The [Perl] Weekly Challenge #062

Once again, Neil Bowers, came up with another exciting task for all Team PWC members. Like always, it was fun task. Thanks to Ryan for providing sample data and expected result based on the definition of the task. Half the job done already. The only thing left for the us, is get on with the job. I noticed Raku needed slightly different approach than the Perl. It could be I am doing something very badly. I am happy to correct myself, if you find anything silly. More on this, later down below.

However the second task of the week, N Queens, turned out to tough nut to crack for me. For the first, since I started contributing, I gave up on this. Technically speaking, I did attempt to solve it with the help of my 11 year old daughter, but it was only limited to 2D rather than 3D as expected in the task. Therefore I decided not to submit my solution. Having said that I didn’t want to lose my work, so just for record, I am sharing in this blog, just in case, if I want to re-visit the code.

New Arel like SQL Manager

Some months ago I started working in a system similar to ActiveRecord. But then it became pretty big so then I centered my attention in a SQL AST manager instead.

So I made a library that is basically an Arel port. You can see the README with most of the basic info. After looking at implementations in CPAN I realized there are many of them already but all of them based on hash structures.

In Rust, we lust: Security-focused super-C++ language still most loved among Stack Overflow denizens
Rust for the fifth year in a row has held its position as the most-loved programming language in Stack Overflow's annual developer survey, even if it's not the primary language for most programmers and not many jobs require it.

Rust, beloved by 86 per cent of respondents this year, recently celebrated five years since its 1.0 release. After years of appreciation for its memory safety features, speed, and other benefits, the language is making the move from an aspirational technology to a growing presence in savvy software organizations.

**Performant Containerized Go* Applications with Intel® Advanced Vector Extensions 512 on Clear Linux* OS**[5]

Major cloud software such as Docker*, etcd*, Istio*, Kubernetes*, Prometheus*, and Terraform* use the Go* programming language for core cloud infrastructure components. Why are they using Go? Compared with many other scripting languages, Go is fast!

This article shows how to develop performant Go applications that leverage Intel® Advanced Vector Extensions 512 (Intel® AVX-512) and a Go container based on Clear Linux* OS to improve the performance potential of Go.

[...]

Go is an open source programming language with concurrency mechanisms that help developers make full use of multicore and networked machines. It is expressive, modular, and efficient. Go based data science and analytic applications typically leverage gonum, a set of libraries for matrices, statistics, and optimization. Libraries like gonum build on top of a lower-level BLAS (Basic Linear Algebra Subroutines) layer.

Gonum / netlib creates wrapper packages that provide an interface to Netlib CBLAS implementations. Because netlib uses C and CBLAS, using gonum/netlib provides indirect use of an Intel processor's Intel AVX-512 capability, if available on the running system. The gonum/netlib recommended BLAS layer for performance on Linux is OpenBLAS.

OpenBLAS is an optimized open source BLAS library based on GotoBLAS2 1.13 BSD version, implemented in C. It provides a BLAS layer implementation with Intel AVX-512 acceleration that is adaptable to Intel® Advanced Vector Extensions 2 (Intel® AVX2) or Intel® Streaming SIMD Extensions (Intel® SSE) only platforms.


One of the latest performance optimizations being pursued by Intel on the open-source Linux side is providing an AVX-512-optimized container for Golang usage.

Intel's Clear Linux crew has assembled a new container providing AVX-512 tuned Go
language support paired with AVX-512 optimized Glibc, OpenMP, and OpenBLAS libraries for operating on Intel's Xeon Scalable servers.

Some notes on Corona [7]

In many ways, very little has changed in the way I work on Free Software projects. I get paid to do so ? partly on Calamares, partly on other things ? and there simply was no switch-to-remote work for me. Sitting at my desk, two monitors, FreeBSD underneath and Linux VMs in my face, with IRC for realtime communication: that?s been part-and-parcel of work for years now and nothing has changed there.

Except that now there?s people in the house.

One thing I notice is that when kid[1] is at the machine next to mine, it?s distracting. But how distracting, depends on what is on-screen. Java code only a little, until I feel the urge to ask what?s the issue ? then I?m the cardboard cutout dog. Geometry Dash also only a little, since the rhythmic clicking of the mechanical keyboard mostly makes the same sound as my own keyboard when I?m doing something derpy like re-indenting chunks of CMakeLists.txt. Minecraft, on the other hand, drives me nuts. I just can?t work sitting next to that.

The Slimbook sees a lot more work now, when I flee to the living room. But that?s where online lessons are happening, so I need to sneak around (sometimes out around the side of the house to cross to the other end of the room) because I don?t want to be broadcast accidentally to 20 students listening to middle-school explanations of quadratic equations. The equations are written on the blackboard painted onto one wall of the room.

kid[0] had final exams cancelled out from under them, so they graduated from school with very little sound or fury. We wrote out a CV together and they now have a job (in ?smart? lockdown times!) until the end of the summer and the start of university.

This'll make you feel old: Uni compsci favourite Pascal hits the big five-oh this year [8]

Pascal, a descendant of ALGOL 60 and darling of computer science courses for decades, turns 50 this year.

For engineers of a certain age, Pascal was hard to avoid in the latter part of the last century. Named for 17th-century French mathematician Blaise Pascal, the language is attributed to Swiss computer scientist Niklaus Wirth and was created in part due to Wirth's frustration with the process to improve the ALGOL 60 language.

Involved in the ALGOL X effort, Wirth proposed ALGOL W, which, while not deemed a sufficient advance over ALGOL 60, became Pascal in 1970.