Programación Concurrente y Distribuida

Contexto e introducción al lenguaje Go

Ms.Carlos Jara García
pcsicjar@upc.edu.pe

Ciencias de la Computación
Universidad Peruana de Ciencias Aplicadas

Agosto del 2024

Contenidos

Algunos términos

- Concurrencia es la habilidad de diferentes partes de un programa de ser ejecutadas en desorden u orden parcial sin afectar el resultado final. Ésto permite la ejecución en paralelo de las partes del programa, lo cual incrementa significativamente la velocidad de ejecución en sistemas multiprocesador y multicore. En términos más técnicos, concurrencia se refiere a la propiedad de descomponibilidad de un programa en componentes independientes o parcialmente independientes del orden de ejecución (**Lamport1978**).

Algunos términos

- Concurrencia es programar como la composición de procesos ejecutables de forma independientemente (procesos en el sentido general).
- Paralelismo es programar como la ejecución en simultáneo de cómputo (posiblemente relacionados).
- Concurrencia vs Paralelismo
  - La concurrencia tiene que ver con la estructura, el paralelismo tiene que ver con la ejecución.
  - La concurrencia proporciona una manera de estructurar una solución para resolver un problema que puede (pero no necesariamente) ser paralelizable.
  - No son lo mismo pero están relacionados.

Rob Pike

Contenidos

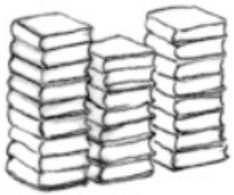
Más términos

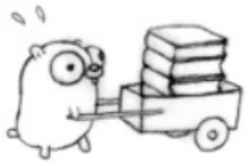
- Concurrencia más comunicación
  - La concurrencia es una forma de estructurar un programa dividiéndolo en partes que se pueden ejecutar de forma independiente.
  - La comunicación es el medio para coordinar las ejecuciones independientes.

Rob Pike

Nuestro problema

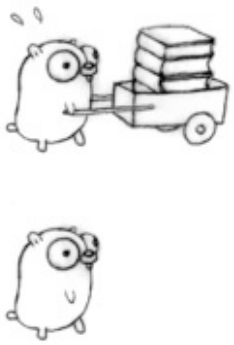
Mover una pila de manuales de lenguajes obsoletos al incinerador.





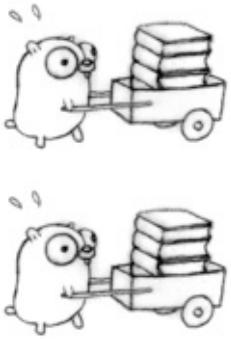
Con solo un gopher esto tomará demasiado tiempo.

Más gophers!



Más gophers no son suficientes; ellos necesitan mas carretillas.

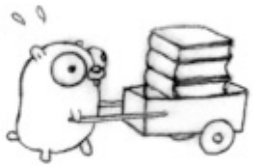
Más gophers y más carretillas

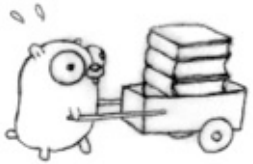


Esto irá rápido, pero esto creará cuello de botella en la pila y el incinedador.
También se necesita sincronizar a los gophers.
Un mensaje (es decir, una comunicación entre los gophers) se hará.

Duplicar todo

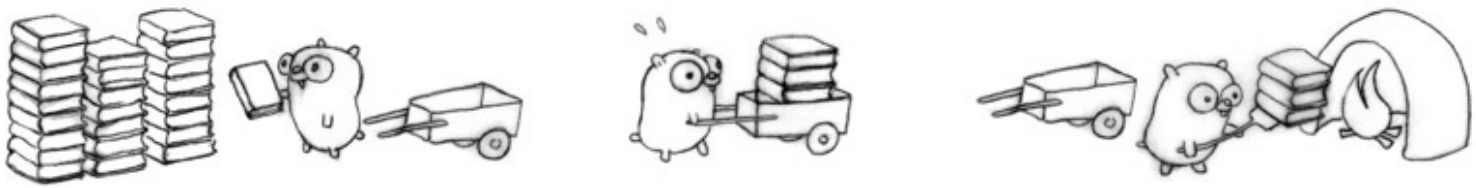
Elimina el cuello de botella; hace ello realmente independiente.


Esto consumirá entradas dos veces más rápido.

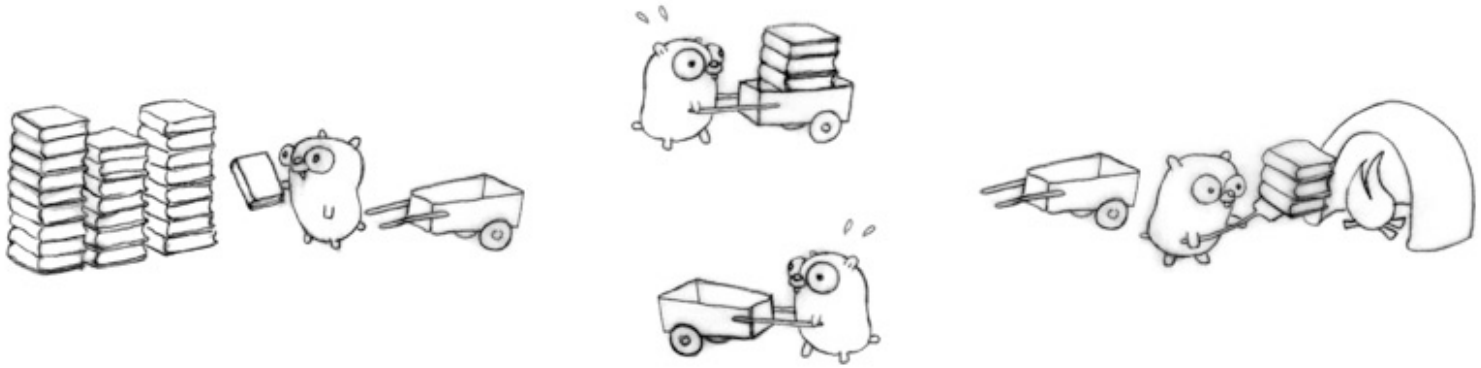
Otro diseño



Tres gophers en acción, pero con probables retrasos.
Cada gopher es un procedimiento que se ejecuta independientemente, más coordinación (comunicación).

Concurrencia más refinada

Agregar otro procedimiento gopher para retornar las carretillas vacias.



Cuatro gophers en acción para mejorar el flujo, cada uno hace una tarea sencilla.
Si organizamos todo correctamente (algo inverosímil pero no imposible), será cuatro veces más rápido que nuestro diseño original de un gopher.

Procedimientos concurrentes

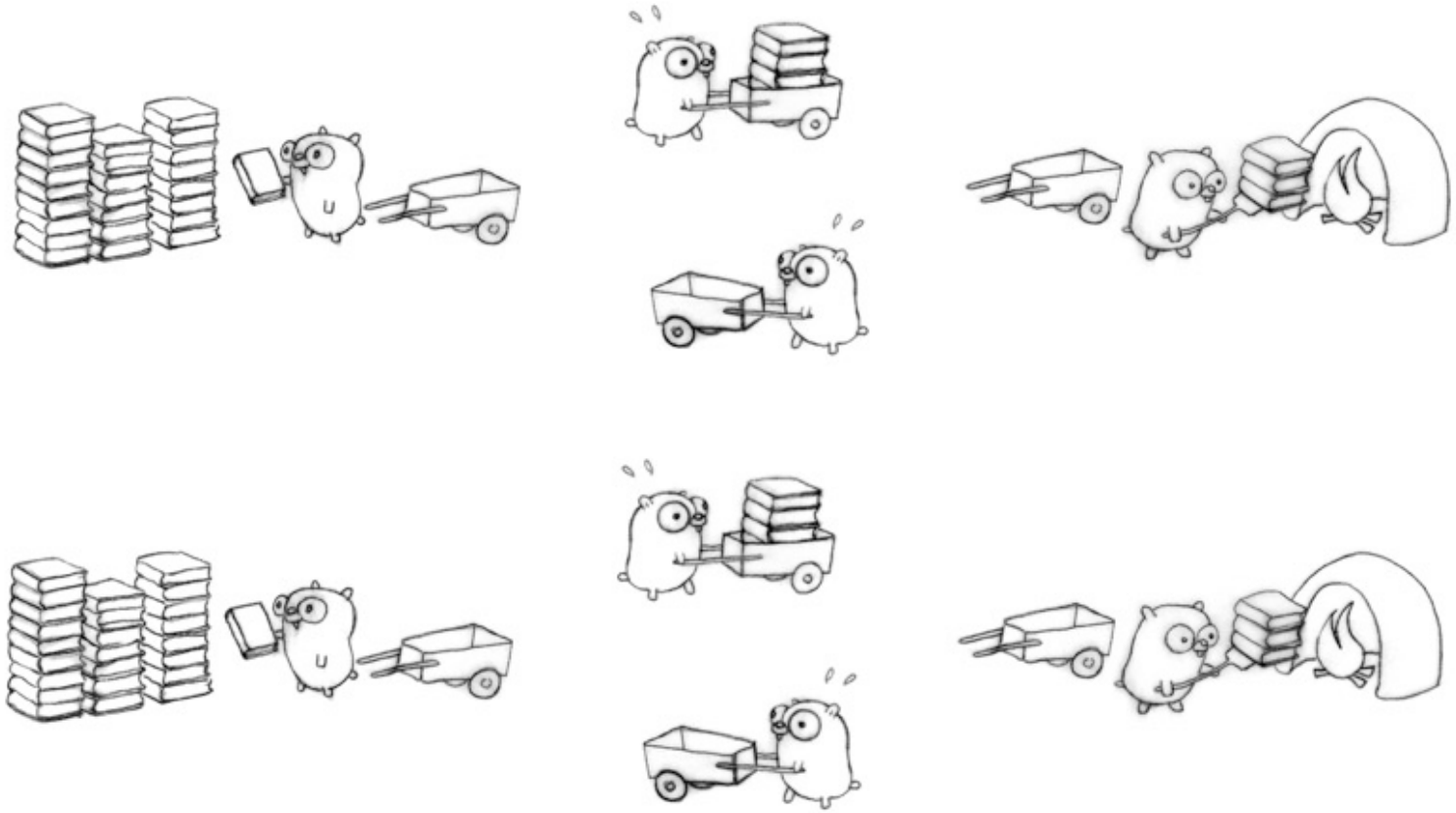
Cuatro procedimientos de distintos gopher.

- Cargar libros en la carretilla.
- Mover la carretilla para el incinerador.
- Descargar el contenido de la carretilla dentro del incinerador.
- Retornar la carretilla vacia.

Diferentes diseños concurrentes habilitan diferentes maneras para paralelizar.

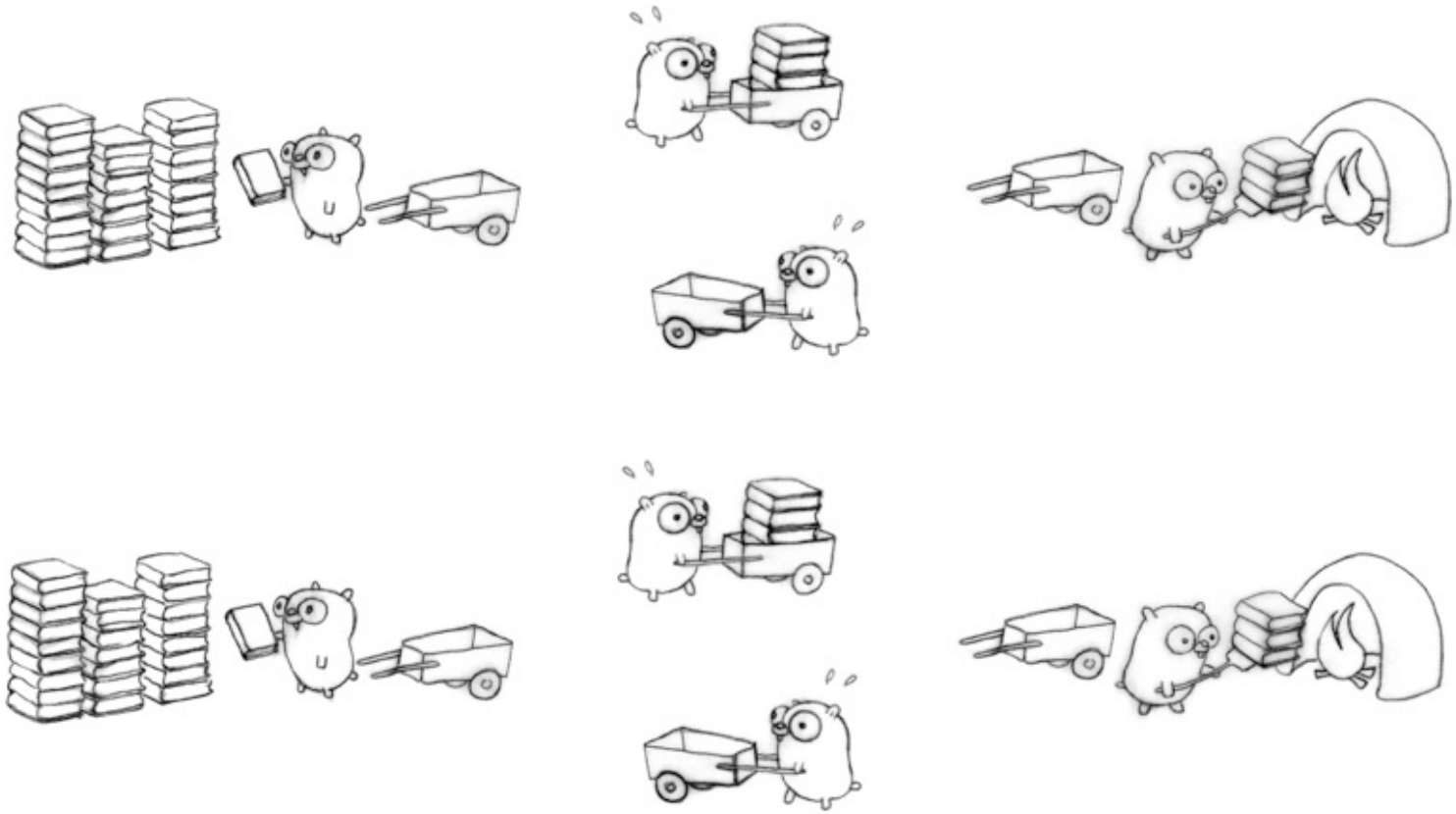
Más Paralelización!

Ahora podemos paralelizar en el otro eje; el diseño concurrente hace esto fácil. Ocho gophers, todos ocupados.



O quizá no haya ninguna paralelización

Tenga en cuenta que incluso si solo hay un gopher activo a la vez (paralelismo cero), sigue siendo una solución correcta y concurrente.

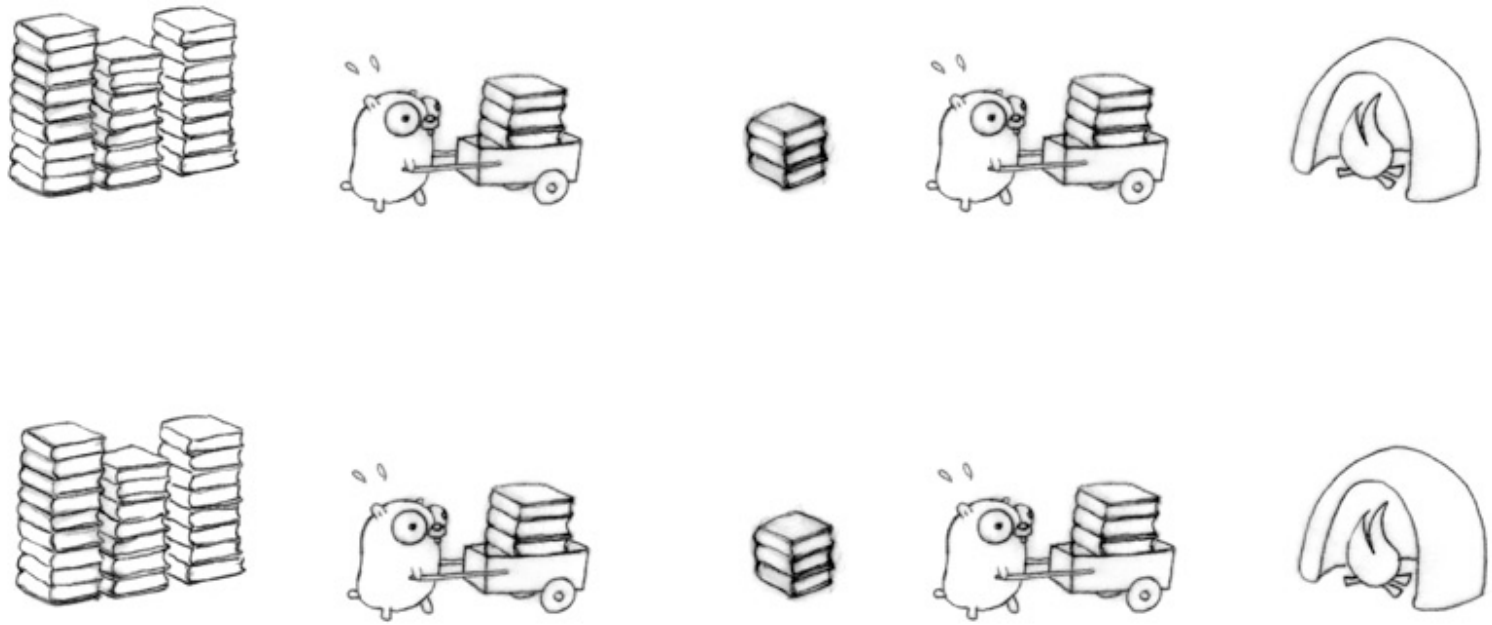


Otro diseño

Aquí está otra manera para estructurar el problema como la composición concurrente de procedimientos gopher.
Dos procedimientos gopher, más una pila de preparación.

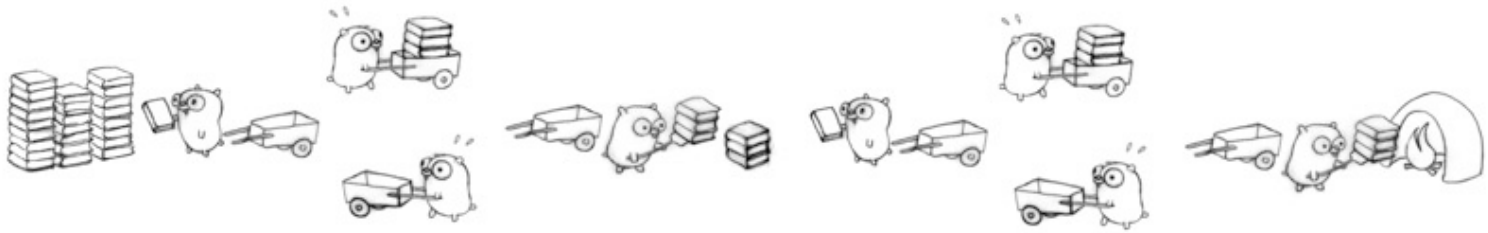
Paralelizable de manera usual

Ejecutar más procedimientos concurrentes para obtener un mayor procesamiento.



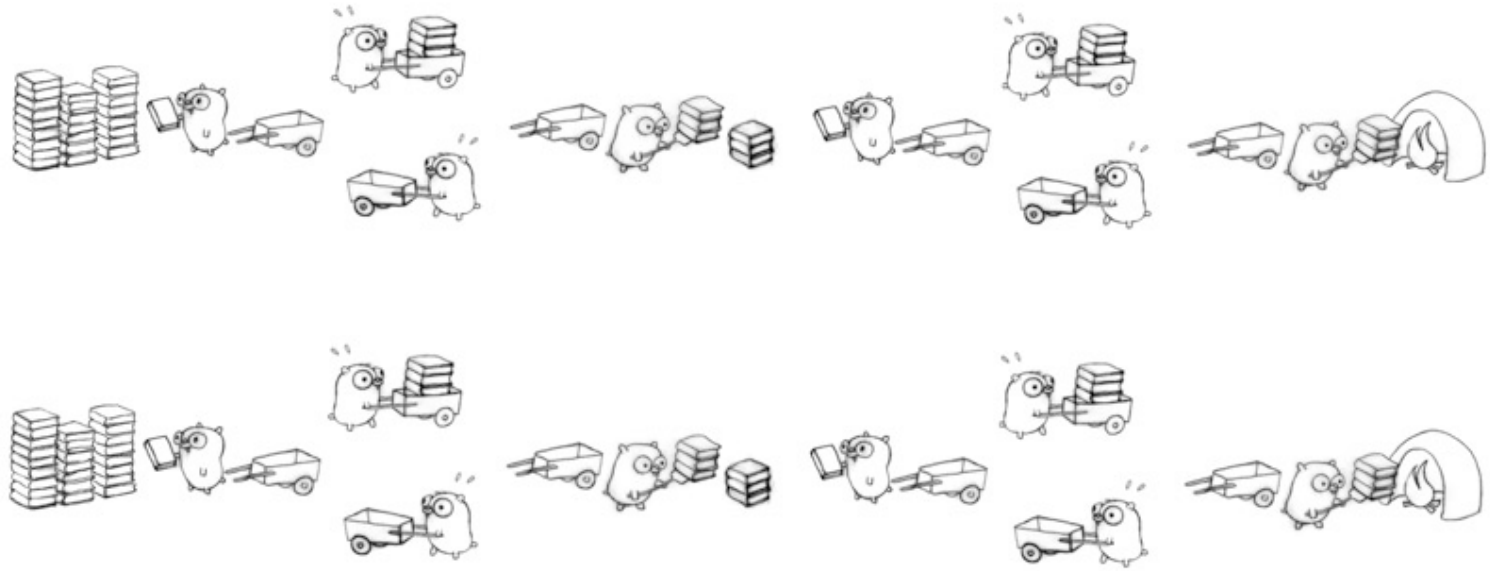
O una manera diferente

Llevando la pila de preparación hacia el modelo concurrente multi-gopher.



Lleno de optimización

Usando todas nuestras técnicas. Dieciséis gophers trabajando duro!



Retornando a computación

En nuestro problema de transporte de libros, sustituimos:

- pila de libros => contenido web
- gopher => CPU
- caretilla => serialización, renderizado, networking
- incinerador => proxy, browser o otro consumidor

Esto se convierte en un diseño concurrente para un escalable servicio Web. Gophers sirviendo contenido Web.

Poniendo en práctica lo aprendido

Elabore el diseño de un modelo óptimo concurrente multi-procedimientos gopher para el siguientes caso:

- Importación y Venta de productos de la china.

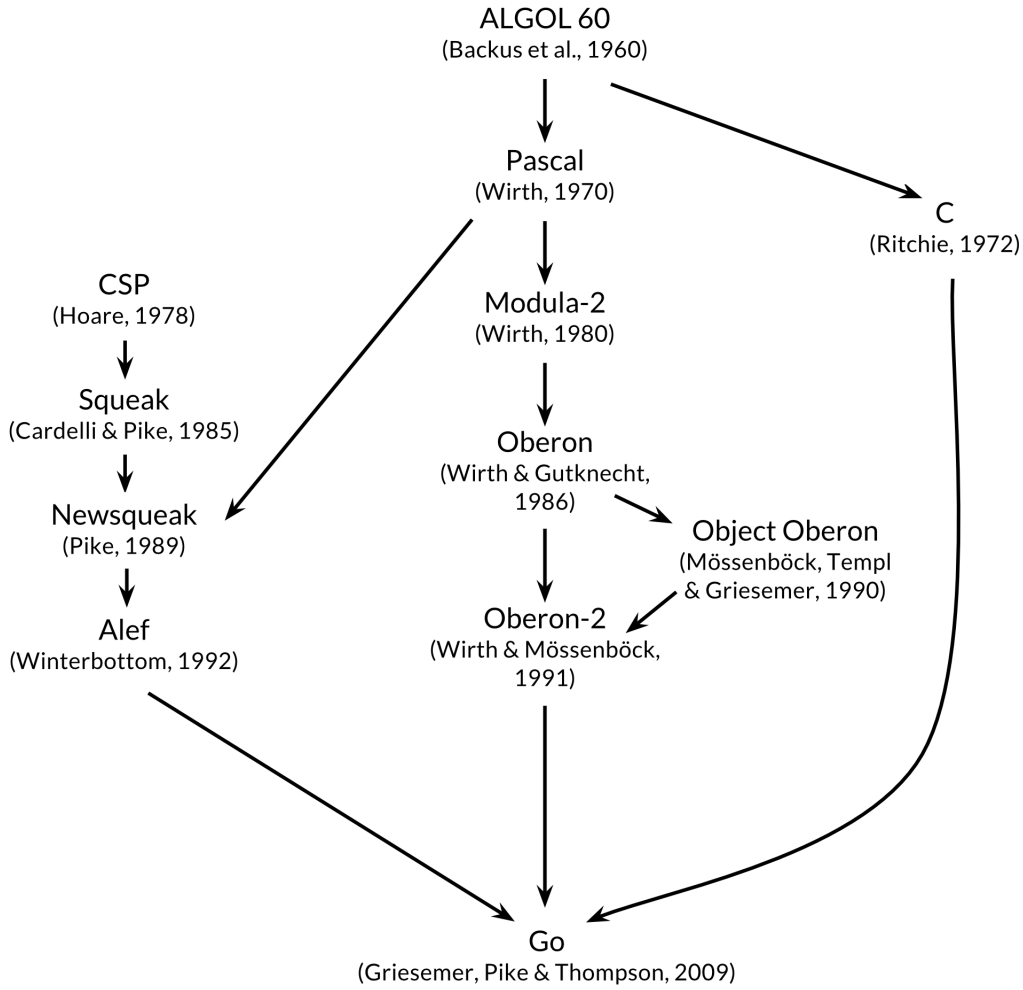
Contenidos

Cómo nace GO

El lenguaje de programación Go (también conocido como Golang) fue creado por Google y fue anunciado por primera vez en noviembre de 2009. El equipo principal de desarrollo incluyó a Rob Pike, Ken Thompson y Robert Griesemer, todos ellos con una larga experiencia en el desarrollo de software y en la creación de lenguajes de programación.

Influencia de los lenguajes de programación

La siguiente figura muestra las influencias más importantes de los lenguajes de programación anteriores en el diseño de Go.



Go realmente soporta concurrencia

- Realmente.
- Es una rutina crear miles goroutines en un solo programa. (Una vez depurado un programa después de haber creado 1,3 millones).
- Las pilas incian pequeñas, pero crecen y se reducen según sea necesario.
- Los goroutines no son gratuitas, pero son muy baratas.

Go realmente soporta concurrencia

- Goroutine es una función que se ejecuta independientemente en un mismo espacio de dirección que otras goroutines.
  Ejemplo:
  f() // call f(); wait for it to return
  go f() // create a new goroutine that calls f(); don't wait
- Goroutine no son Threads
  - Son un poco como threads pero son mucho más económicos.
  - Se multiplexan en subprocesos del sistema operativo según sea necesario.
  - Si un goroutine se bloquea esperando entrada/salida (por ejemplo, al leer de un archivo o al esperar una solicitud de red), en lugar de quedarse inactivo y ocupar recursos, Go puede programar otro goroutine activo en ese subproceso..

Contenidos

Recursos Web

Accedemos al tour de GO: https://go.dev/tour/list