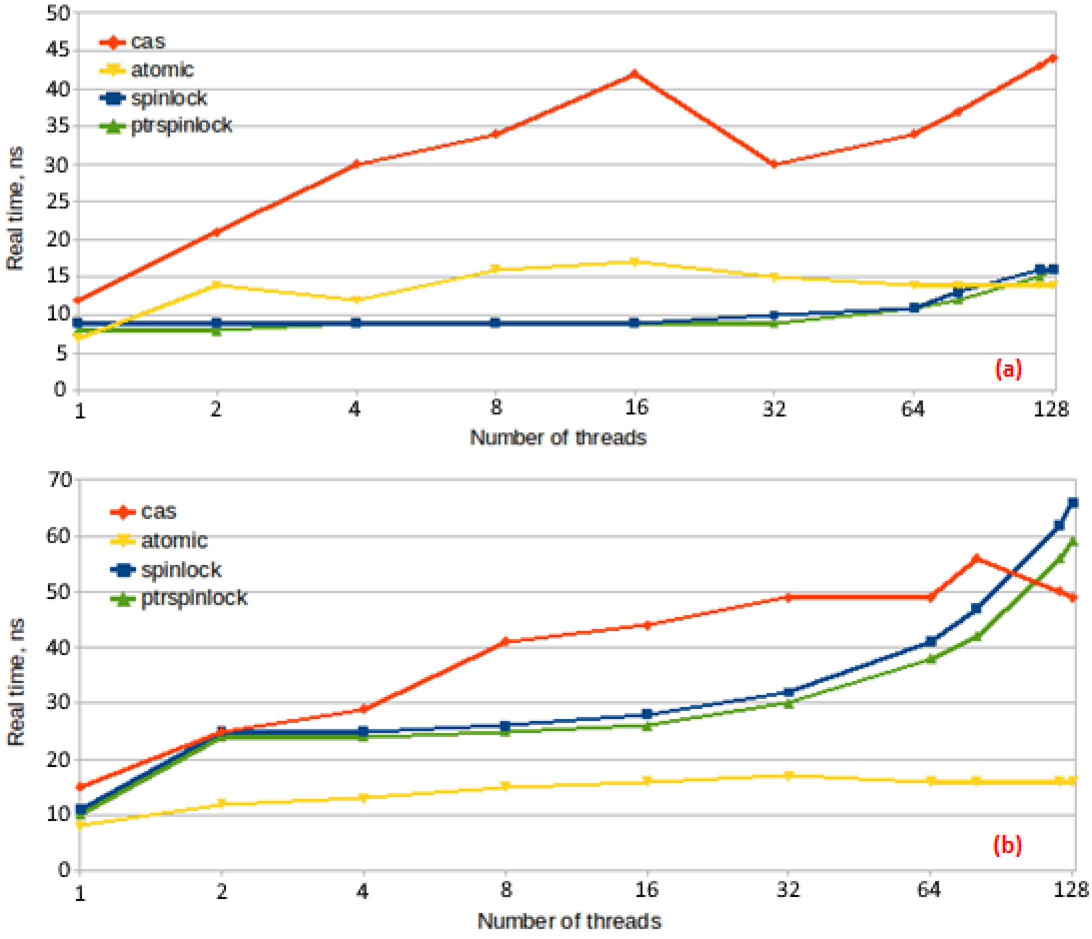


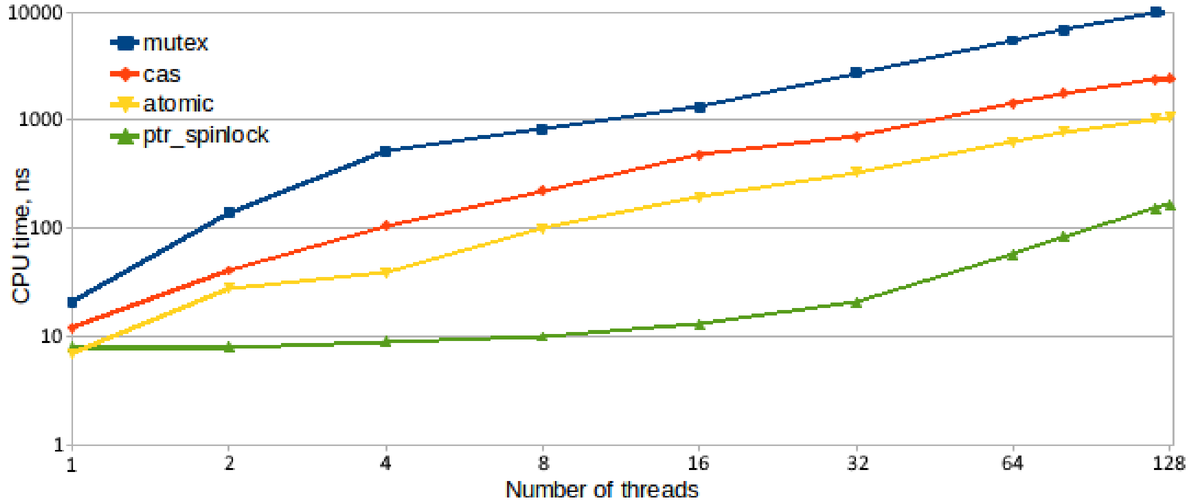
```
BM_acq_rel/real_time/threads:2           6 ns           11 ns  120798788    5.17845G items/s
BM_seq_cst/real_time/threads:2         485 ns          970 ns    1407170    62.8643M items/s
```

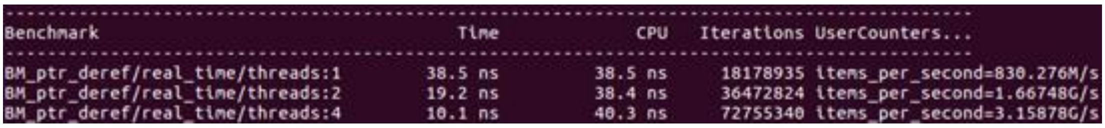
Chapter 6: Concurrency and Performance



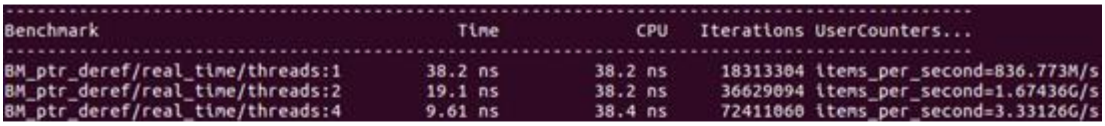




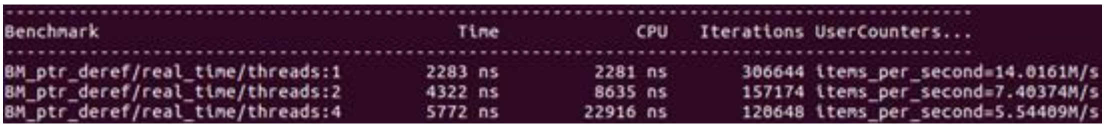




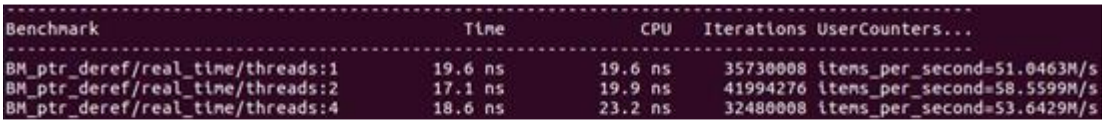
```
---------------------------------------------------------------------------------------------
Benchmark                                   Time             CPU   Iterations UserCounters...
---------------------------------------------------------------------------------------------
BM_ptr_deref/real_time/threads:1         38.5 ns         38.5 ns     18178935 items_per_second=830.276M/s
BM_ptr_deref/real_time/threads:2         19.2 ns         38.4 ns     36472824 items_per_second=1.66748G/s
BM_ptr_deref/real_time/threads:4         10.1 ns         40.3 ns     72755340 items_per_second=3.15878G/s
```



```
---------------------------------------------------------------------------------------------
Benchmark                                   Time             CPU   Iterations UserCounters...
---------------------------------------------------------------------------------------------
BM_ptr_deref/real_time/threads:1         38.2 ns         38.2 ns     18313304 items_per_second=836.773M/s
BM_ptr_deref/real_time/threads:2         19.1 ns         38.2 ns     36629094 items_per_second=1.67436G/s
BM_ptr_deref/real_time/threads:4         9.61 ns         38.4 ns     72411060 items_per_second=3.33126G/s
```

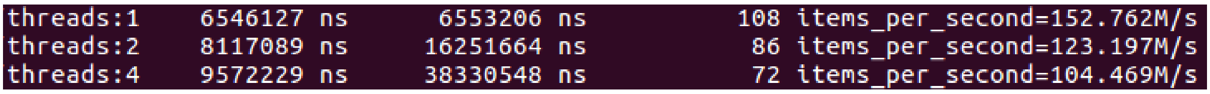


```
------------------------------------------------------------------------------------------------------------
Benchmark                                          Time             CPU   Iterations UserCounters...
------------------------------------------------------------------------------------------------------------
BM_ptr_deref/real_time/threads:1                2283 ns         2281 ns       306644 items_per_second=14.0161M/s
BM_ptr_deref/real_time/threads:2                4322 ns         8635 ns       157174 items_per_second=7.40374M/s
BM_ptr_deref/real_time/threads:4                5772 ns        22916 ns       120648 items_per_second=5.54409M/s
```

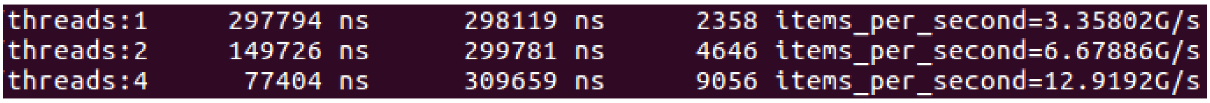


```
------------------------------------------------------------------------------------------------------------
Benchmark                                          Time             CPU   Iterations UserCounters...
------------------------------------------------------------------------------------------------------------
BM_ptr_deref/real_time/threads:1                19.6 ns         19.6 ns     35730008 items_per_second=51.0463M/s
BM_ptr_deref/real_time/threads:2                17.1 ns         19.9 ns     41994276 items_per_second=58.5599M/s
BM_ptr_deref/real_time/threads:4                18.6 ns         23.2 ns     32480008 items_per_second=53.6429M/s
```

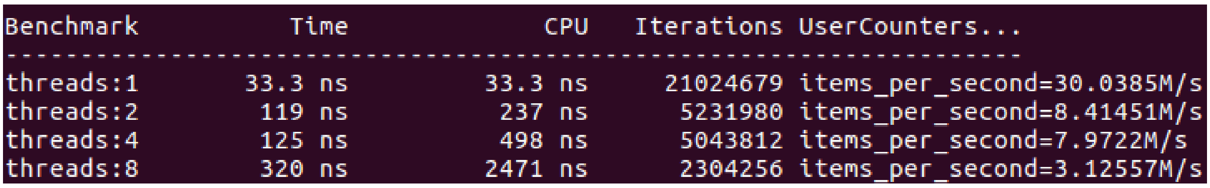
Chapter 7: Data Structures for Concurrency



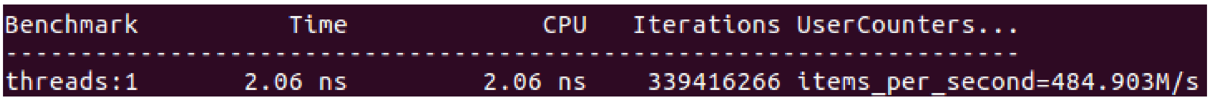
```
threads:1    6546127 ns    6553206 ns          108 items_per_second=152.762M/s
threads:2    8117089 ns   16251664 ns           86 items_per_second=123.197M/s
threads:4    9572229 ns   38330548 ns           72 items_per_second=104.469M/s
```



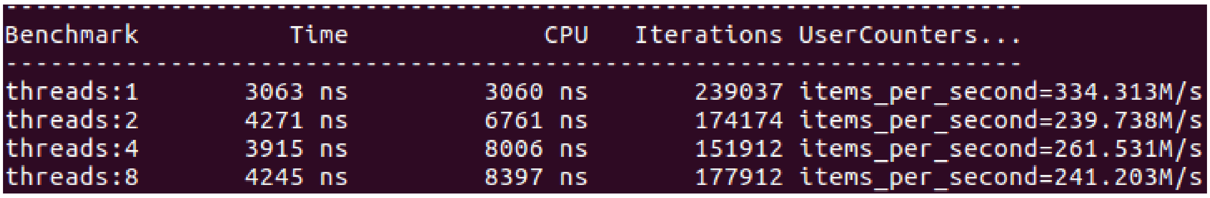
```
threads:1      297794 ns      298119 ns      2358 items_per_second=3.35802G/s
threads:2      149726 ns      299781 ns      4646 items_per_second=6.67886G/s
threads:4       77404 ns      309659 ns      9056 items_per_second=12.9192G/s
```



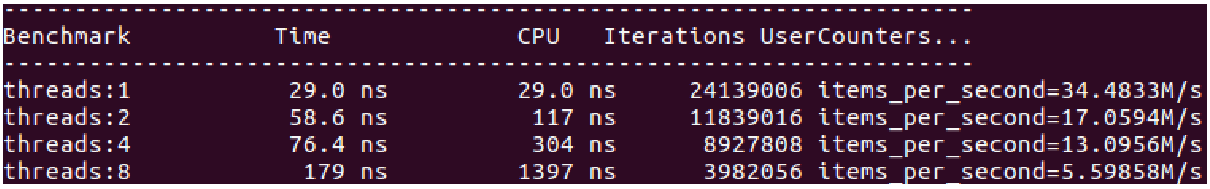
```
Benchmark             Time             CPU   Iterations UserCounters...
------------------------------------------------------------------------
threads:1          33.3 ns         33.3 ns     21024679 items_per_second=30.0385M/s
threads:2           119 ns          237 ns      5231980 items_per_second=8.41451M/s
threads:4           125 ns          498 ns      5043812 items_per_second=7.9722M/s
threads:8           320 ns         2471 ns      2304256 items_per_second=3.12557M/s
```



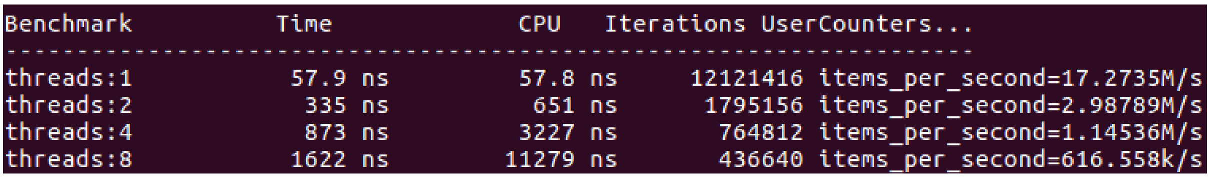
```
Benchmark             Time             CPU   Iterations UserCounters...
------------------------------------------------------------------------
threads:1          2.06 ns         2.06 ns    339416266 items_per_second=484.903M/s
```



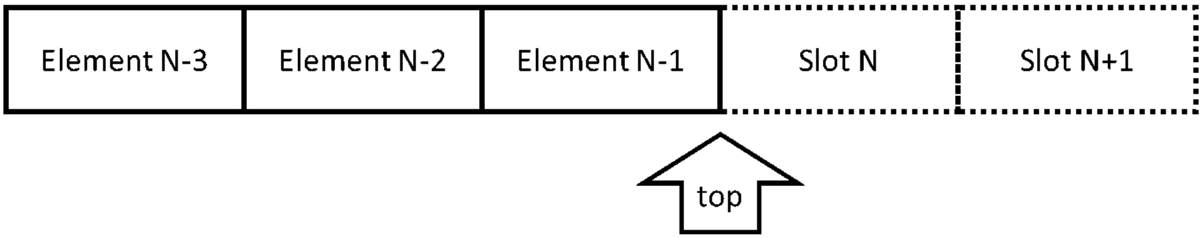
```
------------------------------------------------------------------------
Benchmark             Time             CPU   Iterations UserCounters...
------------------------------------------------------------------------
threads:1          3063 ns         3060 ns       239037 items_per_second=334.313M/s
threads:2          4271 ns         6761 ns       174174 items_per_second=239.738M/s
threads:4          3915 ns         8006 ns       151912 items_per_second=261.531M/s
threads:8          4245 ns         8397 ns       177912 items_per_second=241.203M/s
```

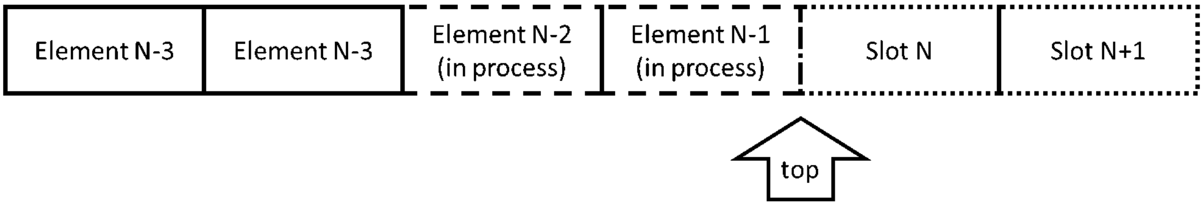


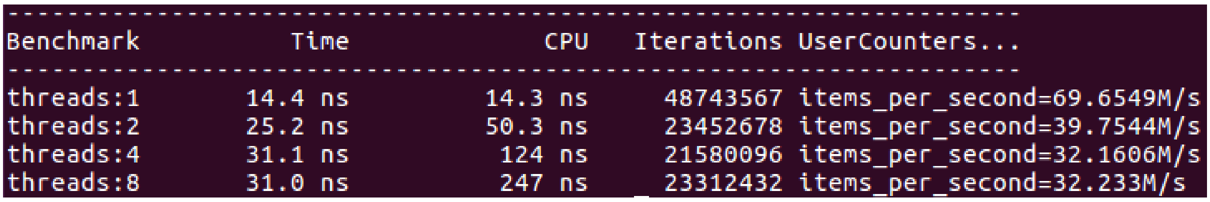
```
------------------------------------------------------------------------
Benchmark               Time             CPU   Iterations UserCounters...
------------------------------------------------------------------------
threads:1            29.0 ns         29.0 ns     24139006 items_per_second=34.4833M/s
threads:2            58.6 ns          117 ns     11839016 items_per_second=17.0594M/s
threads:4            76.4 ns          304 ns      8927808 items_per_second=13.0956M/s
threads:8             179 ns         1397 ns      3982056 items_per_second=5.59858M/s
```



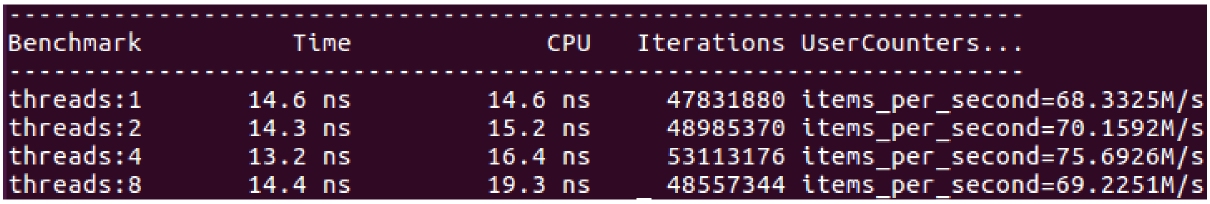
```
Benchmark               Time             CPU   Iterations UserCounters...
------------------------------------------------------------------------
threads:1            57.9 ns         57.8 ns     12121416 items_per_second=17.2735M/s
threads:2             335 ns          651 ns      1795156 items_per_second=2.98789M/s
threads:4             873 ns         3227 ns       764812 items_per_second=1.14536M/s
threads:8            1622 ns        11279 ns       436640 items_per_second=616.558k/s
```



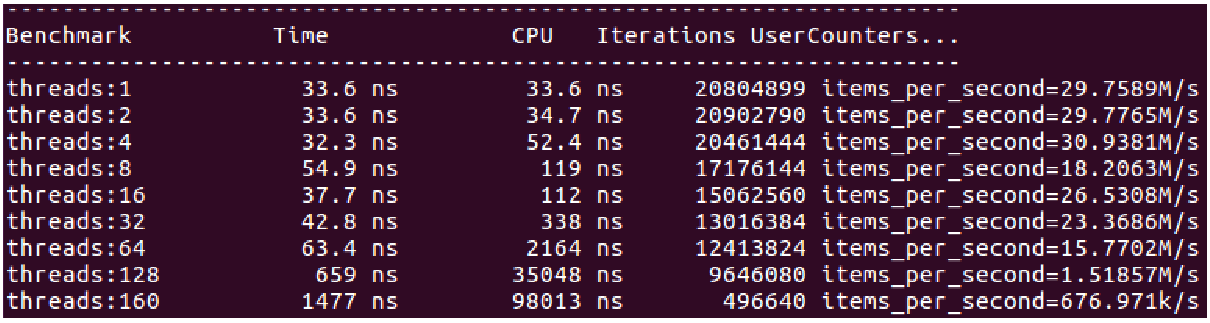




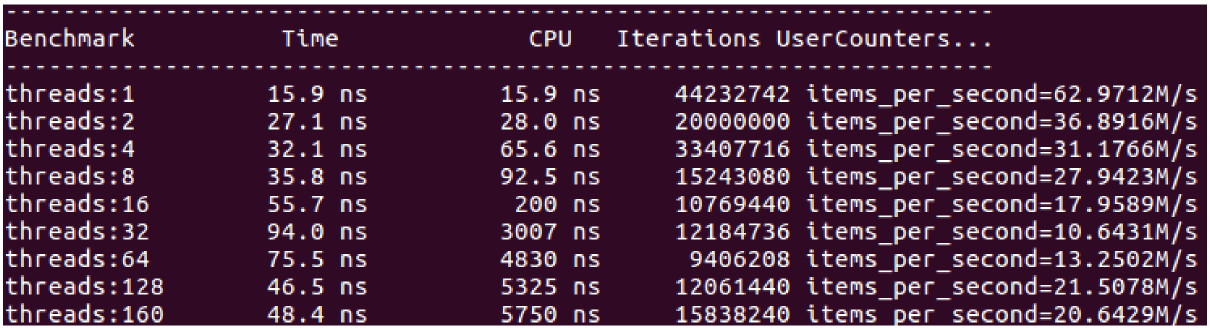
```
--------------------------------------------------------------
Benchmark              Time             CPU   Iterations UserCounters...
--------------------------------------------------------------
threads:1          14.4 ns         14.3 ns     48743567 items_per_second=69.6549M/s
threads:2          25.2 ns         50.3 ns     23452678 items_per_second=39.7544M/s
threads:4          31.1 ns          124 ns     21580096 items_per_second=32.1606M/s
threads:8          31.0 ns          247 ns     23312432 items_per_second=32.233M/s
```



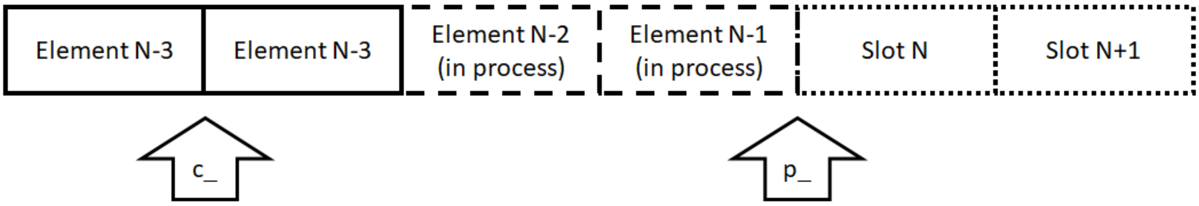
```
--------------------------------------------------------------
Benchmark              Time             CPU   Iterations UserCounters...
--------------------------------------------------------------
threads:1          14.6 ns         14.6 ns     47831880 items_per_second=68.3325M/s
threads:2          14.3 ns         15.2 ns     48985370 items_per_second=70.1592M/s
threads:4          13.2 ns         16.4 ns     53113176 items_per_second=75.6926M/s
threads:8          14.4 ns         19.3 ns     48557344 items_per_second=69.2251M/s
```

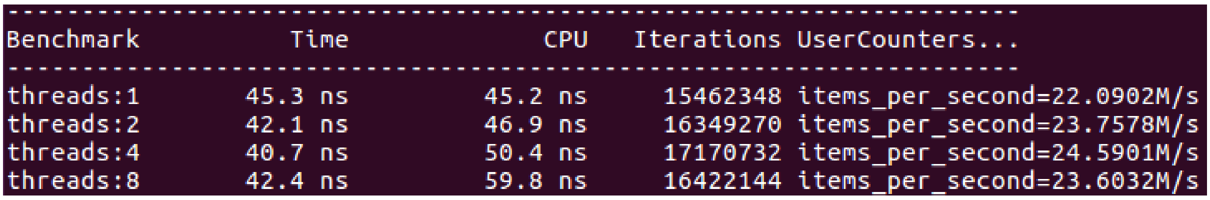


```
--------------------------------------------------------------
Benchmark              Time             CPU   Iterations UserCounters...
--------------------------------------------------------------
threads:1               33.6 ns          33.6 ns     20804899 items_per_second=29.7589M/s
threads:2               33.6 ns          34.7 ns     20902790 items_per_second=29.7765M/s
threads:4               32.3 ns          52.4 ns     20461444 items_per_second=30.9381M/s
threads:8               54.9 ns           119 ns     17176144 items_per_second=18.2063M/s
threads:16              37.7 ns           112 ns     15062560 items_per_second=26.5308M/s
threads:32              42.8 ns           338 ns     13016384 items_per_second=23.3686M/s
threads:64              63.4 ns          2164 ns     12413824 items_per_second=15.7702M/s
threads:128              659 ns         35048 ns      9646080 items_per_second=1.51857M/s
threads:160             1477 ns         98013 ns       496640 items_per_second=676.971k/s
```

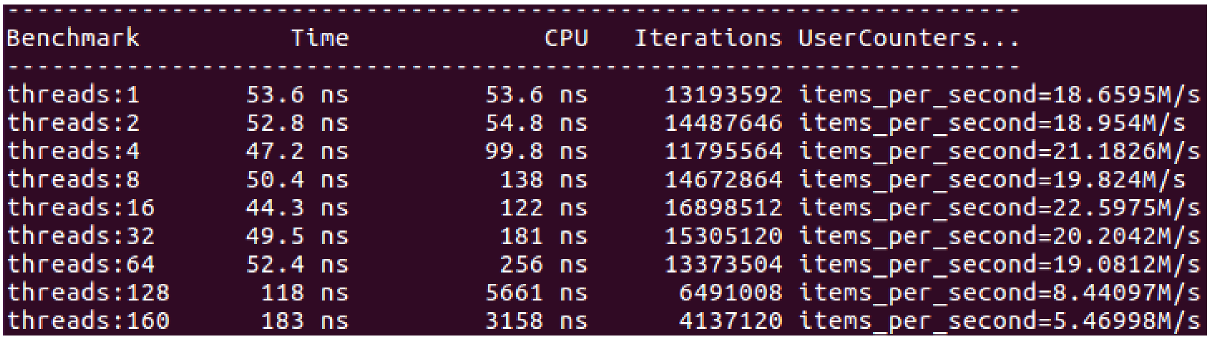


```
--------------------------------------------------------------
Benchmark                Time             CPU   Iterations UserCounters...
--------------------------------------------------------------
threads:1             15.9 ns          15.9 ns     44232742 items_per_second=62.9712M/s
threads:2             27.1 ns          28.0 ns     20000000 items_per_second=36.8916M/s
threads:4             32.1 ns          65.6 ns     33407716 items_per_second=31.1766M/s
threads:8             35.8 ns          92.5 ns     15243080 items_per_second=27.9423M/s
threads:16            55.7 ns           200 ns     10769440 items_per_second=17.9589M/s
threads:32            94.0 ns          3007 ns     12184736 items_per_second=10.6431M/s
threads:64            75.5 ns          4830 ns      9406208 items_per_second=13.2502M/s
threads:128           46.5 ns          5325 ns     12061440 items_per_second=21.5078M/s
threads:160           48.4 ns          5750 ns     15838240 items_per_second=20.6429M/s
```

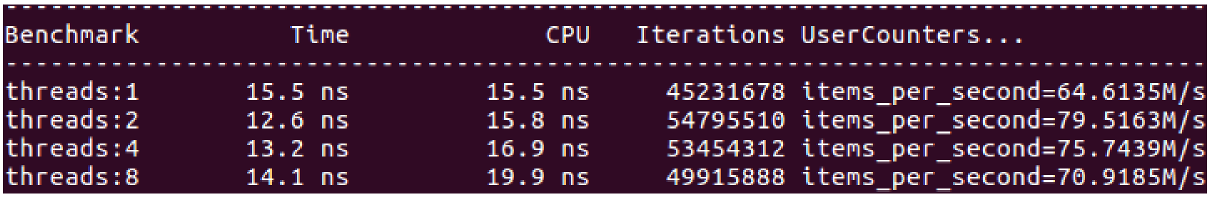




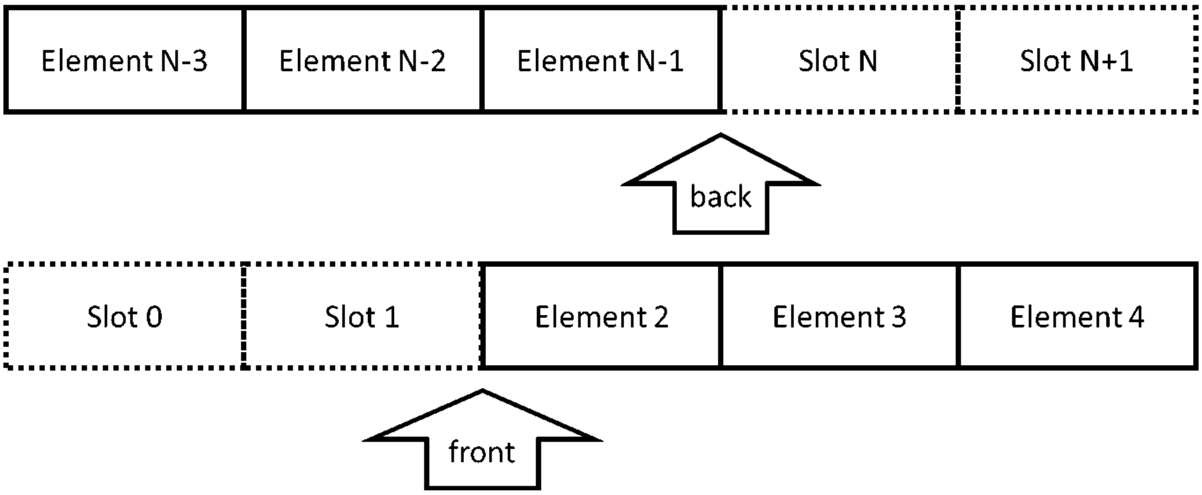
```
--------------------------------------------------------------
Benchmark              Time             CPU   Iterations UserCounters...
--------------------------------------------------------------
threads:1          45.3 ns         45.2 ns     15462348 items_per_second=22.0902M/s
threads:2          42.1 ns         46.9 ns     16349270 items_per_second=23.7578M/s
threads:4          40.7 ns         50.4 ns     17170732 items_per_second=24.5901M/s
threads:8          42.4 ns         59.8 ns     16422144 items_per_second=23.6032M/s
```

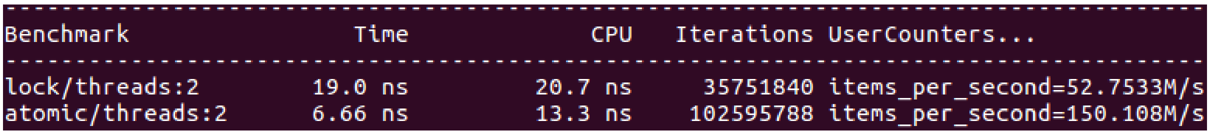


```
--------------------------------------------------------------
Benchmark              Time             CPU   Iterations UserCounters...
--------------------------------------------------------------
threads:1          53.6 ns         53.6 ns     13193592 items_per_second=18.6595M/s
threads:2          52.8 ns         54.8 ns     14487646 items_per_second=18.954M/s
threads:4          47.2 ns         99.8 ns     11795564 items_per_second=21.1826M/s
threads:8          50.4 ns          138 ns     14672864 items_per_second=19.824M/s
threads:16         44.3 ns          122 ns     16898512 items_per_second=22.5975M/s
threads:32         49.5 ns          181 ns     15305120 items_per_second=20.2042M/s
threads:64         52.4 ns          256 ns     13373504 items_per_second=19.0812M/s
threads:128         118 ns         5661 ns      6491008 items_per_second=8.44097M/s
threads:160         183 ns         3158 ns      4137120 items_per_second=5.46998M/s
```

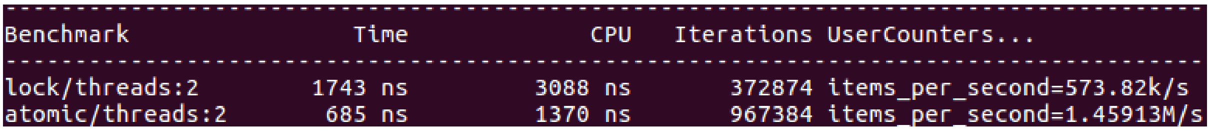


```
-----------------------------------------------------------------------------------
Benchmark              Time             CPU   Iterations UserCounters...
-----------------------------------------------------------------------------------
threads:1         15.5 ns         15.5 ns     45231678 items_per_second=64.6135M/s
threads:2         12.6 ns         15.8 ns     54795510 items_per_second=79.5163M/s
threads:4         13.2 ns         16.9 ns     53454312 items_per_second=75.7439M/s
threads:8         14.1 ns         19.9 ns     49915888 items_per_second=70.9185M/s
```

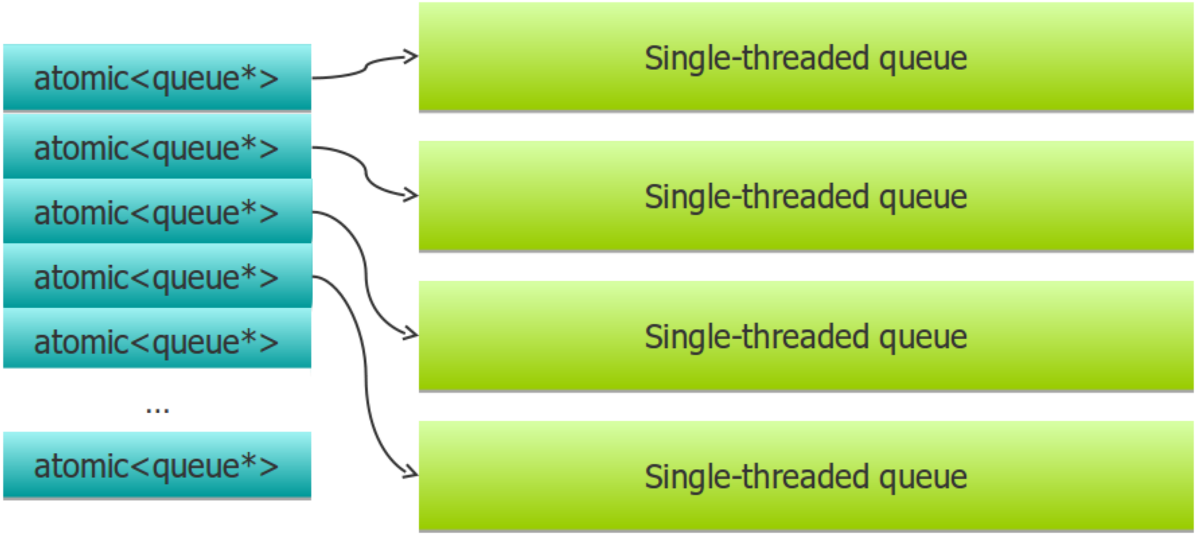


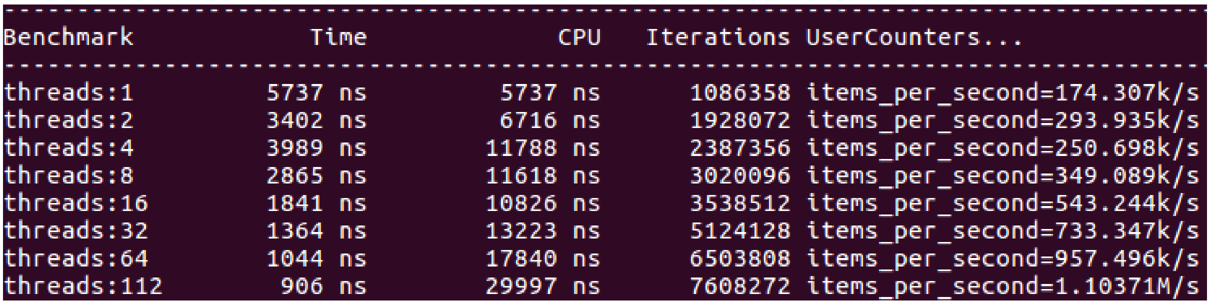


```
-----------------------------------------------------------------------------------
Benchmark                  Time             CPU   Iterations UserCounters...
-----------------------------------------------------------------------------------
lock/threads:2          19.0 ns         20.7 ns     35751840 items_per_second=52.7533M/s
atomic/threads:2        6.66 ns         13.3 ns    102595788 items_per_second=150.108M/s
```

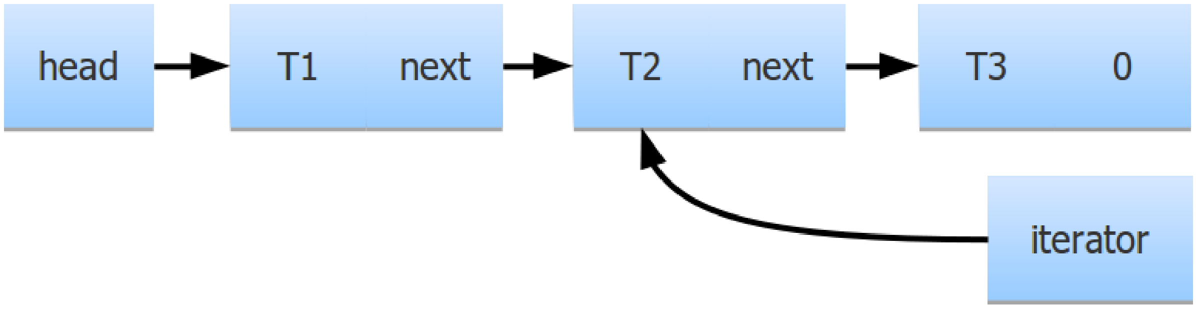


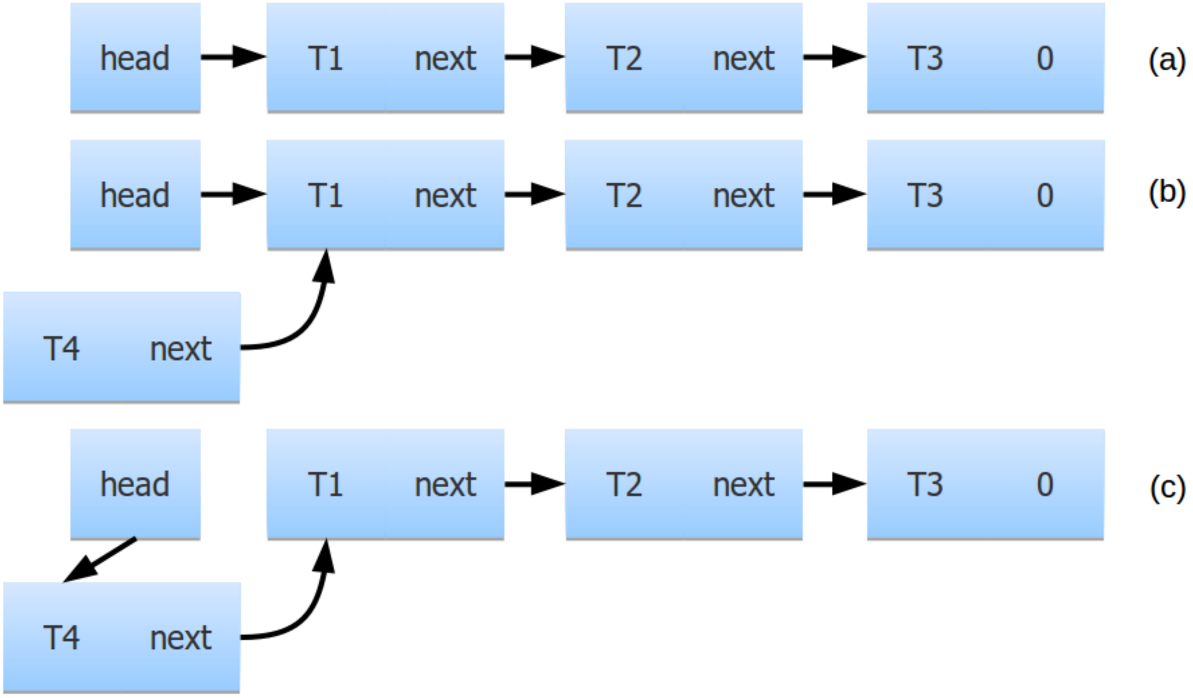
```
-----------------------------------------------------------------------------------
Benchmark                  Time             CPU   Iterations UserCounters...
-----------------------------------------------------------------------------------
lock/threads:2          1743 ns         3088 ns       372874 items_per_second=573.82k/s
atomic/threads:2         685 ns         1370 ns       967384 items_per_second=1.45913M/s
```

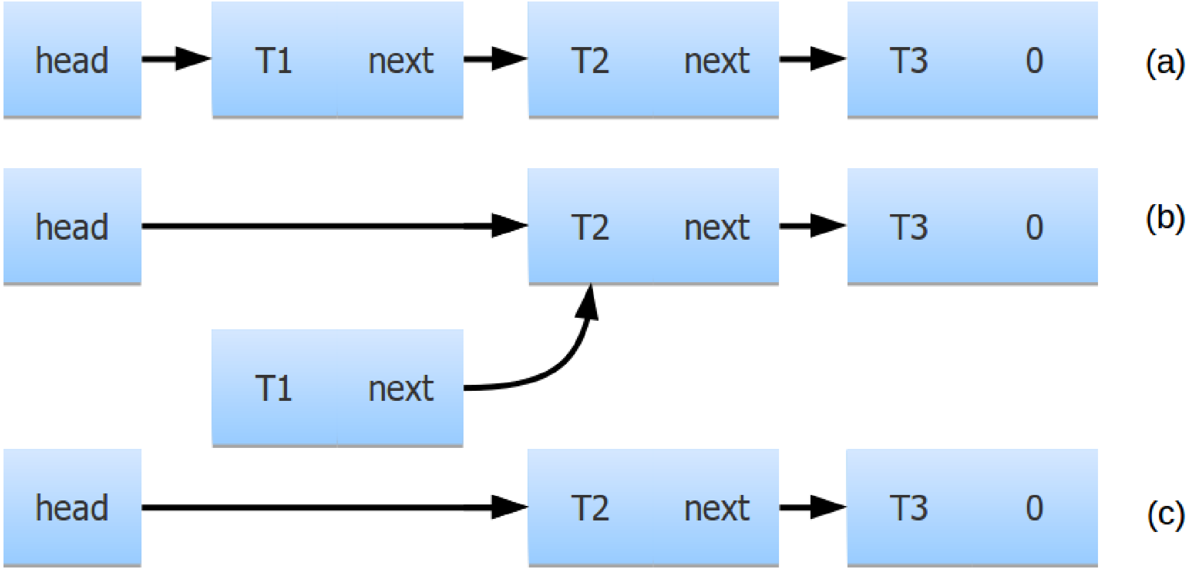


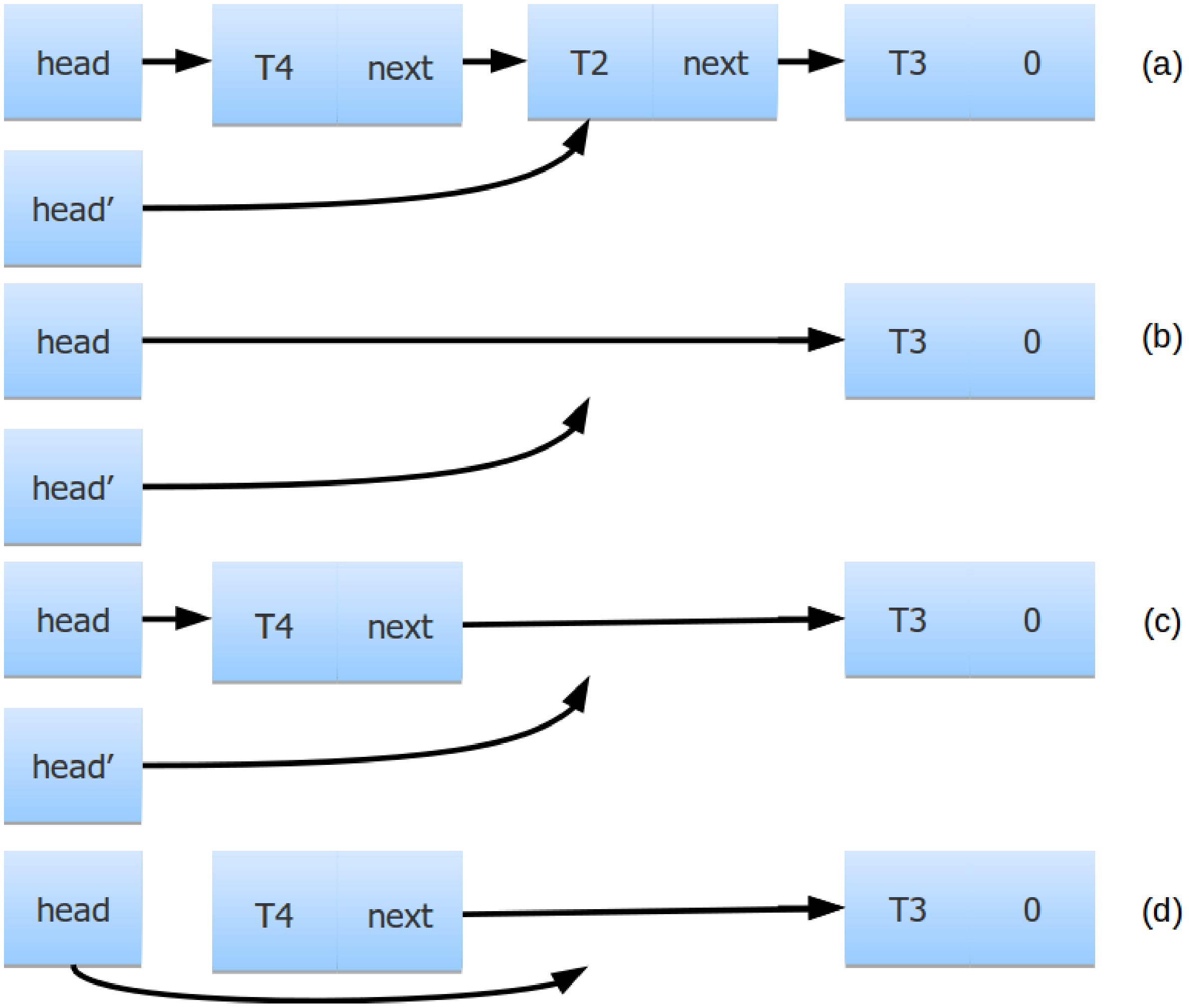


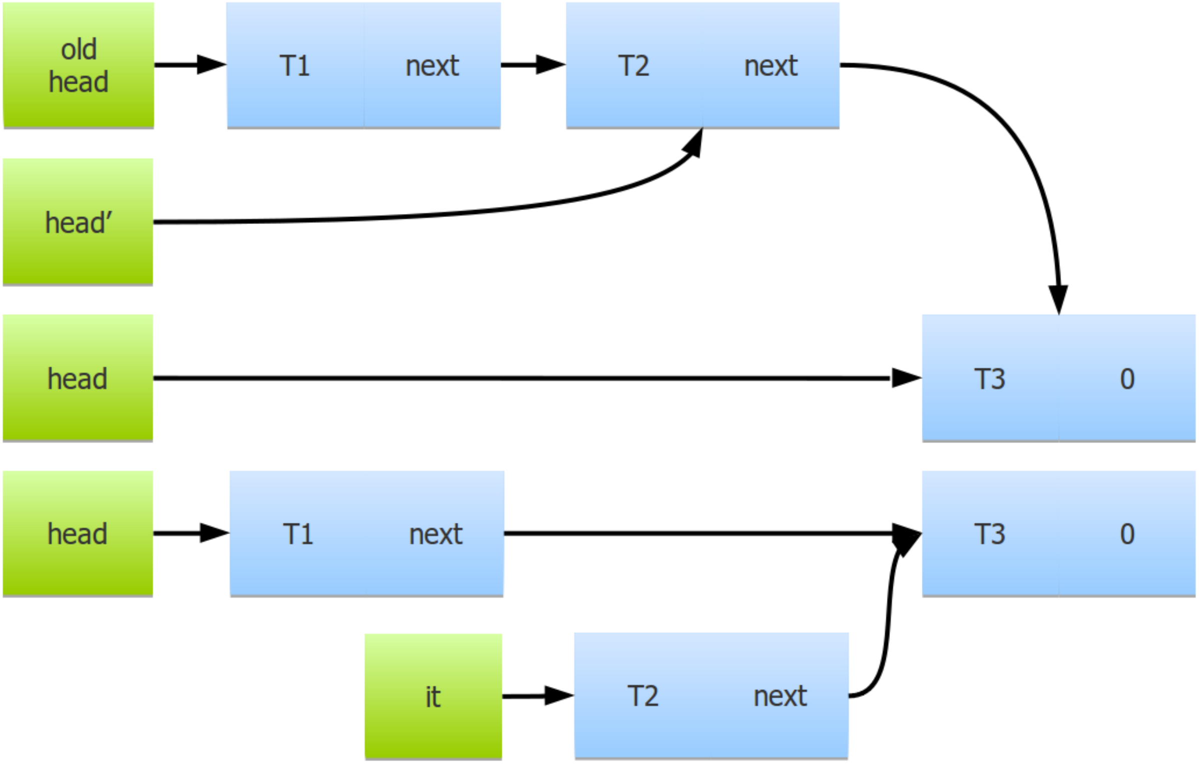
```
-----------------------------------------------------------------------------------
Benchmark              Time             CPU   Iterations UserCounters...
-----------------------------------------------------------------------------------
threads:1           5737 ns         5737 ns      1086358 items_per_second=174.307k/s
threads:2           3402 ns         6716 ns      1928072 items_per_second=293.935k/s
threads:4           3989 ns        11788 ns      2387356 items_per_second=250.698k/s
threads:8           2865 ns        11618 ns      3020096 items_per_second=349.089k/s
threads:16          1841 ns        10826 ns      3538512 items_per_second=543.244k/s
threads:32          1364 ns        13223 ns      5124128 items_per_second=733.347k/s
threads:64          1044 ns        17840 ns      6503808 items_per_second=957.496k/s
threads:112          906 ns        29997 ns      7608272 items_per_second=1.10371M/s
```


head
T1
next
T2
next
T3
0
iterator

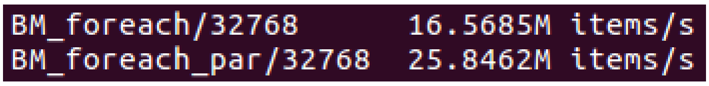

head
T1
next
T2
next
T3
0
(a)
head
T1
next
T2
next
T3
0
(b)
T4
next
head
T1
next
T2
next
T3
0
(c)
T4
next


head
T1
next
T2
next
T3
0
(a)
head
T2
next
T3
0
(b)
T1
next
head
T2
next
T3
0
(c)

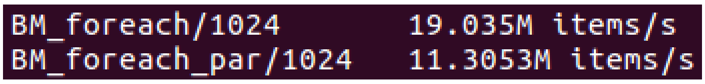

head
T4
next
T2
next
T3
0
(a)
head'
head
T3
0
(b)
head'
head
T4
next
T3
0
(c)
head'
head
T4
next
T3
0
(d)


old head
T1
next
T2
next
head'
head
T3
0
head
T1
next
T3
0
it
T2
next

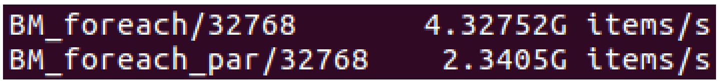
Chapter 8: Concurrency in C++



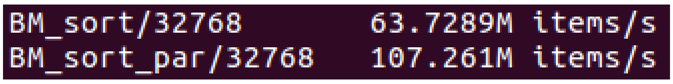
```
BM_foreach/32768       16.5685M items/s
BM_foreach_par/32768   25.8462M items/s
```



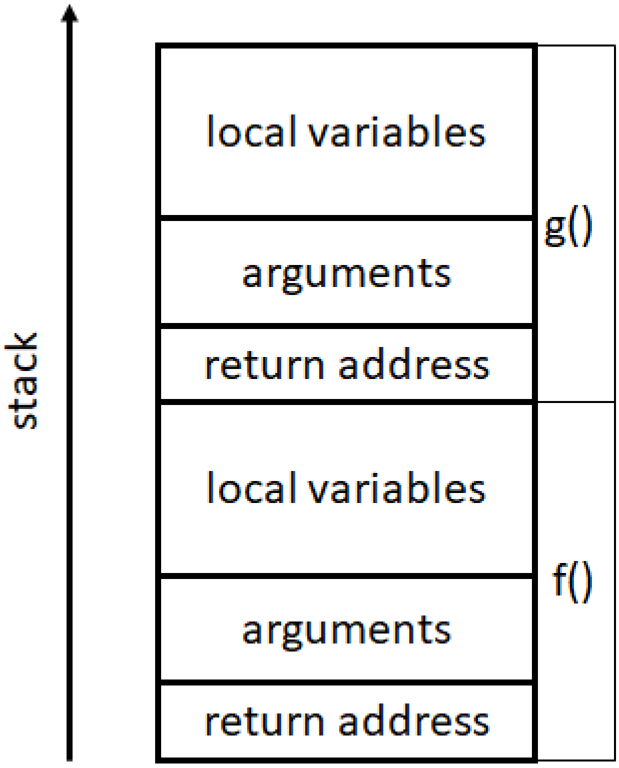
```
BM_foreach/1024        19.035M items/s
BM_foreach_par/1024    11.3053M items/s
```

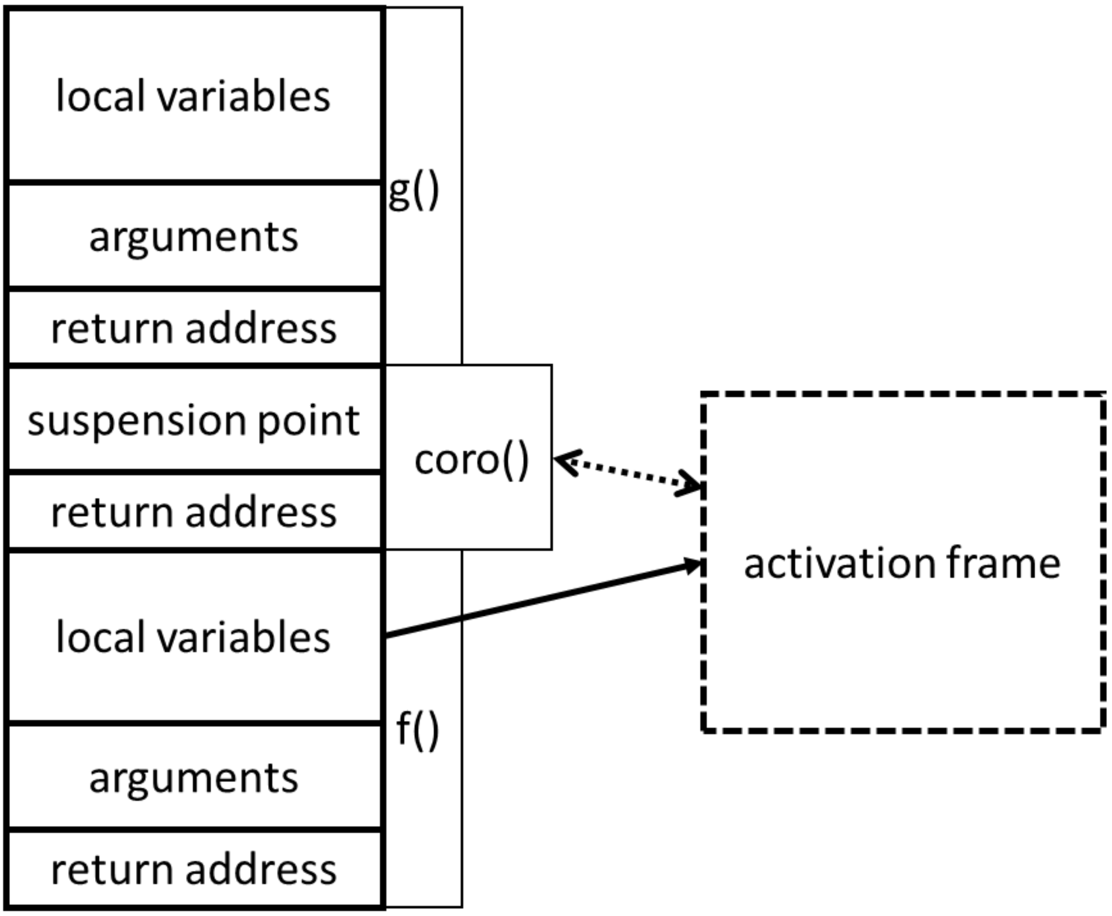


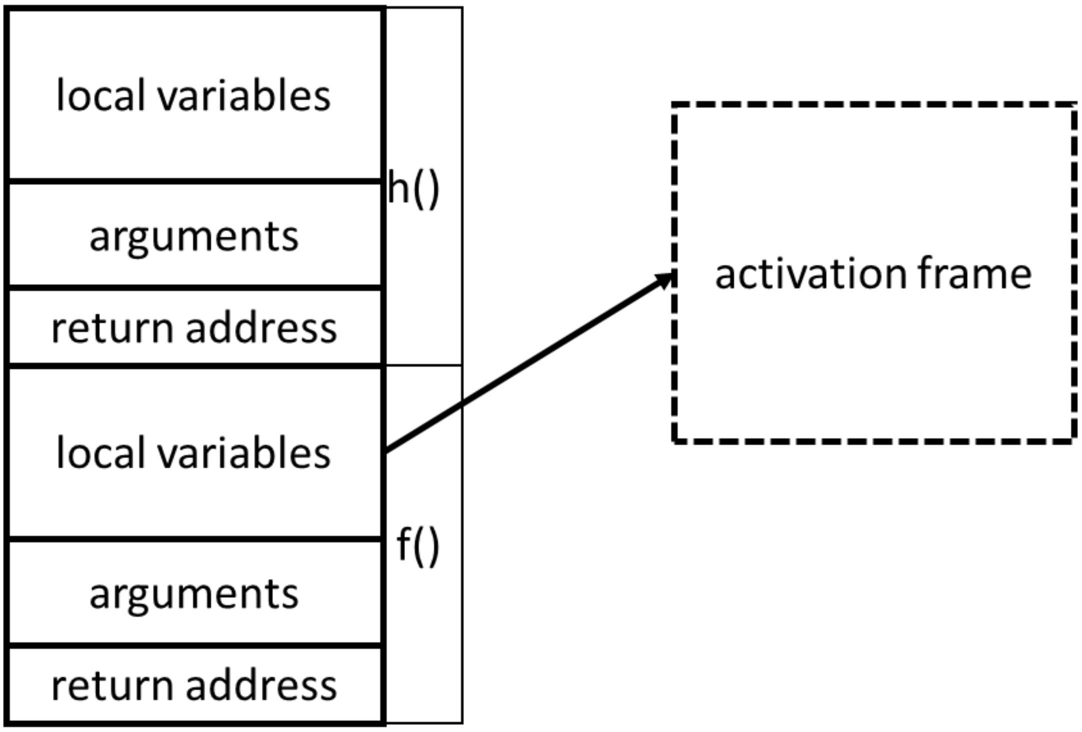
```
BM_foreach/32768        4.32752G items/s
BM_foreach_par/32768     2.3405G items/s
```

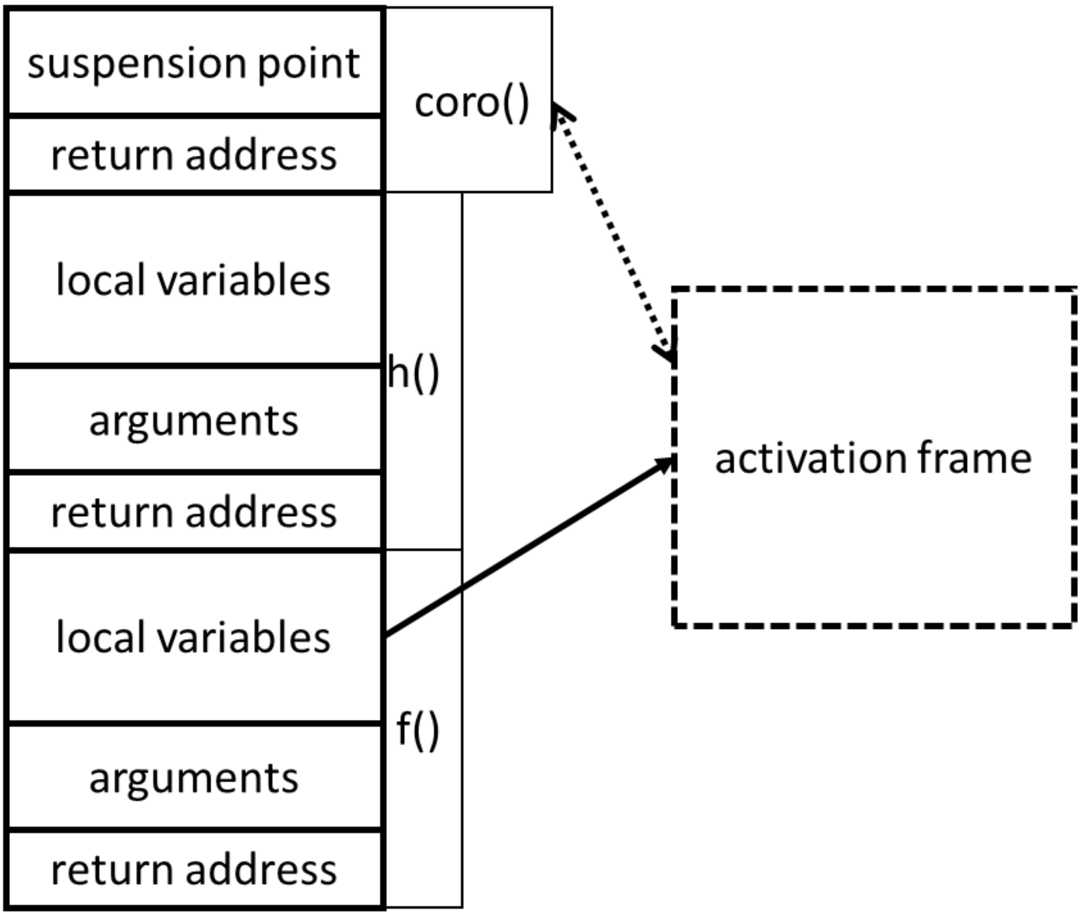


```
BM_sort/32768          63.7289M items/s
BM_sort_par/32768     107.261M items/s
```

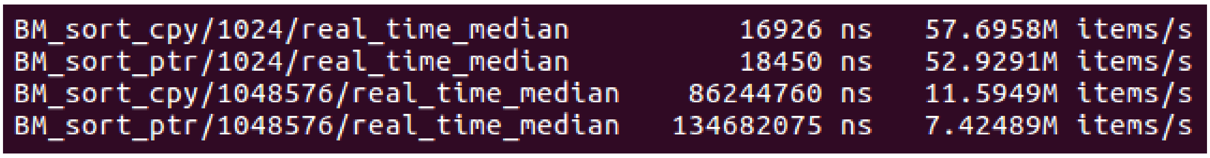



local variables
arguments
return address
g()
suspension point
return address
coro()
activation frame
local variables
arguments
return address
f()

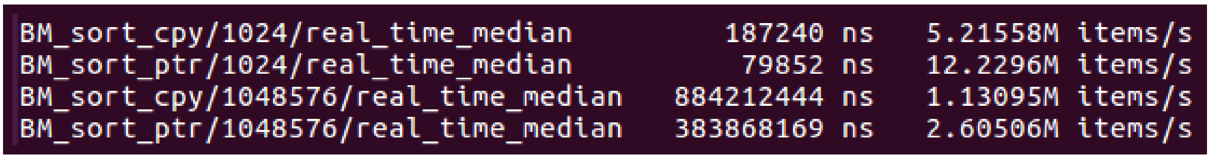

local variables
arguments
return address
h()
activation frame
local variables
arguments
return address
f()


suspension point
return address
coro()
local variables
h()
arguments
return address
local variables
f()
arguments
return address
activation frame

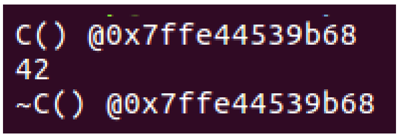
Chapter 9: High-Performance C++



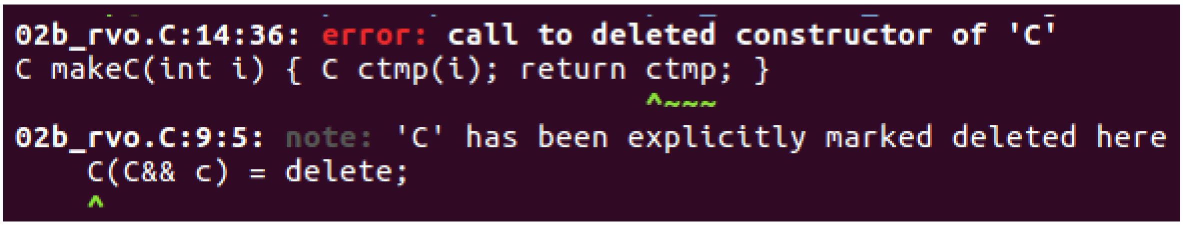
```
BM_sort_cpy/1024/real_time_median          16926 ns    57.6958M items/s
BM_sort_ptr/1024/real_time_median          18450 ns    52.9291M items/s
BM_sort_cpy/1048576/real_time_median    86244760 ns    11.5949M items/s
BM_sort_ptr/1048576/real_time_median   134682075 ns    7.42489M items/s
```



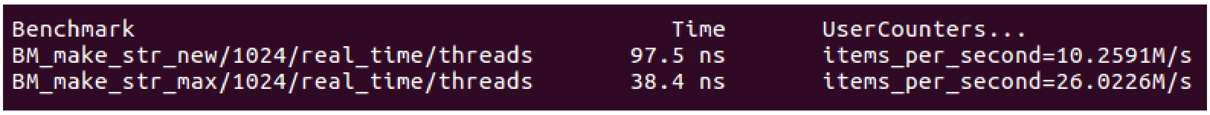
```
BM_sort_cpy/1024/real_time_median         187240 ns    5.21558M items/s
BM_sort_ptr/1024/real_time_median          79852 ns    12.2296M items/s
BM_sort_cpy/1048576/real_time_median   884212444 ns    1.13095M items/s
BM_sort_ptr/1048576/real_time_median   383868169 ns    2.60506M items/s
```



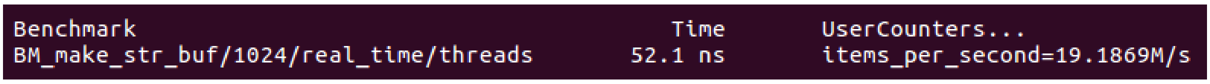
```
C() @0x7ffe44539b68
42
~C() @0x7ffe44539b68
```



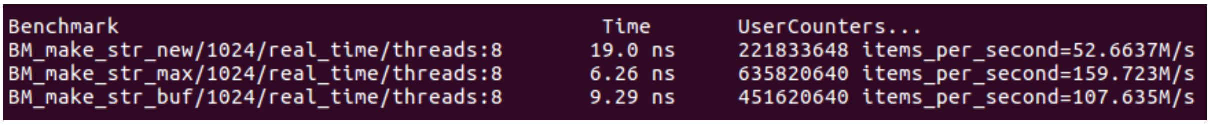
```
02b_rvo.C:14:36: error: call to deleted constructor of 'C'
C makeC(int i) { C ctmp(i); return ctmp; }
                                   ^~~~
02b_rvo.C:9:5: note: 'C' has been explicitly marked deleted here
    C(C&& c) = delete;
    ^
```



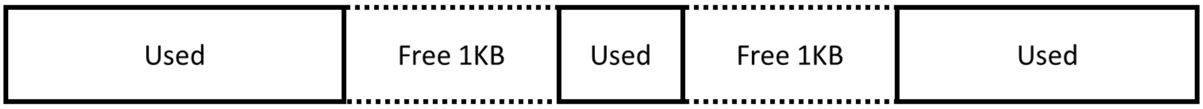
```
Benchmark                                        Time         UserCounters...
BM_make_str_new/1024/real_time/threads        97.5 ns         items_per_second=10.2591M/s
BM_make_str_max/1024/real_time/threads        38.4 ns         items_per_second=26.0226M/s
```

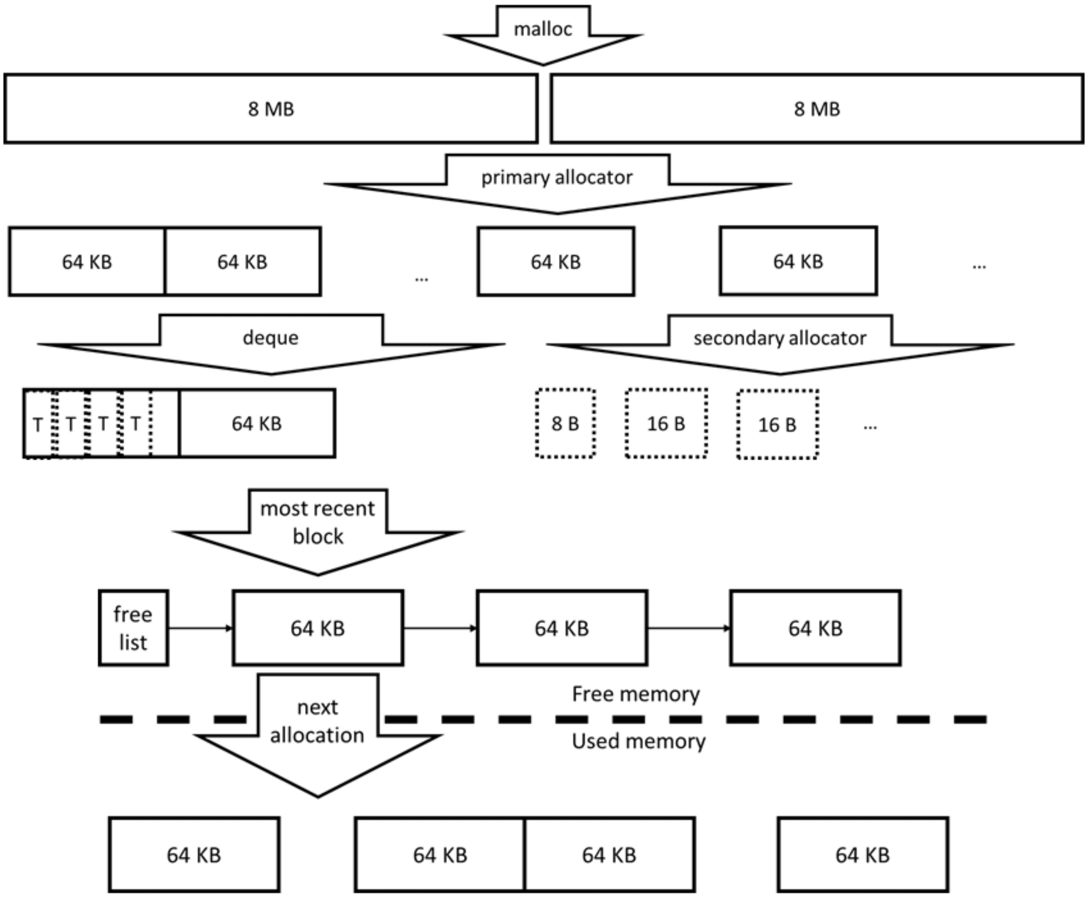


```
Benchmark                                        Time         UserCounters...
BM_make_str_buf/1024/real_time/threads        52.1 ns         items_per_second=19.1869M/s
```

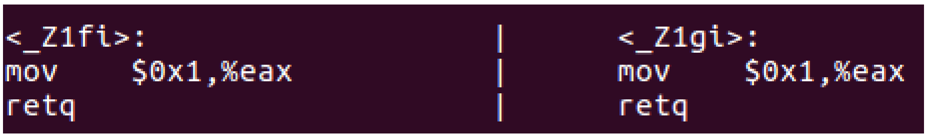


```
Benchmark                                        Time        UserCounters...
BM_make_str_new/1024/real_time/threads:8      19.0 ns        221833648 items_per_second=52.6637M/s
BM_make_str_max/1024/real_time/threads:8      6.26 ns        635820640 items_per_second=159.723M/s
BM_make_str_buf/1024/real_time/threads:8      9.29 ns        451620640 items_per_second=107.635M/s
```

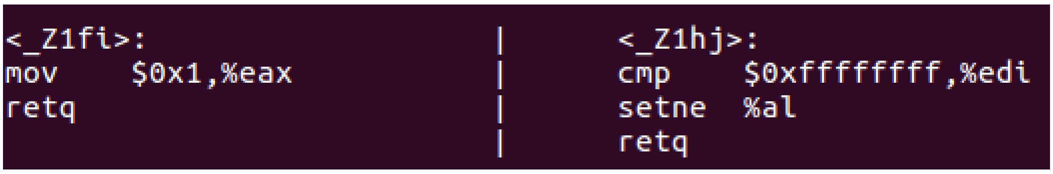



malloc
8 MB
8 MB
primary allocator
64 KB
64 KB
...
64 KB
64 KB
...
deque
secondary allocator
T
T
T
T
64 KB
8 B
16 B
16 B
...
most recent block
free list
64 KB
64 KB
64 KB
next allocation
Free memory
Used memory
64 KB
64 KB
64 KB
64 KB

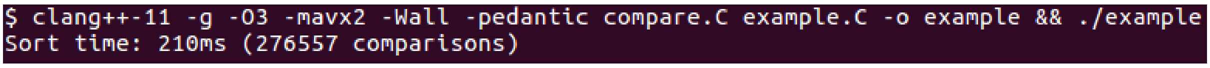
Chapter 10: Compiler Optimizations in C++



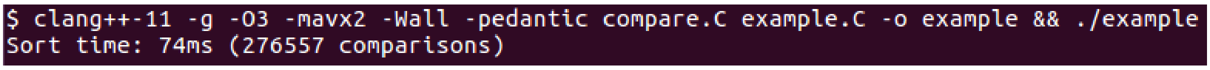
```
<_Z1fi>:                                        |       <_Z1gi>:
mov     $0x1,%eax                               |       mov     $0x1,%eax
retq                                            |       retq
```



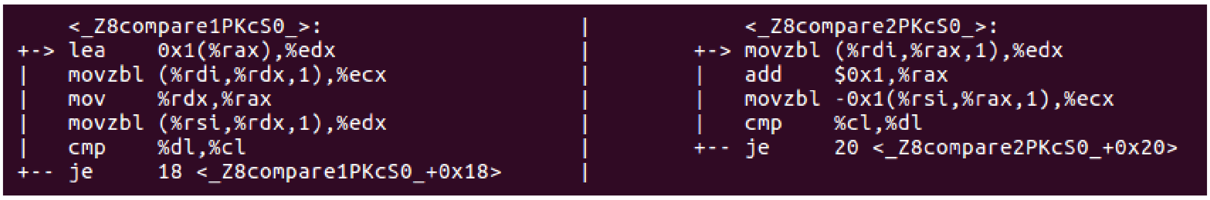
```
<_Z1fi>:                                        |       <_Z1hj>:
mov     $0x1,%eax                               |       cmp     $0xffffffff,%edi
retq                                            |       setne   %al
                                                |       retq
```



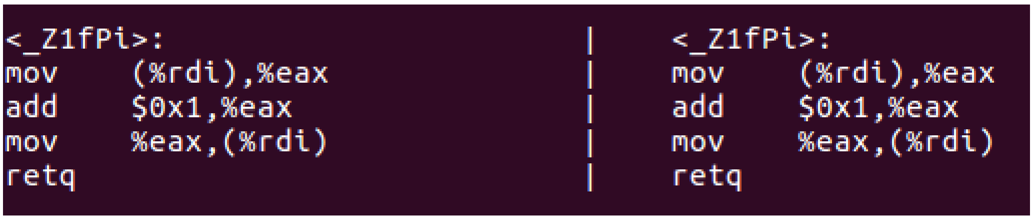
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic compare.C example.C -o example && ./example
Sort time: 210ms (276557 comparisons)
```



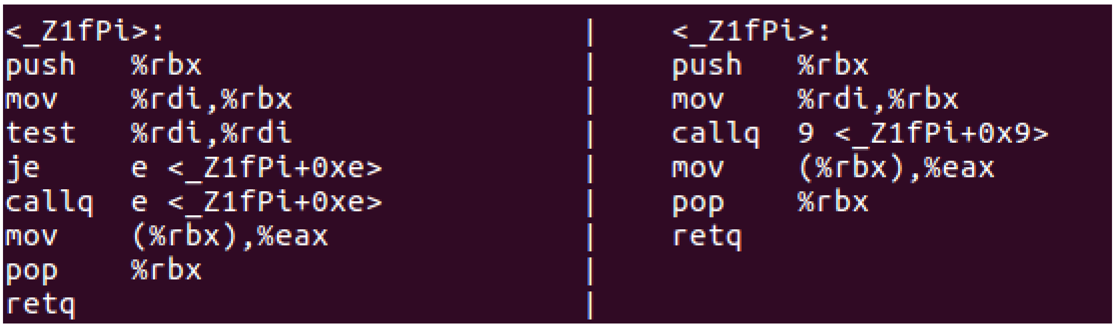
```
$ clang++-11 -g -O3 -mavx2 -Wall -pedantic compare.C example.C -o example && ./example
Sort time: 74ms (276557 comparisons)
```



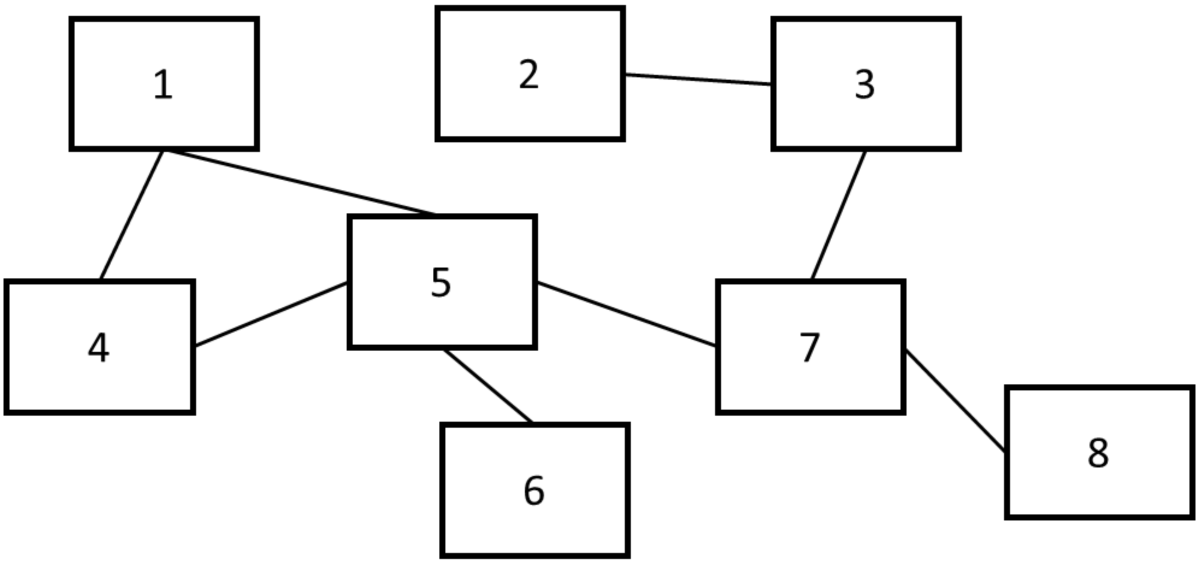
```
     <_Z8compare1PKcS0_>:                          |            <_Z8compare2PKcS0_>:
 +-> lea     0x1(%rax),%edx                        |        +-> movzbl (%rdi,%rax,1),%edx
 |   movzbl (%rdi,%rdx,1),%ecx                     |        |   add     $0x1,%rax
 |   mov     %rdx,%rax                             |        |   movzbl -0x1(%rsi,%rax,1),%ecx
 |   movzbl (%rsi,%rdx,1),%edx                     |        |   cmp     %cl,%dl
 |   cmp     %dl,%cl                               |        +-- je      20 <_Z8compare2PKcS0_+0x20>
 +-- je      18 <_Z8compare1PKcS0_+0x18>           |
```

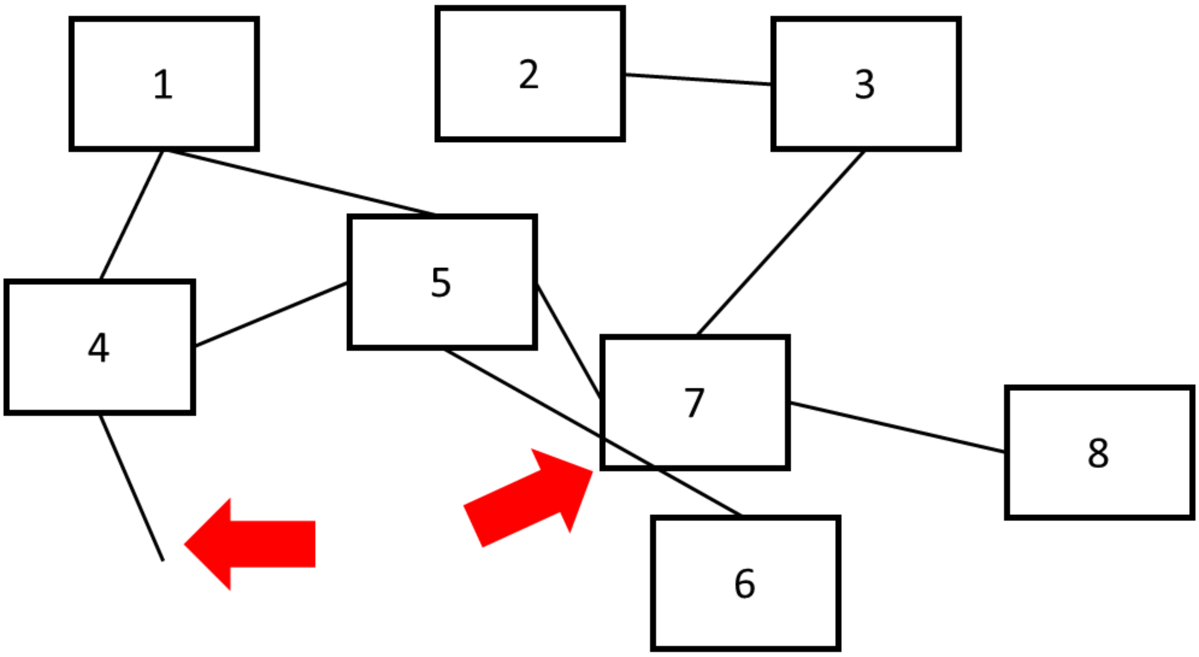


```
<_Z1fPi>:                                       |     <_Z1fPi>:
mov     (%rdi),%eax                             |     mov     (%rdi),%eax
add     $0x1,%eax                               |     add     $0x1,%eax
mov     %eax,(%rdi)                             |     mov     %eax,(%rdi)
retq                                            |     retq
```

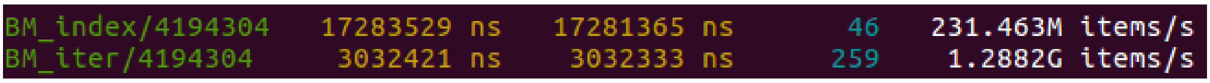


```
<_Z1fPi>:                                       |     <_Z1fPi>:
push    %rbx                                    |     push    %rbx
mov     %rdi,%rbx                               |     mov     %rdi,%rbx
test    %rdi,%rdi                               |     callq   9 <_Z1fPi+0x9>
je      e <_Z1fPi+0xe>                          |     mov     (%rbx),%eax
callq   e <_Z1fPi+0xe>                          |     pop     %rbx
mov     (%rbx),%eax                             |     retq
pop     %rbx                                    |
retq                                            |
```

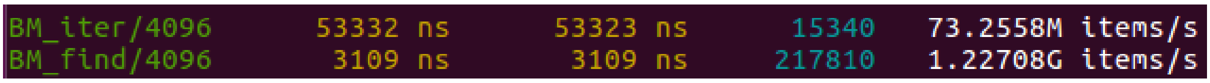

1
2
3
4
5
6
7
8


1
2
3
4
5
7
8
6

Chapter 11: Undefined Behavior and Performance

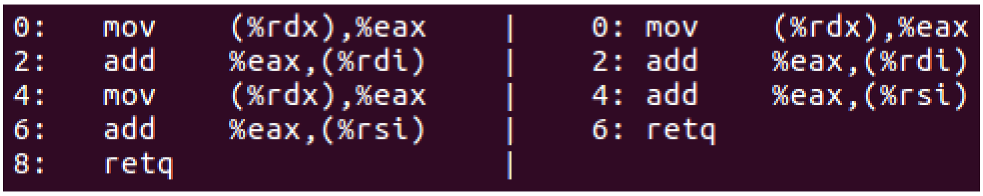


```
BM_index/4194304    17283529 ns    17281365 ns          46    231.463M items/s
BM_iter/4194304      3032421 ns     3032333 ns         259     1.2882G items/s
```

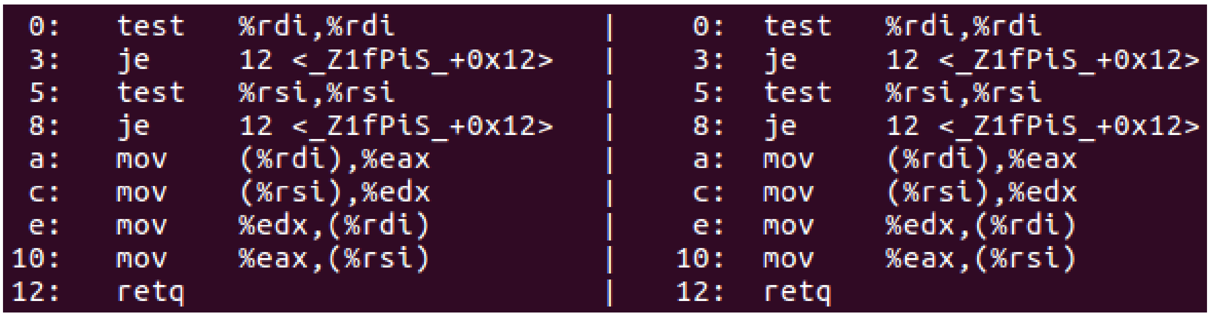


```
BM_iter/4096         53332 ns         53323 ns       15340    73.2558M items/s
BM_find/4096          3109 ns          3109 ns      217810    1.22708G items/s
```

Chapter 12: Design for Performance



```
0:    mov     (%rdx),%eax     |    0: mov    (%rdx),%eax
2:    add     %eax,(%rdi)     |    2: add    %eax,(%rdi)
4:    mov     (%rdx),%eax     |    4: add    %eax,(%rsi)
6:    add     %eax,(%rsi)     |    6: retq
8:    retq                    |
```



```
 0:   test   %rdi,%rdi              |    0:   test   %rdi,%rdi
 3:   je     12 <_Z1fPiS_+0x12>     |    3:   je     12 <_Z1fPiS_+0x12>
 5:   test   %rsi,%rsi              |    5:   test   %rsi,%rsi
 8:   je     12 <_Z1fPiS_+0x12>     |    8:   je     12 <_Z1fPiS_+0x12>
 a:   mov    (%rdi),%eax            |    a:   mov    (%rdi),%eax
 c:   mov    (%rsi),%edx            |    c:   mov    (%rsi),%edx
 e:   mov    %edx,(%rdi)            |    e:   mov    %edx,(%rdi)
10:   mov    %eax,(%rsi)            |   10:   mov    %eax,(%rsi)
12:   retq                          |   12:   retq
```