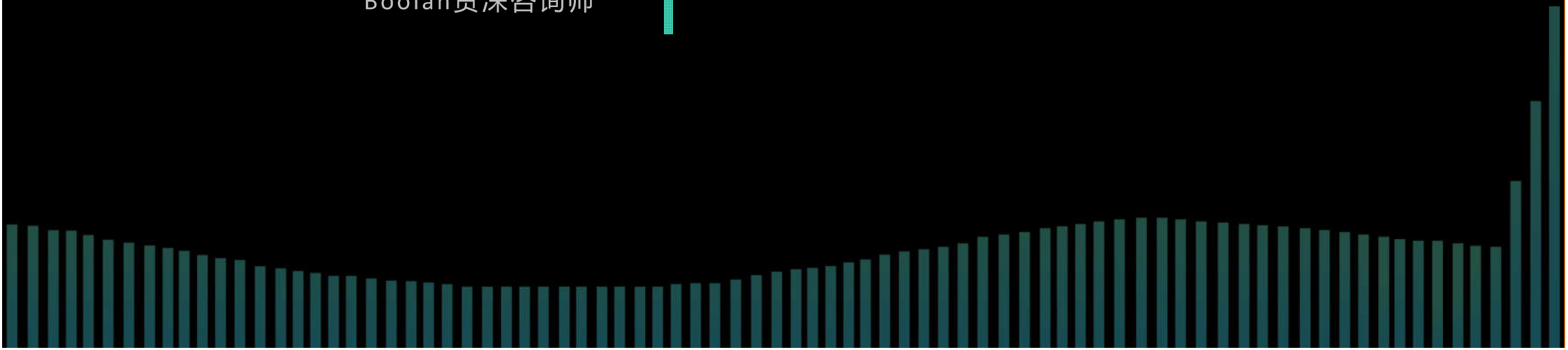
C++ Summit 2020

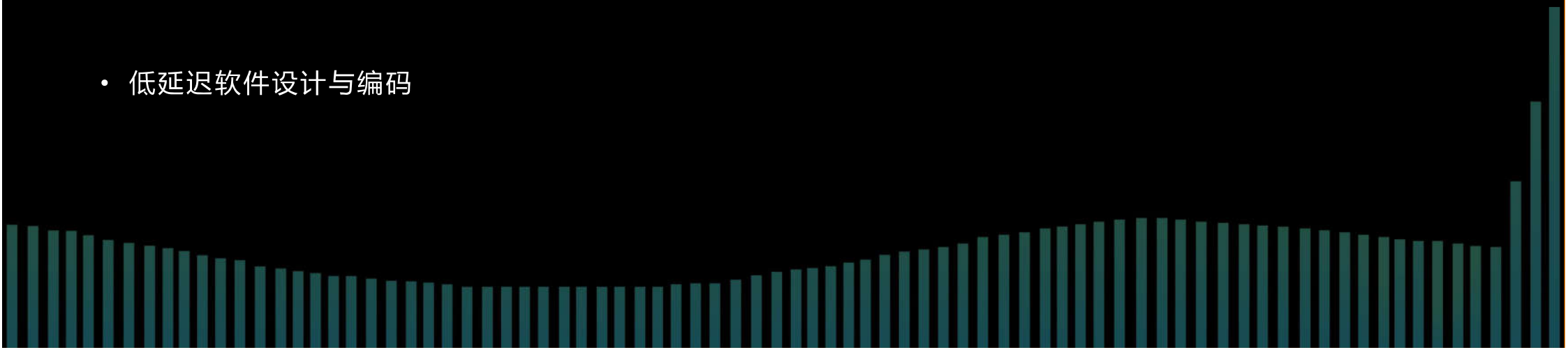
低延迟场景下的性能优化实践

冉 昕
Boolan资深咨询师



Agenda

- 低延迟概述
- 低延迟系统调整
- 低延迟系统编译选项
- 低延迟软件设计与编码



低延迟场景

- 低延迟是第一需求
- 不追求吞吐量
- 不在意资源利用率
- 资源超配
- 案例：快速交易系统

低延迟优化特点

常用性能优化

- 压力测试
- 系统负载
  - CPU使用率
  - 内存占用
  - iowait
- Profile工具找出程序热点
- 优化热点

低延迟性能优化

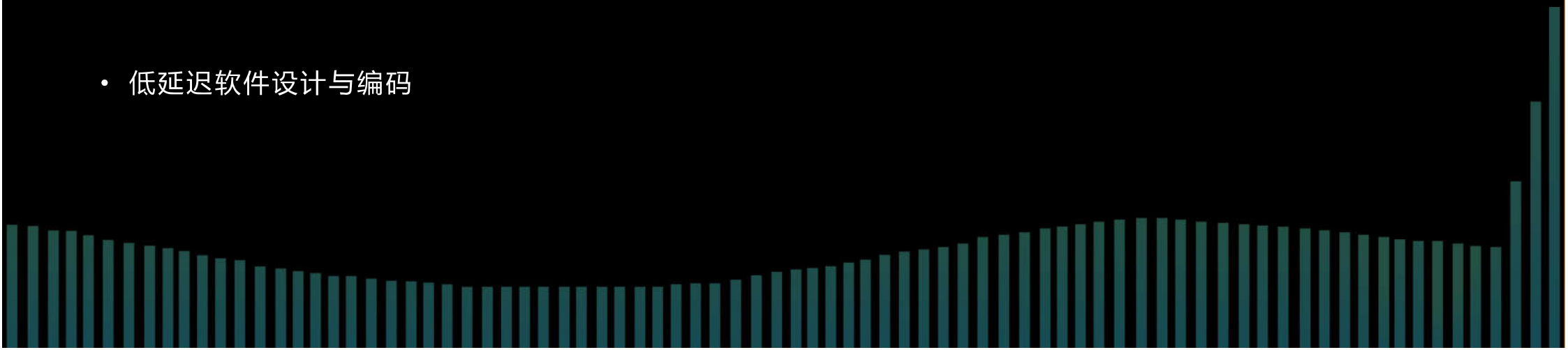
- 系统、设计、编码需要提前考虑低延迟
- 提前规划好critical path
- 测试各单元延迟
- 优化critical path

常见操作时延

Operation	cpu cycle
Add, Sub, And, Or	< 1
memory write	≈ 1
“right" branch of "if"	≈ 1
Mul	3 - 6
L1 read	4
L2 read	10 - 12
"wrong" branch of "if"	10 - 20
L3 read	30 - 50
Div	20 - 100
Function call	25 - 250
Addition polymorphic function call	20 - 30
Mutex lock/unlock	50+
Main RAM read	100 - 150
NUMA: different socket L3 read	100 - 200
NUMA: different socket RAM read	200 - 300
Allocation deallocation pair	200+
User to kernel switch and back	300 - 500
Exception throw + caught	1000 - 2000
Context switch (direct cost)	2000
Context switch (total costs, including cache invalidation)	10K - 1M
Disk read	400K+

Agenda

- 低延迟概述
- 低延迟系统调整
- 低延迟系统编译选项
- 低延迟软件设计与编码



硬件 & 系统

- 物理单机非集群，机器超配：
  - 单核频率高，核数有最低要求
  - X64处理器，执行效率越高越好，不需要虚拟化功能
  - 内存充足
  - NVMe SSD, Optane SSD
  - 低延迟网卡
  - 超频服务器
- 超线程
- 64位linux
  - 最小化安装
  - toolchain升级
  - Rtkernel
  - Tunning with vendor guide

CPU相关优化

- Critical 线程 vs 普通线程
- 根据数量isolate core

  /boot/grub2/grub.cfg

  menuentry 'CentOS Linux (3.10.0-1160.6.1.el7.x86_64) 7 (Core)' --class centos ...

  ……

  linux16 /boot/vmlinuz-3.10.0-1127.13.1.el7.x86_64 root=UUID=…… **isolcpus=3-7 nohz_full=3-7 rcu_nocbs=3-7**
- Critical thread core binding

  sched_setaffinity
- scheduler
  - critical thread: FIFO

    sched_setscheduler
  - normal thread: default(CFS)

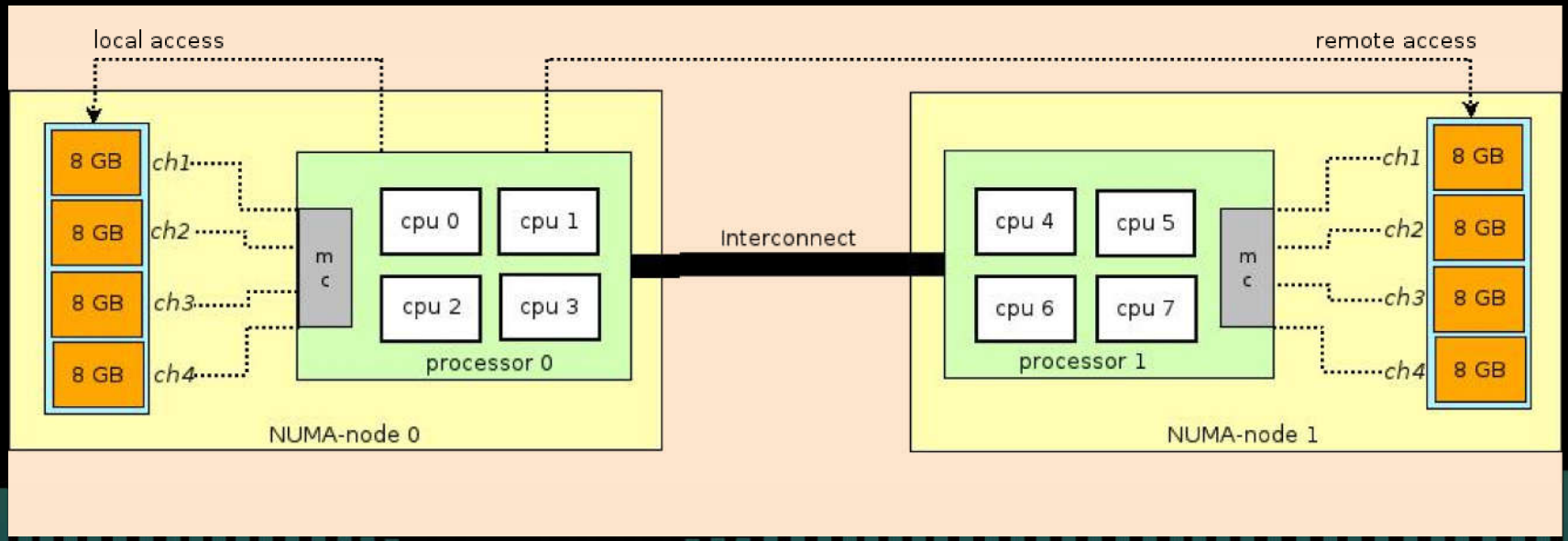
NUMA

- 考虑NUMA

  lscpu

  NUMA node0 CPU(s): 0-3

  NUMA node1 CPU(s): 4-7



*Picture quoted from Boost.Fiber doc

中断

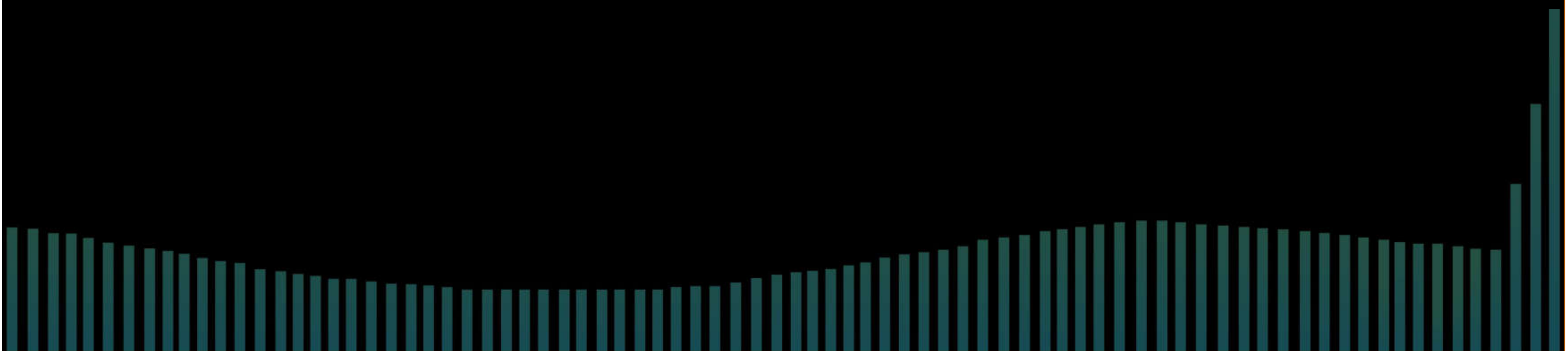
- irqbalance
  - systemctl disable irqbalance.service

- 设置中断affinity到非isolate核心
  - cat /proc/interrupts
    - network 及其他中断　/proc/irq/*/smp_affinity_list
    - LOC　nohz_full
    - RES　rcu_nocbs
    - workqueue　/sys/devices/virtual/workqueue/*/cpumask

中断

- 网卡绑定numa

```
lspci | grep -i "eth"
02:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
cat /sys/bus/pci/devices/0000\:02\:00.0/numa_node
0
```

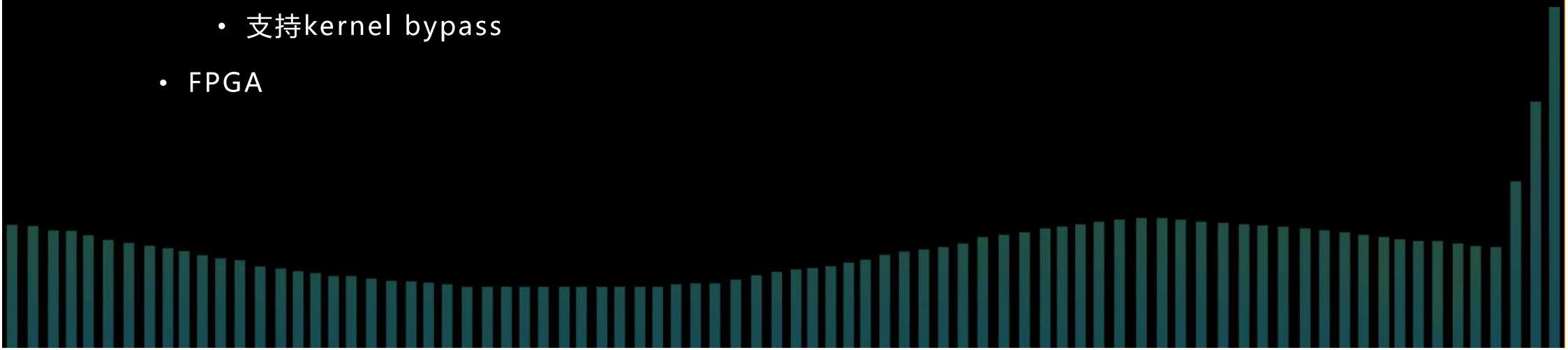


内存

- major fault vs minor fault
- 禁用swap
- mlock
- huge page
  - TLB
- NUMA
  - localalloc
- prefault
- 内存管理器（ptmalloc, tcmalloc, jemalloc）

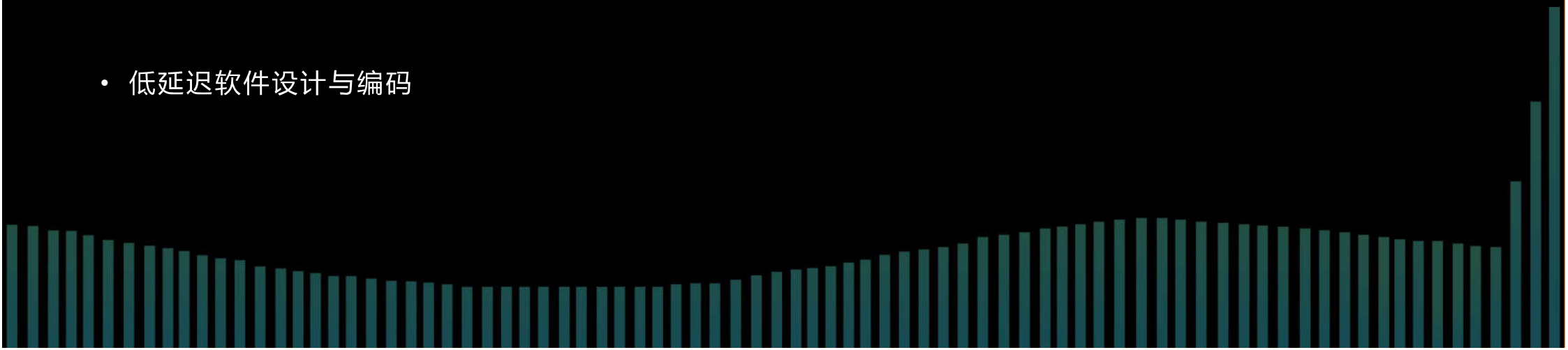
网络

- UDP
- TCP，关闭Nagle和延迟确认
- 带宽越大越好
- DPDK, infiniband, RoCE
- 低延迟网卡
  - 支持kernel bypass
- FPGA



Agenda

- 低延迟概述
- 低延迟系统调整
- 低延迟系统编译选项
- 低延迟软件设计与编码



编译器选择

- gcc, clang, icc
- O2 vs O3
  - -finline-functions (included in O2 since gcc10)
  - -floop-interchange
  - -funswitch-loops
  - -ftree-loop-distribution
  - -ftree-loop-distribute-patterns (included in O2 since gcc10)
  - -ftree-loop-vectorize, -ftree-slp-vectorize (clang O2)
  - -floop-unroll-and-jam
  - -fipa-cp-clone
  - void __attribute__((optimize("O3"))) foo() { // .. }
  - #pragma GCC optimize ("O3")

-floop-interchange

```
for (int j = 0; j < 1024; j++) {
    for(int i = 0; i < 1024; i++) {
        a[i][j] = i * j;
    }
}
```

```
for (int i = 0; i < 1024; i++) {
    for(int j = 0; j < 1024; j++) {
        a[j][i] = i * j;
    }
}
```

```
for (int i = 0; i < 1024; i++) {
    for(int j = 0; j < 1024; j++) {
        a[i][j] = i * j;
    }
}
```

-funswitch-loops

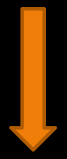
```
for (int i = 0; i < 1024 * 1024; ++i) {
    if (a > 0) {
        result += foo();
    }
    else {
        result += bar();
    }
}
```

```
if (a > 0) {
    for (int i = 0; i < 1024 * 1024; ++i) {
        result += foo();
    }
}
else {
    for (int i = 0; i < 1024 * 1024; ++i) {
        result += bar();
    }
}
```

loop distribution

```
int a[length];
int b[length];

for (int i = 0; i < length; ++i) {
    a[i] = b[i];
    b[i] = 0;
}
```

```
int a[length];
int b[length];

for (int i = 0; i < length; ++i) {
    a[i] = b[i];
}

for (int i = 0; i < length; ++i) {
    b[i] = 0;
}
```



```
int a[length];
int b[length];

memcpy(a, b, length);
memset(a, 0, length);
```

-ftree-loop-vectorize

Why is processing a sorted array faster than processing an unsorted array?

gcc optimization flag -O3 makes code slower than -O2

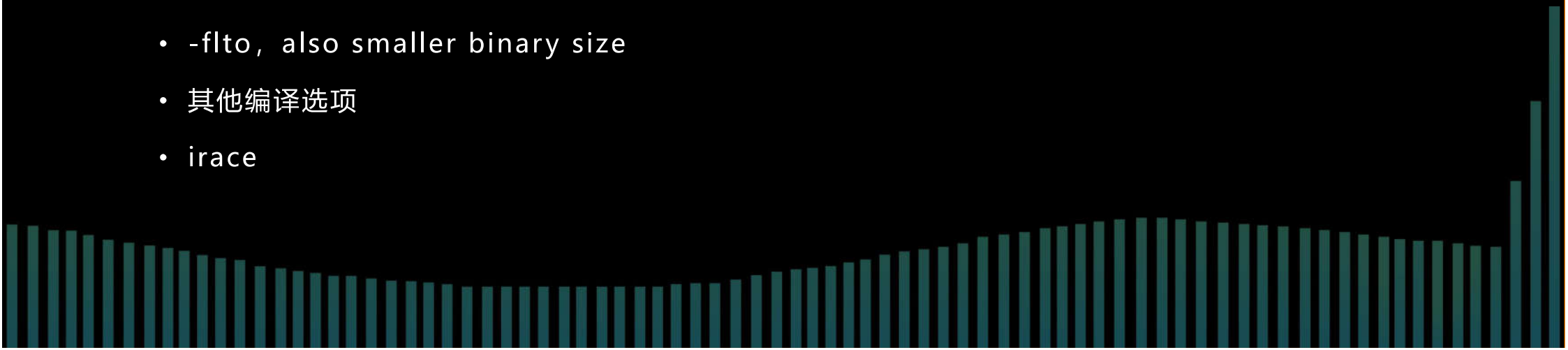
- gcc O1 & O2: cmov
- clang O1: cmov
- clang O2: sse2
- gcc –ftree-loop-vectorize: sse2
- -march

```
const unsigned arraySize = 32768;
int data[arraySize];

for (unsigned c = 0; c < arraySize; ++c)
   data[c] = std::rand() % 256;

// std::sort(data, data + arraySize);

clock_t start = clock();
long long sum = 0;

for (unsigned i = 0; i < 100000; ++i) {
   for (unsigned c = 0; c < arraySize; ++c) {
      if (data[c] >= 128)
         sum += data[c]; ⟶ sum += data[c] + data[c];
   }
}
double elapsedTime =
         static_cast<double>(clock() - start) / CLOCKS_PER_SEC;
```

编译选项

- O3 vs Ofast
  - -ffast-math
- Profile-Guided Optimisations
  - -fprofile-generatge, -fprofile-use & –fprofile-correction
  - -funroll-loops (clang O2)

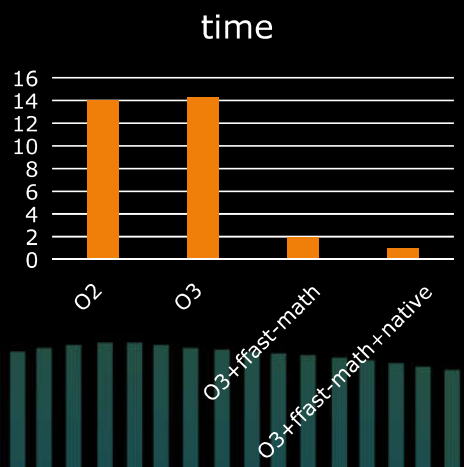
    #pragma GCC unroll n
- -march=native
- -flto，also smaller binary size
- 其他编译选项
- irace



loop-vectorize

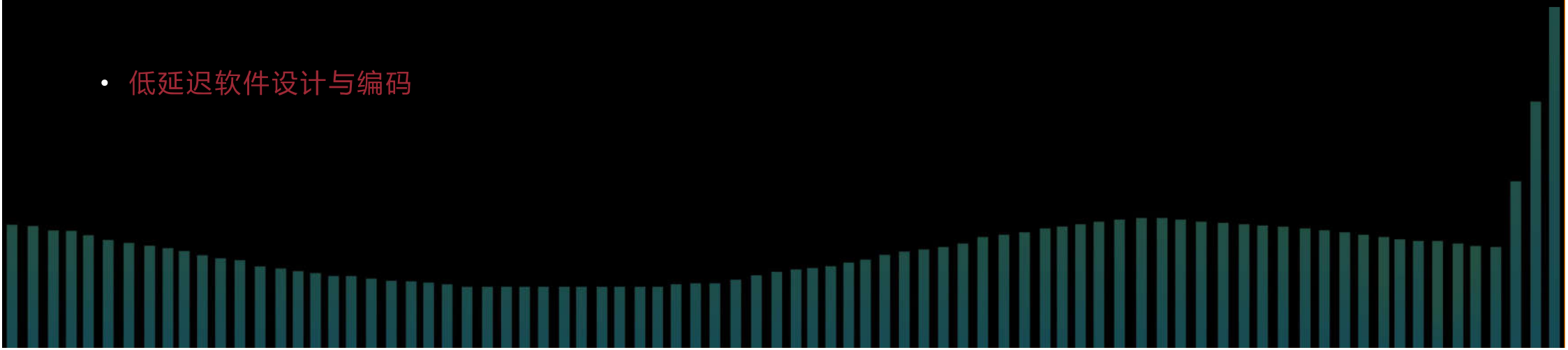
```
double data[arraySize];

for (unsigned c = 0; c < arraySize; ++c)
   data[c] = std::rand() % 256;

// std::sort(data, data + arraySize);

clock_t start = clock();
double sum = 0;

for (unsigned i = 0; i < 100000; ++i) {
   for (unsigned c = 0; c < arraySize; ++c) {
      if (data[c] >= 128)
         sum += data[c] + data[c];
   }
}
```

- floating运算不满足结合律
- -ffast-math
  - -funsafe-math-optimizations
    - -fassociative-math
    - -fno-signed-zeros
    - -fno-trapping-math
- 精度问题



Agenda

- 低延迟概述
- 低延迟系统调整
- 低延迟系统编译选项
- 低延迟软件设计与编码

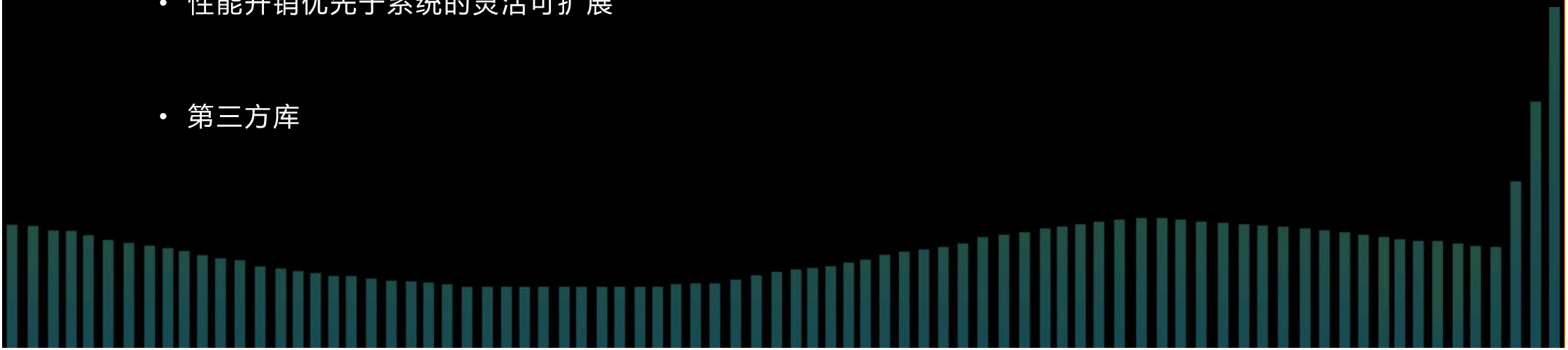


低延迟系统设计与编码

- 单进程多线程 > 多进程
- 提前创建线程
- 线程池
- 静态链接 > 动态链接
- 减少数据拷贝
- 减少数据共享

低延迟系统设计与编码

- 提前计算
- 增量计算
- 尽可能少的间接层
- 性能开销优先于系统的灵活可扩展
- 第三方库



运行时多态 vs 编译时多态

- vptr, vtable
- inline
- CRTP
- Policy based class design
- traits
- if constexpr
- SFINAE/enable_if
- 仅有一个实现 (-fdevirtualize -fdevirtualize-speculatively)
  - 任然有vtable比较
- RTTI

编译时多态

```
class BaseStrategy {
public:
  virtual ~BaseStrategy();
  virtual void OnTick(...);
  ……..

private:
  OMType* _om;
};
```

```
template <typename Derived, template <typename>
typename OMType = OM::OrderManager, bool Critical = true>
class BaseStrategy : public OMType<Derived> {
public:
  void OnTick(...) {
    static_cast<Derived*>(this)->OnTickImpl(...);
  }
  ……..
};
```

```
class ConcreteStrategy : public BaseStrategy {
public:
  void OnTick(...) override;
  ……
};
```

```
class ConcreteStrategy : public BaseStrategy<
ConcreteStrategy> {
public:
  void OnTickImpl(...);
  ……
};
```

编译时多态

```
template <typename T>
concept ControlTraits_ = requires {
  typename T::TraderApi;
  typename T::Spi;
  { T::FTDC_FCC_NotForceClose } -> std::convertible_to<char>;
  { T::FTDC_OPT_LimitPrice } -> std::convertible_to<char>;
   ……
};
template <typename Derived, ControlTraits_ Traits>
class CommonControl {
  CommonControl() {
  if constexpr (std::is_same_v<Traits,
            TradeInterface::Sgit::SgitControlTraits>){
    ……
  }
  typename Traits::TraderApi* _traderApi;
  typename Traits::Spi _spi;
};
```

```
template <typename Derived, template <typename> typename OMType, bool
Critical>
template <bool B>
inline typename std::enable_if_t<B> BaseStrategy<Derived, OMType,
Critical>::AddMessage(const Common::TickMsg& msg){
  _tickQueue.write(msg);
}
```

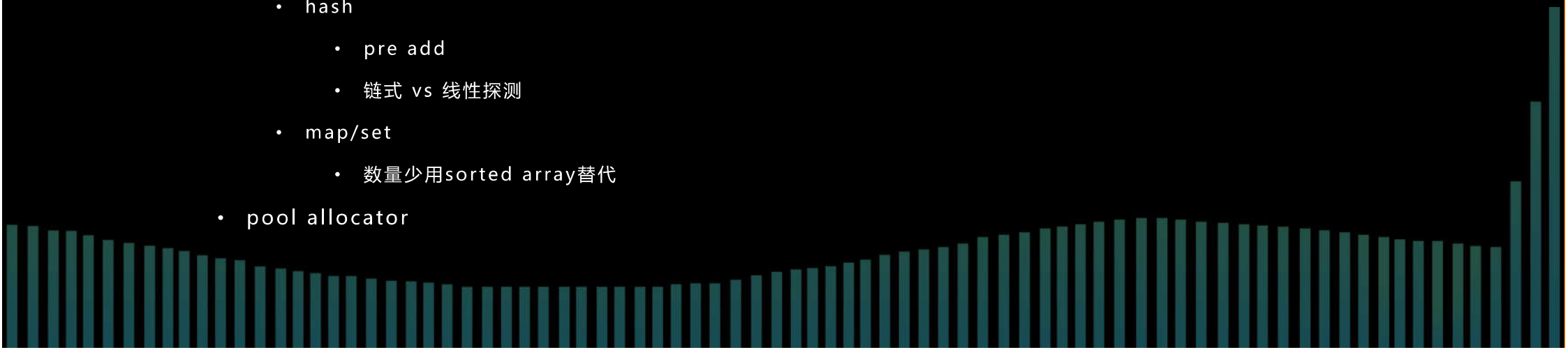
```
template <typename Derived, template <typename> typename OMType, bool
Critical>
template <bool B>
inline typename std::enable_if_t<!B> BaseStrategy<Derived, OMType,
Critical>::AddMessage(const Common::TickMsg& msg){
  _tickQueue.write(msg);
  std::unique_lock lock(_lock);
  _cv.notify_all();
}
```

系统调用 & 日志

- 尽量避免系统调用
  - vdso支持的可以考虑排除
  - strace
- 谨慎打日志
  - 尽量避免打印cache外数据
  - format开销
  - 获取时间开销
    - time
    - clock_gettime
    - rdtsc
      - constant_tsc, nonstop_tsc
  - 低开销日志库
    - 异步打印
    - 离线format

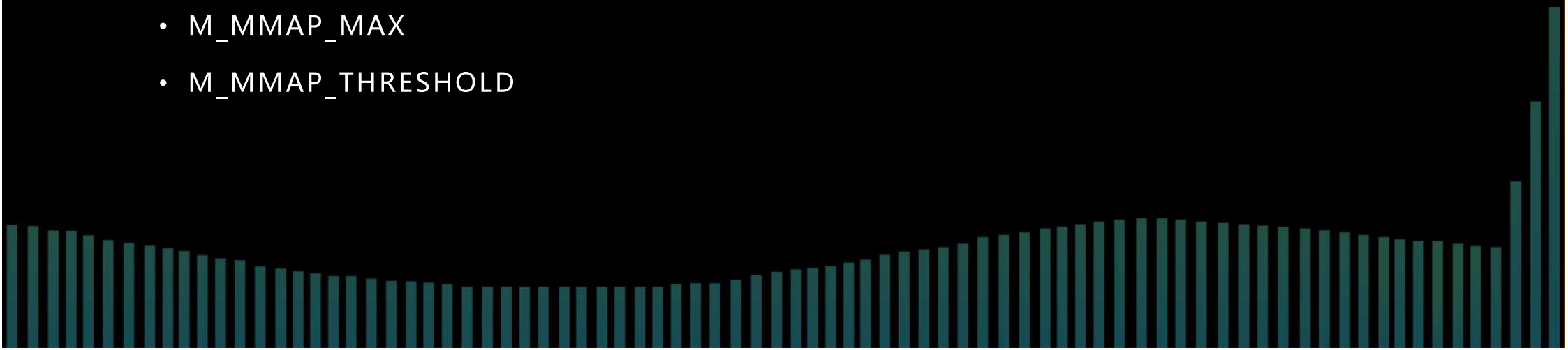
动态内存分配

- 尽量减少动态内存分配
  - 系统调用(可能) + page fault
  - placement new
  - memory pool
  - STL及第三方库带来的内存分配
  - 提前分配内存
    - std::array
    - ring buffer
    - vector
    - hash
      - pre add
      - 链式 vs 线性探测
    - map/set
      - 数量少用sorted array替代
  - pool allocator



ptmalloc 调优

- mallopt
  - M_ARENA_MAX
  - M_ARENA_TEST
  - M_TRIM_THRESHOLD
  - M_MMAP_MAX
  - M_MMAP_THRESHOLD



vector提前分配空间

```
const int num = 1024 * 1024;

std::vector<int> alloc() {
        std::vector<int> vec;
        vec.resize(num);
        for (size_t i = 0; i < vec.size(); i += 1024) {
            vec[i] = 0;
        }
        vec.clear();
        return vec;
}

void use(std::vector<int>& vec) {
        for (size_t i = 0; i < num; ++i) {
            vec.push_back(i);
        }
}
```

```
void* operator new(size_t n) {
        void* p = malloc(n);
        std::cout << "Allocating " << n << " bytes at "
                  << p << "\n";
        return p;
}

void operator delete(void* p) {
        std::cout << "Free " << p << "\n";
        free(p);
}

perf stat -e minor-faults ./test
```

string

- SSO (gcc: 15, clang: 22, msvc: 15)
- char array, string_view

```
std::string string8() {
    return s("12345678");
}
.....
```

```
static void String8(benchmark::State& state) {
  std::string str;
  for (auto _ : state)
    benchmark::DoNotOptimize(str = string8());
}
BENCHMARK(String8);
.....
```

string sso

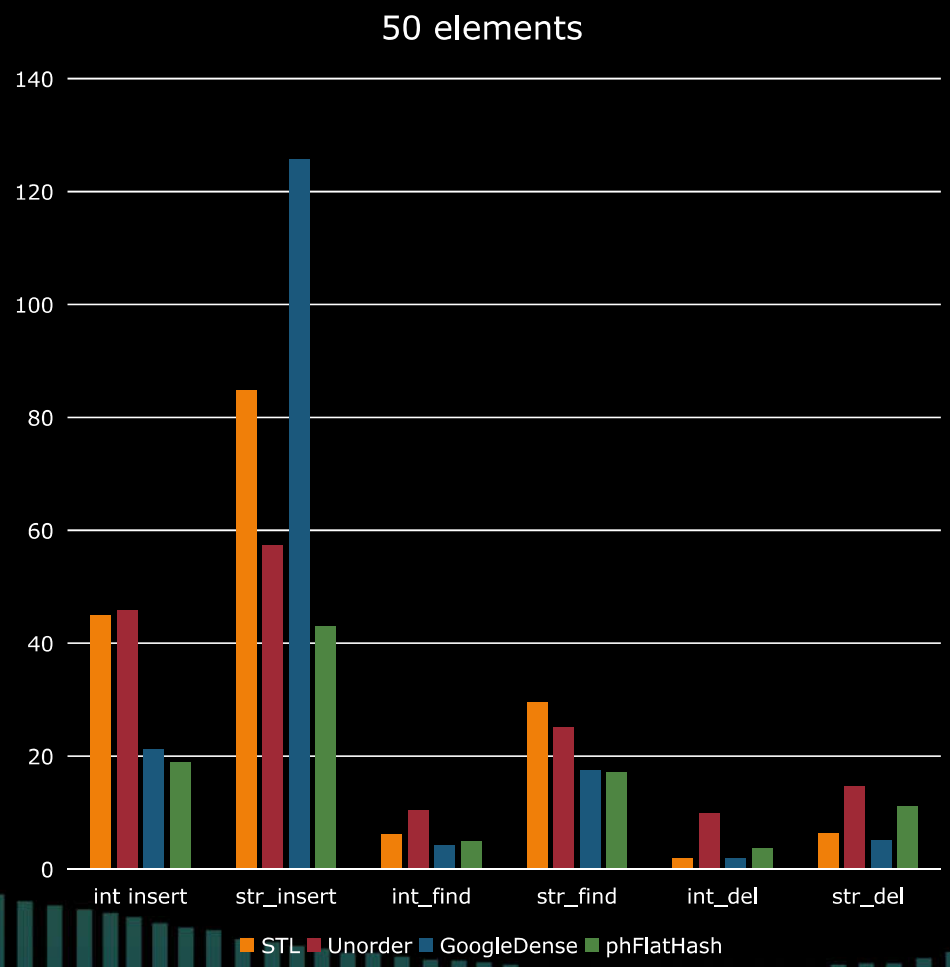
Benchmark	Time	CPU	Iterations
String6	3.45 ns	3.45 ns	202897921
String7	4.07 ns	4.07 ns	172152608
String8	2.82 ns	2.82 ns	248720421
String9	3.47 ns	3.46 ns	201992970
String10	3.55 ns	3.55 ns	197528242
String11	4.10 ns	4.10 ns	170848552
String12	3.45 ns	3.45 ns	203180117
String13	4.09 ns	4.08 ns	171481778
String14	4.07 ns	4.07 ns	171863748
String15	4.69 ns	4.69 ns	149172560
String16	17.6 ns	17.5 ns	39915270
String17	17.7 ns	17.7 ns	39468941

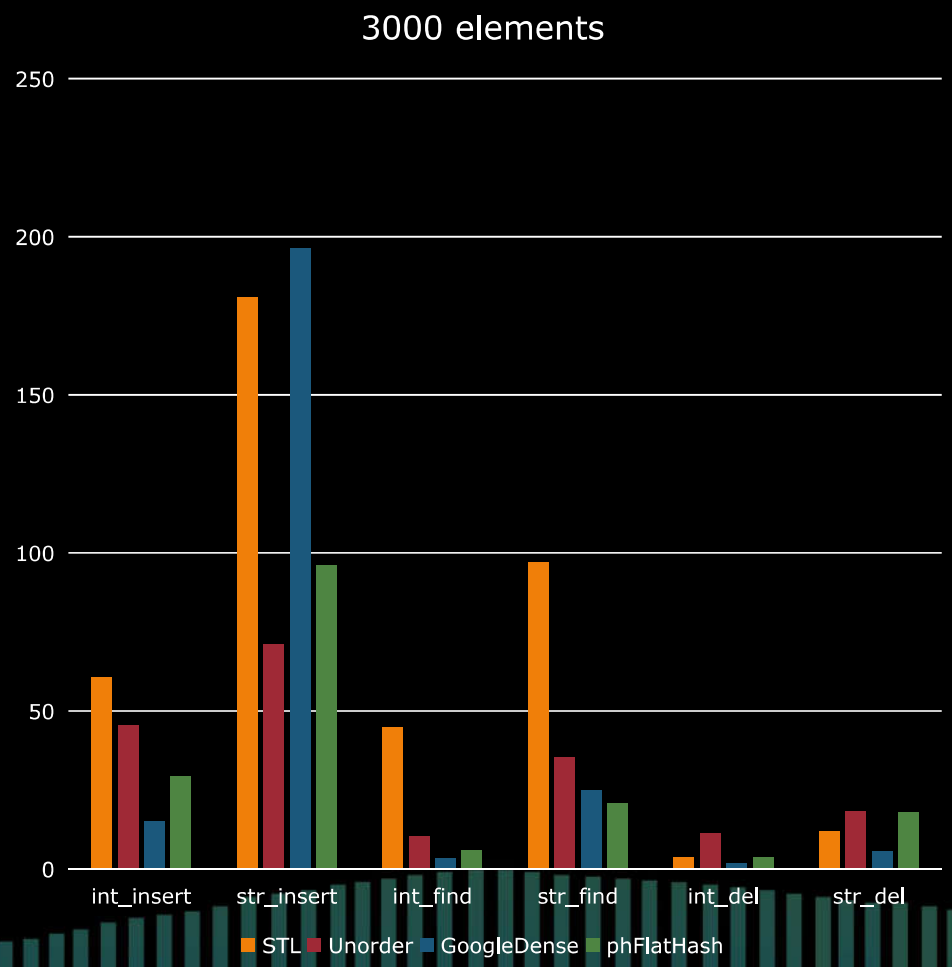
string sso

```
string7[abi:cxx11]():
        leaq    16(%rdi), %rdx
        movb    $55, 22(%rdi)
        movq    %rdi, %rax
        movq    %rdx, (%rdi)
        movl    $13877, %edx
        movl    $875770417, 16(%rdi)
        movw    %dx, 20(%rdi)
        movq    $7, 8(%rdi)
        movb    $0, 23(%rdi)
        ret
string8[abi:cxx11]():
        leaq    16(%rdi), %rdx
        movq    $8, 8(%rdi)
        movq    %rdi, %rax
        movabsq $4050765991979987505, %rcx
        movq    %rdx, (%rdi)
        movq    %rcx, 16(%rdi)
        movb    $0, 24(%rdi)
        ret
string9[abi:cxx11]():
        leaq    16(%rdi), %rdx
        movb    $57, 24(%rdi)
        movq    %rdi, %rax
        movabsq $4050765991979987505, %rcx
        movq    %rdx, (%rdi)
        movq    %rcx, 16(%rdi)
        movq    $9, 8(%rdi)
        movb    $0, 25(%rdi)
        ret
```

```
string16[abi:cxx11]():
        pushq   %r12
        leaq    16(%rdi), %rax
        xorl    %edx, %edx
        movq    %rdi, %r12
        subq    $16, %rsp
        movq    %rax, (%rdi)
        movq    $16, 8(%rsp)
        leaq    8(%rsp), %rsi
        call    std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<ch
ar> >::_M_create(unsigned long&, unsigned long)
        movq    8(%rsp), %rdx
        movdqa  .LC0(%rip), %xmm0
        movq    %rax, (%r12)
        movq    %rdx, 16(%r12)
        movups  %xmm0, (%rax)
        movq    8(%rsp), %rax
        movq    (%r12), %rdx
        movq    %rax, 8(%r12)
        movb    $0, (%rdx,%rax)
        addq    $16, %rsp
        movq    %r12, %rax
        popq    %r12
        ret
```

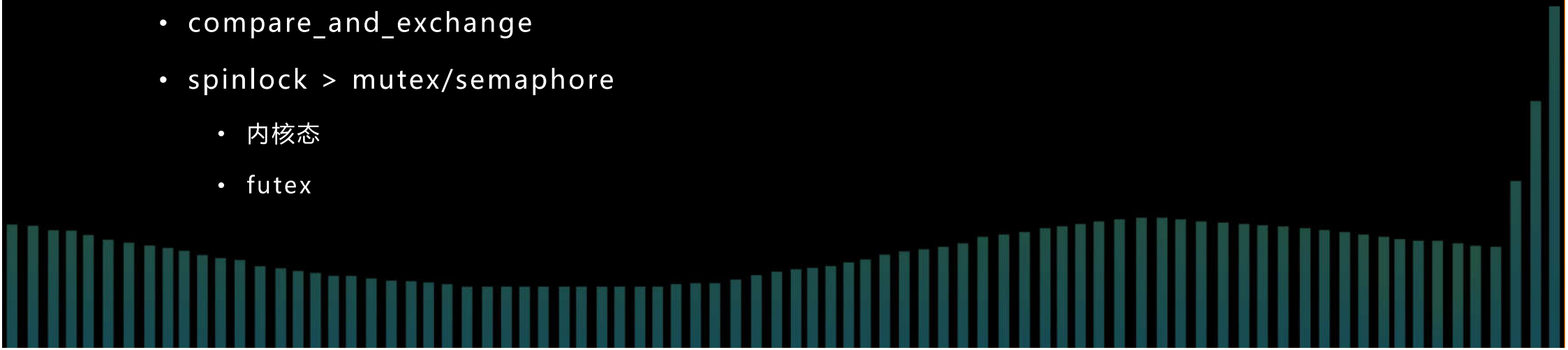
hashmap


50 elements
140
120
100
80
60
40
20
0
int insert
str_insert
int_find
str_find
int_del
str_del
STL
Unorder
GoogleDense
phFlatHash

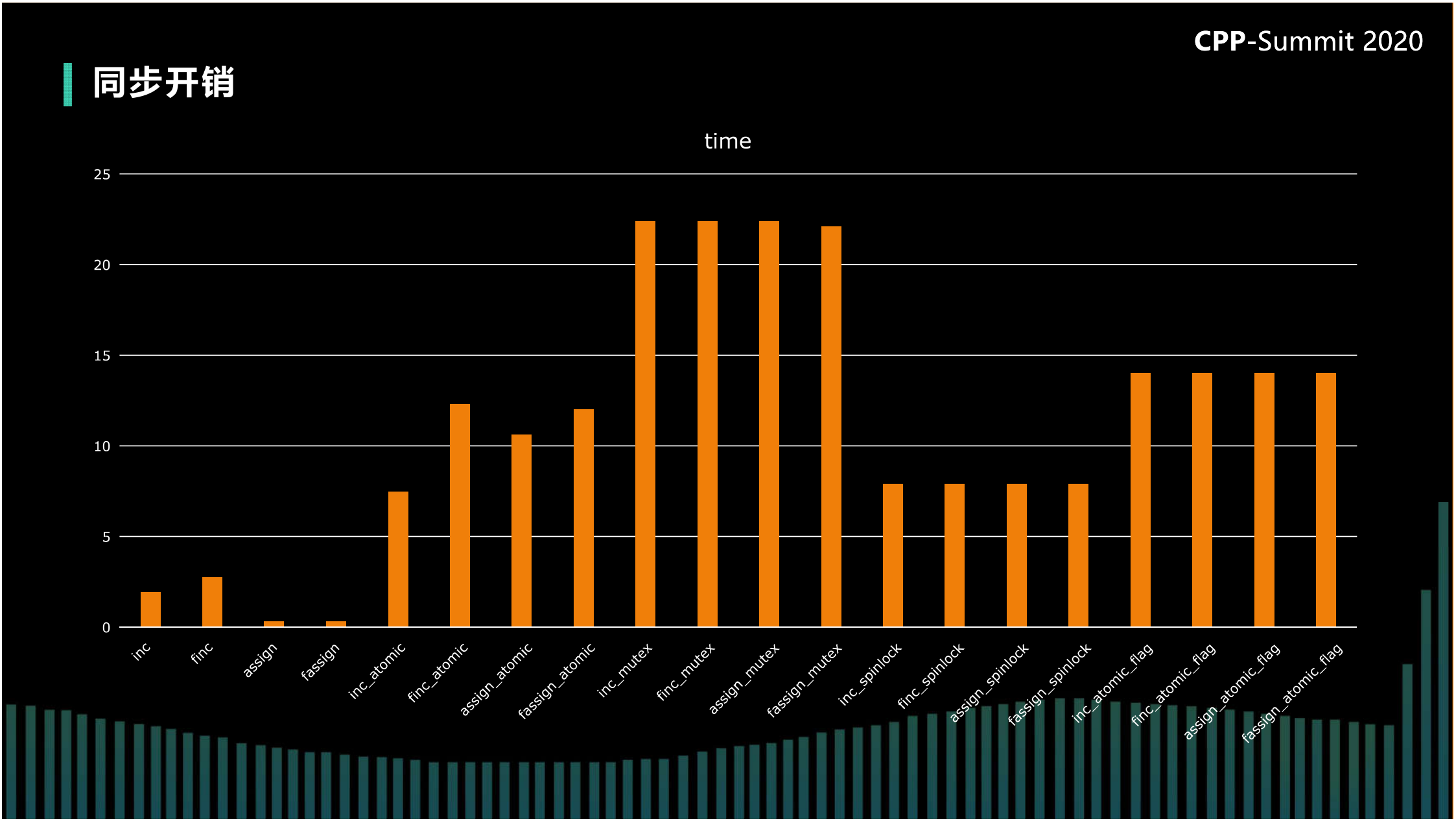

3000 elements
250
200
150
100
50
0
int_insert
str_insert
int_find
str_find
int_del
str_del
STL
Unorder
GoogleDense
phFlatHash

无锁编程

- 消息传递
- lockfree queue
  - spsc
  - mpsc
  - mpmc
- atomic
- atomic_flag
- compare_and_exchange
- spinlock > mutex/semaphore
  - 内核态
  - futex



同步开销


time
25
20
15
10
5
0
inc
finc
assign
fassign
inc_atomic
finc_atomic
assign_atomic
fassign_atomic
inc_mutex
finc_mutex
assign_mutex
fassign_mutex
inc_spinlock
finc_spinlock
assign_spinlock
fassign_spinlock
inc_atomic_flag
finc_atomic_flag
assign_atomic_flag
fassign_atomic_flag

分支处理

- 代码尽可能少用branch
- cmov
- setcc
- sse, avx
- lookup table
- jump table

```
extern void func0();
extern void func1();
// func2, func3, ...

void (*F[])() = { func1, func2, ..... };

void test(int a) {
        switch (a) {
        case 0:
                func0();
                break;
        case 1:
                func1();
                break;
    // ......
}

void test2(int a) {
   F[a]();
}
```

分支处理

```
// probability: condA > condB > condC
if (condA()) {
        return;
}
……..          // sectionA
if (condB()) {
        return;
}
……..          // sectionB
if (condC()) {
        return;
}
……..          // sectionC
……..          // final operation
```

```
if (condC()) {
        return;
}
……..          // sectionC
if (condB()) {
        return;
}
……..          // sectionB
if (condA()) {
        return;
}
……..          // sectionA
……..          // final operation
```

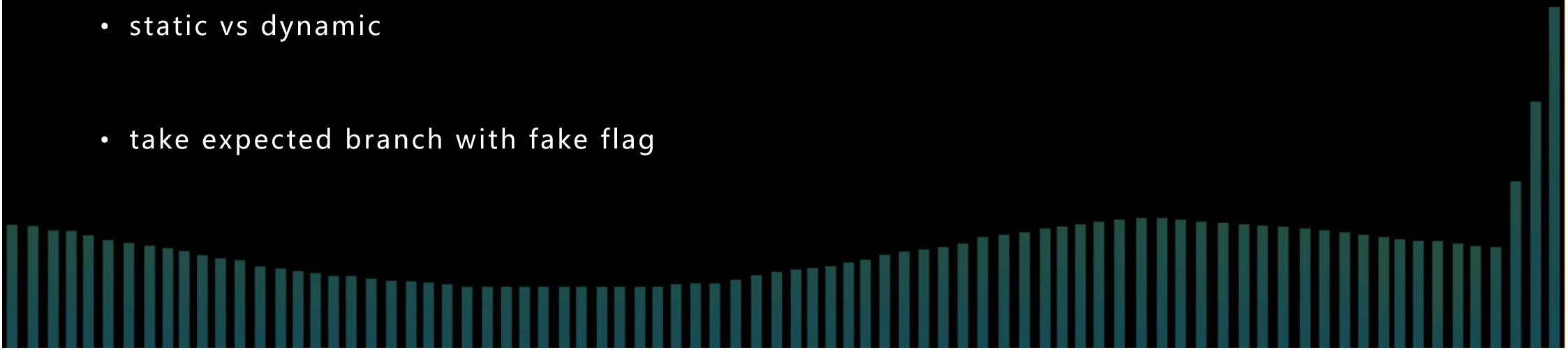
分支处理

```
if (condA()) {
            return;
}
……..                   // sectionA
if (condB()) {
            return;
}
……..                   // sectionB
if (condC()) {
            return;
}
……..                   // sectionC
……..                   // final operation
```

```
……..                   // sectionA
……..                   // sectionB
……..                   // sectionC
if (!(condA() | condB() | condC())) {
……..                   // final operaton
}
```

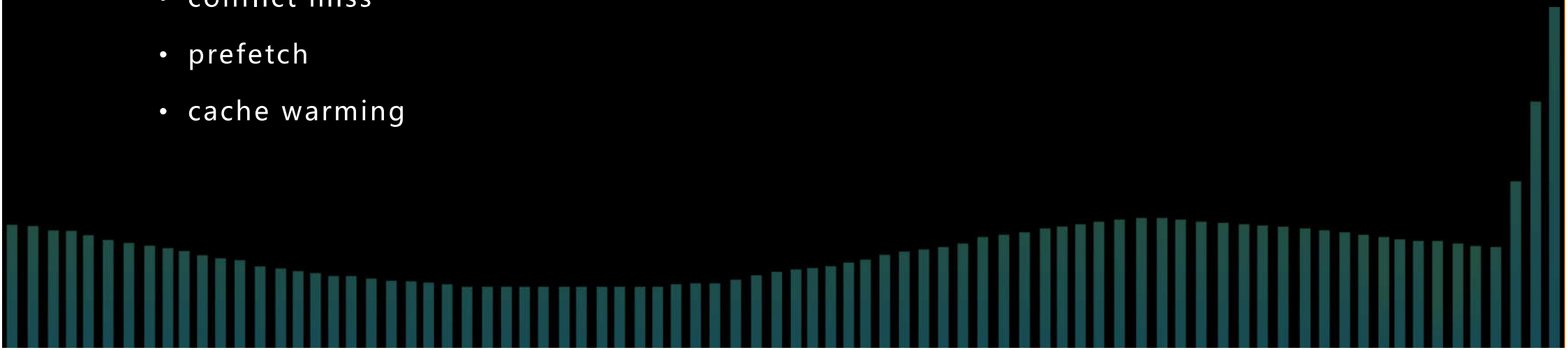
branch prediction

- __builtin_expect(!!(x), 1), __buiiltin_expect(!!(x), 0)
- [[likely]], [[unlikely]]
- reference for compiler
- static vs dynamic
- take expected branch with fake flag



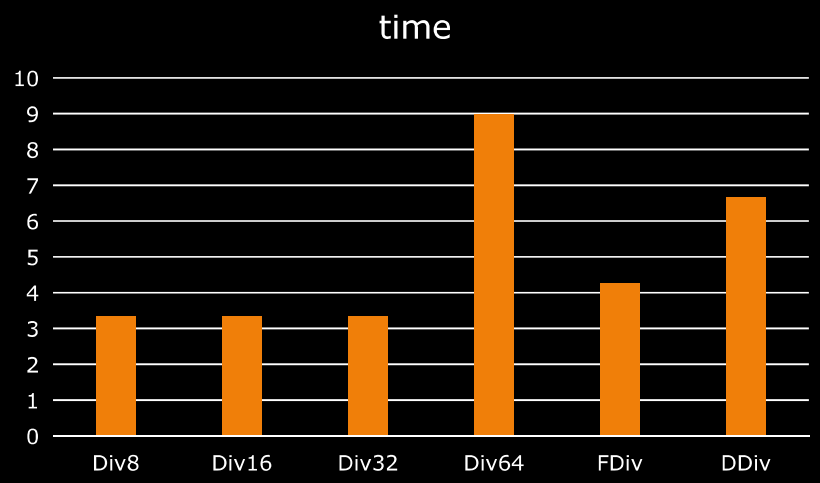
cache优化

- 减少多线程数据写
- alignas(64)
- Padding，避免false sharing
- struct data arrangement
- static，global数据访问
- 代码组织
- conflict miss
- prefetch
- cache warming



运算开销

- Integral除法开销大，尤其是64位
- floating除法优于int64_t
- float vs double
- Packed vs Scalar



类型转换

- signed/unsigned
- Integral: int64_t, long, int, short, char
- float <-> integral
- string <-> integral

```
template <typename F, typename T>
__attribute__((noinline))
T convert(F f)
{
    return static_cast<T>(f);
}
```

```
int convert<signed char, int>(signed char):
        movsbl  %dil, %eax
        ret
int convert<short, int>(short):
        movswl  %di, %eax
        ret
signed char convert<int, signed char>(int):
        movl    %edi, %eax
        ret
short convert<int, short>(int):
        movl    %edi, %eax
        ret
long convert<signed char, long>(signed char):
        movsbq  %dil, %rax
        ret
long convert<short, long>(short):
        movswq  %di, %rax
        ret
long convert<int, long>(int):
        movslq  %edi, %rax
        ret
int convert<long, int>(long):
        movq    %rdi, %rax
        ret
```

类型转换

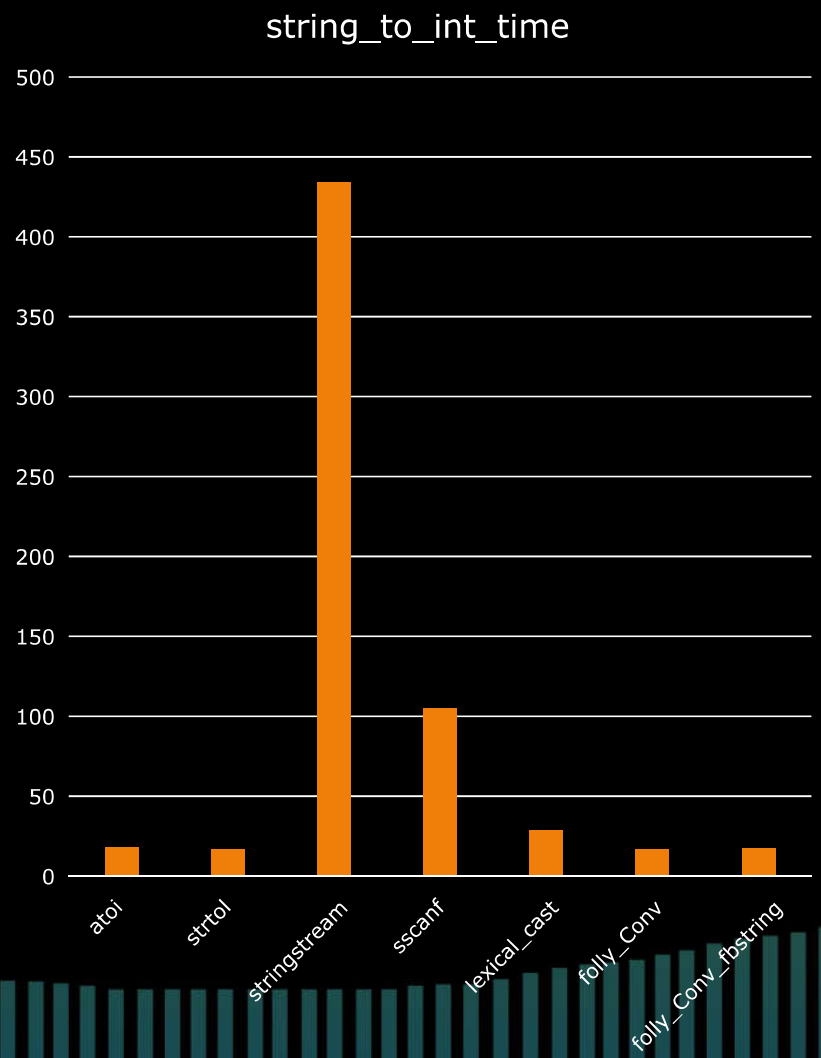
```
int convert<float, int>(float):
        cvttss2sil      %xmm0, %eax
        ret
int convert<double, int>(double):
        cvttsd2sil      %xmm0, %eax
        ret
float convert<int, float>(int):
        pxor    %xmm0, %xmm0
        cvtsi2ssl       %edi, %xmm0
        ret
double convert<int, double>(int):
        pxor    %xmm0, %xmm0
        cvtsi2sdl       %edi, %xmm0
        ret
float convert<long, float>(long):
        pxor    %xmm0, %xmm0
        cvtsi2ssq       %rdi, %xmm0
        ret
double convert<long, double>(long):
        pxor    %xmm0, %xmm0
        cvtsi2sdq       %rdi, %xmm0
        ret
long convert<float, long>(float):
        cvttss2siq      %xmm0, %rax
        ret
long convert<double, long>(double):
        cvttsd2siq      %xmm0, %rax
        ret
```

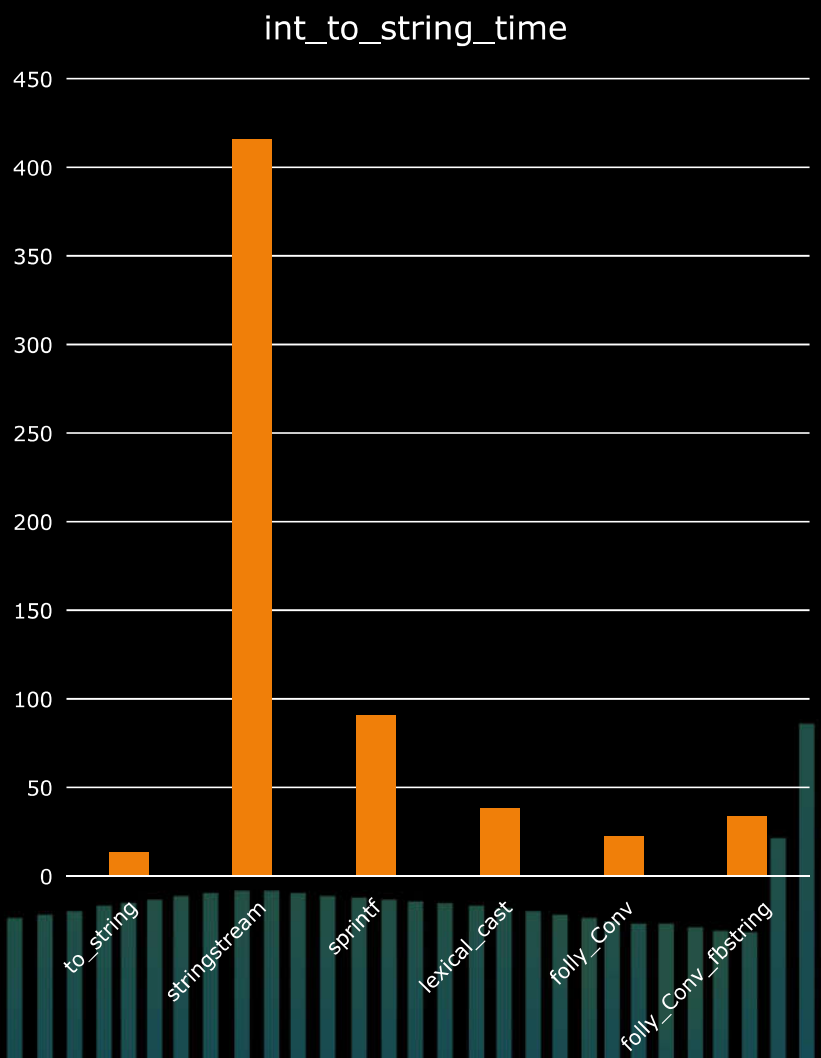
- Intel® Core™ Processor instruction throughput and latency
https://software.intel.com/content/www/us/en/develop/download/10th-generation-intel-core-processor-instruction-throughput-and-latency-docs.html

- Instruction tables
https://www.agner.org/optimize/instruction_tables.ods

类型转换

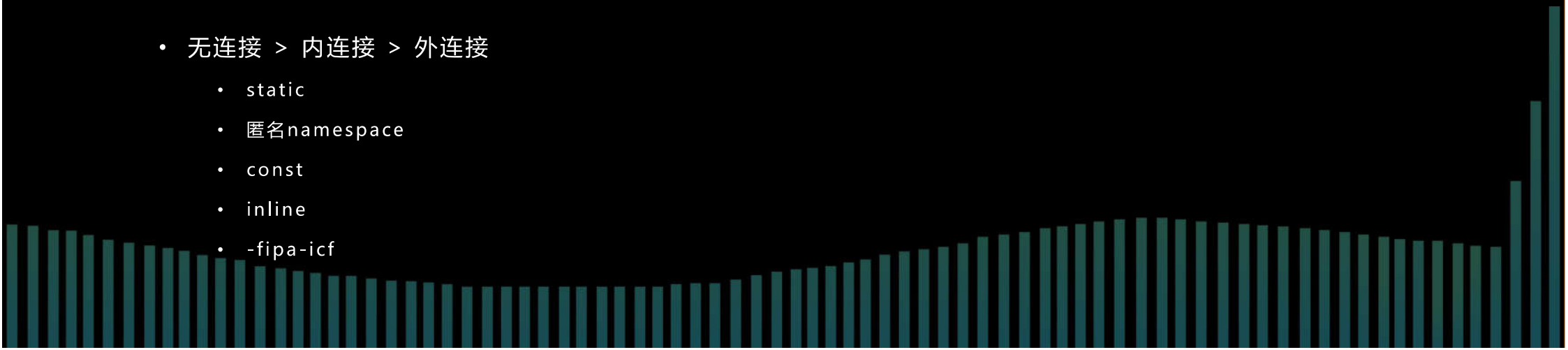
- atoi
- strtol, to_string
- stringsream
- sscanf, sprintf
- lexical_cast
- folly Conv





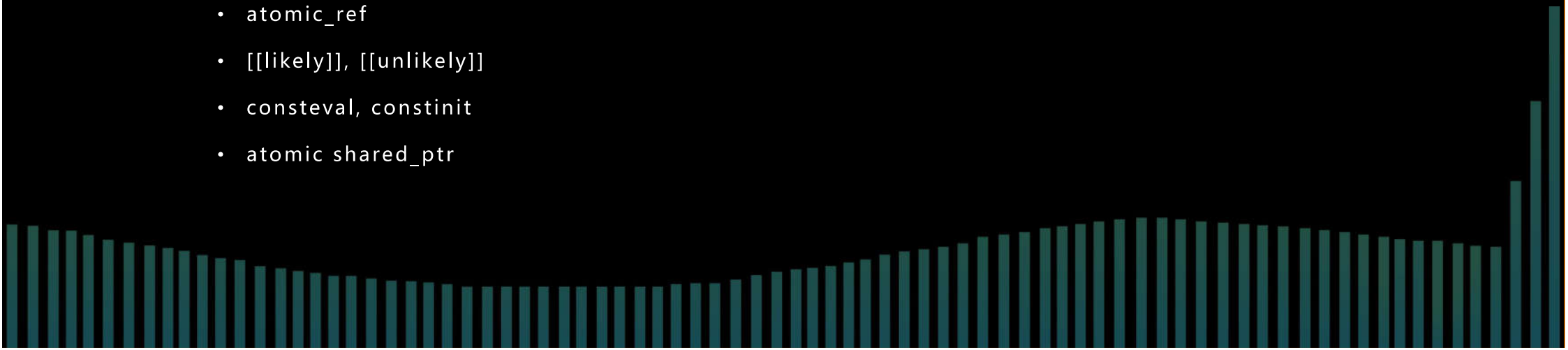
低延迟系统设计与编码

- 注意异常开销
  - 编译器打开异常选项
  - 正常路径不应该用异常
  - 不触发几乎没有开销
  - noexcept
- 作用域尽可能小
- 尽可能使用const
  - 有利于编译器优化
- 无连接 > 内连接 > 外连接
  - static
  - 匿名namespace
  - const
  - inline
  - -fipa-icf



低延迟系统设计与编码

- 智能指针
  - unique_ptr
  - shared_ptr
- C++20
  - atomic float
  - atomic_ref
  - [[likely]], [[unlikely]]
  - consteval, constinit
  - atomic shared_ptr



2020

THANK YOU