پروژه اول – پیاده سازی یک سرور از دیتابیس key-value ها

هدف از این پروژه، آشنایی سریع و مقدماتی با زبان برنامه نویسی Go و قابلیت های آن در زمینه concurrency، و یادآوری مفاهیم Socket programming است. در این پروژه، شما باید یک سرور (server) ساده را با زبان برنامه نویسی Go پیاده سازی نمایید که حاوی یک دیتابیس با درایه هایی به فرم key-value است. در این پروژه، چندین client می‌توانند به طور همزمان با سرور تعامل داشته باشند. هر client یک درخواست (request) که حاوی یک کلید می باشد، را برای سرور ارسال می نماید. در طرف سرور، کلید در پایگاه داده مورد جستجو قرار می‌گیرد و value متناظر با آن از طریق روال ()get به client متناظر متصل شده، برمی گردد.

- مشخصات Server:

کدهای اولیه (framework) برای پیاده سازی این سرور key-value در اختیار شما قرار دارد. این قالب اولیه عملیات های زیر را در اختیار شما قرار خواهد داد:

۱. ()createDB – این تابع یک دیتابیس خالی را در طرف سرور ایجاد می‌نماید.
۲. set (K string, V []byte) – این تابع یک جفت key-value را در دیتابیس ذخیره می نماید. اگر مقدار کلید K قبلا در دیتابیس وجود داشته باشد تنها مقدار V متناظر با آن بروز رسانی می شود.
۳. get (K string) []byte – این تابع مقدار value متناظر ذخیره شده برای کلید K را برمی گرداند.

تمام کلیدها و مقادیر متناظرشان به فرم *[_a-z0-9][a-z] هستند. این سرور باید مشخصات زیر را داشته باشد:

۱- سرور با تمام client ها، تنها از طریق goroutines و channels می تواند تعامل داشته باشد و آنها را مدیریت نماید. چندین client باید قادر باشند به طور همزمان به سرور متصل/غیرمتصل شوند.
۲- وقتی که یک client قصد دارد تا یک value را در سرور ذخیره نماید. client، رشته پیام زیر را برای سرور ارسال و در پایگاه داد ذخیره می‌کند:

set, key, value

وقتی که یک client قصد دارد تا یک value را از سرور دریافت یا بازیابی نماید، client، رشته پیام زیر را برای سرور ارسال می‌نماید:

get, key

این پیام ها، تنها پیام هایی هستند که هر client می تواند برای سرور ارسال نماید. شما در این پروژه، باید هر کدام از این رشته پیام ها را بدرستی تجزیه و تحلیل و عمل مناسب با آن را انتخاب و اجرا نمایید.

۳- وقتی که سرور پیام get ارسال شده توسط یک client را خواند، با رشته پیام زیر به تمام client های متصل شده، از جمله client ای که پیام get را ارسال کرده است، جواب می دهد:

key, value

هیچ پاسخی برای هیچ کدام از client ها در جواب به پیام set ارسال نمی‌شود.

۴- سرور فرض می کند که تمام پیام ها به صورت line-oriented هستند. به عبارت دیگر، هر پیامی که ارسال و یا دریافت می شود با کاراکتر (\n) newline پایان می یابد.

۵- در طرف سرور، هیچ فرضی در رابطه با thread-safe بودن توابع (منظور توابع لیست شده در بالاست که با key-value ها سروکار دارند) لحاظ نشده است. به عبارت دیگر، شما به عنوان قسمتی از پروژه، وظیفه دارید تا mutual exclusion را در هنگام دسترسی به دیتابیس مذکور تضمین، و از رخدادن race condition جلوگیری نمایید.

۶- فرض نمایید، تمام پیام های get ای که توسط سرور دریافت می شوند، کلیدشان قبلا در دیتابیس ذخیره شده است. به عبارت دیگر هیچ client ای یک کلید ذخیره نشده را درخواست نمی نماید.

۷- در سرور باید یک تابع ()Count پیاده سازی گردد تا تعداد clientهایی را که در حال حاضر به سرور متصل هستند را به عنوان خروجی برگرداند.

۸- سرور باید قادر به پاسخگویی به client های slow-reading باشد. برای درک بهتر وضعیت این clientها به سناریوی زیر توجه نمایید. فرض نمایید که یک client برای یک مدت زمان طولانی، روال Read (برای خواندن پیام ارسال شده توسط سرور بر روی اتصال TCP) را فراخوانی نکند. اگر در طول این مدت زمان، سرور به نوشتن پیام بر روی اتصال TCP همان client ادامه دهد، در نهایت، بافر خروجی اتصال TCP به حداکثر ظرفیت ممکن خود می رسد و درخواست های بعدی که توسط سرور برای نوشتن بر روی اتصال صادر می شوند، بلاک خواهند شد.
برای مدیریت این حالت، بهتر است که سرور یک صف به طول حداکثر ۵۰۰ پیام را در اختیار client قرار دهد. این صف از پیام ها توسط client خوانده خواهد شد. اگر تعداد پیام های ارسال شده به یک client که در بافر خروجی سرور ذخیره شده اند، به حداکثر ظرفیت خود (در اینجا ۵۰۰ پیام) برسد، پیام های بعدی حذف خواهند شد. وقتی که این slow-reading client دوباره شروع به خواندن پیام ها نمود، سرور باید از تحویل هر کدام از پیام های بافرشده به client اطمینان حاصل نماید. (توجه: برای پیاده سازی این ویژگی، از یک channel به عنوان بافر استفاده نمایید.)

- موارد ضروری که در انجام پروژه، باید لحاظ گردند:

چندین روش متفاوت برای پیاده سازی این پروژه وجود دارد که روش پیاده سازی شده توسط شما باید تمام ۴ مورد زیر را رعایت نماید:

۱- این پروژه باید بصورت انفرادی انجام گردد. در انتها هم باید یک گزارش کامل از مراحل انجام پیاده سازی به همراه کدها (منظور فایل server.go می باشد)، تحویل داده شود.

۲- در پیاده سازی پروژه حق استفاده از locks و mutexes را ندارید. همگام سازی تمام عملیات ها باید توسط goroutines، channels و دستور select (که مبتنی بر channelها است) پیاده سازی گردد. بنابراین، تمام پیاده سازی هایی که با

استفاده از روش های lock، mutexes و روش هایی که رفتار این دو روش را شبیه سازی می کنند، صورت بگیرد، به کسر نمره شما از پروژه انجام شده ختم خواهد شد.

۳- برای این منظور، تمام package هایی که ممکن است نیاز به استفاده از آنها را داشته باشید عبارتند از: bufio، fmt، net، bytes و strconv.

۴- کدهای برنامه نویسی شما باید با استفاده از go fmt فرمت بندی شده باشد. و از قواعد استانداد نامگذاری در Go تبعیت نمایید.

- راهنمای استفاده از کدهای موجود:

کدهای اولیه برای انجام این پروژه در مسیر src/example.com /p1 قرار دارند، که شامل ۴ فایل می باشد:

۱. server.go تنها فایلی است که شما باید آن را تکمیل و کدهایتان را اضافه نمایید و در نهایت تحویل دهید.
۲. kv.go حاوی توابعی است که برای انجام عملیات در دیتابیس موردنظر استفاده می شوند. حاوی ۳ تابع است که به طور مستقیم در فایل server.go برای پیاده سازی سرور استفاده می شوند.
۳. interface.go حاوی یک اینترفیس است که شما در این پروژه توابع آن را باید پیاده سازی نمایید. توجه نمایید خود این فایل نباید تغییر کند.
۴. server_test.go حاوی مجموعه تست هایی است که برای نمره دهی به پروژه پیاده سازی شده توسط شما، اجرا خواهند شد. شما هم می توانید آنها را برای تست پروژه خود، قبل از تحویل، اجرا نمایید.

برای آشنایی بیشتر با نحوه‌ی ساخت، اجرا و تست پروژه تان فایل README.txt را مطالعه نمایید.

- نکات:

۱- پیاده سازی پروژه را هر چه سریعتر آغاز نمایید، چرا که مهلت انجام پروژه ۲ هفته است و این زمان قابل تمدید نمی باشد.

۲- هدف اصلی این پروژه، ایجاد یک فرصت برای پیاده سازی و برنامه نویسی همزمان است. در این پروژه، تجزیه وتحلیل مساله از منظر پیدا کردن تکه کدهایی که نیاز به قفل کردن دارند، نادیده گرفته می شود. و قصد ما اینست که شما با استفاده از ساختارهای Go، که به طور ذاتی قابلیت mutual exclusion را فراهم می کنند، به این هدف دست پیدا کنید. چرا که شما برای بسیاری از پروژه هایی که در آینده انجام خواهید داد، باید از این روش استفاده نمایید. برای تمرکز و تسلط بر این موضوع و پیاده سازی این پروژه، بخش Tour of Go کفایت می نماید.

۳- رویکرد این پروژه، یک روش ترتیبی است. ابتدا سعی نمایید تا یک client و سرور را به هم متصل کنید تا با یکدیگر ارتباط برقرار نمایند. در مرحله بعد، سعی نمایید تا از طریق goroutineها و channelها، ارتباط را برقرار نمایید. برای این منظور، سایت Tour of Go را مطالعه نمایید و از کلاس های استاندارد موجود استفاده نمایید. در نهایت شما می توانید به سادگی ویژگی های دیگری را به طور پیوسته، به پروژه اضافه نمایید.

۴- همراه با کدهای اولیه، دو برنامه دیگر srunner و crunner هم در فایل پروژه قرار دارند. با اجرای srunner سرور شما آغاز به کار می کند و تا ابد اجرا می گردد. برنامه crunner می تواند توسط شما پیاده‌سازی گردد تا client های ساده و آزمایشی را اجرا نماید. البته پیاده سازی crunner در نمره دهی این پروژه لحاظ نمی شود ولی توصیه ما اینست که در مراحل اولیه توسعه پروژه خود از آن استفاده نمایید.

۵- نوع داده های (data types) استفاده شده، در توابع مربوطه را به خوبی به یاد داشته باشید و به هیچ عنوان آنها را تغییر ندهید.