**گزارش مربوط به نحوه مدریت کاربران و ذخیره سازی اطلاعات جانبی در redis**

**مقدمه:**

بر اساس پروپوزال نیازمند آن هستیم که بتوانیم برای کاربران سطح دسترسی تعیین کنیم و بتوانیم میان اپلیکیشن های سرور، سرور پردازشی ، و اپلیکشین کلاینت هماهنگی ایجاد کنیم. همانطور که در گزارش های پیشین دیدم یکی از راهکارهای یکپارچه سازی ذخیره کردن اطلاعات در پایگاه داده است. به همین منظور نیاز داریم که یک پایگاه داده به پروژه اضافه کنیم. ویژگی پایگاه داده ای قصد داریم به پروژه اضافه کنیم سریع بودن و توسعه پذیری و قابلیت کنترل کاربران است. سریع ترین پایگاه داده که در حال حاضر موجود است ، Redis است. Redis اطلاعات را در حافظه (رم) نگهداری میکند و بر اساس آن سرویس میدهد بنابرین میتوان که سریعترین پایگاه داده موجود است. در ادامه به بررسی redis میپردازیم

**redis چیست:**

یک ذخیره کننده ساختار اطلاعات، داخل حافظه‌ای متن‌باز است که می­تواند به‌عنوان پایگاه­داده، حافظه نهان و ابزاری برای یکپارچه‌سازی عمل کند. Redis از ساختارهای اطلاعات متفاوتی ازجمله؛ رشته­ها، هش­ها، لیست­ها، مجموعه­ها، مجموعه­های مرتب‌شده برای کاربردهای متفاوت و جریان­های اطلاعات (سری زمانی‌ها) پشتیبانی می­کند و توانایی اجرا بر روی خوشه­ای از کامپیوترها را دارا است که بزرگ‌ترین مزیت استفاده از Redis است.

در این نرم‌افزار بعضی از امکانات محاسباتی ازجمله اضافه کردن به رشته‌ها، تغییر در اعداد، تغییر در لیست‌ها، محاسبه اشتراکات دو مجموعه، بررسی المان‌های منحصربه‌فرد و ...را انجام داد. Redis به زبان **ANSI C** نوشته‌شده و بر روی همه‌ی سیستم‌عامل‌ها بدون هیچ‌گونه پیش‌نیاز قابل‌اجرا است. اگرچه این برنامه بر روی همه‌ی سیستم‌عامل‌ها توانایی اجرا دارد ولی خود برنامه، استفاده از سری سیستم‌عامل‌های Linux را پیشنهاد می­دهد.

درست است که Redis اطلاعات را در حافظه ذخیره می‌کند اما به این معنی نیست که پس از خاموش شدن و یا هر اتفاقی که باعث خالی شدن حافظه شود، داده‌های ما پاک می‌شوند. بلکه Redis برای نگه‌داری دائمی داده‌ها آن‌ها را با توجه به تنظیماتی که برای آن مشخص کرده‌ایم به دیسک اصلی سیستم منتقل کرده و بعد از پاک شدن حافظه مجدد می‌تواند آن‌ها را منتقل کند و کار را از سر بگیرد. این ویژگی باعث شده اصطلاحاً به آن **on-disk persistence** بگویند.

زمانی از ذخیره‌سازی موقت[[1]](#footnote-1) استفاده می‌شود که قصد داشته باشیم دسترسی به هارددیسک کمتر انجام شود. به‌عبارت‌دیگر در ذخیره­سازی موقت، اطلاعات در حافظه موقت ذخیره می‌شود که این فرآیند سرعت دسترسی به اطلاعات و بارگذاری آن‌ها را افزایش می‌دهد. از این طریق در کنار صرفه‌جویی در زمان و افزایش سرعت، دسترسی کمتری به منابع موردنیاز انجام می‌شود. این امر نیز به بهینه‌سازی بیشتر کمک می‌کند. به این نکته نیز باید اشاره کرد که در Redis اطلاعات در حافظه ذخیره می‌شوند، این امر باعث می‌شود دسترسی به آن‌ها با سرعت بسیار بیشتری انجام شود؛ اما این سکه روی دیگری نیز دارد و امکان ذخیره‌سازی دائمی اطلاعات را در Redis وجود نخواهد داشت. در این پژوهش ازآنجاکه پس از تجمیع اطلاعات نیازمند پردازش این اطلاعات هستیم تا به رانندگی افراد امتیازی اختصاص دهیم از این نرم‌افزار استفاده شد.

در Redis دو عنصر، کلید و مقدار[[2]](#footnote-2) داریم. عنصر مقدار می­تواند انواع مختلفی داشته باشد. اجازه دهید به برخی این مقادیر و کاربردهای آن‌ها نگاهی بیندازیم.

**رشته‌ها:** اگر مقدار از نوع رشته بود، می­توان عملیات درج، بهنگام سازی، حذف، دریافت را از آن کلید انجام داد. مثلاً با دستور زیر می­توان یک کلید ساخت و یک‌رشته در آن درج کرد:

|  |  |
| --- | --- |
| یک کلید به اسم user\_1 ساخته‌شده و  book1 به‌عنوان مقدار برای این کلید در نظر گرفته می­شود. | SET user\_1 book1 |
| مقدار مورد (book1) نظر برای کلید user\_1 را برگردانده می­شود. | GET user\_1 |

**لیست:** با استفاده از لیست می­توان یک آرایه دلخواه داشت که بتوان در این آرایه، عنصری را اضافه یا کم کرد و عملیات مختلف دیگر را انجام داد.

**دسته‌ها**[[3]](#footnote-3)**:** اگر مقدار موجود از نوع دسته‌ها باشد درواقع یک لیست وجود دارد که هیچ‌کدام از عناصر آن تکراری نیستند.

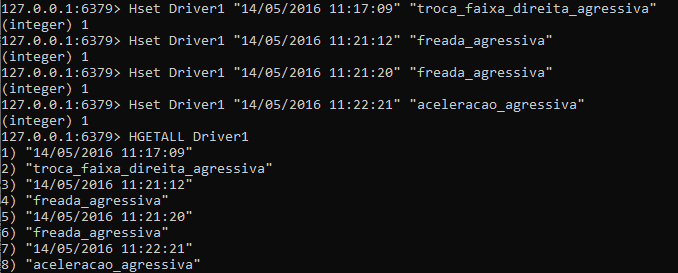
**دسته‌های منظم شده**[[4]](#footnote-4)**:** همانند مقدارهایی از نوع دسته است با این تفاوت که هر عنصر از مجموعه دارای وزن است و این وزن می­تواند به‌صورت مرتب نگهداری شود. (مثلاً از وزن کم به زیاد)

**هش­ها[[5]](#footnote-5):** اگر با انواع JSON آشنایی داشته باشید درک مقادیر هش‌ها ساده­تر است. این مقادیر می­توانند اشیایی مانند JSON را در خود جای دهند. بدین شکل که ابتدا کلیدی برای ذخیره‌سازی اطلاعات دریافت کرده و سپس به ذخیره‌سازی اطلاعات به‌صورت زوج مرتب‌هایی از کلید و مقدار می­پردازد.

علاوه بر موارد فوق، Redis می­تواند مقادیر دیگری را نیز ذخیره و بازیابی کند. یکی از انواع آن‌ها، HyperLogLog است. فرض کنید می­خواهید تعداد تکرار یک عنصر خاص از یک لیست را به دست آورید. اگر این لیست بسیار بزرگ باشد، این کار به‌راحتی انجام نمی­پذیرد. مقادیر HyperLogLog می‌تواند با دقت بسیار بالا (اما نه ۱۰۰ درصد) تعداد تکرار یک عنصر خاص را حدس بزند. این کار با استفاده از الگوریتم‌های خاصی امکان‌پذیر است. انواع دیگری مانند ذخیره‌سازی عناصری از جنس موقعیت‌های مکانی و یا Bitmap های نیز در Redis وجود دارند که کاربردهای خاص خود را دارند.

در این پژوهش ازآنجاکه میان رانندگان و اطلاعات ارسالی آن‌ها تمایز قائل هستیم و تنها قصد انتقال رویدادهای رانندگی و برچسب زمانی آن‌ها را داریم، می­توانیم اطلاعات خود را به‌صورت هش‌ها در Redis درآورده و به این شکل ذخیره‌سازی اطلاعات را پیاده­سازی کنیم. اطلاعات با دستور و ساختار زیر ذخیره‌شده‌اند.

کلید ذخیره‌سازی اطلاعات بر اساس شماره شناسایی راننده تعیین می­شود. ازآنجایی‌که رشته­های هر کلید باید نسبت به یکدیگر منحصربه‌فرد باشند، از برچسب زمان به‌عنوان رشته و در بخش مقدار آن رویداد رانندگی موردنظر نوشته می­شود. با دستور HSET می­توان یک سری اطلاعات هش ساخت و اطلاعات رشته و مقدار را به آن، مانند لیست، اضافه کرد و با دستور HGETALL می­توان اطلاعات ذخیره‌شده در یک هش را به دست آورد. در شکل ‏زیر نمونه­ای از استفاده از این دستورات و بارگذاری این اطلاعات در سرویس‌دهنده Redis می­توانیم مشاهده کنیم.



در شکل فوق با برنامه redis-cli به سرور Redis متصل شدیم و سعی کردیم دستورات را در خط دستور وارد کنیم. در ادامه با یک برنامه پایتون، سعی می­شود اطلاعات را از نرم‌افزار RabbitMQ خوانده و به سرویس‌دهنده Redis منتقل کرد.

**راه اندازی Redis**

از گزارش های پیشین به یاد داریم که می­توان ایمیج های استاندارد را از داکرهاب، بارگذاری و مورداستفاده قرارداد، Redis یکی از ایمیج های استاندارد است که می­توان به‌سادگی از آن استفاده نمود، برای استفاده از این ایمیج با داکر کامپوز، تنها کافی است، پورت مخصوص به ارتباط با کانتینر آن را به کامپیوتر میزبان متصل کرده و فضایی در کامپیوتر میزبان برای ذخیره‌سازی اطلاعات برای آن اختصاص دهیم تا با شروع به کار مجدد کانتینر، اطلاعات ذخیره‌شده در آن از بین نرود، همچنین می­توان فایل تنظیمات مربوط به Redis را تغییر داد و در کانتینر قرارداد و دستور داد تا برنامه Redis بر اساس تنظیماتی که در کانتینر قرارگرفته، اجرا شود. در شکل زیر می­توان دستوراتی که در فایل docker-compose.yaml برای بالا آمدن کانتینر Redis نوشته‌شده را مشاهده کرد.

cache:

    image: redis:latest

    container\_name: redis

    ports:

        - 6379:6379

    volumes:

        - ./redis/config/redis.conf:/redis.conf

        - ./redis/Data:/data

    command: [ “redis-server”, “/redis.conf” ]

همچنین برای دسترسی و استفاده از این سرویس به سادگی میتوان از کتابخانه ای که برای آن در پایتون نوشته استفاده نمود.

<https://redis-py.readthedocs.io/en/stable/>

همانطور که در مقدمه یاد شد، از این سرویس برای دو منظور استفاده شده

1. مدریت کاربران
2. هماهنگ سازی اپلیکیشن ها
3. **مدریت کاربران :**
   1. **مقدمه:**

برای مدریت کاربران سیستم از ویژگی ای که این نرم افزار در اختیار ما برای دسترسی به اطلاعات در اختیارمان میگذاشت استفاده شده این سرویس به اختصار ACL یا ‏ ‏Access Control List‏ ، قابلیتی است که به اتصالات خاصی اجازه می دهد تا از نظر دستورات قابل اجرا و کلیدهای قابل دسترسی محدود شوند. روش کار به این صورت است که پس از اتصال، یک کلاینت باید با ارائه یک نام کاربری و یک رمز عبور معتبر احراز هویت کند: اگر مرحله احراز هویت موفقیت آمیز بود، اتصال با یک کاربر معین و محدودیت هایی که کاربر دارد مرتبط می شود.

بدین منظور در اپلیکیشن سرور هنگام تعریف کاربر جدید ، آن کاربر را وارد لیست میکنیم که به اطلاعات بتواند دسترسی داشته باشد. همچنین در اپلیکیشن کلاینت هنگامی که میخواهد به اطلاعات دوربین ها دسترسی پیدا کند و وارد صفحه لاگین میشود، در حقیقت به این اطلاعات دسترسی پیدا کرده است.

* 1. نحوه به کارگیری

همانطور که پیشتر گفته شد برای اینکار از کتابخانه پاتون مربوطه و تنظیمات مربوط به ردیس استفاده میکنیم. در تصویر زیر میتوان این api ها را مشاهده نمود.

import redis

REDIS\_IP='localhost'

REDIS\_PORT=6379

Redis\_client = redis.Redis(host=REDIS\_IP, port=REDIS\_PORT)

Redis\_client.acl\_setuser("person1", enabled=True, nopass=False, passwords="+salam",

                            commands=["+HGETALL","+ACL"],categories=['+@hash'],

                            keys=["\*"])

print(Redis\_client.acl\_users())

Redis\_client\_p1 = redis.Redis(host=REDIS\_IP, port=REDIS\_PORT,

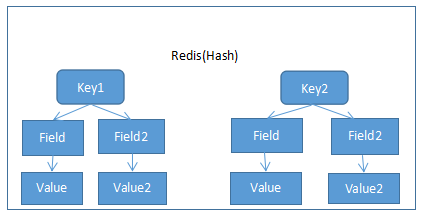
                               username="person1",password="salam")

1. **هماهنگ سازی اپلیکیشن ها**
   1. **مقدمه**

اپلیکیشن ها هنگامی که در کنار یک دیگر کار میکنند، نیاز داردند که نسبت به هم آپدیت باشند و از تغییرات همدیگر آگاه شوند، به عنوان مثال هنگامی که در برنامه سرور یک دوبین جدید تعریف میشود، برنامه کاربر نیاز دارد که نیست به آن آگاهی پیدا کند. برای همین منظور نیاز داریم تا هنگامی که دوبین جدید تعریف حذف و فعال و غیر فعال میشود. این اطلاعات در جایی ذخیره شوند. همچنین هنگامی که لایه پردازشی جدیدی تعریف میشود اپلیکیشن کاربر باید بداند که اطلاعات پردازش شده را از کجا باید دریافت کند. به همین منظور در مواردی که بالا گفته شد باید اطلاعات در ردیس ذخیره شوند.

* 1. **نحوه بکارگیری**

برای اینکه بتوان به سادگی از ردیس بهره برد، همانطور که پیشتر data Type های درون ردیس را مشاهده نمودیم، برای اپلیکیشن ما بهترین گزینه میتواند دیتاتایپ hash باشد. چرا که میتوان key را به نام دوربین وصل کنیم و در مقادیرها key value های مستقل دیگری داشته باشیم.



به این شکل Key را نام دورین و field ها را ویژگی های دوربین ها همانند ip ، نام exchange که در rabbitmq برای آن تعریف شده سطح دسترسی دوربین مورد نظر و البته فیلد لایه پردازشی در نظر میگیریم. برای جزئیات لایه پردازشی از یک مکانیزم دیگری که یک json را در داخل value ها قرار گیرند استفاده شد و نام field آن alg به مخفف الگوریتم قرار گرفت.

در رشته کد زیر میتوان تابعی که برای استفاده از آن در نظر گرفته شده را مشاهده نمود:

import redis

import json

REDIS\_IP='localhost'

REDIS\_PORT=6379

def update\_alg(redis\_cli,came\_name,alg,active,lvl):

    current\_json=redis\_cli.hget(came\_name,"alg")

    if current\_json ==None:

        current\_json ="{}"

    current\_dict=json.loads(current\_json)

    if active !=-1:

        current\_dict[alg]=(active,lvl)

    else:

        if alg in current\_dict:

            del current\_dict[alg]

    updated\_json=json.dumps(current\_dict)

    redis\_cli.hset(came\_name,"alg",updated\_json)

Redis\_client = redis.Redis(host=REDIS\_IP, port=REDIS\_PORT)

print(Redis\_client.hgetall("DEF\_MINIO"))

update\_alg(Redis\_client,"c4","eye",-1,1)

1. Caching [↑](#footnote-ref-1)
2. Value [↑](#footnote-ref-2)
3. Sets [↑](#footnote-ref-3)
4. Sorted Sets [↑](#footnote-ref-4)
5. Hashes [↑](#footnote-ref-5)