New Language, Go, Promising for Scientific Programming

Sonia Keys
Smithsonian Astrophysical Observatory
keys@cfa.harvard.edu

Potential use of Go, a free and open source computer language sponsored by Google, is explored by porting a user program and comparing to other languages. Ported was Digest2, a program developed at the Minor Planet Center to screen submitted but unconfirmed asteroid astrometry tracks for potential detections of Near Earth Objects. Go seems a promising new choice for scientific programming. It was found to have execution speed comparable to Fortran and C, source code simplicity comparable to Python, and a concurrency model that is unique and intuitive.

Features Enabling Scientific Programming

Easy
• Imperative, procedural programming style is the style familiar to most scientists. It is the style of Fortran and C.
• A clean, new language design without historical baggage.

Fast
• Native code compiler generates fast object code. Performance is comparable to other compiled languages like Fortran and C.
• Modern concurrency features designed for today’s multi-core CPUs.
• IEEE floating point, native complex types.

Amenable
• Free and open source.
• Runs on Linux, Darwin, Windows operating systems. SWIG and FFI support.
• SWIG support. This allows Go to link to existing well tested and highly optimized C and C++ libraries like BLAS/LAPACK, GNU Scientific Library, FFTW.
• Two compiler modes, one a GCC front end and the other stand-alone.

A language comparison can be seen below. Go is functional style characteristic of Haskell, Erlang.

Go's Synopsis

Simplicity
• Clean library mechanism with no header files.
• Type intersecting.
• Complexity of modern object oriented languages consciously avoided.

Safety
• Strong static typing.
• Garbage collection.
• No pointer arithmetic.

Concurrency
• Based on CSP, but innovative in a new direction.
• “Channels” are analogous to Unix pipes, but in memory and more flexible. Synchronous and asynchronous communication, buffered and unbuffered, are possible with multiple consumer, multiple writer, channel types.

Language Comparison

Go keeps
• Speed of compiled languages like Fortran, C, and C++
• Light declaration syntax of dynamically typed languages like Python or JavaScript.
• Memory safety typical of interpreted languages.

Go adds
• Innovative concurrency model.
• Innovative abstraction mechanism, interfaces.

Go rejects
• Header files of C and C++, with their long dependency chains and long compilation times.
• Type inference of C++ and most OOL languages.
• Interpreters and V8s as with Python, Scala.
• Functional style characteristic of Haskell, Erlang.

Performance is comparable to other compiled languages like Fortran and C. SWIG support. This allows Go to link to existing well tested and highly optimized C and C++ libraries like BLAS/LAPACK, GNU Scientific Library, FFTW.

Concurrency

A parallelizable problem.
• Given a list of asteroid "tracklets", Digest2 computes a "RED" score for each. The computations are independent and are trivially parallelized.

Method
Start "workers." Gunsmores are not tied to an OS thread, but in this case makes sense to have the same number of working CPU cores.

A channel of channels will be used to accumulate results.
Start "dispatcher." On a four core CPU, six gunsmores are now running: main, four workers, and dispatcher.

Dispatcher attaches a "return ticket" to each tracklet. Annotated tracklets go on one channel, return tickets in another.
All workers get annotated tracklets from the same source channel.

They work independently, and may complete out of order. Results are sent by return ticket.
Scores arrive by return ticket, in order, on a single channel.

Safety
• Strong static typing.