ارزیابی کارایی matching engine

امیرمحمد پیرحسینلو

401443029

بهمن ماه 1402

فهرست مطالب

۱- خلاصه

دراین پروژه کارایی یک matching engine مورد ارزیابی قرار می‌گیرد. پارامترهای مورد اندازه‌گیری شامل latency و throughput است. ارزیابی بر روی یک سرور و یک سیستم شخصی انجام می‌شود. در انتها دو نمودار ارائه می‌شود: نمودار اندازه‌گیری‌های انجام شده روی سرور و نمودار اندازه‌گیری‌های صورت گرفته در سیستم شخصی. در نهایت نتایج و روش‌های بهبود کارایی بیان می‌شوند.

۲- مقدمه

در ابتدا اینکه یک matching engine چیست شرح داده می‌شود سپس سراغ اجزای آن می‌رویم. در ادامه، پارامترهای مورد اندازه‌گیری بیان می‌شوند و پس از آن در مورد مفهوم percentile صحبت می‌شود.

۲-۱- موتور تطبیق‌دهنده (Matching Engine)

matching engine یک سیستم برنامه‌ریزی است که در بازارهای مالی مانند بازار بورس، بازار رمزارزها و بازار فارکس عمل می‌کند. وظیفه اصلی آن تطبیق سفارشات خرید و فروش است که به صورت خودکار انجام می‌شود.
این سیستم دو نوع سفارش را دریافت می‌کند: سفارش خرید (Bid) و سفارش فروش (Ask). با توجه به قیمت، حجم و دیگر شرایط مشخص شده در سفارش، matching engine تلاش می‌کند تا سفارشات متناسب را با یکدیگر تطبیق دهد. هدف اصلی این سیستم ایجاد انطباق بین سفارشات خرید و فروش به گونه‌ای است که قیمت‌ها همگرا شود و معاملاتی انجام شود. سفارشات ممکن است در حالت تطبیق نشده (Not Yet Matche) مانده یا به حالت صف (Queue) در آید. این سیستم باید به صورت پیوسته و realtime عمل کند و باید قابلیت پردازش حجم بالایی از سفارشات را داشته باشد. از آنجا که سرعت و دقت امر حیاتی در این سیستم است، بهینه‌سازی و بهبود عملکرد از جمله چالش‌های مهمی است که برای طراحان و مهندسان این سیستم وجود دارد.

۲-۲- انواع سفارشات (Order Types)

- سفارش بازار (Market Order)

در این نوع سفارش، مشتری درخواست خرید یا فروش با هر قیمتی که در بازار موجود است را ارسال می‌کند. سفارش بازار، به سرعت و با اولویت اجرا برای matching engine منتقل می‌شود و بلافاصله با بهترین قیمت موجود در بازار تطبیق داده می‌شود.

- سفارش حدی (Limit Order)

در این نوع سفارش، مشتری یک قیمت خاص برای خرید یا فروش تعیین می‌کند. سفارش حدی تنها در صورتی اجرا می‌شود که قیمت بازار به قیمت مشتری یا بهترین قیمت توسط یک سفارش بازار برسد. در غیر این صورت، سفارش به صورت معلق در صف قرار می‌گیرد و در انتظار تطبیق با قیمت مورد نظر می‌ماند.

- سفارش شرطی (Conditional Order)

در این نوع سفارش، مشتری یک شرط خاص را برای انجام معامله تعیین می‌کند. به عنوان مثال، مشتری می‌تواند تعیین کند که سفارش خرید اجرا شود در صورتی که قیمت ارز به یک سطح خاص برسد. این نوع سفارشات برای انجام معاملات در شرایط خاص یا برای کاهش ریسک‌های بازار مورد استفاده قرار می‌گیرند.

- سفارش توقف (Stop Order)

در این نوع سفارش، مشتری یک قیمت توقف (قیمت Trigger) مشخص می‌کند. اگر قیمت بازار به قیمت توقف مشتری برسد، سفارش توقف به یک سفارش بازار تبدیل می‌شود و بلافاصله تطبیق داده می‌شود.

- سفارشی ترکیبی (Combination Order)

در این نوع سفارش، مشتری می‌تواند سفارشات مختلفی را با هم ترکیب کند. به عنوان مثال، یک مشتری ممکن است یک سفارش خرید حدی و یک سفارش فروش حدی را به صورت یک سفارش ترکیبی ارسال کند که هر دو باید به صورت همزمان اجرا شوند.

انواع و اقسام سفارشات دیگری نیز وجود دارد که به همین موارد بالا اکتفا می‌شود.

۲-۳- کارایی (Performance)

matching engine برای داشتن عملکرد بهینه و انجام معاملات به سرعت و با دقت بالا نیاز به طراحی و پیاده‌سازی مناسب دارند. در زیر به برخی از جنبه‌های مهم که تاثیر مستقیم بر عملکرد matching engine دارند، پرداخته می‌شود:

- ساختار داده‌ها (Data Structures)
برای ذخیره و مدیریت سفارشات و داده‌های بازار، استفاده از ساختارهای داده‌های بهینه حیاتی است. برخی از ساختارهای داده‌ای معمول شامل صفوف، درخت‌ها، گراف‌ها و هش‌تیبل‌ها هستند. ساختار داده‌های مناسب موجب بهبود عملکرد و کاهش زمان پاسخ‌گویی می‌شود.
- چندنخی (Multithreading)
برای پردازش همزمان سفارشات و اجرای عملیات متعدد، استفاده از چندنخی ضروری است. با استفاده از چندنخی، موتور تطبیق سفارشات می‌تواند سفارشات را همزمان پردازش کند و این امر منجر به افزایش توانایی پاسخ‌گویی و بهبود عملکرد می‌شود.
- قفل‌گذاری (Locking)
در مواقعی که چندین نخ به داده‌های مشترک دسترسی دارند، استفاده از قفل‌گذاری ضروری است. با اعمال قفل بر روی مناطق حساس کد، می‌توان از رخداد مشکلات همگام‌سازی و تداخل‌های داده‌ها جلوگیری کرد که می‌تواند منجر به کاهش خطاهای ناخواسته شود اما عملکرد را نیز کاهش می‌دهد.
- ثبت رویداد (Journaling)
برای اطمینان از ایمنی و قابلیت بازیابی داده‌ها، موتورهای تطبیق سفارشات معمولاً از روش‌های ثبت رویداد استفاده می‌کنند. با ثبت هر عملیات معامله، اطمینان حاصل می‌شود که حالت سیستم همیشه قابل بازیابی است و اطلاعات دقیق و کاملی از تاریخچه معاملات وجود دارد. با این حال زمان دسترسی به دیسک خیلی بیشتر از زمان دسترسی به ممووری است در نتیجه این روش باعث کاهش کارایی می‌شود.
- فشرده‌سازی (Compression)
برای کاهش حجم داده‌ها و افزایش سرعت انتقال، استفاده از روش‌های فشرده‌سازی از اهمیت بالایی برخوردار است. با استفاده از فشرده‌سازی، حجم داده‌های انتقالی و ذخیره شده کاهش می‌یابد که می‌تواند به بهبود عملکرد و کاهش مصرف منابع سیستم کمک کند اما در مواقعی ممکن است باعث کاهش کارایی شود مانند زمانی که عکس عمل فشرده‌سازی، زمان زیادی ببرد.

این عوامل تنها بخشی از اقداماتی هستند که موتورهای تطبیق سفارشات برای داشتن عملکرد بهینه و اجرای موثر معاملات مورد استفاده قرار می‌دهند. توجه به هریک از این جنبه‌ها و بهبود آنها به کمک توسعه دهندگان و مهندسان در بهبود عملکرد و کارایی این سیستم‌ها کمک می‌کند.

۳- مشخصات نرم‌افزار مورد تست

۳-۱- مشخصات کلی

- ماژول کنترل ریسک و حسابداری
- ماژول ثبت رویدادها (Journaling) و نمونه‌برداری دیسک (Snapshot)

- رابط برنامه‌نویسی (API) برای معاملات، مدیریت و گزارشات

۳-۲- مشخصات جزیی

- وضعیت کاری حافظه‌ای (In-memory State) برای داده‌های حسابداری و دفاتر سفارش (Order Book).
- event-sourcing - disk journaling and journal replay support, state snapshots (serialization) and restore operations, LZ4 compression).
- الگوریتم‌های تطبیق و کنترل ریسک بدون قفل (locking)
- عدم استفاده از اعداد اعشاری

از دست دادن دقت امکان‌پذیر نیست. برای مثال از دست رفتن واحدهای کوچک پول ممکن نیست زیرا از اعداد اعشاری استفاده نمی‌شود.

- عملیات‌ها atomic و deterministic هستند.

اگر یک سناریو از ثبت سفارشات را هر تعداد بار اجرا کنیم، وضعیت نهایی متغیرها در سیستم (متغیرهای در حافظه، فایل‌های روی دیسک) یکسان خواهد بود.

- پردازش چند هسته‌ای با استفاده از pipeline

هر هسته پردازنده مسئول یک مرحله پردازش خاص است. برای مثال، یک هسته وظیفه ثبت سفارشات را بر عهده دارد و یک هسته‌ی دیگر وظیفه‌ی مدیریت حساب‌ها.

- کارمزد Maker و Taker

می‌توان کارمزد دریافتی از Taker ها و Maker ها را مشخص کرد.

Taker:

فرد یا شرکتی است که سفارش خود را به بازار ارسال می‌کند و تصمیم می‌گیرد که به قیمت موجود در بازار عمل کند. به عبارت دیگر، Taker سفارش خود را به صورت فوری با قیمت موجود در بازار اجرا می‌کند.

Maker:

فرد یا شرکتی است که سفارش خود را به بازار ارسال می‌کند، اما به انتظار قیمت مورد نظر خود در بازار می‌ماند. به عبارت دیگر، Maker قصد دارد سفارش خود را با یک قیمت خاص وارد بازار کند و در انتظار اجرای آن با یک Taker است. وقتی که یک Taker سفارش Maker را با یک قیمت مطابقت داد، معامله انجام می‌شود و Maker به عنوان دریافت کننده سفارش شناخته می‌شود.

به طور کلی، Takerها به سرعت معامله و تسریع در فرایند معاملات علاقه دارند، در حالی که Makerها به جذب معاملات با قیمت‌های مطلوب و کاهش هزینه‌های معاملاتی تمایل دارند.

- دو پیاده‌سازی از order book
  - پیاده‌سازی ساده با استفاده از AVL Tree
  - پیاده‌سازی کارا با استفاده از Raddix Tree
- تست
  - تست‌های واحد (Unit Test)
  - تست‌های یکپارچه (Integration Test)
  - تست‌های فشار (Stress Test)
  - تست‌های اصلاحی/یکپارچگی (Integrity/Consistency Test)

- عملیات تعلیق/ادامه کاربر
- API گزارش‌های اصلی (حساب کاربر، سود/زیان)
- انواع سفارشات
  - IOC

سفارش فوری یا لغو (Immediate-or-Cancel) به معنای این است که مشتری سفارش را با قیمت موجود در بازار فوراً اجرا می‌کند. اگر سفارش به صورت کامل یا بخشی اجرا شود، معامله انجام می‌شود، اما اگر سفارش نتواند به صورت کامل اجرا شود، **قسمت‌های باقی‌مانده** فوراً لغو می‌شود.

  - GTC

سفارش خوب تا لغو (Good-till-Cancel) به معنای این است که مشتری سفارش را با یک قیمت معین وارد بازار می‌کند، اما تا زمانی که سفارش اجرا نشده و مشتری آن را لغو نکند، اعتبار دارد. به این معناست که سفارش در بازار باقی‌مانده و در انتظار اجرا می‌ماند تا زمانی که مشتری تصمیم به لغو آن بگیرد یا تا زمانی که سفارش به صورت کامل اجرا شود.

  - FOK-B

سفارش پر کردن یا لغو بودجه‌ای (Fill-or-Kill Budget) به معنای این است که مشتری یک سفارش را با یک قیمت معین وارد بازار می‌کند و درخواست می‌دهد که سفارش باید به صورت کامل اجرا شود. اگر سفارش نتواند به صورت کامل اجرا شود، کل سفارش فوراً لغو می‌شود و هیچ بخشی از آن به صورت پاره‌پاره اجرا نمی‌شود. این نوع سفارش به عنوان یک مکانیزم حفاظتی استفاده می‌شود تا از وقوع معامله ناقص یا ناکافی جلوگیری شود.

۴- کارایی

به منظور ارزیابی عملکرد یک سیستم، عموماً از مفاهیمی مانند تاخیر (Latency)، ظرفیت (Throughput)، تست‌های ناهماهنگ (Hiccups Test) و تست‌های سریال‌سازی (Serialization Test) استفاده می‌شود. در زیر به توضیح هر یک از این مفاهیم می‌پردازیم:

۴-۱- پارامترهای کارایی

- تاخیر (Latency)

تاخیر به مدت زمانی اشاره دارد که یک سیستم یا سرویس برای پاسخ به یک درخواست مشخص نیاز دارد. این میزان زمانی است که طول می‌کشد تا یک درخواست از زمانی که به سیستم وارد می‌شود تا زمانی که پاسخ به کاربر ارسال می‌شود، پردازش شود. به طور معمول، تاخیر بر اساس زمانی که سیستم به پردازش درخواست نیاز دارد، محاسبه می‌شود و به عنوان یک عامل مهم در ارزیابی کارایی سیستم مورد توجه قرار می‌گیرد.

- زمان پاسخ (Response Time)

زمان پاسخ به مدت زمانی اشاره دارد که میزان زمانی است که یک کاربر از زمانی که یک درخواست ارسال می‌کند تا زمانی که پاسخ به آن درخواست دریافت می‌شود، طول می‌کشد. این شامل تاخیر سیستم (Latency) برای پردازش درخواست و همچنین هر گونه تاخیر دیگری که ممکن است در مسیر ارتباط بین کاربر و سیستم وجود داشته باشد، می‌شود. زمان پاسخ از زمانی که کاربر درخواست خود را ارسال می‌کند تا زمانی که نتیجه به کاربر ارسال می‌شود، محاسبه می‌شود.

به طور خلاصه، تاخیر (Latency) به مدت زمانی اشاره دارد که سیستم برای پردازش یک درخواست مشخص نیاز دارد، در حالی که زمان پاسخ (Response Time) به مدت زمانی اشاره دارد که یک کاربر برای دریافت پاسخ به درخواست خود طول می‌کشد. زمان پاسخ شامل تاخیر سیستم (Latency) به عنوان یکی از عوامل، همچنین هرگونه تاخیر دیگری است که ممکن است در مسیر ارتباط بین کاربر و سیستم وجود داشته باشد.

- ظرفیت (Throughput)

ظرفیت نشان‌دهنده تعداد درخواست‌ها یا تراکنش‌هایی است که یک سیستم می‌تواند در یک بازه زمانی معین پردازش کند. برای بسیاری از سیستم‌ها، بهبود ظرفیت بدین معناست که سیستم قادر به پشتیبانی از بیشترین تعداد کاربران یا تراکنش‌ها با کمترین تاخیر ممکن است.

- تست‌های ناهماهنگ (Hiccups Test)

تست‌های ناهماهنگ برای بررسی و ارزیابی پایداری سیستم در مقابل تغییرات ناگهانی یا عدم پیش‌بینی شده استفاده می‌شود. در این تست‌ها، سیستم به شدت آزمایش می‌شود تا مشخص شود که آیا در صورت بروز مشکلات یا شرایط نامناسب، عملکرد آن به طور قابل قبول باقی می‌ماند یا خیر.

- تست‌های سریال‌سازی (Serialization Test)

تست‌های سریال‌سازی به بررسی و ارزیابی توانایی یک سیستم در پردازش داده‌ها و انجام تراکنش‌ها با سرعت و بهره‌وری بالا می‌پردازند. این تست‌ها به ما کمک می‌کنند تا اطمینان حاصل کنیم که سیستم قادر به پردازش حجم بزرگی از داده‌ها با کمترین تاخیر ممکن است و در مقابل افزایش ترافیک و بار کاری، عملکرد خود را حفظ می‌کند.

از ترکیب این مفاهیم می‌توان به عملکرد کلی یک سیستم پیچیده پی برد و مشکلات و مسائلی که ممکن است برای آن بروز کنند را شناسایی و برطرف کرد

۲-۴- مفهوم percentile

مفهوم percentile یک اندازه‌گیری آماری است که برای توصیف توزیع مجموعه داده‌ها استفاده می‌شود. در این مفهوم، مجموعه داده به صد بخش مساوی تقسیم می‌شود، به طوری که هر بخش نمایندهٔ یک درصد از مجموع مشاهدات است. درصدها کمک می‌کنند تا درک کنیم که نقاط داده چگونه پخش شده‌اند و چه نسبتی از داده‌ها زیر یک مقدار مشخص قرار می‌گیرند.

یک مثال از اندازه‌گیری‌های تاخیر (Latency) را در نظر بگیریم که زمان لازم برای سیستم برای پاسخ به یک درخواست را نشان می‌دهد. فرض کنید که ما دارای اندازه‌گیری‌های زیر در میلی‌ثانیه هستیم:

real meassurements: 35, 40, 30, 50, 15, 20, 45, 25, 55, 10

sorted: 10,15,20,25,30,35,40,45,50,55

برای محاسبه percentileها:

- 50 درصد (میانه)

برای پیدا کردن 50 درصد (همچنین به نام میانه)، ابتدا اندازه‌گیری‌های تاخیر را به صورت صعودی مرتب می‌کنیم. 50 درصد متناظر با مقدار در موقعیت پنجم است، که معادل با 30 میلی‌ثانیه است.

- 90 درصد

نشان دهنده مقداری است که زیر آن 90 درصد از مشاهدات قرار دارد که معادل 50 میلی ثانیه است.

- 99 درصد

مشابه محاسبه 90 درصد، ما 99 درصد از تعداد کل مشاهدات را محاسبه می‌کنیم. از آنجایی که نمی‌توانیم به عنوان یک مشاهدهٔ جزئی از یک مشاهده استفاده کنیم، به مشاهده دهم گرد می‌کنیم که معادل با 55 میلی‌ثانیه است. بنابراین، 99 درصد معادل با 55 میلی‌ثانیه است.

۴-۳- نکات و ترفندهای percentileها

- زمان استفاده

percentileها برای درک توزیع داده‌ها و شناسایی نقاط نامتعادل یا رفتارهای نامعمول در مجموعه داده مفید هستند. آنها به طور معمول در نظارت بر عملکرد مورد استفاده قرار می‌گیرند، به ویژه در سیستم‌هایی که زمان پاسخ اهمیت دارد مانند سرورهای وب، پایگاه‌های داده یا پلتفرم‌های معاملات مالی.

- جایی که نباید اعتماد کرد

اگرچه percentileها اطلاعات مفیدی از توزیع داده‌ها ارائه می‌دهند، اما ممکن است نسبت به داده‌های نامتوازن و یا توزیع‌های نامتقارن حساس باشند. در مجموعه‌های داده با مقادیر استثنایی یا دم‌های سنگین (Thick Tailed)، درصدها ممکن است توزیع واقعی داده را به خوبی نشان ندهند. به علاوه، percentileها اطلاعاتی درباره شکل توزیع فراتر از یکسری درصدها فراهم نمی‌کنند، بنابراین این ممکن است کلیت تنوع داده‌ها را به خوبی نشان ندهد.

به عنوان مثال، در نظر بگیرید که دارای مجموعه داده‌ای از اندازه‌گیری‌های تاخیر هستیم که بیشتر مقادیر آن‌ها در یک محدوده خاص جمع شده‌اند، اما تعدادی از نقاط outlier به طور قابل توجهی مقادیر بسیار بالایی دارند. در این حالت، فقط اعتماد به درصدها برای نشان دادن عملکرد واقعی سیستم، مناسب نیست. در این موارد، احتمالاً نیاز به استفاده از ابزارهای آماری دیگر یا تجزیه و تحلیل بیشتر می‌باشد.

۵- محک (Benchmark)

۵-۱- مشخصات matching engine

- order book با یک نماد (symbol)

برای مثال، منظور از symbol می‌تواند BTC/USDT باشد که قیمت بیتکوین را بر اساس تتر نشان می‌دهد.

- تعداد پیام‌های ورودی
- 3000000 درخواست به شکل زیر توزیع می‌شوند:
  - GTC 9%
  - IOC 3%
  - Cancel Commands 6%
  - Move Commands 82%

دستور حرکت، یکی از عملیات مهم است که در فرآیند اجرای معاملات مورد استفاده قرار می‌گیرد. این دستور به معنای جابه‌جایی یک سفارش موجود در کتاب سفارشات به یک قیمت دیگر است. زمانی که یک معامله جدید با قیمتی مختلف در جریان است یا تقاضایی در سمت خرید یا فروش وجود دارد، سفارشات ممکن است نیاز به جابه‌جایی داشته باشند تا با نوسانات قیمت جدید سازگار شوند.
به طور مثال، فرض کنید یک سفارش خرید برای 10 واحد از یک سهم با قیمت 100 ریال وجود دارد ولی به دلیل افزایش ناگهانی قیمت، قیمت سهم به 105 ریال افزایش یافته است. در این صورت، با اجرای یک دستور حرکت، سفارش خرید به قیمت 105 ریال جابه‌جا می‌شود تا با شرایط جدید مطابقت داشته باشد.

  - حدود 6% از تمام دستورها باعث انجام یک یا چند معامله می‌شوند.

- 1000 حساب کاربری فعال
- به طور میانگین حدودا 1000 عدد limit order در حالت فعال هستند که در حدود 750 اسلات قیمت مختلف قرار گرفته‌اند.
- نتایج زمان تاخیر (Latency) فقط برای پردازش ریسک و order matching است. موارد دیگر مانند تاخیر رابط شبکه، IPC، ژورنالینگ در آن شامل نمی‌شود.
- داده‌های آزمایشی، پرشی (burst mode) نیستند که به معنای وقفه ثابت بین دستورات است (بین 0.2 تا 8 میکروثانیه بسته به throughput مورد نظر).
- GC قبل/بعد از اجرای هر چرخه بنچمارک (3000000 پیام) فعال می‌شود. منظور از GC، Java Garbage Collector است.

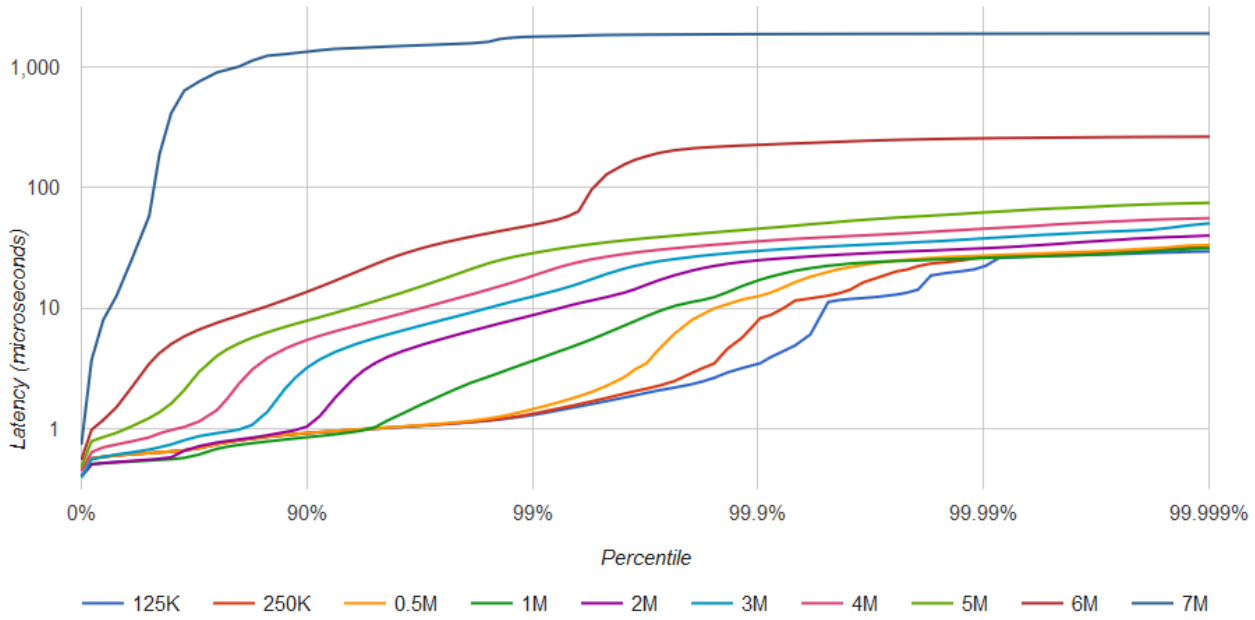
۵-۲- نتیجه‌ی benchmark روی Intel® Xeon® X5690

rate	50.0%	90.0%	95.0%	99.0%	99.9%	99.99%	worst
125K	0.6µs	0.9µs	1.0µs	1.4µs	4µs	24µs	41µs
250K	0.6µs	0.9µs	1.0µs	1.4µs	9µs	27µs	41µs
500K	0.6µs	0.9µs	1.0µs	1.6µs	14µs	29µs	42µs
1M	0.5µs	0.9µs	1.2µs	4µs	22µs	31µs	45µs
2M	0.5µs	1.2µs	3.9µs	10µs	30µs	39µs	60µs
3M	0.7µs	3.6µs	6.2µs	15µs	36µs	45µs	60µs
4M	1.0µs	6.0µs	9µs	25µs	45µs	55µs	70µs
5M	1.5µs	9.5µs	16µs	42µs	150µs	170µs	190µs
6M	5µs	30µs	45µs	300µs	500µs	520µs	540µs
7M	60µs	1.3ms	1.5ms	1.8ms	1.9ms	1.9ms	1.9ms

تصویر ۱: نتایج latency benchmark روی Intel® Xeon® X5690

محور عمودی به معنای تعداد دستوراتی است که به matching engine ارسال می‌شود و محور افقی اندازه‌گیری latency برای percentile های مختلف است.
برای مثال، 50 درصد دستورات هر کدام کمتر از 0.6 میکروثانیه طول کشیده‌اند در صورتی که در کل 125000 دستور ارسال کرده‌ایم (خانه‌ی (2و2)).

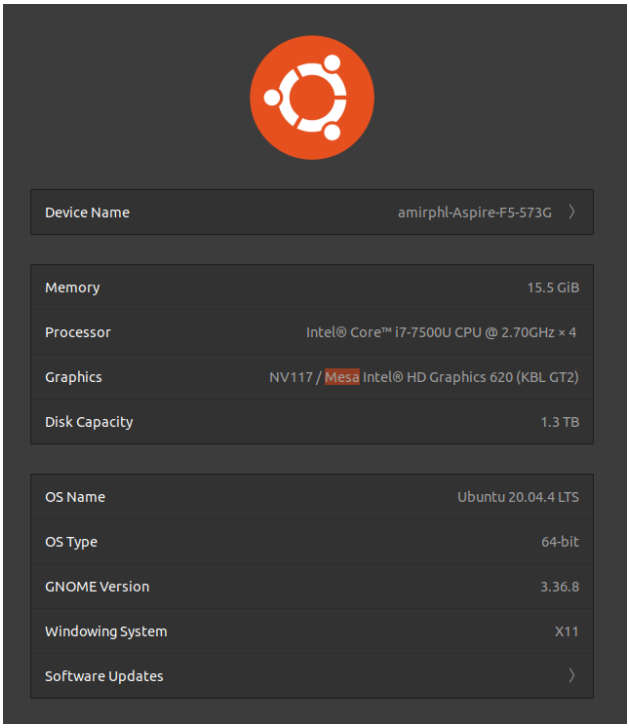
نمایش جدول بالا در قالب نمودار:



تصویر ۲: نتایج latency benchmark روی Intel® Xeon® X5690 در قالب نمودار

۵-۳- نتایج اجرا روی سیستم شخصی

۵-۳-۱- تست latency

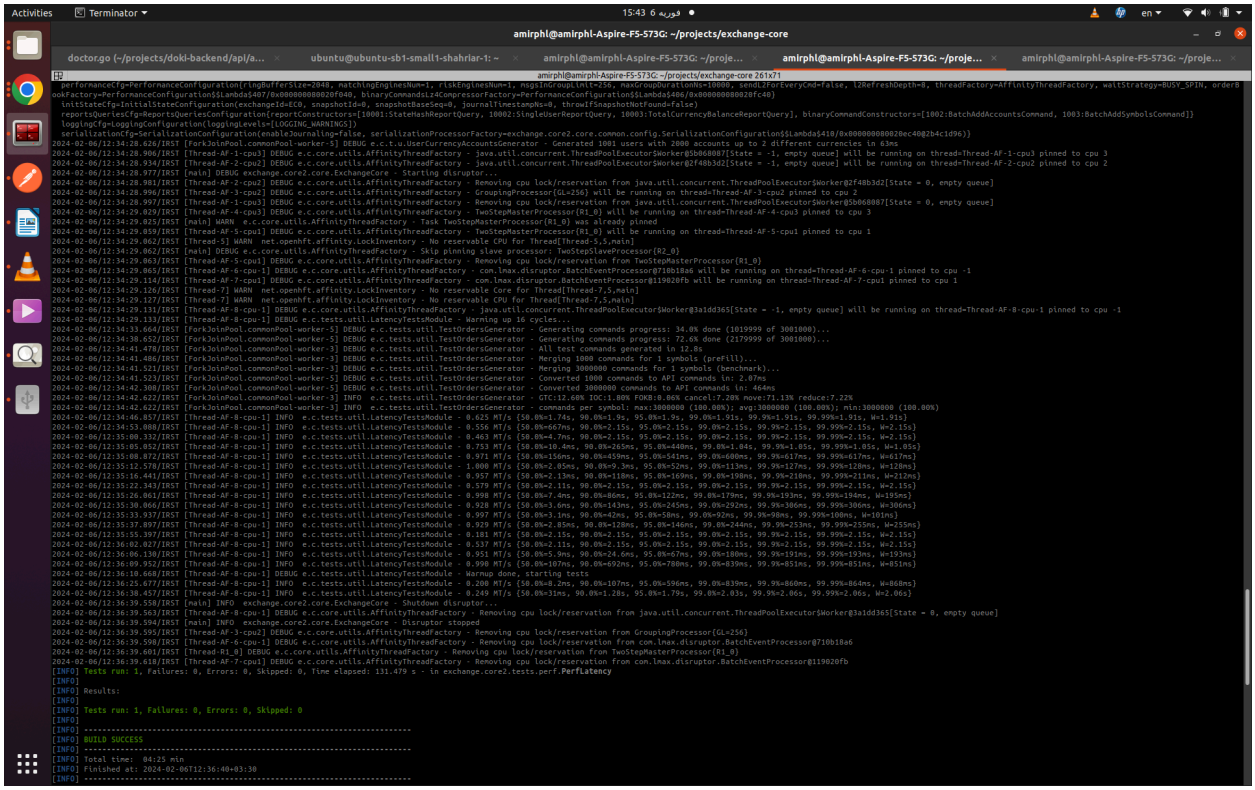


تصویر ۳: مشخصات سیستم شخصی

با این دستور می‌توانید تست latency را بر روی سیستم خود انجام دهید (باید از قبل java و maven روی سیستم شما نصب باشد):

```
mvn -Dtest=PerfLatency#testLatencyMargin test
```

نمونه‌ای از خروجی دستور بالا:

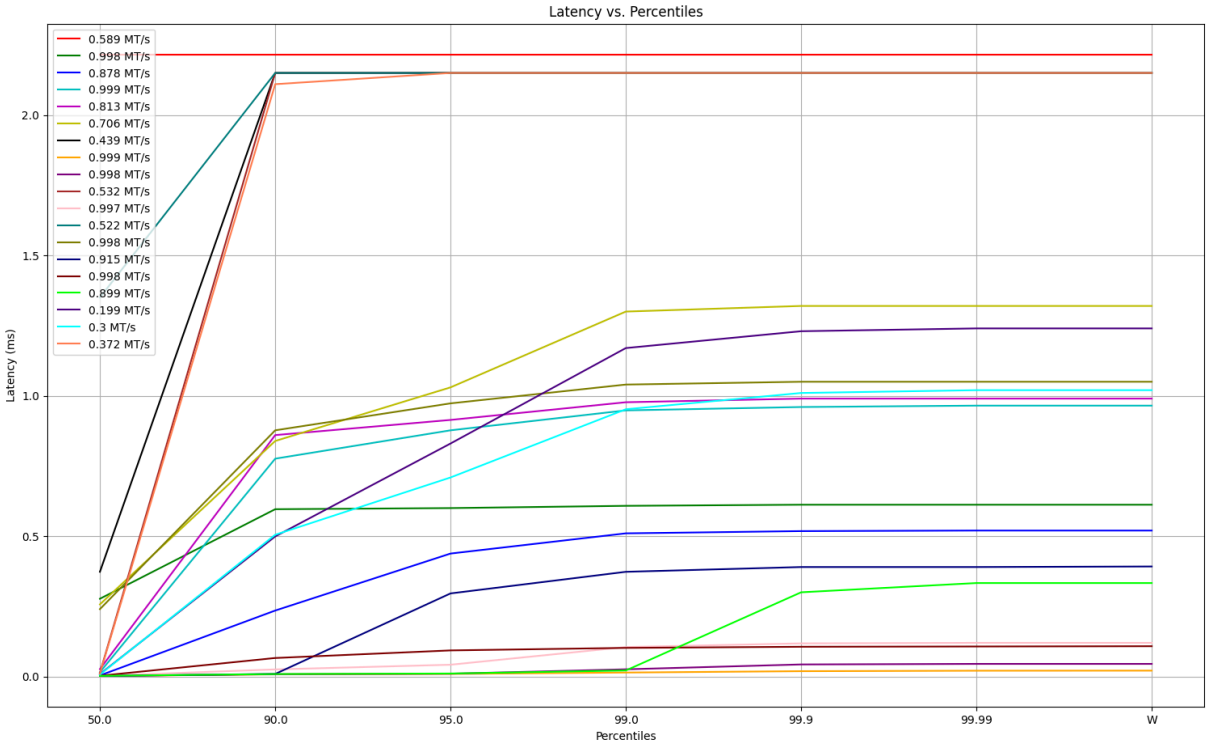


تصویر ۴: خروجی ترمینال اجرای تست latency روی سیستم شخصی

```
0.589 MT/s {50.0%=2.15s, 90.0%=2.15s, 95.0%=2.15s, 99.0%=2.15s, 99.9%=2.15s, 99.99%=2.15s, W=2.15s}
0.998 MT/s {50.0%=277ms, 90.0%=596ms, 95.0%=600ms, 99.0%=608ms, 99.9%=612ms, 99.99%=612ms, W=612ms}
0.878 MT/s {50.0%=3.8ms, 90.0%=235ms, 95.0%=438ms, 99.0%=510ms, 99.9%=518ms, 99.99%=520ms, W=520ms}
0.999 MT/s {50.0%=15.3ms, 90.0%=776ms, 95.0%=877ms, 99.0%=948ms, 99.9%=960ms, 99.99%=965ms, W=965ms}
0.813 MT/s {50.0%=26.3ms, 90.0%=860ms, 95.0%=914ms, 99.0%=977ms, 99.9%=990ms, 99.99%=990ms, W=990ms}
0.706 MT/s {50.0%=257ms, 90.0%=839ms, 95.0%=1.03s, 99.0%=1.3s, 99.9%=1.32s, 99.99%=1.32s, W=1.32s}
0.439 MT/s {50.0%=373ms, 90.0%=2.15s, 95.0%=2.15s, 99.0%=2.15s, 99.9%=2.15s, 99.99%=2.15s, W=2.15s}
0.999 MT/s {50.0%=2.04ms, 90.0%=7.7ms, 95.0%=9.3ms, 99.0%=14.2ms, 99.9%=19.0ms, 99.99%=20.7ms, W=21.0ms}
0.998 MT/s {50.0%=2.08ms, 90.0%=8.3ms, 95.0%=10.0ms, 99.0%=25.7ms, 99.9%=43ms, 99.99%=45ms, W=45ms}
0.532 MT/s {50.0%=12.6ms, 90.0%=2.15s, 95.0%=2.15s, 99.0%=2.15s, 99.9%=2.15s, 99.99%=2.15s, W=2.15s}
0.997 MT/s {50.0%=2.47ms, 90.0%=25.3ms, 95.0%=42ms, 99.0%=105ms, 99.9%=118ms, 99.99%=120ms, W=120ms}
0.522 MT/s {50.0%=1.35s, 90.0%=2.15s, 95.0%=2.15s, 99.0%=2.15s, 99.9%=2.15s, 99.99%=2.15s, W=2.15s}
0.998 MT/s {50.0%=240ms, 90.0%=877ms, 95.0%=973ms, 99.0%=1.04s, 99.9%=1.05s, 99.99%=1.05s, W=1.05s}
0.915 MT/s {50.0%=2.11ms, 90.0%=9.2ms, 95.0%=296ms, 99.0%=373ms, 99.9%=390ms, 99.99%=390ms, W=392ms}
0.998 MT/s {50.0%=2.7ms, 90.0%=66ms, 95.0%=93ms, 99.0%=102ms, 99.9%=106ms, 99.99%=107ms, W=108ms}
0.899 MT/s {50.0%=2.33ms, 90.0%=8.8ms, 95.0%=10.7ms, 99.0%=21.5ms, 99.9%=300ms, 99.99%=333ms, W=333ms}
Warmup done, starting tests
0.199 MT/s {50.0%=8.7ms, 90.0%=499ms, 95.0%=830ms, 99.0%=1.17s, 99.9%=1.23s, 99.99%=1.24s, W=1.24s}
0.300 MT/s {50.0%=7.0ms, 90.0%=505ms, 95.0%=709ms, 99.0%=952ms, 99.9%=1.01s, 99.99%=1.02s, W=1.02s}
0.372 MT/s {50.0%=10.2ms, 90.0%=2.11s, 95.0%=2.15s, 99.0%=2.15s, 99.9%=2.15s, 99.99%=2.15s, W=2.15s}
```

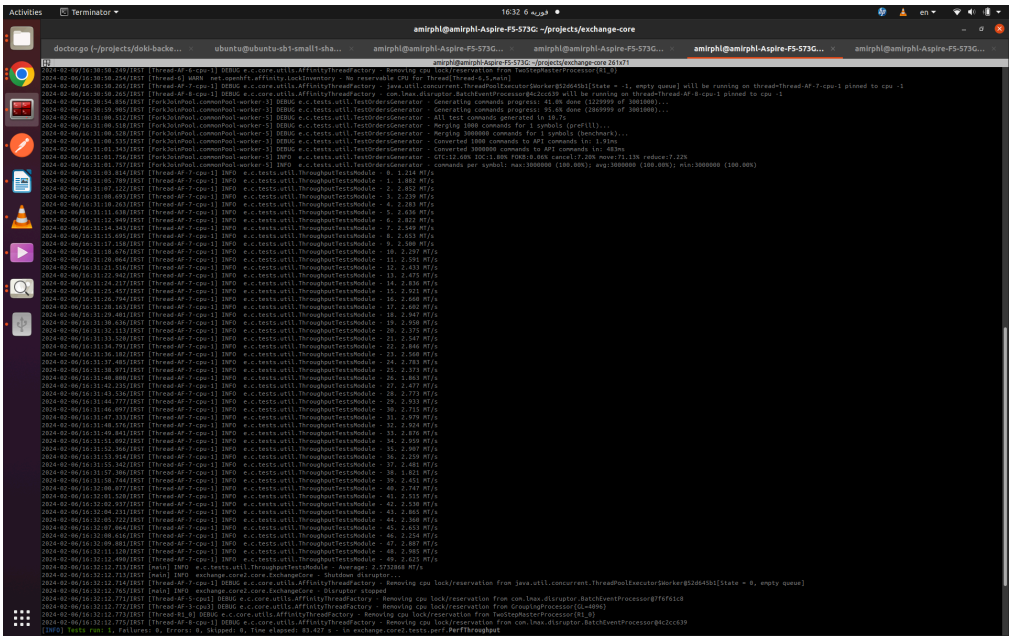
تصویر ۵: نتایج latency benchmark روی سیستم شخصی

نمایش خروجی ترمینال در قالب نمودار (با اجرای دستور python3 plot-latency.py نمودار زیر ساخته می‌شود):



تصویر ۶: نتایج latency benchmark روی سیستم شخصی در قالب نمودار

۵-۳-۲- تست throughput



تصویر ۷: خروجی ترمینال اجرای تست throughput روی سیستم شخصی

خروجی بیانگر این است که میانگین thoughput برابر 2.5732868 میلیون دستور در ثانیه است.

در سیستم خود می‌توانید با دستور زیر، تست throughput را اجرا کنید:

```
mvn -Dtest=PerfThroughput#testThroughputMargin test
```

۶- نتایج و بهبود کارایی

همانطور که از نمودارها مشاهده می‌شود، ترند latency ها یکسان است. در زمینه‌ی matching engineها، گلوگاه اصلی cpu است. همچنین مشکل اصلی کاهش کارایی در matching engineها، وجود lock برای خواندن و نوشتن در صف‌های مختلف است. در یک نگاه کلی می‌توان هر matching engine را به شکل تعدادی صف متصل به هم در قالب یک pipeline دید. وجود صف‌های concurrent به دلیل استفاده از lock ها که خود به یک سری system call ترجمه می‌شوند، موجب افزایش شدید latency می‌شود. برای هر این مشکل، ساختار داده‌ی LMAX Disruptor ارائه شده‌است. برای مطالعه‌ی بیشتر به اینجا مراجعه کنید. طبعا در بخش‌های زیادی می‌توان کارایی را زیاد کرد. در زیر به تعدادی از این موارد اشاره می‌شود.

- استفاده از ساختار داده‌های مناسب
  جستجو در زمان log(n) یا loglog(n))
- بخش بندی قیمت‌ها
  استفاده از تردها و ساختارداده‌های جداگانه برای قیمت‌های hot (بازه‌هایی از قیمت‌ها که سفارشات در آنها بیشتر است.).
- کاهش اثر locking در قسمت‌های مختلف با استفاده از LMAX Disruptor
- کاهش دسترسی به دیسک با بافر کردن write ها