Virtual vs CRTP in High-Frequency Trading Signal Calculation

Shuen WU

October 3, 2025

Abstract

In this report, we benchmark three implementations of a simple trading signal used in high-frequency trading (HFT) systems: a free function baseline, a runtime polymorphic strategy (via virtual functions), and a compile-time polymorphic strategy (via the Curiously Recurring Template Pattern, CRTP). The objective is to quantify the performance differences in terms of nanoseconds per tick, throughput, and their implications in low-latency trading environments.

1 Introduction

High-frequency trading (HFT) systems process tens of millions of market data ticks per second. The design of per-tick signal computation is critical, as even small differences in latency can accumulate into significant trading advantages or disadvantages. In this experiment, we evaluate:

1. A free function implementation (control).
2. Runtime polymorphism using virtual functions.
3. Static polymorphism using CRTP.

2 Methodology

The trading signal is defined as:

$$signal = \alpha_1 \cdot (microprice - mid) + \alpha_2 \cdot imbalance,$$

where mid, microprice, and imbalance are computed from top-of-book quotes. We generated 10^7 synthetic ticks using a xorshift PRNG and benchmarked each implementation with one iteration. Timing was measured in nanoseconds per tick and throughput in ticks per second.

3 Results

Table 1 shows the benchmark results.

Implementation	Time (ms)	ns/tick	Throughput (M ticks/s)
Free function	10.701	1.070	934.48
Virtual call	13.702	1.370	729.81
CRTP call	11.013	1.101	908.05

Table 1: Performance comparison of free function, virtual, and CRTP implementations on 10M ticks.

4 Discussion

The free function and CRTP variants exhibit nearly identical performance, both achieving sub-1.1 ns per tick and close to 1 billion ticks per second. This is expected, as CRTP allows the compiler to inline calls and optimize away dispatch overhead, effectively matching the free function baseline.

By contrast, the virtual function implementation is slower by roughly 28%. This overhead stems from the vtable lookup and indirect branch, which cannot be inlined and impose penalties on branch prediction and instruction pipelines. In a high-frequency trading context, such differences can be material, as they directly affect per-tick latency and system throughput.

The results confirm that CRTP is highly suitable for latency-critical hot paths in HFT, while virtual functions may still be preferable for higher-level strategy composition or plugin-based architectures where flexibility outweighs microsecond-level performance.

5 Conclusion

This experiment demonstrates that static polymorphism (CRTP) matches the efficiency of free functions and significantly outperforms runtime polymorphism in per-tick trading signal calculation. In HFT systems where every nanosecond matters, CRTP is the preferred abstraction technique for hot-path logic. Virtual functions remain useful in non-latency-critical contexts, where runtime flexibility is required.