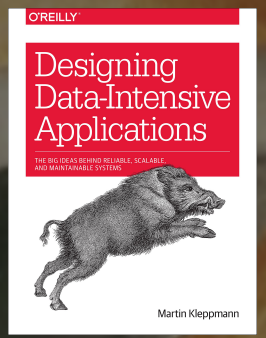
MSFT Sys Meetup



http://

Policy Against Harassment at ACM Activities

https://www.acm.org/about-acm/policy-against-harassment

MSFT Sys Meetup wants to encourage and preserve this open exchange of ideas, which requires an environment that enables all to participate without fear of personal harassment. We define harassment to include specific unacceptable factors and behaviors listed in the ACM's policy against harassment. Unacceptable behavior will not be tolerated.

Freely accessible resources

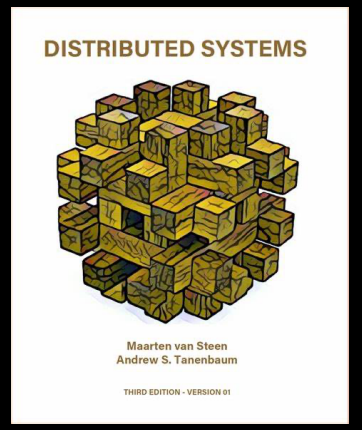
[Code]

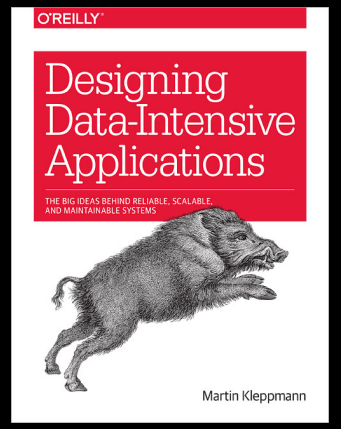
[Zoom]

[Course]

[DDIA (O'Reilly)]

[Distributed System 3rd edition]





Calendar: https://docs.google.com/spreadsheets/d/1RsbGpq1cwNSmYn5hcmT8Hv5O4qssl2HXsTcG82RHVQk/edit?usp=sharing

(Internal) Teams: g078pwd

(Public) Discord

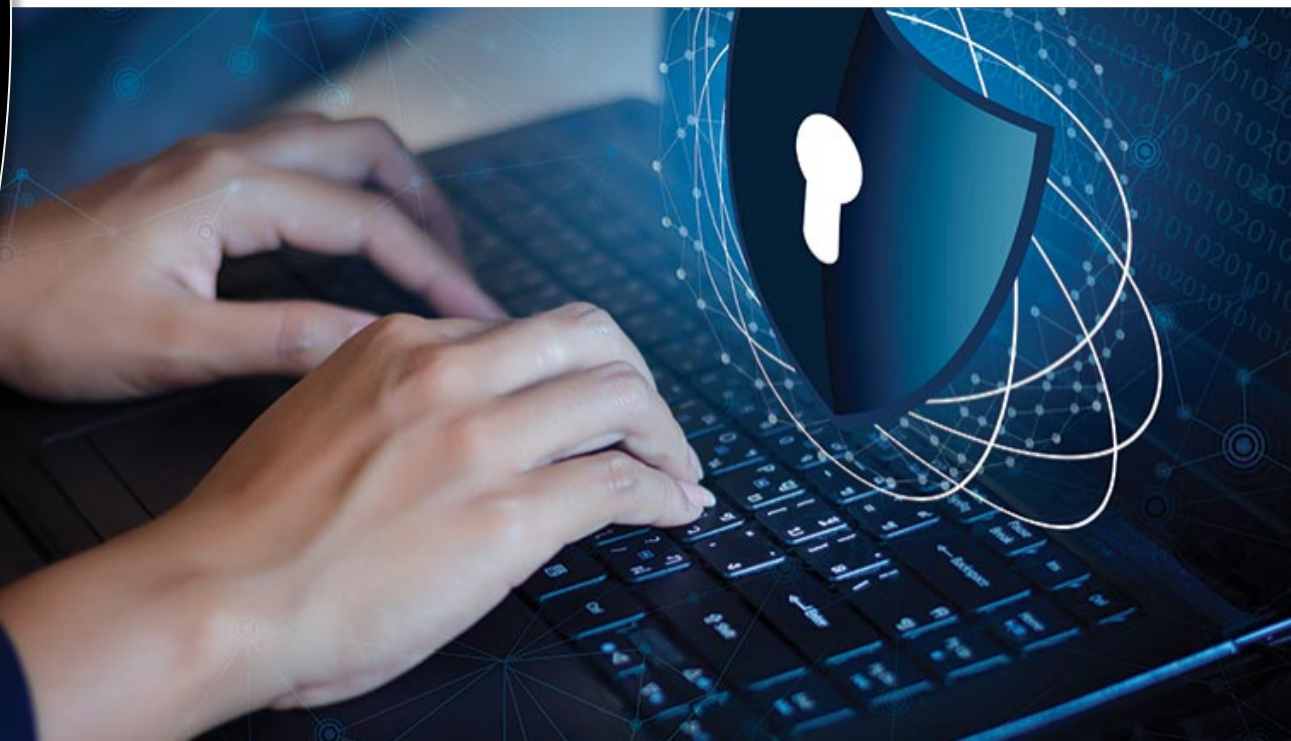
(Public) WeChat: add mossaka or Lin1991Wen

Notion: https://www.notion.so/invite/cd6df70a94e7f67f6d21f4c509783d3c9cfd0e69

YouTube: https://www.youtube.com/playlist?list=PL1voNxn5MODMJxAZVvgFHZ0jZ-fuSut68

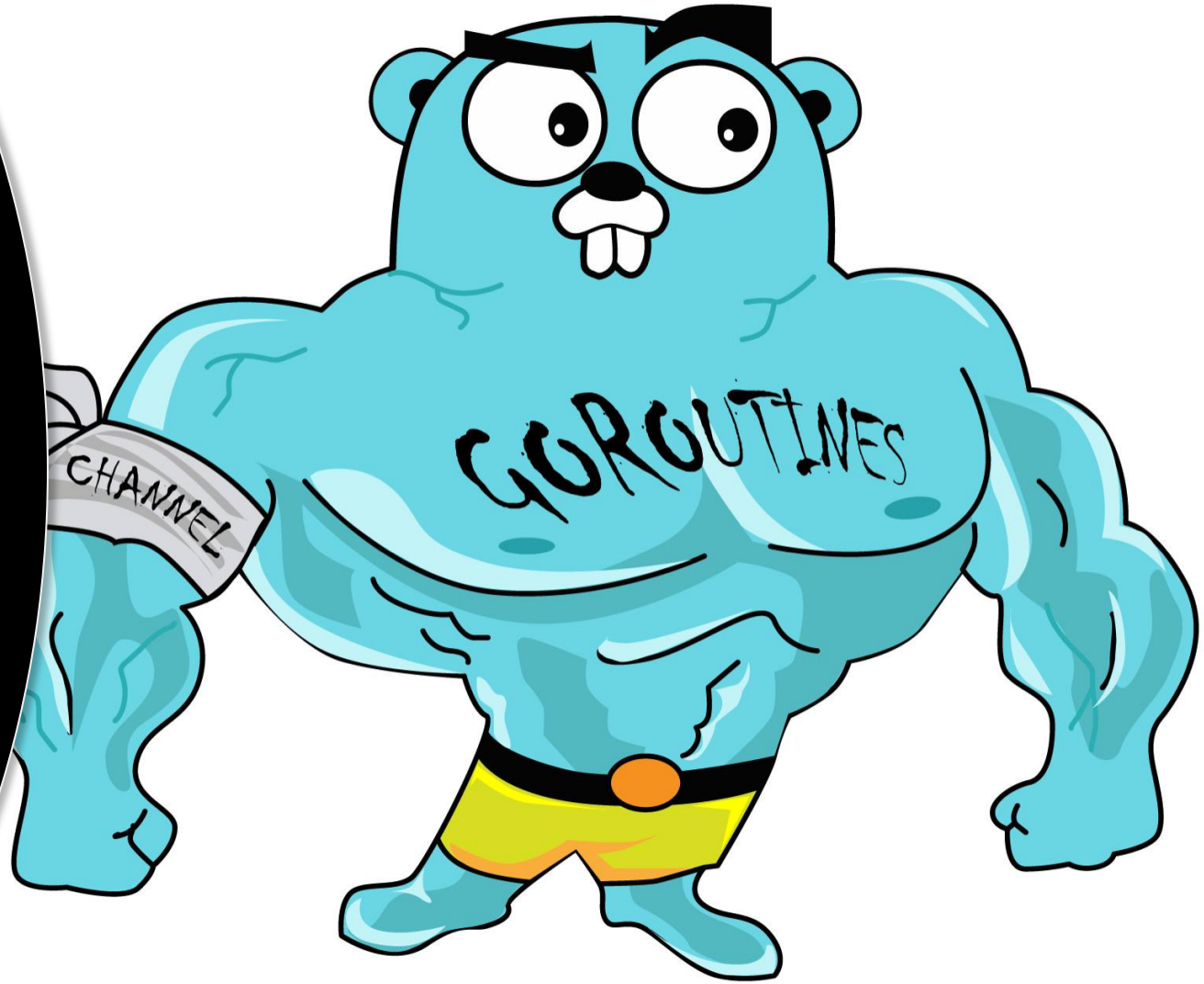
Company Privacy

Microsoft



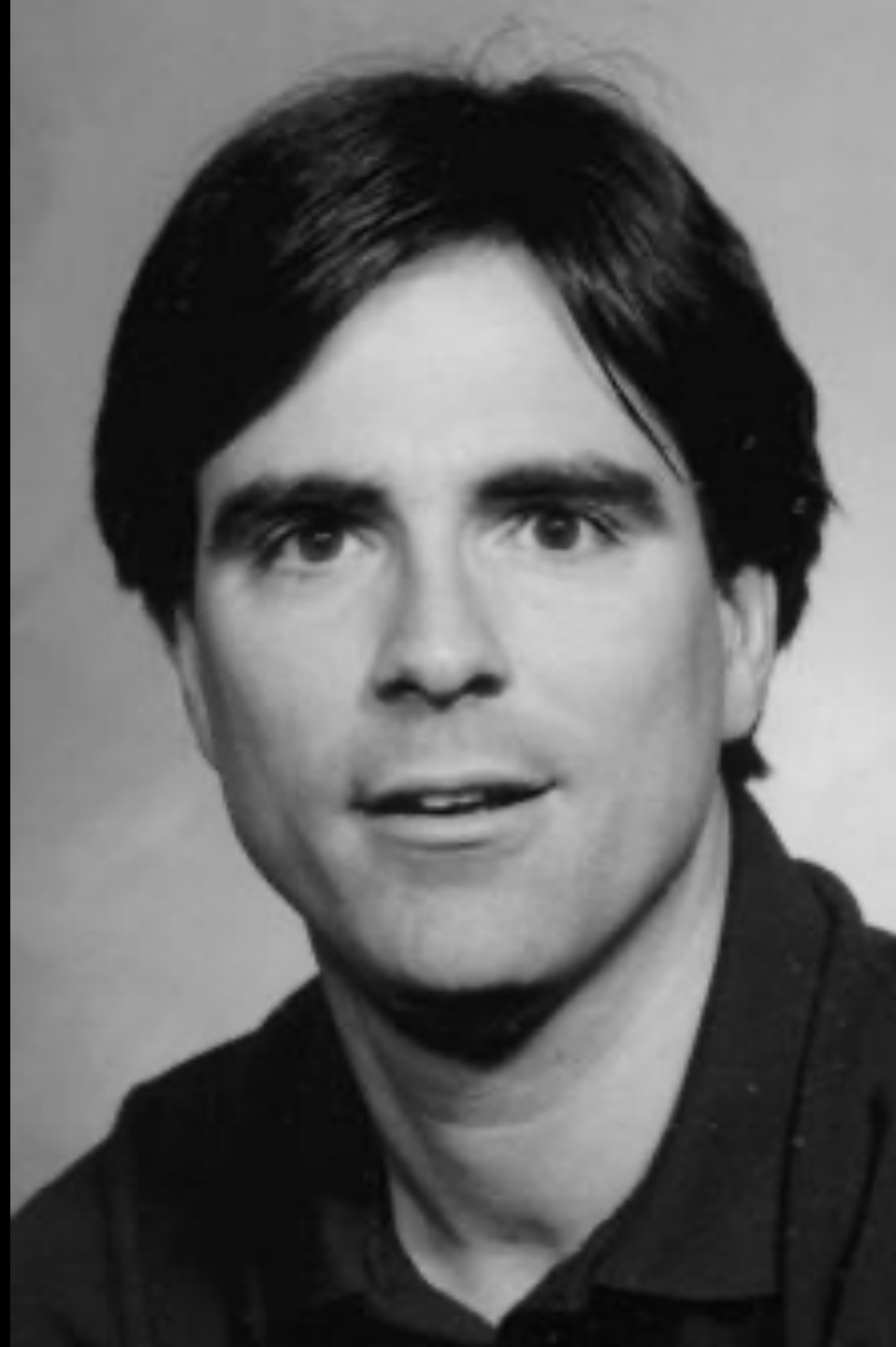
Topic Covered

- Why Threads?
- Threading challenges
- **Goroutine and Synchronization**
- **Goroutine vs OS Thread**
- What is Remote Procedure Call?
- **How RPC works in Go?**
- RPC example



“If Things aren’t going **WELL**, That probably means you’re **LEARNING** a lot and it’ll go **BETTER** later.”

- Randy Pausch



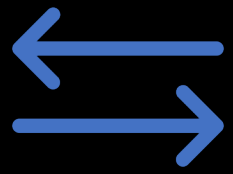
Why Threads?

- Multicore performance
- Concurrency
  - Concurrency is the ability of different parts or units of a program, algorithm, or problem to be executed out-of-order or in partial order, without affecting the final outcomes.
- Convenience
  - In background, once per second, check whether each worker is still alive.

Threading challenges?

Shared data

Critical Section



Coordination between threads

Producer-consumer problem

Deadlock

Goroutine & Synchronization

Shared data

- Sync: sync.Mutex; sync.Cond; sync.Once (Self study)

Goroutine

- How to create a goroutine?

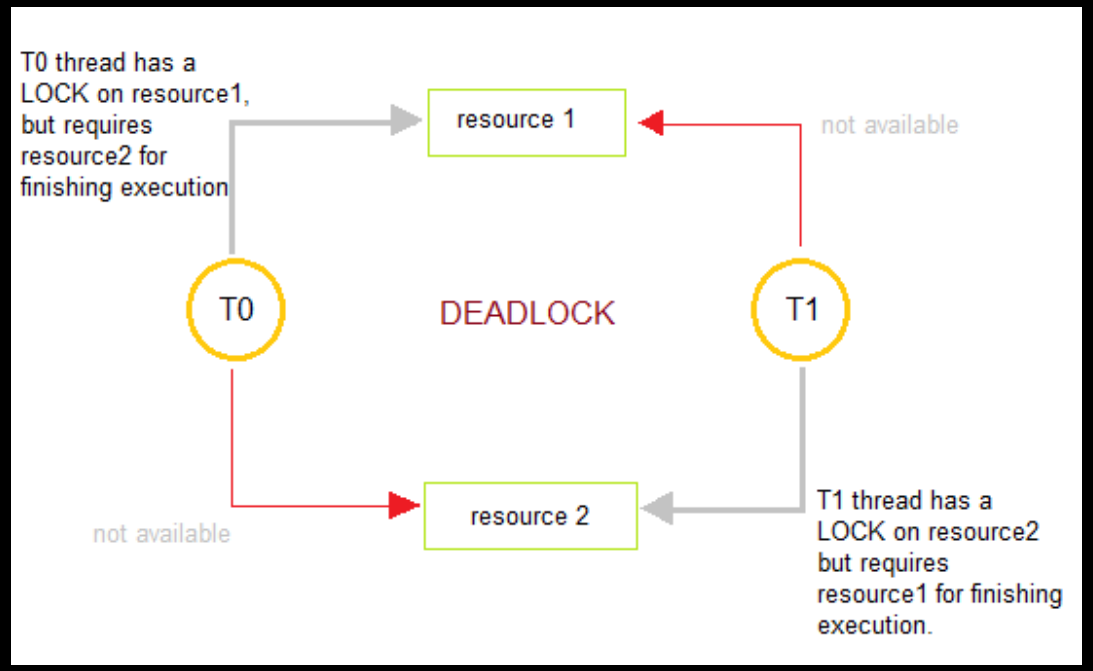
Coordination between Goroutines

- Sync: sync.WaitGroup
- Channel: Unbuffered Channel; Buffered Channel; Channel Pipelines; Unidirectional Channel

Deadlock

- In concurrent computing, a deadlock is a state in which each member of a group waiting for another member, including **itself**, to take actions, such as sending a message or more commonly releasing a lock.

Goroutine & Synchronization



Deadlock

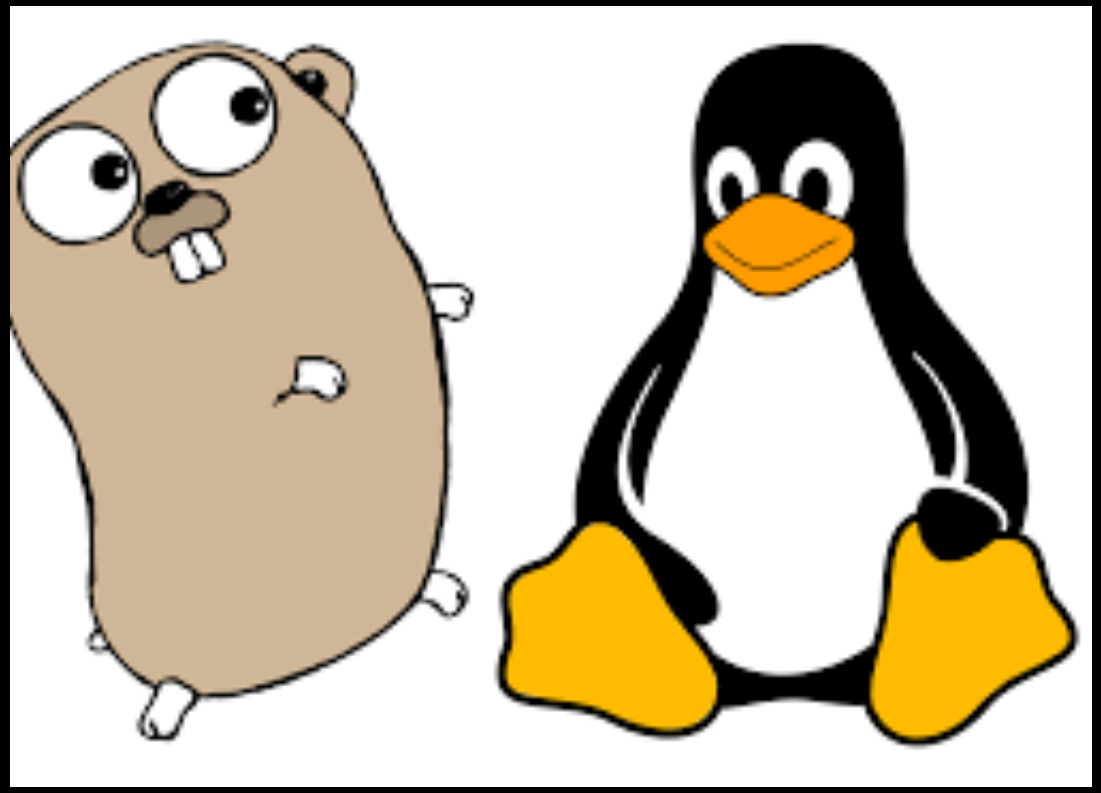
- In concurrent computing, a deadlock is a state in which each member of a group waiting for another member, including **itself**, to take actions, such as sending a message or more commonly releasing a lock.
- Race detector

  Add -race flag to your go build, go run, or go test command.

Goroutine & Synchronization DEMO

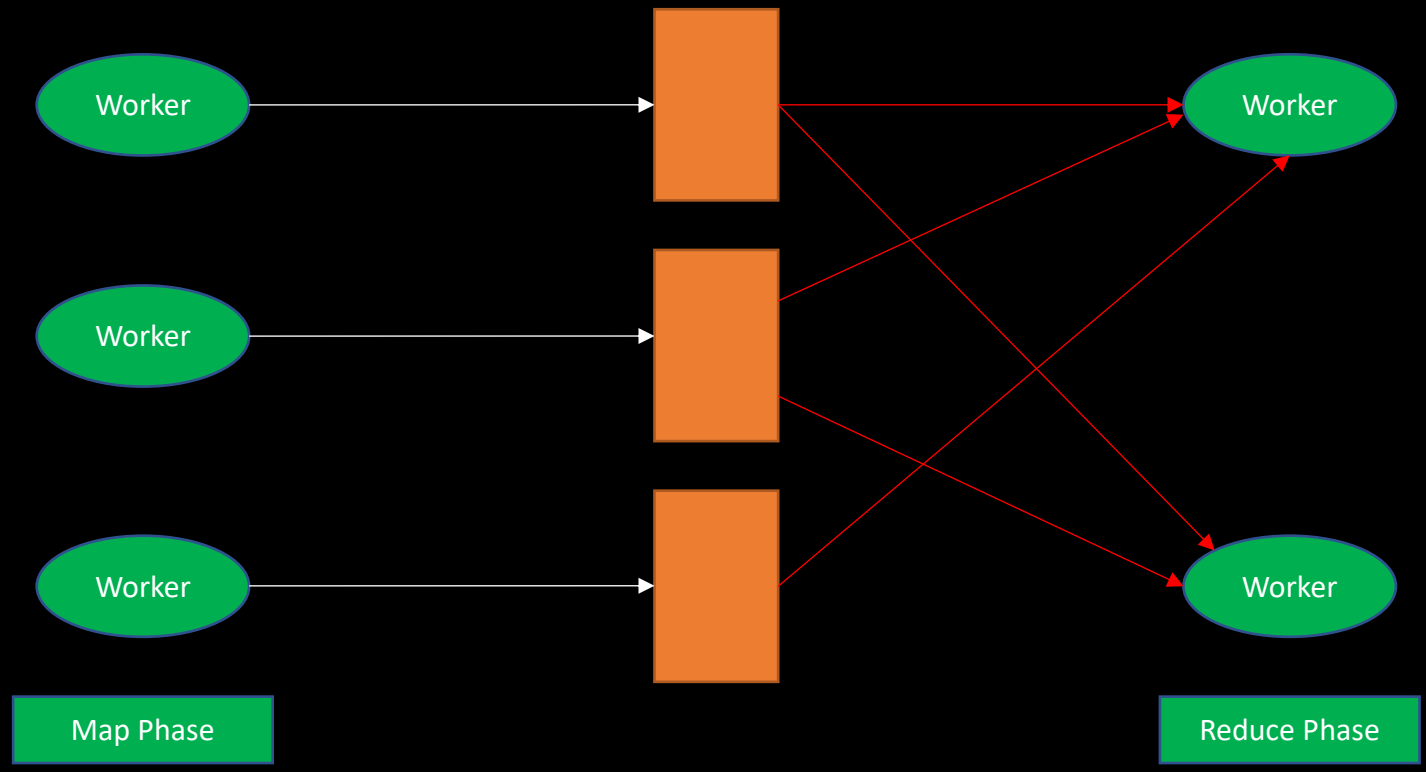
Goroutine VS. OS Thread

- Stack Size:
  - OS: Stack has fixed-size block of memory (2MB)
  - GO: Stack size non-fixed (2KB – 1GB)
- Scheduling:
  - OS threads are scheduled by the OS kernel
  - Go has its own scheduler: m:n scheduling
- Identity:
  - An OS thread has identity
  - A goroutine has no identity

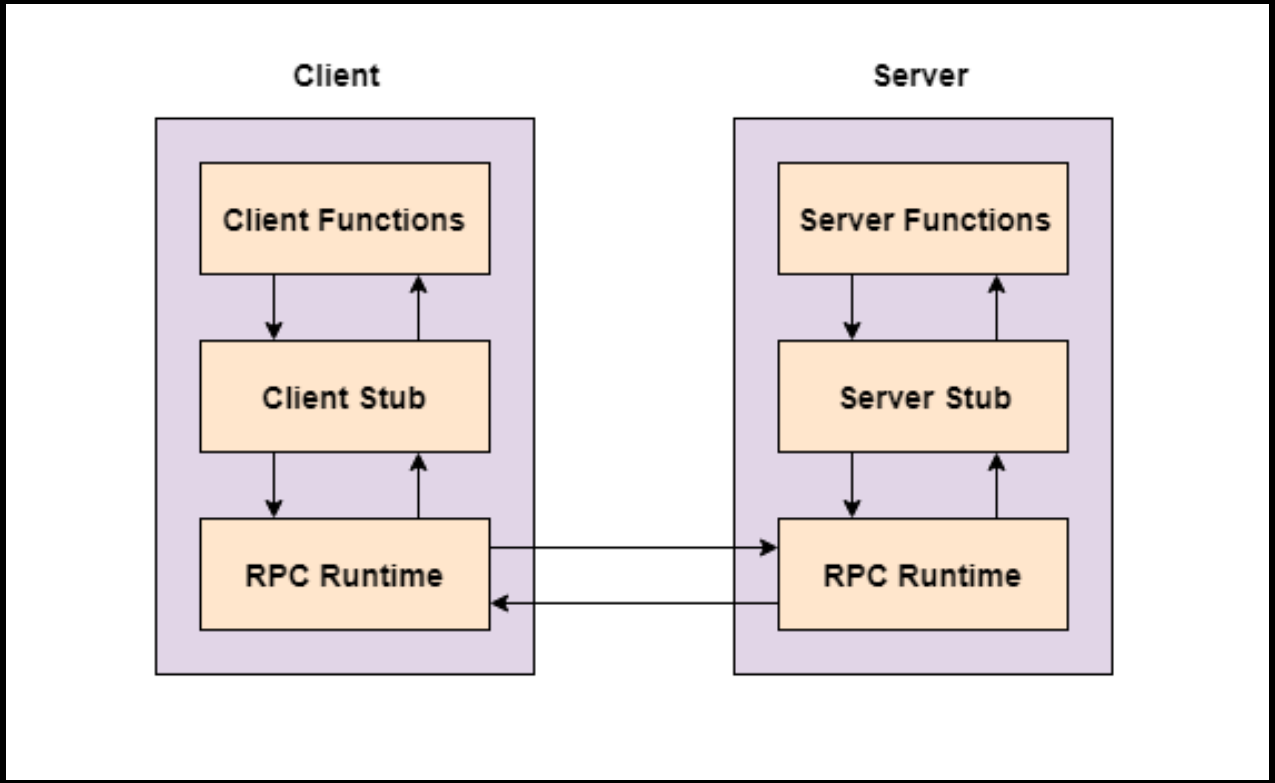


What is remote procedure call (RPC)?

A remote procedure call (RPC) is when a computer program causes a procedure (subroutine) in a different address space (commonly on another computer on a shared network), which is coded as if it were a normal (local) procedure call, without the programmer explicitly coding the details for the remote interaction.



What is remote procedure call (RPC)?



Client stub: Push parameters to the stack;

Client stub: Packs the parameters into a message and make a system call to send the message. (Marshalling)

Client OS: Sends the message from the client machine to the server machine.

Server OS: Passes the incoming packets to the server stub.

Server stub: Unpacks the parameters from the message. (Unmarshalling)

Server stub: Calls the server procedure.

(The reply traces the same steps in the reverse direction.)

RPC failures

Failures in RPC:

- Lost packet, broken network, slow server, crashed server

Best effort:

- Call() waits for response for a while;
- If none arrives, re-send the request;
- Repeat;
- Give up and return error;

Better RPC behavior: “at most once”

- Never resend a duplicate request

Another RPC behavior: “exactly once”

- Lab 3

How RPC works in Go?

A server register an object, making it visible as a service with the name of the type of the object. After registration, exported methods of the object will be accessible remotely.

- Register multiple objects (services) of different types;
- The method’s type is exported;
- The method is exported;
- The method has two arguments, both exported (or builtin) types;
- The method’s second argument is a pointer;
- The method has return type error;

RPC example

Reference

Concurrency in go

The Go Programming Language (Chapter 8 & Chapter 9)

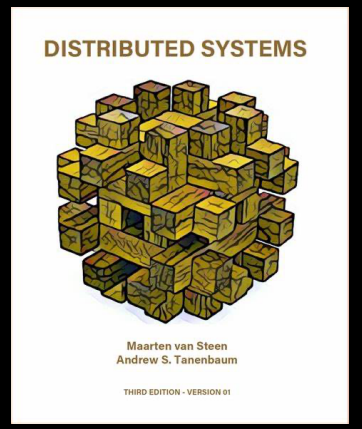
Deadlock

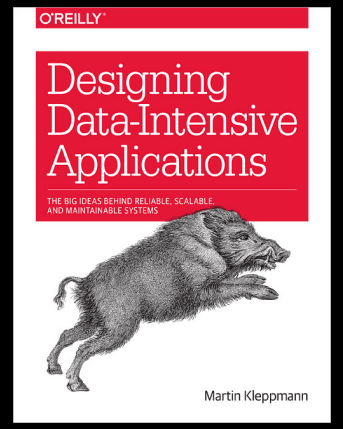
Concurrency

Remote Procedure Call

Go Package rpc

Freely accessible resources





[Code]

[Zoom]

[Course]

[DDIA (O'Reilly)]

[Distributed System 3rd edition]

Calendar: https://docs.google.com/spreadsheets/d/1RsbGpq1cwNSmYn5hcmT8Hv5O4qssl2HXsTcG82RHVQk/edit?usp=sharing

(Internal) Teams: g078pwd

(Public) Discord

(Public) WeChat: add mossaka or Lin1991Wen

Notion: https://www.notion.so/invite/cd6df70a94e7f67f6d21f4c509783d3c9cfd0e69

YouTube: https://www.youtube.com/playlist?list=PL1voNxn5MODMJxAZVvgFHZ0jZ-fuSut68

Questions?