Iterative Design For Performant Code

“Software is getting slower more rapidly than hardware becomes faster.”

We often hear that our programs "should be efficient" and we should "squeeze every bit of performance" out of them. But what does that really mean? What's the difference between efficiency and performance? When does this really matter? Are these two goals ever in conflict with each other?

Let's spend some time deeply thinking about data structures and memory access patterns. Just a bit of theory and a whole lot of code analysis (C++). We'll get to build a simple autocomplete suggestion engine in 2 fundamentally different ways and figure out together the pros & cons of each.

Spoiler:
No AI/ML or any other fancy stuff: just "Algorithms + Data Structures = Programs".