1. Record Nr. UNINA9911008982603321

Autore	Serdar Burak
Titolo	Effective Concurrency in Go : Develop, Analyze, and Troubleshoot High Performance Concurrent Applications with Ease
Pubbl/distr/stampa	Birmingham : , : Packt Publishing, Limited, , 2023 ©2023
ISBN	9781804615980 1804615986
Edizione	[1st ed.]
Descrizione fisica	1 online resource (212 pages)
Disciplina	005.133
Soggetti	Go (Computer program language) Computer multitasking
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Description based upon print version of record.
Nota di contenuto	Cover -- Title Page -- Copyright and Credit -- Dedicated -- Contributors -- Table of Contents -- Preface -- Chapter 1: Concurrency - A High-Level Overview -- Technical Requirements -- Concurrency and parallelism -- Shared memory versus message passing -- Atomicity, race, deadlocks, and starvation -- Summary -- Question -- Further reading -- Chapter 2: Go Concurrency Primitives -- Technical Requirements -- Goroutines -- Channels -- Mutex -- Wait groups -- Condition variables -- Summary -- Questions -- Chapter 3: The Go Memory Model -- Why a memory model is necessary -- The happened-before relationship between memory operations -- Synchronization characteristics of Go concurrency primitives -- Package initialization -- Goroutines -- Channels -- Mutexes -- Atomic memory operations -- Map, Once, and WaitGroup -- Summary -- Further reading -- Chapter 4: Some Well-Known Concurrency Problems -- Technical Requirements -- The producer-consumer problem -- The dining philosophers problem -- Rate limiting -- Summary -- Chapter 5: Worker Pools and Pipelines -- Technical Requirements -- Worker pools -- Pipelines, fan-out, and fan-in -- Asynchronous pipeline -- Fan-out/fan-in -- Fan-in with ordering -- Summary -- Questions -- Chapter 6: Error Handling -- Error handling -- Pipelines -- Servers -- Panics -- Summary -- Chapter 7: Timers and

Tickers -- Technical Requirements -- Timers - running something later -- Tickers - running something periodically -- Heartbeats -- Summary -- Chapter 8: Handling Requests Concurrently -- Technical Requirements -- The context, cancelations, and timeouts -- Backend services -- Distributing work and collecting results -- Semaphores - limiting concurrency -- Streaming data -- Dealing with multiple streams -- Summary -- Chapter 9: Atomic Memory Operations -- Technical Requirements -- Memory guarantees -- Compare and swap. Practical uses of atomics -- Counters -- Heartbeat and progress meter -- Cancellations -- Detecting change -- Summary -- Chapter 10: Troubleshooting Concurrency Issues -- Technical Requirements -- Reading stack traces -- Detecting failures and healing -- Debugging anomalies -- Summary -- Further reading -- Index -- Other Books You May Enjoy.

Sommario/riassunto

Gain a deep understanding of concurrency and learn how to leverage concurrent algorithms to build high-throughput data processing applications, network servers and clients that scale. Key Features Learn about the Go concurrency primitives, Go memory model, and common concurrency patterns Develop the insights on how to model solutions to real-life problems using concurrency Explore practical techniques to analyze how concurrent programs behave Book Description The Go language has been gaining momentum due to its treatment of concurrency as a core language feature, making concurrent programming more accessible than ever. However, concurrency is still an inherently difficult skill to master, since it requires the development of the right mindset to decompose problems into concurrent components correctly. This book will guide you in deepening your understanding of concurrency and show you how to make the most of its advantages. You'll start by learning what guarantees are offered by the language when running concurrent programs. Through multiple examples, you will see how to use this information to develop concurrent algorithms that run without data races and complete successfully. You'll also find out all you need to know about multiple common concurrency patterns, such as worker pools, asynchronous pipelines, fan-in/fan-out, scheduling periodic or future tasks, and error and panic handling in goroutines. The central theme of this book is to give you, the developer, an understanding of why concurrent programs behave the way they do, and how they can be used to build correct programs that work the same way in all platforms. By the time you finish the final chapter, you'll be able to develop, analyze, and troubleshoot concurrent algorithms written in Go. What you will learn Understand basic concurrency concepts and problems Learn about Go concurrency primitives and how they work Learn about the Go memory model and why it is important Understand how to use common concurrency patterns See how you can deal with errors in a concurrent program Discover useful techniques for troubleshooting Who this book is for If you are a developer with basic knowledge of Go and are looking to gain expertise in highly concurrent backend application development, then this book is for you. Intermediate Go developers who want to make their backend systems more robust and scalable will also find plenty of useful information. Prior exposure Go is a prerequisite.