SPECTRAL:
TECHNOLOGIES

WEEK 7

Perf

SPECTRAL:
TECHNOLOGIES

Perf

- perf stat
- perf record/report
- perf syscall

SPECTRAL:
TECHNOLOGIES

perf stat

- Allows you to measure various counters of different executables (`perf list`)
  - branches, branch-misses
  - cycles, instructions, context-switches
  - major-faults, minor-faults, faults
  - cache-misses, cache-references, L1-dcache-load-misses, L1-dcache-loads
- `perf stat -e <events>` to specify concrete events

SPECTRAL:
TECHNOLOGIES

Example

- Lets make a program, that takes sorted array and then processes multiple queries of "find lower_bound(x) for given x"
- Simple solution (binary search)

```
int Query(int x) const {
    auto it = std::lower_bound(data_.begin(), data_.end(), x);
    return it == data_.end() ? 0 : *it;
}
```

- Assume the length of array is 10^8 and there are 10^7 queries
- With this array size the expected bottleneck is gonna be random memory jumps, that binary search does

perf stat

- perf stat by default doesn't show much:

```
> perf stat ./bin
 Performance counter stats for './bin':

         8,401.37 msec task-clock                #    0.999 CPUs utilized
               76      context-switches          #    0.009 K/sec
                0      cpu-migrations            #    0.000 K/sec
          195,440      page-faults               #    0.023 M/sec
   26,730,393,359      cycles                    #    3.182 GHz
       44,953,734      stalled-cycles-frontend   #    0.17% frontend cycles idle
       85,475,051      stalled-cycles-backend    #    0.32% backend cycles idle
   11,219,240,562      instructions              #    0.42  insn per cycle
                                                 #    0.01  stalled cycles per insn
    1,060,599,655      branches                  #  126.241 M/sec
                0      branch-misses             #    0.00% of all branches
```

- So lets try to check cache events

SPECTRAL:
TECHNOLOGIES

Example

```
>perf stat -e cache-references,cache-misses,L1-dcache-loads,L1-dcache-load-misses ./bin
 Performance counter stats for './bin':

       317,474,639      cache-references:u
       131,958,911      cache-misses:u            #   41.565 % of all cache refs
     1,620,186,411      L1-dcache-loads:u
       165,314,470      L1-dcache-load-misses:u   #   10.20% of all L1-dcache hits

       8.634930995 seconds time elapsed

       8.375269000 seconds user
       0.255977000 seconds sys
```

SPECTRAL:
TECHNOLOGIES

Optimization

– Lets split the initial array into blocks of size 10^4 and save maximum in each block separately

```
void InitIndex() {
    block_size = sqrt(data_.size());
    index_.reserve(data_.size() / block_size);
    for (size_t i = 0; i < data_.size(); i += block_size) {
        size_t end = std::min(i + block_size, data_.size());
        index_.push_back(data_[end - 1]);
    }
}
```

SPECTRAL:
TECHNOLOGIES

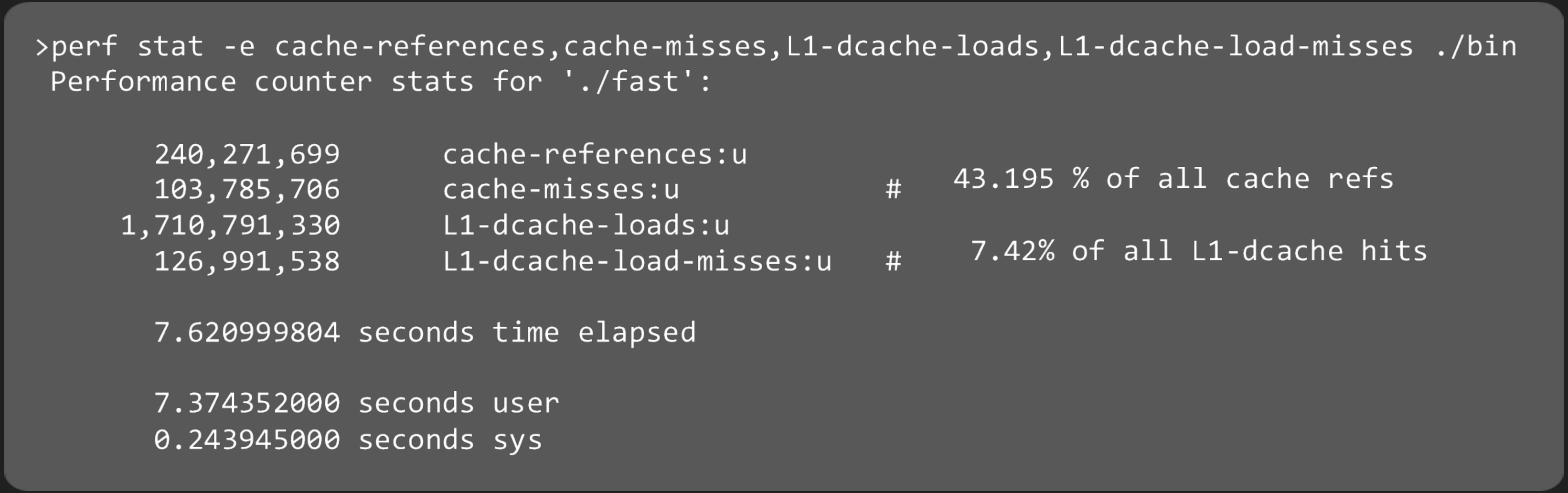
Optimization

– To process a query, perform a binary search on `index_` first to find the corresponding block and then another binary search in this block

```
int Query(int x) const {
    auto it = std::lower_bound(index_.begin(), index_.end(), x);
    if (it == index_.end()) {
        return 0;
    }
    size_t block_num = it - index_.begin();
    size_t left = block_size * block_num;
    size_t right = std::min(data_.size(), left + block_size);
    return *std::lower_bound(data_.begin() + left, data_.begin() + right, x);
}
```

SPECTRAL:
TECHNOLOGIES

Optimization



```
>perf stat -e cache-references,cache-misses,L1-dcache-loads,L1-dcache-load-misses ./bin
 Performance counter stats for './fast':

      240,271,699      cache-references:u
      103,785,706      cache-misses:u              #   43.195 % of all cache refs
    1,710,791,330      L1-dcache-loads:u
      126,991,538      L1-dcache-load-misses:u     #    7.42% of all L1-dcache hits

      7.620999804 seconds time elapsed

      7.374352000 seconds user
      0.243945000 seconds sys
```

SPECTRAL:
TECHNOLOGIES

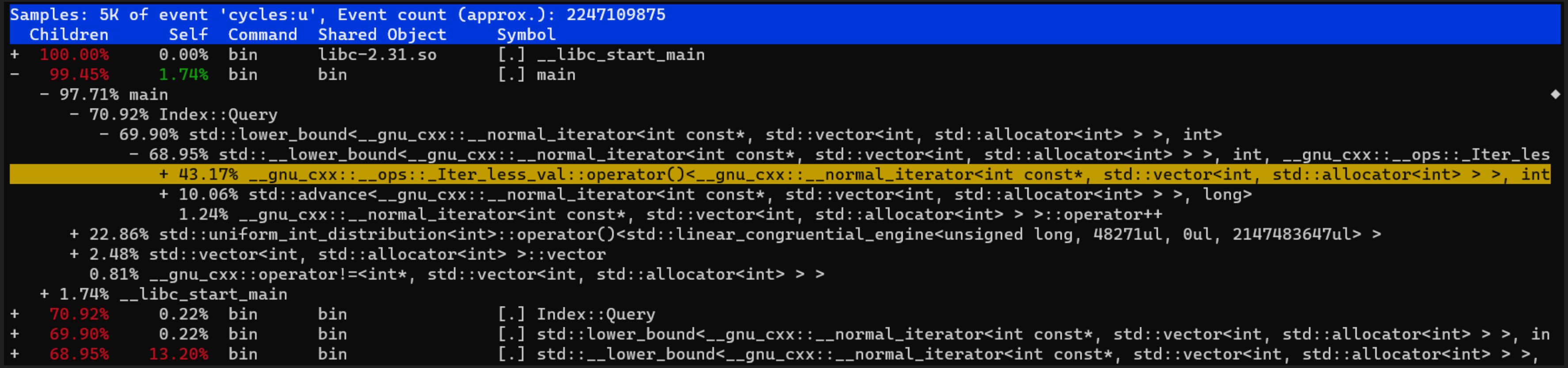
perf record

- `perf record` allows to profile given executable
- `perf report` builds a report based on the profiling's result
- Build with `-g -fno-omit-frame-pointer`
  - Sometimes `-O0` is needed if you want to see a proper call-graph
- `perf report` uses `cycles` by default, however you given specify an event with `-e`

SPECTRAL:
TECHNOLOGIES

Example

```
> clang++-15 -std=c++17 -O0 -g -fno-omit-frame-pointer bin.cpp -o bin
>perf record -g ./bin
took 15.9982
[ perf record: Woken up 3 times to write data ]
[ perf record: Captured and wrote 0.535 MB perf.data (5521 samples) ]
> perf report
```

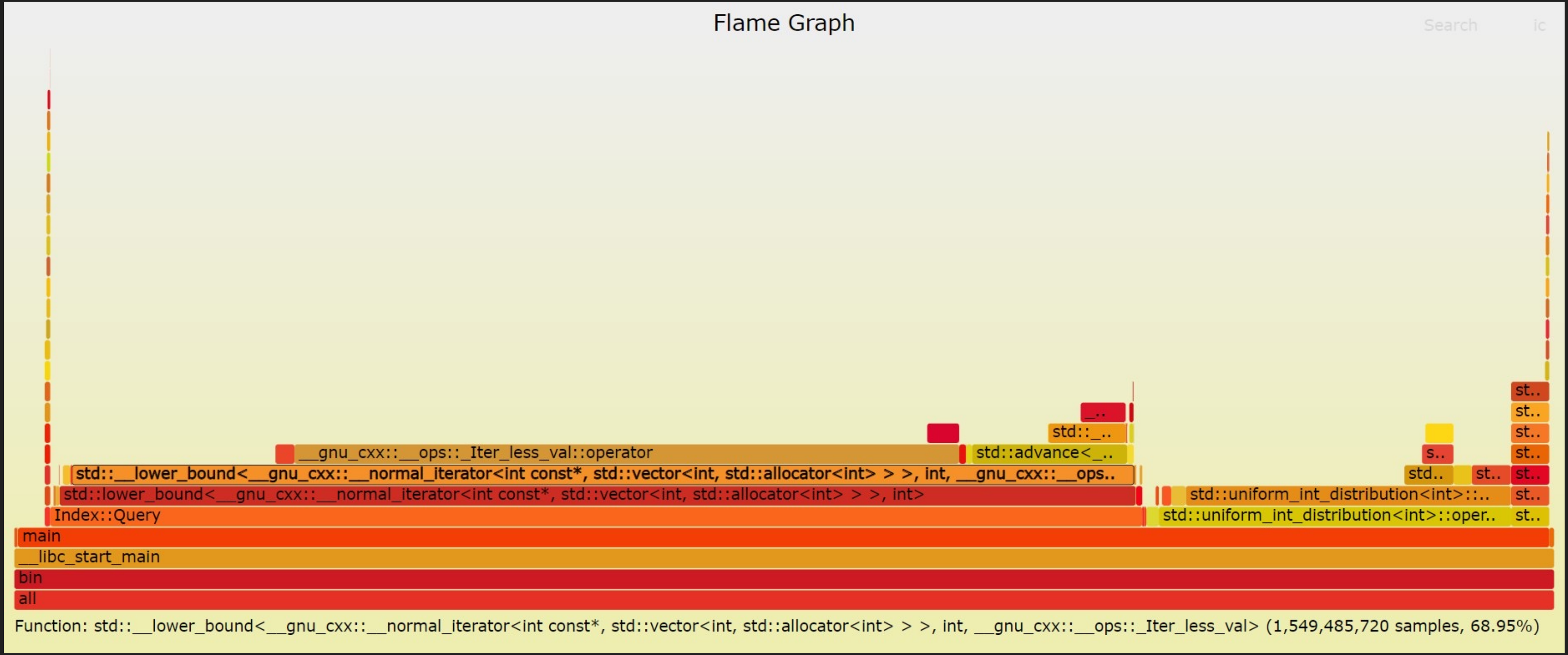


```
Samples: 5K of event 'cycles:u', Event count (approx.): 2247109875
  Children      Self  Command  Shared Object      Symbol
+  100.00%     0.00%  bin      libc-2.31.so       [.] __libc_start_main
-   99.45%     1.74%  bin      bin                [.] main
   - 97.71% main
      - 70.92% Index::Query
         - 69.90% std::lower_bound<__gnu_cxx::__normal_iterator<int const*, std::vector<int, std::allocator<int> > >, int>
            - 68.95% std::__lower_bound<__gnu_cxx::__normal_iterator<int const*, std::vector<int, std::allocator<int> > >, int, __gnu_cxx::__ops::_Iter_les
               + 43.17% __gnu_cxx::__ops::_Iter_less_val::operator()<__gnu_cxx::__normal_iterator<int const*, std::vector<int, std::allocator<int> > >, int
               + 10.06% std::advance<__gnu_cxx::__normal_iterator<int const*, std::vector<int, std::allocator<int> > >, long>
                 1.24% __gnu_cxx::__normal_iterator<int const*, std::vector<int, std::allocator<int> > >::operator++
      + 22.86% std::uniform_int_distribution<int>::operator()<std::linear_congruential_engine<unsigned long, 48271ul, 0ul, 2147483647ul> >
      + 2.48% std::vector<int, std::allocator<int> >::vector
        0.81% __gnu_cxx::operator!=<int*, std::vector<int, std::allocator<int> > >
   + 1.74% __libc_start_main
+   70.92%     0.22%  bin      bin                [.] Index::Query
+   69.90%     0.22%  bin      bin                [.] std::lower_bound<__gnu_cxx::__normal_iterator<int const*, std::vector<int, std::allocator<int> > >, in
+   68.95%    13.20%  bin      bin                [.] std::__lower_bound<__gnu_cxx::__normal_iterator<int const*, std::vector<int, std::allocator<int> > >,
```

SPECTRAL:
TECHNOLOGIES

Flamegraph

- Allows to view `perf report` in a more convinient and graphic way
  - https://github.com/brendangregg/FlameGraph



Function: std::__lower_bound<__gnu_cxx::__normal_iterator<int const*, std::vector<int, std::allocator<int> > >, int, __gnu_cxx::__ops::_Iter_less_val> (1,549,485,720 samples, 68.95%)

SPECTRAL:
TECHNOLOGIES

perf syscall

– perf itself uses `perf_event_open` syscall

```
int syscall(SYS_perf_event_open, struct perf_event_attr *attr,
pid_t pid, int cpu, int group_fd, unsigned long flags);
```

– Allows to analyze some block of code, not the entire program
– https://man7.org/linux/man-pages/man2/perf_event_open.2.html