HPS

https://valerius.me

Valerius Mattfeld

Go Programming in High-Performance Computing

Applications, Scalability and Speed

Table of contents

Origins

- Google has developed the **Go** programming language, aka. Golang

Documentation - The Go Programming Language, Pike, *Another Go at Language Design*

Origins

- Google has developed the **Go** programming language, aka. Golang
- Engineers wanted to address criticism of other languages, but maintain their useful features
  - Static typing (C)
  - Readability (Python)

Documentation - The Go Programming Language, Pike, *Another Go at Language Design*

Origins

- Google has developed the **Go** programming language, aka. Golang
- Engineers wanted to address criticism of other languages, but maintain their useful features
  - Static typing (C)
  - Readability (Python)
  - Add networking and multiprocessing (new)

Documentation - The Go Programming Language, Pike, *Another Go at Language Design*

Origins

- Google has developed the **Go** programming language, aka. Golang
- Engineers wanted to address criticism of other languages, but maintain their useful features
  - Static typing (C)
  - Readability (Python)
  - Add networking and multiprocessing (new)
  - Version 1.0 in 2012

Documentation - The Go Programming Language, Pike, *Another Go at Language Design*

The Go Programming Language

- Open source

The Go Programming Language, *Documentation - The Go Programming Language*

The Go Programming Language

- Open source
- Simple and clean syntax

The Go Programming Language, Documentation - The Go Programming Language

The Go Programming Language

- Open source
- Simple and clean syntax
- Concurrency via `goroutines`

The Go Programming Language, Documentation - The Go Programming Language

The Go Programming Language

- Open source
- Simple and clean syntax
- Concurrency via `goroutines`
- Auto-typing at variable declaration

The Go Programming Language, Documentation - The Go Programming Language

The Go Programming Language (cont.)

- Fast compilation

Documentation - The Go Programming Language,Go: Source Code

The Go Programming Language (cont.)

- Fast compilation
- Build-in garbage collection

Documentation - The Go Programming Language,Go: Source Code

The Go Programming Language (cont.)

- Fast compilation
- Build-in garbage collection
- Big standard library

Documentation - The Go Programming Language,Go: Source Code

The Go Programming Language (cont.)

- Fast compilation
- Build-in garbage collection
- Big standard library
- Many helper / Q.O.L. tools

Documentation - The Go Programming Language,Go: Source Code

The Go Programming Language (cont.)

- Go Modules for dependencies.

Documentation - The Go Programming Language

The Go Programming Language (cont.)

- Go Modules for dependencies.
- comparable to `pip`, `cargo`, `npm`, etc.

Documentation - The Go Programming Language

The Go Programming Language (cont.)

- Go Modules for dependencies.
- comparable to `pip`, `cargo`, `npm`, etc.
- `go.mod`

Documentation - The Go Programming Language

Go Syntax Example

Logging an add()-function implemented in Go

main.go

```go
package main // package scope definition
import (
    "github.com/rs/zerolog" // using a third-party package
    "github.com/rs/zerolog/log"
)

func add(a, b int) int { // function implementation
    return a + b
}
func main() { // entry point
    n := 5 // variable declaration
    log.Println(add(n, 5)) // logging library call
}
```

The Go Programming Language (cont.)

- Language specific Memory Model

Documentation - The Go Programming Language, Pike, *Another Go at Language Design*

The Go Programming Language (cont.)

- Language specific Memory Model
- Data Race Detector

Documentation - The Go Programming Language, Pike, *Another Go at Language Design*

The Go Programming Language (cont.)

- Language specific Memory Model
- Data Race Detector
- Syntax and language design force you to slow down

Documentation - The Go Programming Language, Pike, *Another Go at Language Design*

Goroutine Example

main.go

```go
func printNumbers(ch chan int) {
        for i := 1; i <= 5; i++ {
                ch <- i
                time.Sleep(1 * time.Second)
        }
        close(ch) // close the channel, preventing infinite blocking
}
func main() {
        ch := make(chan int)
        go printNumbers(ch) // invocate the function via a routine
        for num := range ch { // awaits values until the channel is closed
                fmt.Println("Received:", num)
        }
}
```

Go in Virtualization: Docker / Moby

- **Efficient**: No virtualized operating system required

The Moby Project, Efficiency Meets Flexibility

Go in Virtualization: Docker / Moby

- **Efficient**: No virtualized operating system required
- **Isolation**: Self-contained environments

The Moby Project, Efficiency Meets Flexibility

Go in Virtualization: Docker / Moby

- **Efficient**: No virtualized operating system required
- **Isolation**: Self-contained environments
- **Portability**: Containers can run on almost any operating system

The Moby Project, Efficiency Meets Flexibility

Go in Virtualization: Docker / Moby (cont.)

- **Memory Safety**: Built-In garbage collection and language design reduce bugs

The Moby Project, *Efficiency Meets Flexibility*, Batta, *Golang Working with goroutines*

Go in Virtualization: Docker / Moby (cont.)

- **Memory Safety**: Built-In garbage collection and language design reduce bugs
- **Simplicity**: Makes it easy to extend programs in a short amount of time

The Moby Project, *Efficiency Meets Flexibility*, Batta, *Golang Working with goroutines*

Go in Virtualization: Docker / Moby (cont.)

- **Memory Safety**: Built-In garbage collection and language design reduce bugs
- **Simplicity**: Makes it easy to extend programs in a short amount of time
- **Concurrency**: Go routines are lightweight threads (2 kB each)

The Moby Project, *Efficiency Meets Flexibility*, Batta, *Golang Working with goroutines*

Go in Virtualization: Docker / Moby (cont.)

- **Memory Safety**: Built-In garbage collection and language design reduce bugs
- **Simplicity**: Makes it easy to extend programs in a short amount of time
- **Concurrency**: Go routines are lightweight threads (2 kB each)
- **Fast**: Compiled executables

The Moby Project, *Efficiency Meets Flexibility*, Batta, *Golang Working with goroutines*

Go in Infrastructure: Apptainer / Singularity

- Apptainer, formerly Singularity

Apptainer - The Container System for Secure HPC, Apptainer - Repository

Go in Infrastructure: Apptainer / Singularity

- Apptainer, formerly Singularity
- Almost completely written in Go

Apptainer - The Container System for Secure HPC, Apptainer - Repository

Go in Infrastructure: Apptainer / Singularity

- Apptainer, formerly Singularity
- Almost completely written in Go
- Non-Root-Container system for HPC

Apptainer - The Container System for Secure HPC, *Apptainer - Repository*

Go in Infrastructure: Apptainer / Singularity

- Apptainer, formerly Singularity
- Almost completely written in Go
- Non-Root-Container system for HPC
- Minimal virtualization, Docker compatible

Apptainer - The Container System for Secure HPC, *Apptainer - Repository*

Go in Infrastructure: Apptainer / Singularity (cont.)

- Container integrity-guarantee at runtime

Go in Infrastructure: Apptainer / Singularity (cont.)

- Container integrity-guarantee at runtime
- Container encryption, and secret management compatibility (e.g. Vault)

Go in Infrastructure: Apptainer / Singularity (cont.)

- Container integrity-guarantee at runtime
- Container encryption, and secret management compatibility (e.g. Vault)
- Tighter integration to system resources, e.g. GPUs

Go in Infrastructure: Apptainer / Singularity (cont.)

- Container integrity-guarantee at runtime
- Container encryption, and secret management compatibility (e.g. Vault)
- Tighter integration to system resources, e.g. GPUs
- Custom container repositories possible

Go in Container Orchestration: Kubernetes

- Kubernetes (k8s) is written in Go and builds upon Docker.
- Automates deployment and scaling
- Manages container orchestrations

Kubernetes Documentation

Go in Container Orchestration: Kubernetes in HPC

- Can be tightly integrated with HPC
- Fine-grained scheduling policies containerized workloads possible
- Models for rootless, unprivileged Runtime Environment Containers are proposed
- **But:** K8s was originally designed for microservices, not HPC workloads

Dockendorf, Baer, and Johnson, “Early Experiences with Tight Integration of Kubernetes in an HPC Environment”,Liu and Guitart, *Fine-Grained Scheduling for Containerized HPC Workloads in Kubernetes Clusters*, Hursey, “A separated model for running rootless, unprivileged PMIx-enabled HPC applications in Kubernetes”

K8s for Serverless Functions

- Notable examples for Kubernetes-based self-hostable platforms are:
  - knative.dev (supporting languages like Go, Elixir, Java, etc.)

Knative documentation, *Nuclio - "Serverless" for Real-Time Events and Data Processing*, *openfaas/faas: OpenFaaS - Serverless Functions Made Simple*, *fission/fission: Fast and Simple Serverless Functions for Kubernetes*

K8s for Serverless Functions

- Notable examples for Kubernetes-based self-hostable platforms are:
  - knative.dev (supporting languages like Go, Elixir, Java, etc.)
  - nuclio.io (completely written in Go)

Knative documentation, *Nuclio - "Serverless" for Real-Time Events and Data Processing*, *openfaas/faas: OpenFaaS - Serverless Functions Made Simple*, *fission/fission: Fast and Simple Serverless Functions for Kubernetes*

K8s for Serverless Functions

- Notable examples for Kubernetes-based self-hostable platforms are:
  - knative.dev (supporting languages like Go, Elixir, Java, etc.)
  - nuclio.io (completely written in Go)
  - openfaas.com (also using Go)

Knative documentation, *Nuclio - "Serverless" for Real-Time Events and Data Processing*, *openfaas/faas: OpenFaaS - Serverless Functions Made Simple*, *fission/fission: Fast and Simple Serverless Functions for Kubernetes*

K8s for Serverless Functions

- Notable examples for Kubernetes-based self-hostable platforms are:
  - knative.dev (supporting languages like Go, Elixir, Java, etc.)
  - nuclio.io (completely written in Go)
  - openfaas.com (also using Go)
  - fission.io (built with Go)

Knative documentation, *Nuclio - "Serverless" for Real-Time Events and Data Processing*, *openfaas/faas: OpenFaaS - Serverless Functions Made Simple*, *fission/fission: Fast and Simple Serverless Functions for Kubernetes*

Using Go in "working" HPC applications

- No *native* implementation of OpenMPI so far

Beifuss, *A Golang Wrapper for MPI*,Bromberger, *sbromberger/gompi*, *mpi package - github.com/cpmech/gosl/mpi - Go Packages*, Weging, *go-mpi*

Using Go in "working" HPC applications

- No *native* implementation of OpenMPI so far
- Wrapper packages to OpenMPI C++ library
  - GoMPI, feature complete with MPI v2
  - Gosl/MPI, most popular wrapper

Beifuss, *A Golang Wrapper for MPI*,Bromberger, *sbromberger/gompi*, *mpi package - github.com/cpmech/gosl/mpi - Go Packages*, Weging, *go-mpi*

Using Go in "working" HPC applications

- No *native* implementation of OpenMPI so far
- Wrapper packages to OpenMPI C++ library
  - GoMPI, feature complete with MPI v2
  - Gosl/MPI, most popular wrapper
  - yoo/Go-MPI faster than Gosl, less message latency
- Currently not Rust-FFI wrapper

Beifuss, *A Golang Wrapper for MPI*,Bromberger, *sbromberger/gompi, mpi package - github.com/cpmech/gosl/mpi - Go Packages*, Weging, *go-mpi*

Using Go in "working" HPC applications (cont.)

- Implementations C are faster

Beifuss, *A Golang Wrapper for MPI*

Using Go in "working" HPC applications (cont.)

- Implementations C are faster
- Better scaling for non-blocking communication

Beifuss, *A Golang Wrapper for MPI*

Using Go in "working" HPC applications (cont.)

- Implementations C are faster
- Better scaling for non-blocking communication
- There is a lack of better/native implementations

Beifuss, *A Golang Wrapper for MPI*

Using Go in "working" HPC applications (cont.)

Thoughts: Use Go when...

- Heavy use of parallelization is required

Using Go in "working" HPC applications (cont.)

Thoughts: Use Go when...

- Heavy use of parallelization is required
- Many developers are involved

Using Go in "working" HPC applications (cont.)

Thoughts: Use Go when...

- Heavy use of parallelization is required
- Many developers are involved
- Changes need to happen quickly

Conclusion

- Go is currently the most used language for infrastructure and containerization
- Memory Management, and state-of-the-art parallelization
- OpenMPI C++ Wrappers are most common
- Go is slower than Rust and C++, but faster than Python
- Go should be used in deployments, Rust or C++ for calculations in HPC
- Easy to learn, fast initial contributions

References

Apptainer - Repository. original-date: 2021-11-30T13:45:16Z. July 2023. URL: https://github.com/apptainer/apptainer (visited on 07/06/2023).

Apptainer - The Container System for Secure HPC. en. URL: https://apptainer.org/ (visited on 07/06/2023).

Batta, Anjaneyulu. *Golang Working with goroutines*. en. URL: https://learnbatta.com//course/golang/working-with-goroutines/ (visited on 07/06/2023).

Beifuss. *A Golang Wrapper for MPI*. 2014.

Bromberger, Seth. *sbromberger/gompi*. original-date: 2019-10-15T17:52:52Z. June 2023. URL: https://github.com/sbromberger/gompi (visited on 07/06/2023).

Dockendorf, Trey, Troy Baer, and Doug Johnson. "Early Experiences with Tight Integration of Kubernetes in an HPC Environment". In: *Practice and Experience in Advanced Research Computing*. PEARC '22. New York, NY, USA: Association for Computing Machinery, July 2022, pp. 1–4. ISBN: 978-1-4503-9161-0. DOI: 10.1145/3491418.3535150. URL: https://doi.org/10.1145/3491418.3535150 (visited on 07/05/2023).

Documentation - The Go Programming Language. en. URL: https://go.dev/doc/ (visited on 07/04/2023).

Efficiency Meets Flexibility: The Advantages of Using Go in Your Project. en. URL: https://polcode.com/resources/blog/efficiency-meets-flexibility-the-advantages-of-using-go-in-your-project/ (visited on 07/05/2023).

fission/fission: Fast and Simple Serverless Functions for Kubernetes. URL: https://github.com/fission/fission (visited on 05/31/2023).

Go: Source Code. URL: https://cs.opensource.google/go/go/+/master:src/ (visited on 07/05/2023).

Hursey, Joshua. "A separated model for running rootless, unprivileged PMIx-enabled HPC applications in