Lecture 20 — Self-Optimizing Software

Jeff Zarnett
jzarnett@uwaterloo.ca

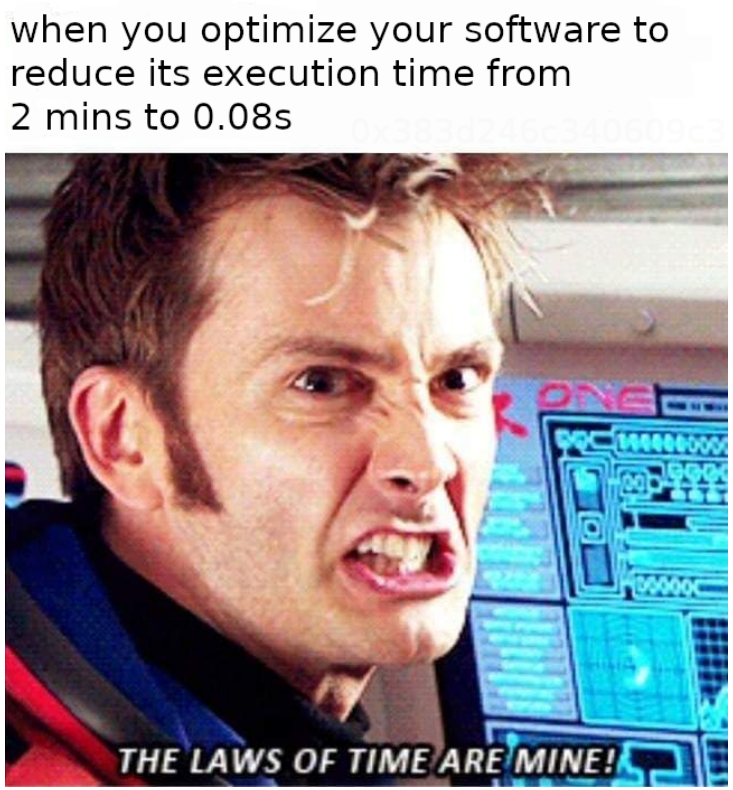
Department of Electrical and Computer Engineering
University of Waterloo

September 4, 2022

Self-Optimizing Software

The previous discussion was at compile-time... what about runtime?

The compiler can’t help, but maybe a program can change itself to be better.



We’ll start with the simple things and get more complex.

Caching

Your first thought about how a program might change itself for speed might be something like caching!

Keep track of the popular exchange rates?

If something becomes popular, it goes in the cache and becomes faster to access.

Too Easy?

You *do* technically get different behaviour at runtime, but this is not quite what we wanted to talk about in this topic.

It’s too easy, and your program should probably be doing it already.

Observe and Change

Next idea: your program's configuration changes at runtime to adapt to observed behaviour

This works becausse our initial guess might not be correct, but also because conditions can change at any time.

Database Example

Query processing: given a certain query, the database server comes up with an execution plan for how to carry it out.

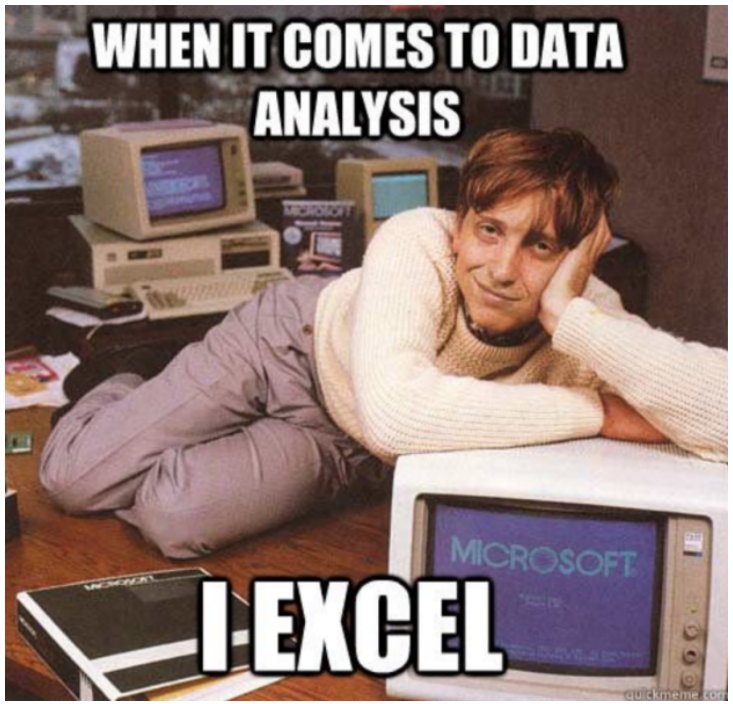
A complex query may have multiple valid routes to executing it.

The server guesses what is correct, but may change its guess based on observation.

Storage?

Also, the server can update how it stores data based on how it's used.

Analogy: Excel sheet with student grades.



This can apply to file storage on disk too.

External Services

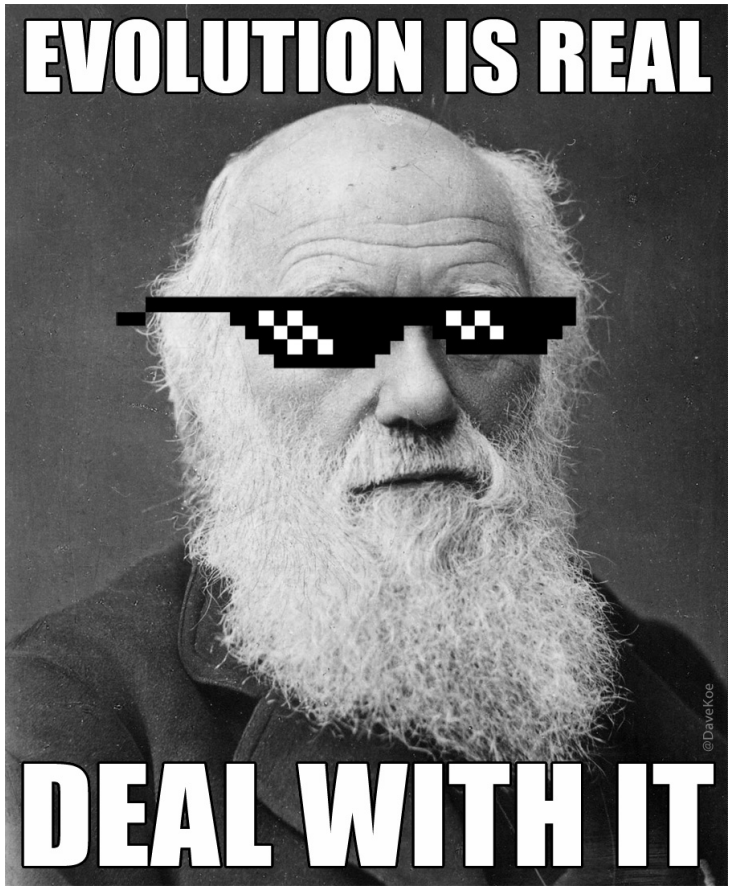
There are three different servers where we can send messages and we'll measure and remember how long it took to get a response from that server.

Geographically closest might be the fastest? Test this assumption!

Genetic Algorithm

If you've taken a course on algorithms, you might know this.

Inspired by natural selection.



Create candidate solutions, evaluate for fitness, keep successful ones.

If Applicable...

This requires some parameters to configure, of course.

The problem also can’t be pass/fail, but needs to have degrees of success.

The fitness function might find a local maximum rather than a global one.

Finding the solution might be fairly slow.

Genetic Algorithms Do What?

How does it help? Optimize our configuration parameters.

Recall our earlier discussion of Google Maps: decision based on parameters.

- Number of routes to evaluate
- Heuristic for generating routes to evaluate
- Decay of traffic information reported by other motorists
- Time of day and month and if it's a holiday
- Search radius for alternate routes

Does Google Maps Work Like This?

This isn’t necessarily how Google does it.

Choosing correct parameters by hand is very difficult.

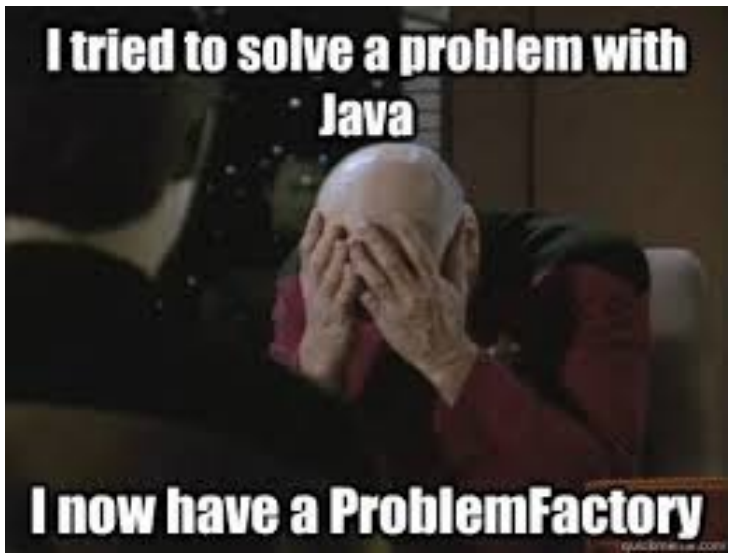
The problem is likely non-linear.

Hotspot Compiler

Some compiler optimizations are always a clear win; others only sometimes.

Inlining is sometimes good, but can make things worse.

Compile-time decisions usually are permanent, but there's also the JVM...



Java Can Be Good?

There's the JIT (Just-in-Time) compiler; so decisions can be made at runtime what to compiler.

The runtime can observe the actual runtime behaviour and change what decision it makes.

That's more helpful in some scenarios (inlining) than others (branch prediction).

Bonus Features

Some other things that can happen at runtime worth talking about:

- Escape Analysis
- On-Stack Replacement
- Intrinsics

Escape Analysis

“Are there any side effects visible outside a method?”

If no, we can skip some heap allocation and maybe change the scope of locks.

Lock elision: don’t use a lock if there’s no need for one.

Lock coarsening: make critical sections bigger; reduce lock/unlock events.

On-Stack Replacement

“I know a shortcut.”

The VM can switch between different implementations of the same method.

If a function is hot, use a more optimized version.

There’s a cost to swapping in the stack frame, but long-term savings.

Ever Debugged in Java/Kotlin?

When you have the debugger attached you can swap in frames with new code.

Very helpful when it takes a lot of work to reproduce a problem...
Or the app takes a long time to start.

Intrinsics

“Prepare in advance the best version for this architecture.”

There are precompiled segments for specific platforms like x86.

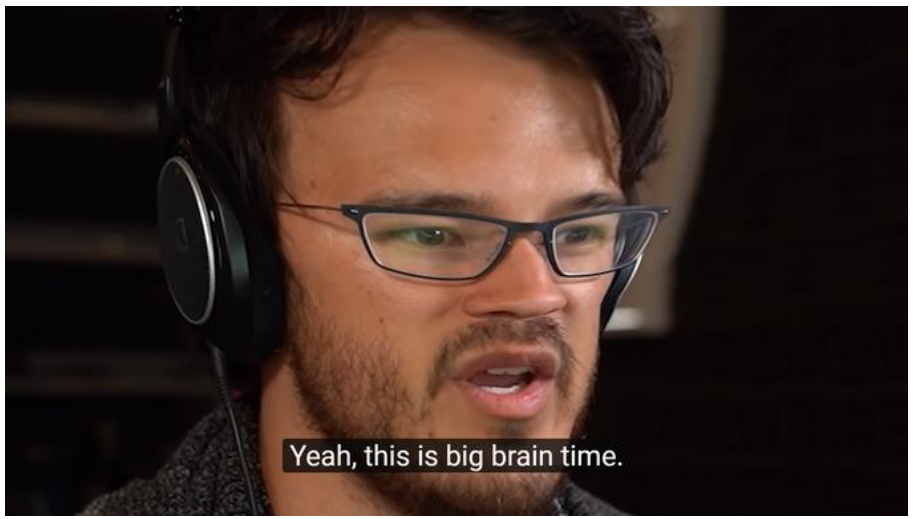
Originally in C++ but now you can convert Java bye code directly.

Intrinsics are complicated... But closest to what you maybe imagined.

Rust and the Hotspot

Rust doesn’t have a runtime as a design choice.

Decisions made at compile-time will stand... Unless...



Precompiled

One possibility is to have different versions of the code compiled and swap them in as needed.

Of course we have to prepare them all and it increases compile time.

We’re still guessing what we think, but we get more than one guess!

Swapping it in at runtime requires some coordination.

But why not if-statements?

Code Generation

So, you want to compile some code.

Don’t write a compiler... Use the one on the current system!

Requiring a specific compiler might be limiting in deployment.

We’re In Business

First, take the binary code of the segment that we want to optimize and then have the compiler take a look at it.

The compiler will then take the binary, convert it to its internal representation (the intermediate representation), then optimize it, and compile it.

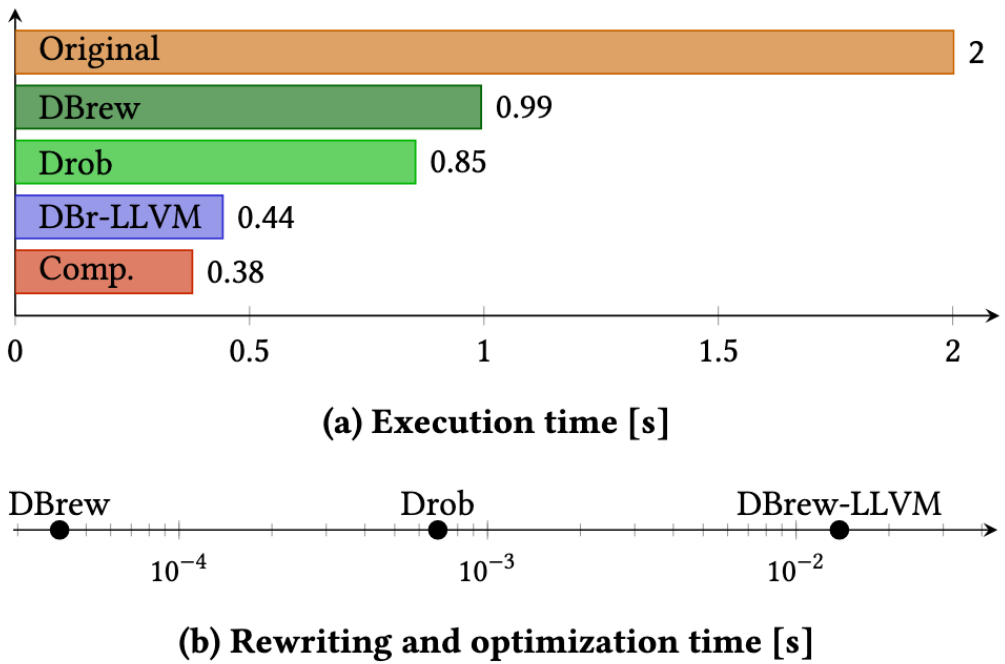
The new binary code can get swapped in when it’s ready.

Does it Work?

Does this work? Yes!

A calculation on a matrix of size 649 $\times$ 649 for 20 000 iterations.

Here's a graph of the performance results from the improved code:



Given the right workload, this is worth doing!

Unfortunately...

“Program sus.”

The Arms Race Continues

Programs that rewrite themselves are frequently judged as suspicious by anti-virus and anti-malware software.

This is part of the arms race between anti-virus software and software.

Anti-malware software end-users have installed may affect the experience.