Low Latency Programming

Talk Outline

Introduction

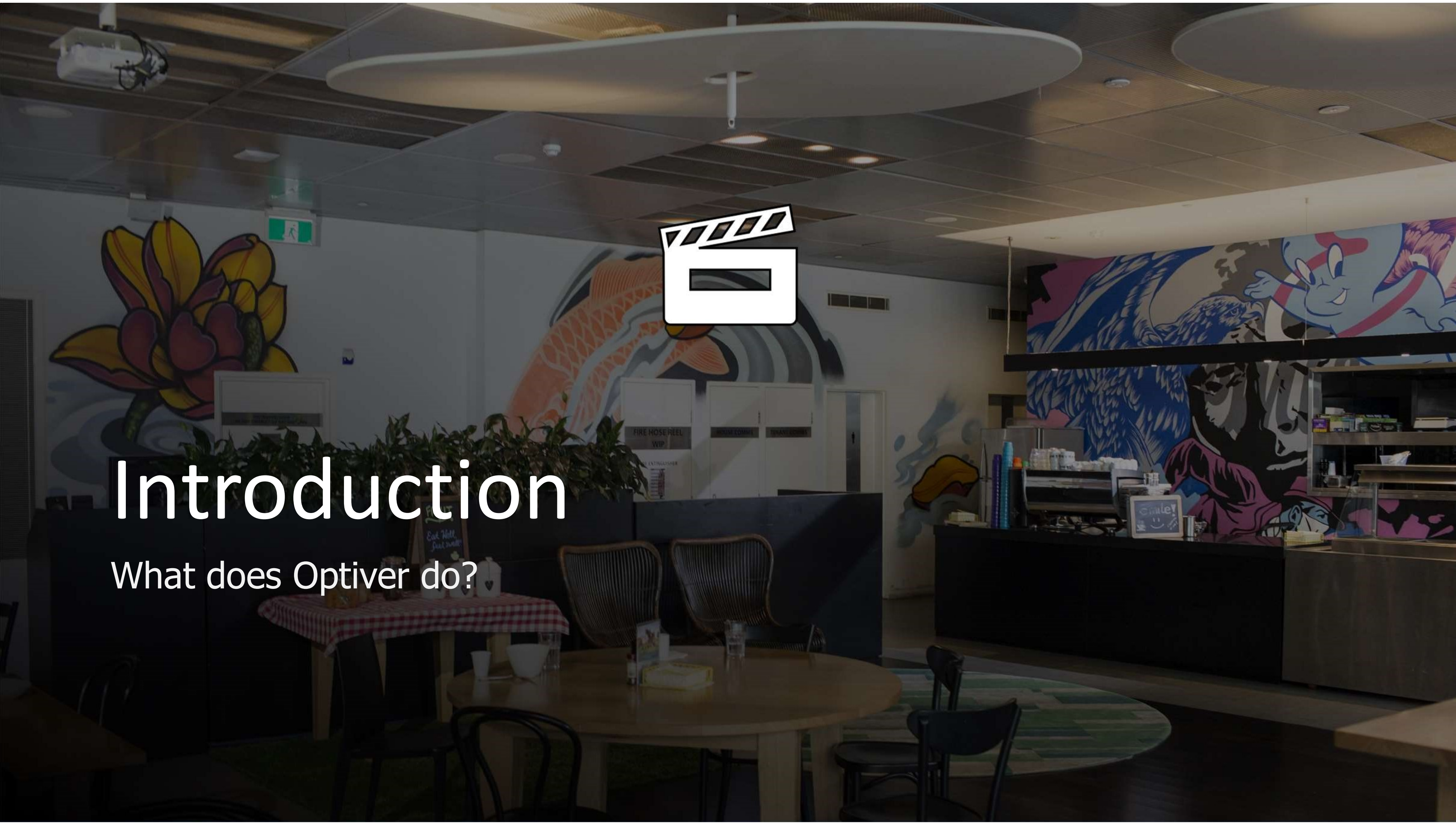
Who does Optiver do?

Improving Latency

The process of improving latency and some useful tools.

Join Us

The application process and new-starter program.



Introduction

What does Optiver do?

The art of Market-Making

market-maker
/ˈmɑkɪt-ˈmeɪkə/

noun

1. A trading party that provides ‘liquidity’ to help others trade.

It’s about estimating the price of a product right *now* instead of in the future.

And offering a lower price than anyone else to buyers

$

And a higher price than anyone else to sellers.

We Improve the Market

Trading

Traders can buy and sell whenever they want to.

Price Discovery

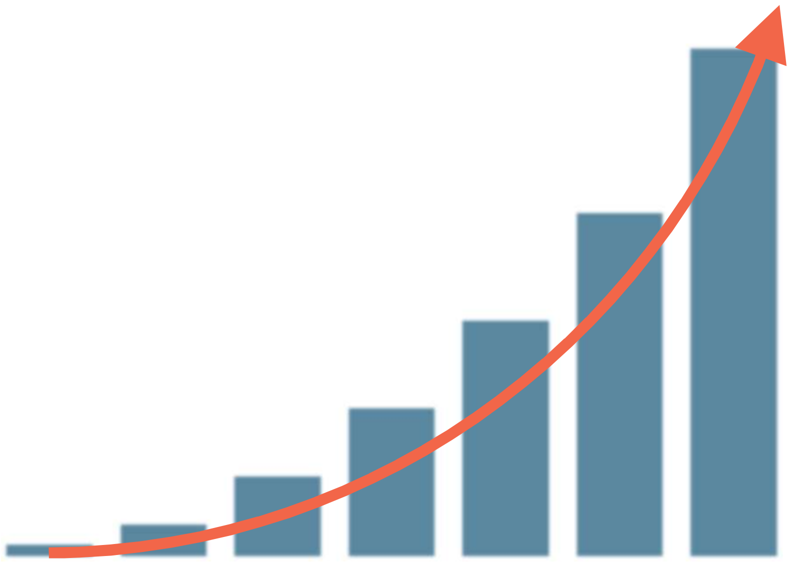
Accurate prices are continuously available.

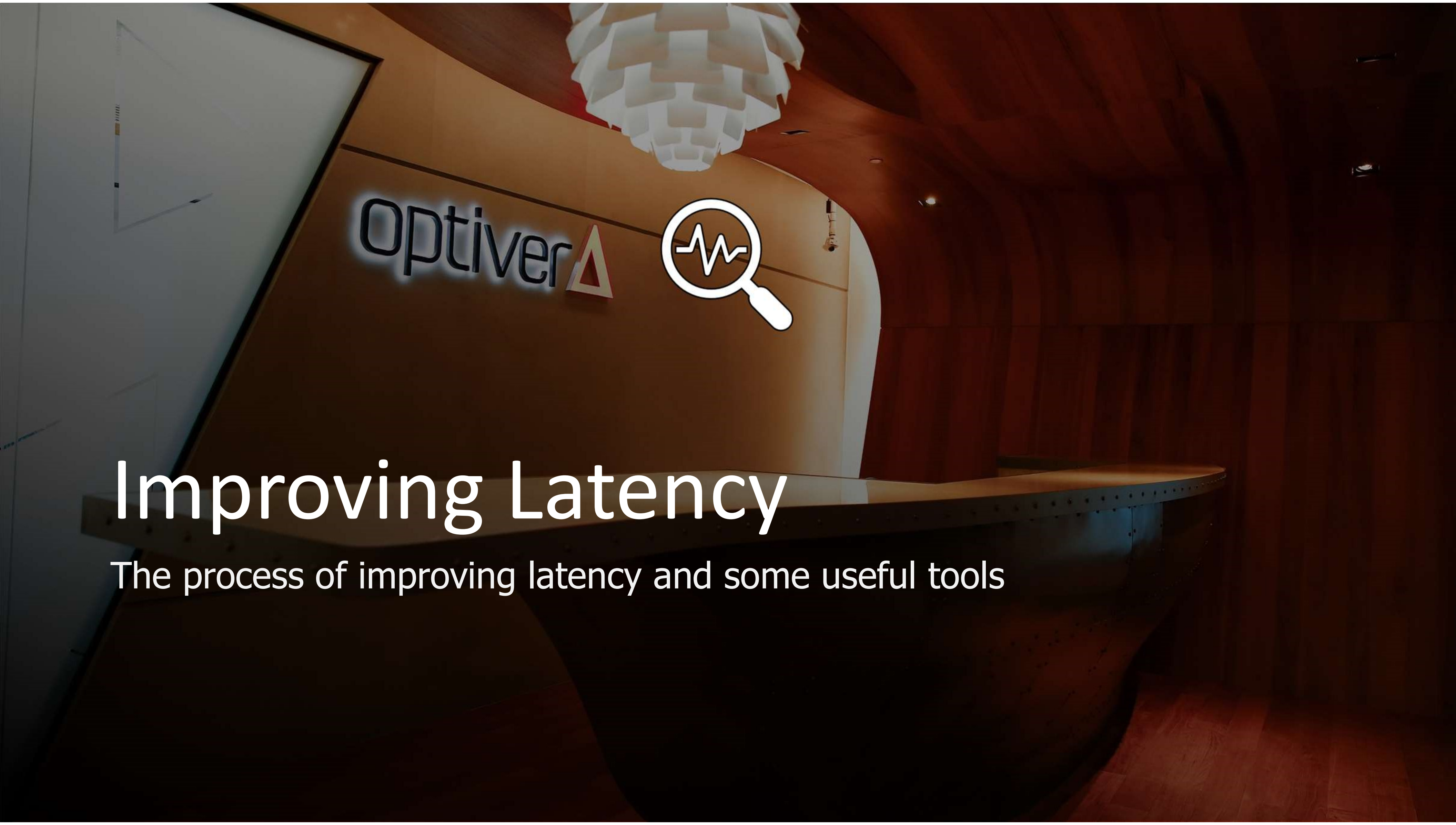
Liquidity

Assets can be readily converted to cash.

Tight Spreads

Lower trading costs.

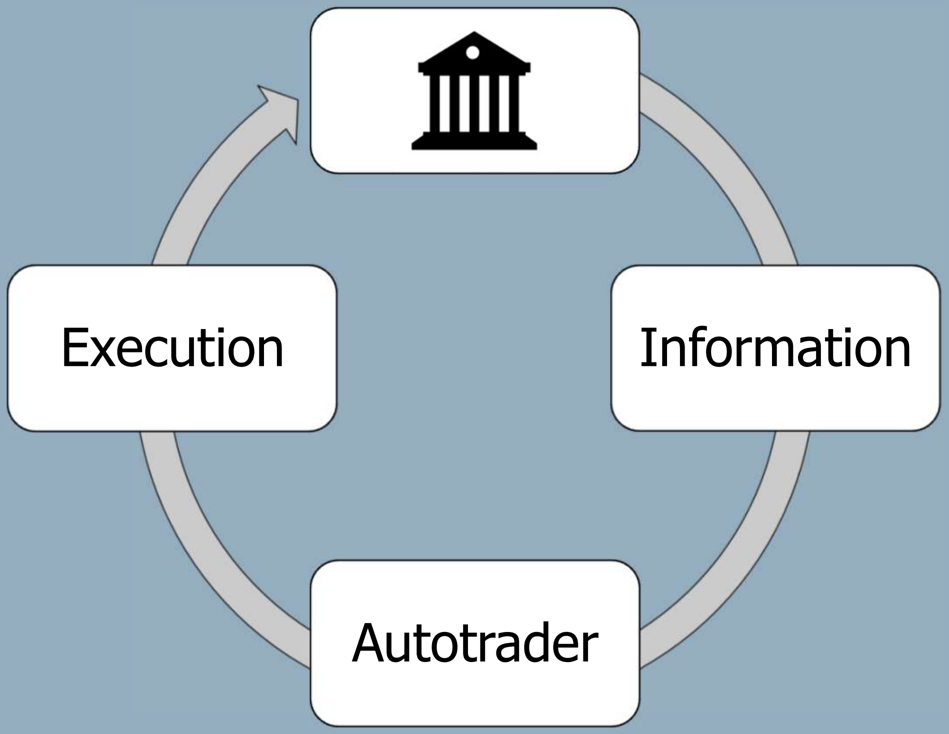




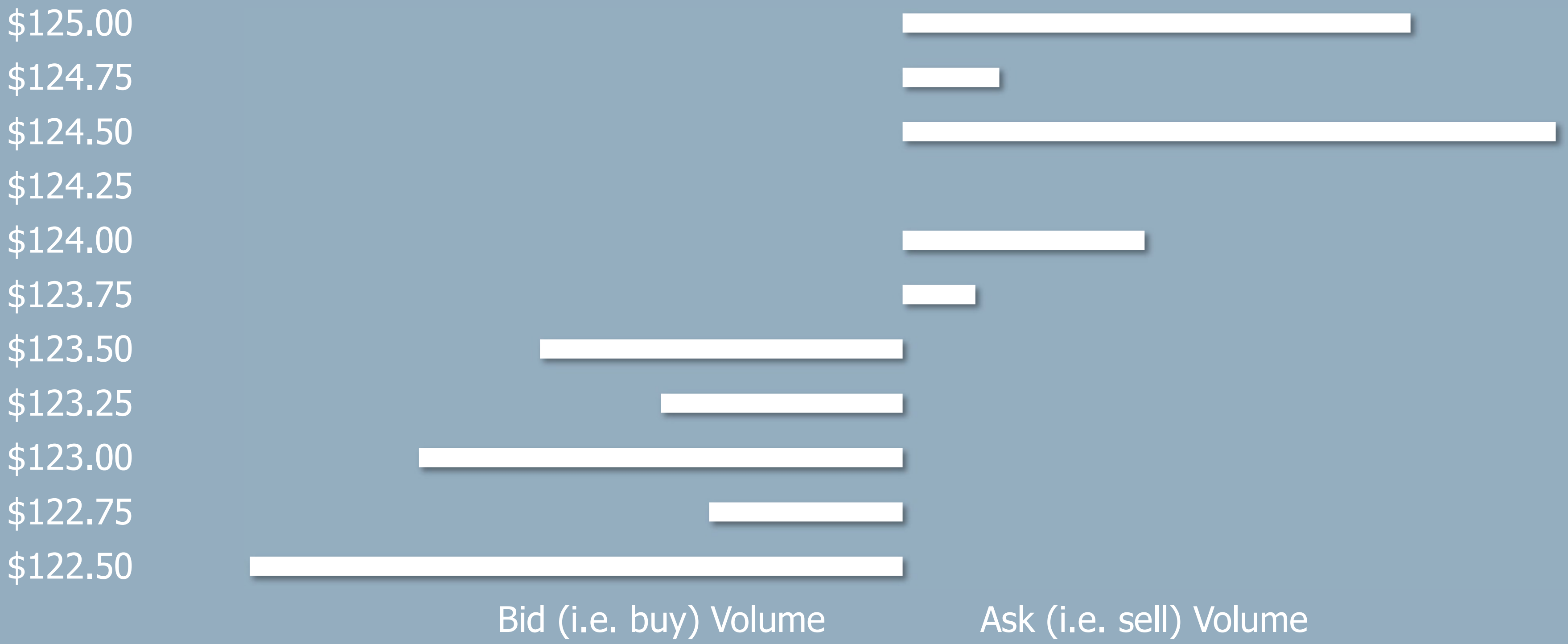
Improving Latency

The process of improving latency and some useful tools

The Money Loop



What Kind of Information?



The Golden Rules

Measure

All the things.

Avoid

Avoid premature optimization.

Problem

Understand the problem you're trying to solve.

Libraries

Know the available tools.

Hardware

Be aware of the characteristics of your hardware.

Identify the Hot Path

An Orderbook

```
class OrderBook {
public:
	OrderBook(std::vector<std::pair<std::string, unsigned long>> levels);

	volume_type get_volume(const std::string_view price_level);
};
```

Measuring the Hot Path

- Google Benchmark
- https://github.com/google/benchmark

Measuring the Hot Path

```
static void BM_existing_price_level(benchmark::State& state)
{
        unsigned long idx = 0;
        OrderBook low_latency{VOLUMES};

        for (auto _ : state)
        {
                std::string price = std::get<0>(VOLUMES[idx]);
                idx = (idx + 1) % VOLUMES.size();
                benchmark::DoNotOptimize(low_latency.get_volume(price));
        }
}
BENCHMARK(BM_existing_price_level);
```

Measuring the Hot Path

```
static void BM_non_existing_price_level(benchmark::State& state)
{
	unsigned long idx = 0;
	OrderBook low_latency{VOLUMES};

	for (auto _ : state)
	{
		std::string missing_price = MISSING_PRICES[idx];
		idx = (idx + 1) % VOLUMES.size();
		benchmark::DoNotOptimize(low_latency.get_volume(missing_price));
	}
}
BENCHMARK(BM_non_existing_price_level);
```

Measuring the Hot Path

```
----------------------------------------------------------------------------
Benchmark                                       Time             CPU   Iterations
----------------------------------------------------------------------------
BM_existing_price_level                       850 ns          798 ns       884919
BM_non_existing_price_level                   898 ns          868 ns       812241
```

First Attempt

```
class OrderBook {
public:
	OrderBook(std::vector<std::pair<std::string, unsigned long>> levels);

	volume_type get_volume(const std::string_view price_level)
	{
		auto it = mVolumes.find({price_level.data(), prive_level.size()});
		if (it != mVolumes.end())
		{
			return it->second;
		}
		return 0;
	}

private:
	std::map<std::string, unsigned long> mVolumes;
};
```

Use the right Algorithm

Callgrind

```
$ valgrind –tool=callgrind ./bmark
$ callgrind_annotate callgrind.out.19254


…


20,776,453     /build/glibc-YYA7BZ/glibc-2.31/string/…
19,559,988     /usr/include/c++/9/bits/stl_tree.h:std::_Rb_tree<…
```

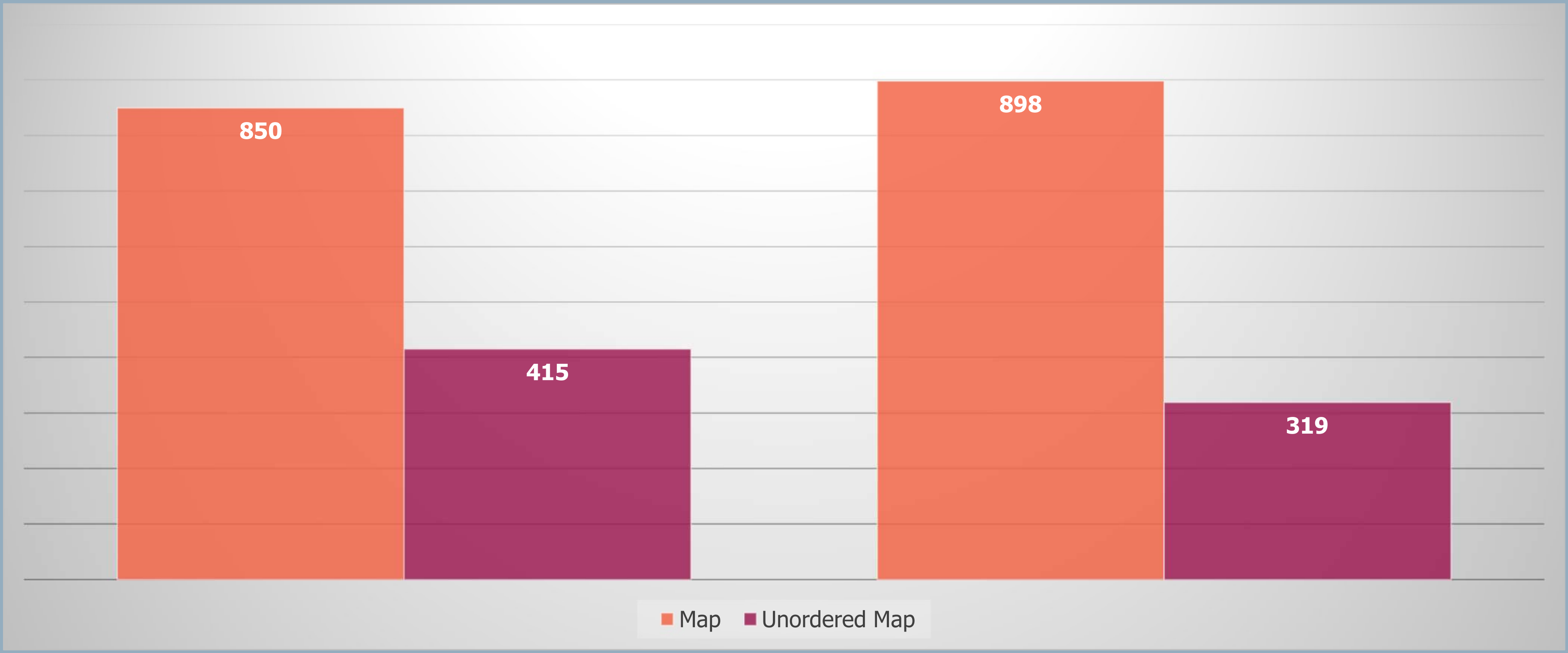
Second Attempt

```
class OrderBook {
public:
        OrderBook(std::vector<std::pair<std::string, unsigned long>> levels);

        volume_type get_volume(const std::string_view price_level)
        {
                auto it = mVolumes.find({price_level.data(), prive_level.size()});
                if (it != mVolumes.end())
                {
                        return it->second;
                }
                return 0;
        }

private:
        std::unordered_map<std::string, unsigned long> mVolumes;
};
```

Measuring the Hot Path Again

```
---------------------------------------------------------------------
Benchmark                                Time             CPU   Iterations
---------------------------------------------------------------------
BM_existing_price_level                415 ns          411 ns      1703025
BM_non_existing_price_level            319 ns          314 ns      2226261
```

Benchmark: Right Algorithm



Avoid Allocation

Callgrind Again

```
$ valgrind –tool=callgrind ./bmark
$ callgrind_annotate callgrind.out.19346


…


8,004,152      libs/low_latency/order_book.h:OrderBook::get_volume(…
7,117,412      ???:std::_Hash_bytes(void const*, unsigned long, unsigned …
5,664,900      /usr/include/c++/9/bits/basic_string.tcc:void
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>
>::_M_construct<char const*>(char const*, char const*, std::forward_iterator_tag)
…
```

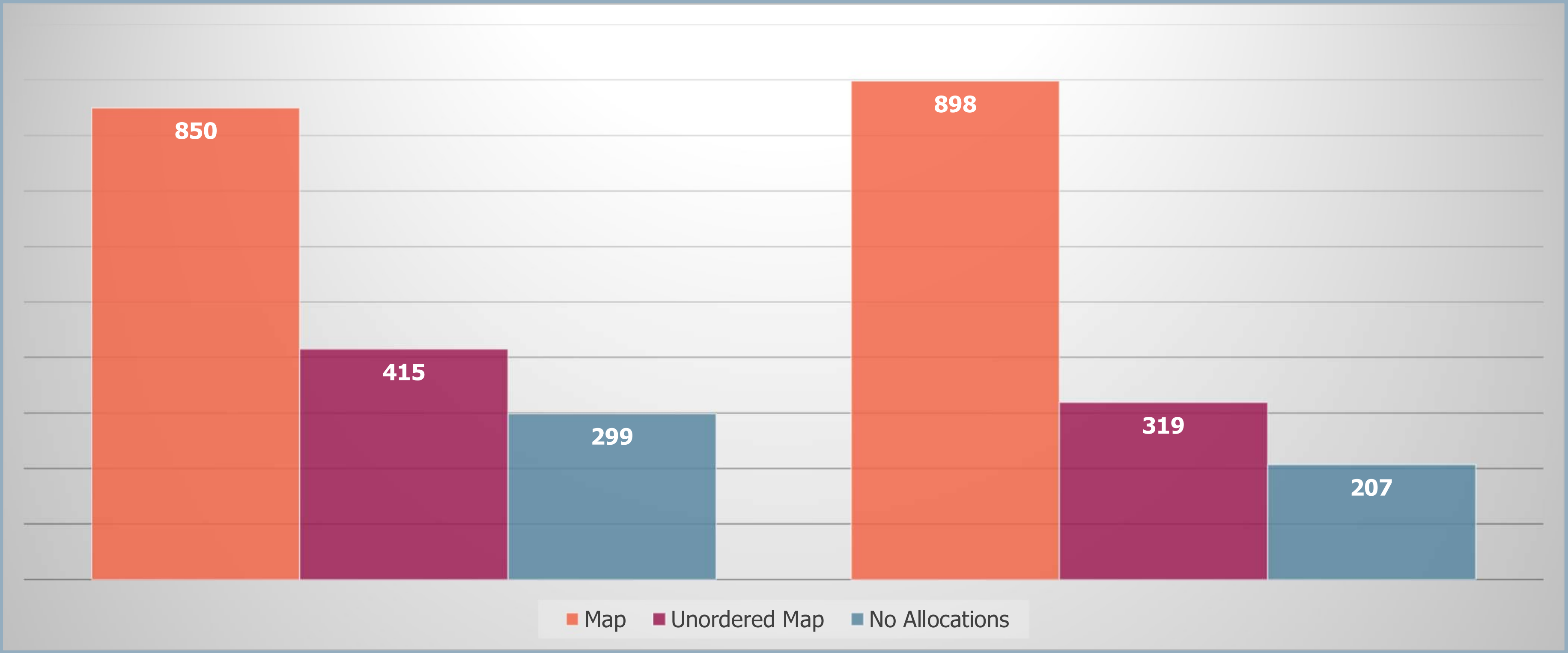
Third Attempt

```
class OrderBook {
public:
        OrderBook(std::vector<std::pair<std::string, unsigned long>> levels);

        volume_type get_volume(const std::string_view price_level)
        {
                auto it = mVolumes.find(price_level); // <- No allocations
                if (it != mVolumes.end())
                {
                        return it->second;
                }
                return 0;
        }

private:
        std::vector<std::string> mPrices;
        std::unordered_map<std::string_view, unsigned long> mVolumes;
};
```

Measuring the Hot Path Again

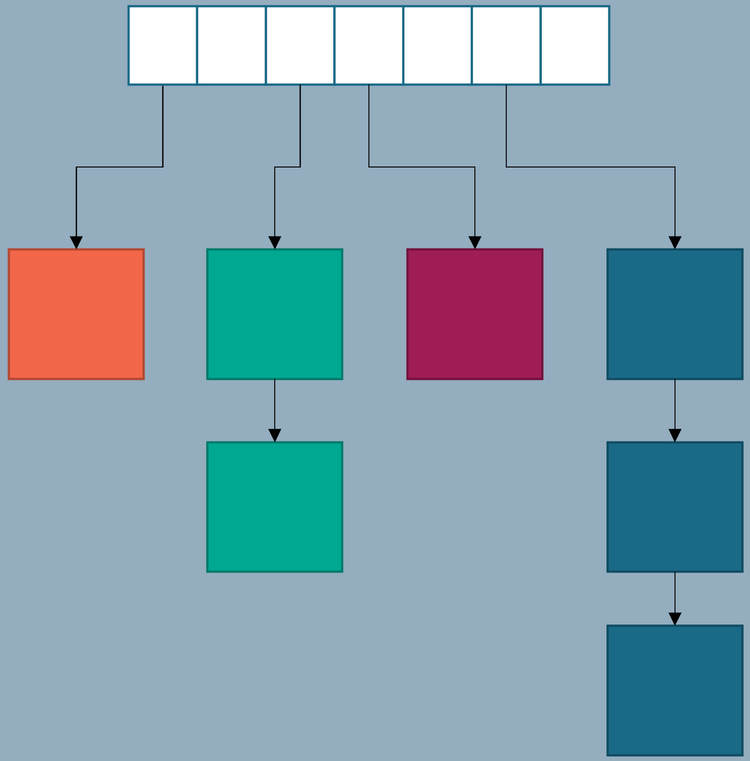
```
---------------------------------------------------------------------------
Benchmark                                     Time             CPU   Iterations
---------------------------------------------------------------------------
BM_existing_price_level                     299 ns          299 ns      2405024
BM_non_existing_price_level                 207 ns          207 ns      3316498
```

Benchmark: No Allocations

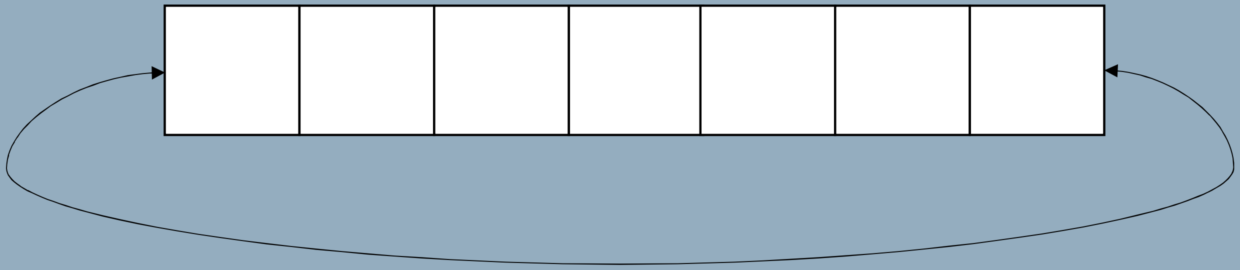


Utilise the Cache

Std::Unordered_map



A circular Buffer



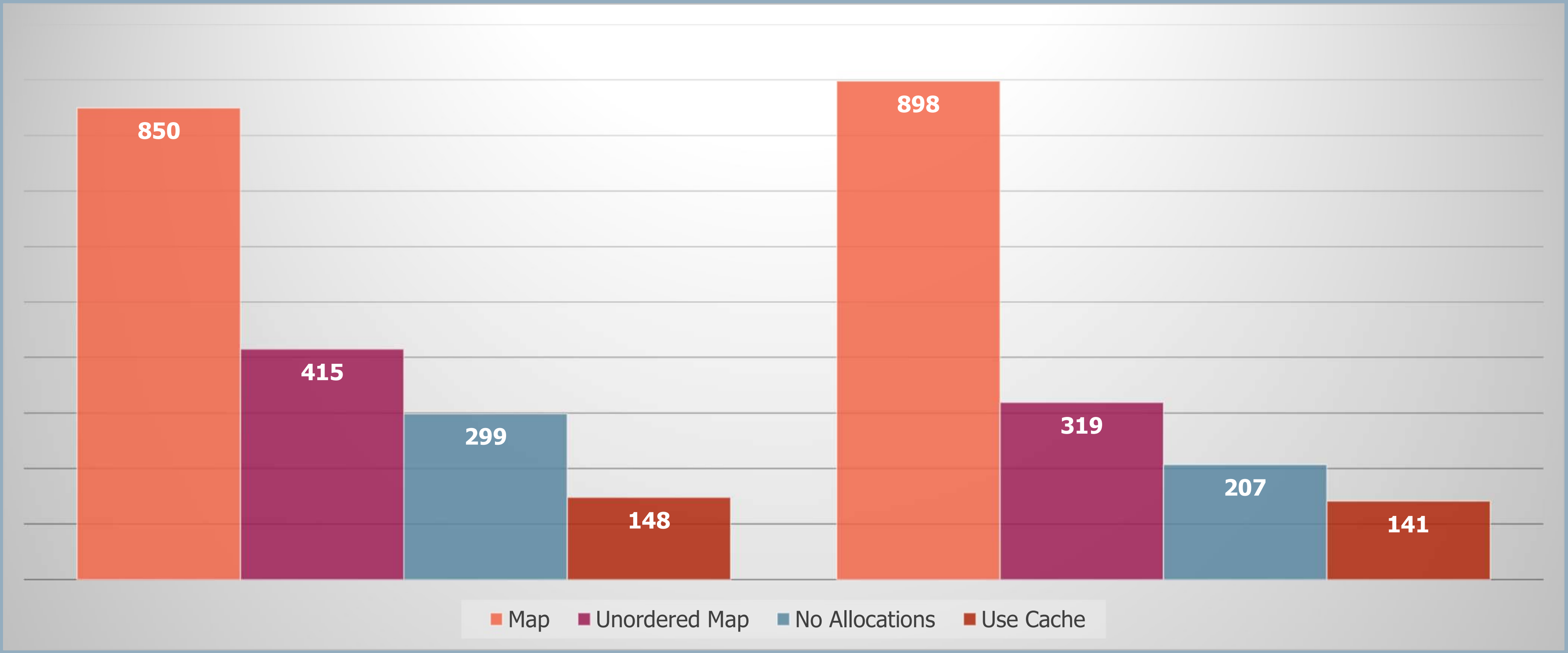
Fourth Attempt

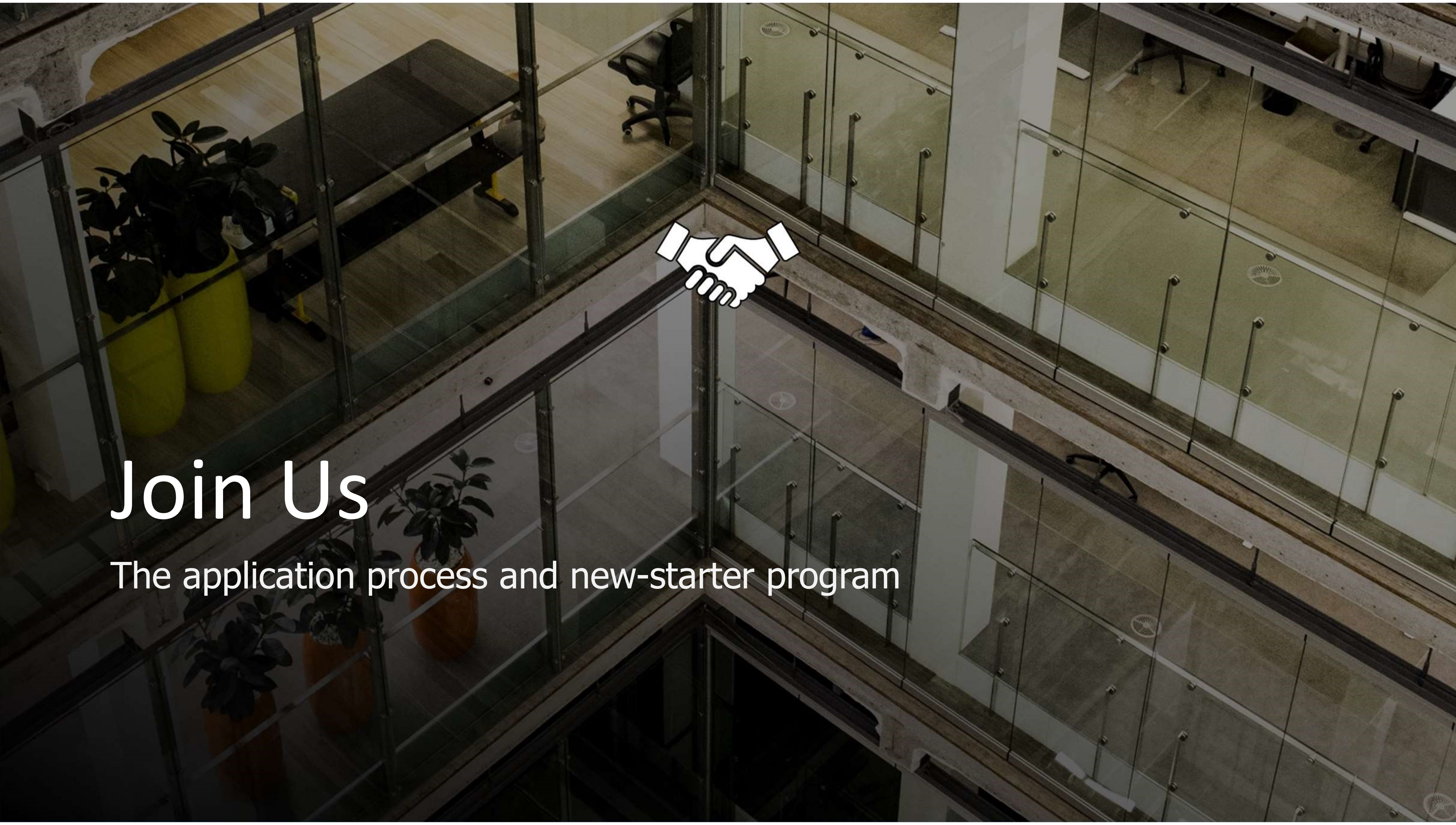
```
class OrderBook {
public:
	OrderBook(std::vector<std::pair<std::string, unsigned long>> levels);

	volume_type get_volume(const std::string_view price_level)
	{
		auto price = (unsigned long)(std::strtod(price_level.data(),
		                                     nullptr)*100);
		return mBuffer[price % mBuffer.size()];
	}

private:
	std::vector<unsigned long> mBuffer;
};
```

Measuring the Hot Path Again

```
----------------------------------------------------------------------------
Benchmark                                     Time             CPU   Iterations
----------------------------------------------------------------------------
BM_existing_price_level                     148 ns          147 ns      4736432
BM_non_existing_price_level                 141 ns          141 ns      4982592
```

Benchmark: No Allocations





Join Us

The application process and new-starter program

Trading Roles at Optiver

Trading	Research	Risk Manager
• Undertake trading through our auto traders • Identify profitable opportunities in the market • Identify trends in market data	• Identify trends in market data • Identify solutions to increase our trading execution success	• Manage market, credit and technology related risks • Providing risk opinions and views to Trading and Management

Technology Roles at Optiver

Software Developer	Production Engineer	FPGA Developer
• Design & develop our trading systems • Maximise speed, reliability & scalability • C++, C# & Python	• Optimise & maintain our trading platform • Maximise uptime & efficiency • Architect & implement automation	• Accelerate our networks & trading systems • Explore mechanisms for faster communications • Work with the fastest devices & platforms

Software Application Process

Apply

Visit our website.

Review

Does your application meet our criteria?

HackerRank

Online programming test.

Phone Interviews

We'll call you for a couple of quick chats.

On-site

Come to our office for your final interviews.

IT New-Starter Program

Bootcamp

Hit the ground running.

Mentorship

Learn from the best.

Rotations

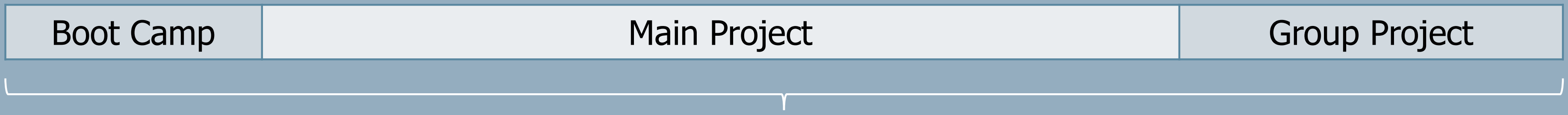
Join a team, do amazing things.

Ongoing Training

Get to the next level.

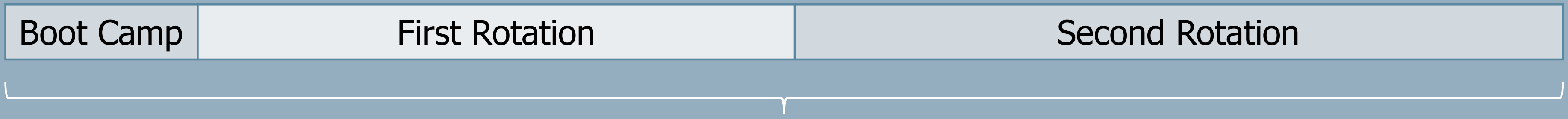
IT Intern & Graduate Programs

Intern Program



12 Weeks

Graduate Program



10 Months

Questions

“No one is dumb who is curious.” – Neil deGrasse Tyson