بسمه تعالی

توضیحات پیاده سازی تسک پیاده سازی خرید و فروش پارشیالی کاربران

ثبت سفارش:

در زمان ثبت سفارش فروش، مهم هست که چک شود میزان موجودی طلای کاربر، از میزان سفارش کمتر نباشد

همینطور مهم هست که در زمان ثبت سفارش خرید، مقدار کیف پول کاربر، نباید کمتر از مبلغ خرید به اضافه کمیسون باشد؛ البته بحث کیف پول جز تسک نبود ولی اگر فرصت میشد حتما انجامش میدادم

مچ کردن سفارشات:

این قسمت در واقع بخش اصلی سیستم ما هست و باید در حد امکان بهینه پیاده سازی شود.

اولین راه حلی که نوشتم به این شکل بود که بلافاصله بعد از ثبت یک سفارش، سفارش برای یک job ارسال شود و job پس از dispatch شدن، سفارشات متناظر هم قیمت را پیدا کند و بعد درون یک حلقه، سفارشات پیمایش شود تا سفارش مورد نظر completed شود.

راه حل خوبی بود و مزایای خودشو داشت، ولی مشکلش این بود که در صورت افزایش حجم سفارشات، احتمال race condition و تداخل سفارشات هم قیمت، زیاد میشد، ضمن اینکه به هر دلیلی به خطا میخورد کل عملیات متوقف میشد تا زمانی که سفارش بعدی ای با همون قیمت ثبت میشد.

راه حل دومی که داشتم این بود که مستقل از ثبت سفارشات، یک command هر دقیقه اجرا بشه و همه سفارشات open را بررسی و مچ کنه، مشکل این راه حل این بود که ممکن بود بین ثبت سفارش و اجرای command یک دقیقه فاصله بیفته

راه حل سوم این بود سفارش های سبک (مثلا زیر 10 گرم) فورا مچ بشن، ولی سفارشات سنگین تر رو بفرستیم برای job؛ به این دلیل که ممکن بود برای کامل کردن یک سفارش مثلا فروش 30 گرمی، احتیاج به مچ شدن با 10 تا سفارش خرید با میانگین 3 گرم باشه!

در نهایت راه حل چهارم (راه حل انتخاب شده): بعد از ثبت یک سفارش، یک event اجرا میشه سپس درون Listener، کلاس Job مورد نظر فراخوانی میشه و درون job، سفارشات هم قیمت به صورت گروه بندی شده با هم بررسی میشن

صد البته که بهترین راه حل به الگوی استفاده، حجم تراکنش ها و انتظارات کاربران ما بستگی داره، انجام تست هایی مثل load testing میتونه به ما کمک کنه تا بهترین راه حل را انتخاب کنیم
نکته اصلی اینجاست که این سیستم یک سیستم مالی و فینتک هست و هر اشتباهی، میتونه آسیب های زیادی به برند و وجهه سایت وارد کنه، بدیهی هست بیشتر از هر پروژه دیگه ای نیازمند دقت و توجه هست

تراکنش ها (Transactions):

بعد از مچ شدن سفارشات، تراکنش در جدول transaction ذخیره میشه
در مورد transaction ها، ایده اولیه این بود که id خریدار/فروشنده در جدول سفارشات ذخیره بشه، و فقط id سفارشات در جدول تراکنش ها ذخیره بشه ولی برای نمایش تراکنش های هر فرد و نمایش جزئیات آن، احتیاج به join بین جدول سفارشات و تراکنش ها بود. راه حل join برای پروژه های کوچک و متوسط کاملا مناسب هست ولی برای پروژه های بزرگ، که این دو جدول ممکنه چند میلیون رکورد داشته باشن، join کند میشه و باید از راه حل denormalization استفاده کرد. یعنی id کاربران هم در جدول سفارشات ذخیره بشه و هم در جدول تراکنش ها تکرار بشه تا نیازی به join نداشته باشیم

کمیسیون ها:

در مورد کمیسیون ایده اولیه این بود که به ازای وزن هر تراکنش (هر سفارش مچ شده)، کمیسیون محاسبه بشه و برای هر دو طرف معامله کمیسیون یکسان باشه. ولی بعدا متوجه شدم این راه حل ناعادلانه هست، چون ممکنه احمد یک سفارش فروش 12 گرمی داشته باشه و انتظار کمیسیون 1 درصد داشته باشه، ولی به صورت اتفاقی، سفارشش با 3 تا سفارش خرید زیر 10 گرم، مچ میشه و عملا همه تراکنش هاش با کمیسیون 1.5 درصد حساب میشه
راه دوم این بود که یک روش دومرحله ای داشته باشیم، با هر بار تراکنش، کمیسیون همون تراکنش حساب بشه، هر زمان سفارشش کامل یا کنسل شد، کارمزد نهایی محاسبه بشه و کمیسیون مازادی که قبلا گرفته شده، به کیف پول کاربر بازگردونده بشه، ولی این روش سربار بیشتری روی دیتابیس داشت
راه حل سوم و راه حل مرسوم این هست که کمیسیون بر حسب میزان سفارش خرید/فروش اولیه محاسبه میشه و همون قطعی میشه و فروش تدریجی/ کامل نشدن سفارش باعث تغییر در کارمزد نمیشه؛ تنها مشکل ظاهری این راه حل اینه که ممکنه کاربر یه سفارش حجم بالا بذاره تا بتونه کمیسیون کم رو ثبت بکنه، بعد بیاد بخش اعظم سفارشش رو کنسل کنه، که با توجه به این

موضوع که بلافاصله پس از ثبت سفارش، سفارش وارد matcher میشه و Lock میشه و عملا تا پایان matching امکان تغییر یا کنسل کردنش نیست تا حدی موضوع حل میشه، مگر اینکه سفارش هم قیمتی وجود نداشته باشه و سفارش باز بمونه و قابل کنسل کردن باشه
در نهایت انتخاب راه حل میتونه بستگی به سیاست سایت، و بحث بهینه بودن دیتابیس و کارایی بهتر نرم افزار داره

کنسل کردن سفارش:
سفارش تا زمانی که در وضعیت open/partial باشد امکان کنسل شدن دارد، و فقط ایجاد کننده سفارش میتواند آن را کنسل کند

نمایش به تومان، ذخیره به ریال:
موضوع ساده ای هست ولی یک نکته داره. برای این کار، دو روش کلی داریم
روش اول اینکه همه جا محاسبات و ذخیره سازی ها بر اساس ریال باشه، صرفا هنگام خروجی به کاربر (کلاس های Resource) و همینطور زمان گرفتن ورودی از کاربر در کنترلرها، به صورت دستی این تبدیل ها انجام بشن
روش دوم استفاده از راه حل هایی که به صورت اتوماتیک مقادیر را تبدیل کنن، که برای این روش، دو راه حل توی لاراول داریم، استفاده از Mutator/Accessor ها و همینطور راه حل استفاده از کلاس های Cast
نکته اینجاست که در روش دوم، زمانی که مثلا از دیتای یک جدول، توی کوئری های دیگه داریم استفاده میکنیم باید مواظب باشیم که دیتایی که گرفتیم به تومان هست ولی مقایسه با ریال انجام میشه! باید قبل از اجرای کوئری تبدیل انجام بشه

در ادامه:
قطعا این پیاده سازی، یک پیاده سازی اولیه هست و جای کار خیلی زیادی داره و میشه بهش موارد دیگه ای رو اضافه کرد، از جمله:

- پیاده سازی کیف پول برای کاربران
- پیاده سازی cache با کمک Redis برای اطلاعات پرکاربرد مثل لیست معاملات
- اضافه کردن سیستم اعلان برای اطلاع رسانی به کاربران بعد از اتمام سفارشات
- استفاده از message queue هایی مثل RabbitMQ در جهت مقیاس پذیری سیستم

- مانیتورینگ خطاها با ابزارهایی مثل Prometheus و Grafana
- استفاده از لاگ ها با ابزارهایی مثل ElasticSearch و Kibana

با احترام
عباس گودرزی