Go là một ngôn ngữ lập trình mới do Google thiết kế và phát triển. Nó được kỳ vọng sẽ giúp ngành công nghiệp phần mềm khai thác tối đa nền tảng đa lõi của bộ vi xử lý và hoạt động đa nhiệm tốt hơn.

Sự ra đời của ngôn ngữ Go

Vào ngày 21 tháng 9 năm 2007 bộ ba Robert Griesemer, Rob Pike và Ken Thompson bắt đầu phát thảo những tiêu chí cho một ngôn ngữ lập trình mới trên bảng trắng. Vài ngày sau đó, những tiêu chí đó được chấp nhận và một kế hoạch xây dựng một ngôn ngữ lập trình mới được định hình. Sau đó, công việc thiết kế Go(Còn có tên là Golang) được tiến hành song song với các công việc không liên quan khác. Tháng Một năm 2008, Kem Thompson bắt tay xây dựng một trình biên dịch ngôn ngữ C để thử nghiệm các ý tưởng. Vào giữa năm 2008, ngôn ngữ mới này trở thành một dự án toàn thời gian và được đầu tư đầy đủ để trở thành một trình biên dịch dùng cho sản phẩm. Vào tháng 5/2008, Ian Taylor đã độc lập phát triển một GCC front-end cho Go dựa trên bản nháp của đặc tả ngôn ngữ này. Russ Cox đã tham gia vào cuối năm 2008 và giúp hiện thực hoá ngôn ngữ này cũng như các thư viện của nó. Vào ngày 10/11/2009, Go trở thành một dự án mã nguồn mở, và sau đó có rất nhiều cá nhân trong cộng đồng đã đóng góp ý tưởng cũng như mã nguồn cho dự án này.[11]

Việc khai thác tối đa sức mạnh của các bộ xử lý đa lõi và phần cứng thế hệ mới đối với các ngôn ngữ hiện có được xem như là việc không thể làm được. Bởi những giới hạn vốn có của các ngôn ngữ lập trình trên máy tính như C, C++, Java,... Bấy lâu nay, các vấn đề xử lý đa lõi vẫn là chuyện của hệ điều hành.

Google đưa ra ngôn ngữ Go như là một cách tiếp cận khác về vấn đề xử lý đa lõi. Thay vì chỉ có hệ điều hành được phép cung cấp tài nguyên và xử lý, thì các phần mềm cũng có thể tương tác trực tiếp với nền tảng đa lõi giúp cho việc xử lý nhanh hơn rất nhiều.

Đặc điểm thiết kế

Cú pháp

Về mặt cú pháp thì Go rất giống ngôn ngữ C, tuy nhiên nó có nhiều thay đổi trong thiết kế để an toàn hơn và có cú pháp súc tích và dễ đọc. Go cho phép lập trình viên vừa khai báo và khởi tạo biến cùng một lúc mà không cần phải chỉ định kiểu dữ liệu i:=3 hoặc name:="Hello, world!", điều này trái ngược với cú pháp của ngôn ngữ C int i = 3; và const char *s = "Hello, world!". Ở Cuối mỗi dòng lệnh cũng không cần kết thúc bằng dâu chấm phẩy và mỗi hàm có thể trả về nhiều hơn một giá trị.

Dành cho tất cả mọi người

Tài liệu về Go, mô tả Go là "một ngôn ngữ biên dịch nhanh, static type, compiled language (ngôn ngữ biên dịch), nhưng lại giống như một dynamic, interpreted language (ngôn ngữ thông dịch)". Ngay cả khi một chương trình Go lớn, cũng sẽ được biên dịch chỉ trong vòng vài giây. Thêm vào đó, Go còn tránh được những điểm hạn chế của C liên quan đến các file và thư viện. Nói tóm lại, Go giúp cuộc sống của lập trình viên trở nên dễ dàng bằng nhiều cách:

Tiện lợi:

Go được so sánh với các scripting language (ngôn ngữ kịch bản) như Python với khả năng đáp ứng nhiều nhu cầu lập trình phổ biến. Một số tính năng này được tích hợp vào trong chính ngôn ngữ, chẳng hạn như "goroutines" là một hàm cho concurrency và kiểu giống như behavior, ngoài ra các tính năng bổ

sung được có sẵn trong các package thư viện Go chuẩn, như http package của Go. Giống như Python, Go cung cấp khả năng quản lý bộ nhớ tự động bao gồm việc garbage collection (dọn file rác).

Không giống các ngôn ngữ kịch bản như Python, Go biên dịch (compile) code ra nhị phân một cách nhanh chóng. Và không giống như C hoặc C ++, Go biên dịch cực nhanh, nhanh đến mức khiến bạn cảm thấy khi làm việc với Go giống như là làm việc với một ngôn ngữ kịch bản hơn là một ngôn ngữ biên dịch.

Tốc độ:

Run nhị phân chậm hơn so với C, nhưng sự khác biệt về tốc độ này không đáng kể đối với hầu hết các ứng dụng. Hiệu suất của Go tốt ngang với C trong phần lớn công việc và nói chung là nhanh hơn so với các ngôn ngữ khác nổi tiếng về tốc độ (ví dụ: JavaScript, Python và Ruby).

Linh hoạt:

Các file executable được tạo bằng toolchain của Go có thể hoạt động độc lập mà không cần external dependencies mặc định. Toolchain hỗ trợ cho nhiều hệ điều hành, hardware platform (chuẩn phần cứng của máy tính) khác nhau và có thể được sử dụng để biên dịch các chương trình nhị phân qua các nền tảng.

Khả năng tương thích:

Go cung cấp tất cả những điều trên mà không bị mất quyền truy cập vào hệ thống bên dưới (underlying system). Phương mềm Go có thể liên kết với thư viện C bên ngoài hoặc thực hiện các lệnh call hệ thống native. Ví dụ trong Docker, Go interface với các chức năng Linux low-level, cgroups và namespace (tạm dịch: không gian tên), để hoạt động với container.

Hỗ trợ:

Toolchain Go có sẵn dưới dạng binary của Linux, MacOS hoặc Windows hoặc như là một container trong Docker. Go được đặt mặc định trong nhiều bản phát hành phổ biến của Linux, như Red Hat Enterprise Linux và Fedora, giúp cho việc triển khai Go source trở nên dễ dàng hơn đối với các nền tảng trên. Go cũng hỗ trợ mạnh mẽ cho nhiều development environment (môi trường phát triển) của bên thứ ba, từ Microsoft Visual Studio Code đến Komodo IDE của ActiveState.

Khi nào nên dùng Golang?

khi nào nên dùng golang? Và dùng trong trường hợp nào? Không có ngôn ngữ nào phù hợp với mọi loại công việc, nhưng có một số ngôn ngữ phù hợp với nhiều mục đích hơn những ngôn ngữ khác. Mạnh mẽ khi phát triển một số loại ứng dụng chính:

Phân phối các network service (dịch vụ mạng).

Các chương trình ứng dụng mạng (network application) sống hay chết là dựa vào concurrency và các tính năng native concurrency của Go, các goroutines và các channel, rất phù hợp cho các tác vụ đó. Do đó, có nhiều dự án Go dành cho mạng, các chức năng distributed (phân phối) và dịch vụ đám mây: API, web server, minimal frameworks cho các web application và các loại tương tự.

Sự phát triển của cloud-native.

Các tính năng concurrency và network của Go và tính linh hoạt cao của nó làm cho nó phù hợp với việc xây dựng các ứng dụng cloud-native. Trên thực tế, Go đã được sử dụng để xây dựng một trong những nền tảng phát triển ứng dụng dựa trên cloud-native, ứng dụng hệ thống containerization Docker.

Thay thế cho cơ sở hạ tầng hiện có.

Phần lớn các phần mềm của chúng tôi phụ thuộc vào cơ sở hạ tầng Internet đã lạc hậu. Việc viết lại những thứ như vậy bằng Go mang lại nhiều lợi ích, như giữ an toàn bộ nhớ tốt hơn, triển khai trên nhiều nền tảng dễ dàng hơn và một code base "sạch" để hỗ trợ bảo trì trong tương lai. Một server SSH mới được gọi là Teleport và một phiên bản mới của Network Time Protocol được viết bằng Go, được cung cấp như phương pháp thay thế cho các đối tác thông thường của họ.

khi nào nên dùng golang

Go không phù hợp với việc gì?

Go được thiết kế nhỏ gọn và dễ hiểu, vì vậy dẫn đến một số tính năng nhất định bị bỏ qua. Thế nên một số tính năng phổ biến có trong các ngôn ngữ khác thì lại không có trong Go.

Một trong những tính năng Go không có là generics, là kiểu biểu diễn của Types dưới dạng tham số khi định nghĩa lớp, hàm và interfaces. Go không bao gồm generics và steward của ngôn ngữ này ngăn cản việc thêm generics vào vì cho rằng điều đó sẽ làm giảm tính đơn giản. Tuy vẫn có thể làm việc tốt với Go, nhưng rất nhiều lập trình viên vẫn muốn thêm generics vào nó.

Nhược điểm khác của Go là kích thước của các chương trình. Code được biên dịch kiểu static (tĩnh) theo mặc định. Cách này làm đơn giản hóa quá trình xây dựng và triển khai, nhưng dẫn đến việc chỉ một đoạn code đơn giản "Hello, world!" lại nặng đến khoảng 1,5MB trên Windows 64-bit. Nhóm nghiên cứu của Go đang cố gắng để giảm kích thước của những chương trình này trong những bản phát hành kế tiếp. Có những giải pháp cho việc này là nén file hoặc xoá bỏ thông tin về debug của Go.

Tuy nhiên, một tính năng khác của Go, quản lý bộ nhớ tự động (AMM), có thể được xem như là một nhược điểm, vì garbage collection (quá trình thu gom file rác) đòi hỏi một số memory nhất định để xử lý. Theo thiết kế, Go không thể quản lý bộ nhớ bằng tay và việc dọn dẹp file rác ở Go bị chỉ trích là không thể giải quyết tốt các loại memory load (bộ nhớ tải) xuất hiện trong các ứng dụng của doanh nghiệp.

Xét về mặt tích cực, Go 1.8 mang lại nhiều cải tiến trong quản lý bộ nhớ và dọn dẹp file rác để giảm độ trệ (lag). Tất nhiên, các nhà phát triển Go có thể sử dụng phân bố (allocation) bộ nhớ bằng tay trong một extension của C hoặc bằng cách sử dụng thư viện quản lý bộ nhớ thủ công của bên thứ ba.

Tình hình xung quanh việc xây dựng một GUI (giao diện đồ họa người dùng) phong phú cho các ứng dụng của Go, chẳng hạn như trong các ứng dụng dành cho desktop, vẫn còn rải rác.

Hầu hết các sản phẩm từ Go là các command-line tool hoặc các dịch vụ network. Mặc dù vậy, vẫn có rất nhiều dự án đang được thực hiện để mang lại một GUI phong phú cho các ứng dụng của Go. Có các framework như GTK và GTK3. Một dự án khác nhằm cung cấp platform-native UI, mặc dù các giao diện này dựa vào các binding của C và không được viết bằng Go. Ngoài ra, bởi vì Go được thiết kế là nền tảng độc lập, nên không có bất kỳ cái gì được nêu ở trên có thể trở thành một phần của package chuẩn.

[1] Click text view file