معماری جامع جهت مقیاس‌پذیری و افزایش بهره‌وری و ثبات سامانه‌های پرداخت

A grand architecture to improve large-scale payment system's performance and consistency

مصطفی رستگار، رئیس گروه توسعه نرم‌افزار شرکت خدمات انفورماتیک، M_Rastgar@ISC.CO.IR
عاطفه احمدی، کارشناس ارشد توسعه نرم‌افزار شرکت خدمات انفورماتیک، A_Ahmady@ISC.CO.IR

چکیده

استقبال از سامانه های نظام پرداخت نظیر ساتنا، چکاوک و پایا در دهه اخیر به شکل فزاینده ای همه‌گیر شده است. چرا که انتقال وجه بین بانکی همواره پایه‌ای‌ترین نیاز هر نظام بانکی می باشد. لذا با ظهور این سامانه‌ها، به دلیل افزایش سرعت نقل و انتقالات از یک سو و کاهش چشمگیر هزینه‌های آن از سوی دیگر، حجم تراکنش‌های انتقال بین بانکی به شدت افزایش یافت. بنابراین طراحی بهینه این سامانه‌ها از نظر کارایی از حساسیت ویژه‌ای برخوردار می باشد. جهت افزایش کارایی، یا به عبارت دیگر افزایش سرعت پردازش تراکنش‌ها، یک راه حل عمومی، تولید سامانه‌های مقیاس‌پذیر می‌باشد. در این جا منظور از مقیاس‌پذیری، توزیع‌پذیری سامانه در سرورهای مختلف می باشد به گونه‌ای که هر یک از آن‌ها مسئولیت بخشی از کسب و کار سامانه را بر عهده دارد. تا کنون همه طراحی‌های سامانه‌های مقیاس‌بزرگ پرداخت، مبتنی بر پایگاه داده رابطه‌ای بوده که اغلب محدودیت‌هایی در زمینه سرعت، مقیاس‌پذیری و پایداری در کنار هم را به همراه دارند. از طرفی، در سامانه‌های نظام پرداخت به دلیل بار مالی تراکنش‌ها، همواره پایداری سیستم حائز اهمیت است، که به نوعی با مقیاس‌پذیری و سرعت در تناقض می‌باشد. در این مطالعه طرحی ارائه می‌گردد تا در این سامانه‌ها پایداری را در کنار مقیاس‌پذیری و سرعت به همراه داشته باشد که شامل مولفه‌های مختلفی نظیر پایگاه داده رابطه‌ای جهت ثبات، پایگاه داده درون حافظه‌ای مبتنی بر NoSQL جهت سرعت و پردازش‌های موازی در بستر توزیع شده برای مقیاس‌پذیری، می‌باشد. چالش اصلی در این طراحی، هماهنگی بین مولفه‌های مذکور می‌باشد. در این مطالعه با ارائه راهکارهایی برای چالش‌های مذکور، طرح نهایی در قالب سامانه شبیه‌سازی شده نظام پرداخت، مورد ارزیابی قرار گرفته و نشان داده شد که نتایج مطلوب حاصل گردید.

واژگان کلیدی: ثبات، مقیاس‌پذیری، پایگاه داده رابطه‌ای، NoSQL، پردازش موازی، پرداخت بین‌بانکی

طبقه بندی JEL: C63، O33، M38

Abstract

As far as interbank payment orders play a fundamental role in banking systems, these systems, such as Real-Time Gross Settlement (RTGS), Automated Clearing House (ACH) and CIS (Check Imaging System), has become extraordinarily popular. Therefore, by the emergence of these systems, due to the increase in the speed of transfers, on the one hand, and the significant reduction of its costs, on the other hand, the number of interbank transactions has grew impressively in the recent decade. Therefore, devising an optimized architecture to make payment systems throughput robust, become so important. In order to

increase the efficiency, or, in other words, processing much more transactions in a limited period of time, a general solution is the production of scalable systems. Here, ا scalability ب means the distribution of the system on different servers, so that each of them is responsible for part of the system's business. So far, all large scale payment systems, have been established based on relational database, which in most of them, making fast processing, scalability and durability plan simultaneously, is disastrous. On the other hand, in payment systems due to the financial burden of transactions, the durability of the system is always important, which is against scalability and high throughput. In this study, a super-design is proposed to provide durability in these systems along with scalability and high performance, which includes various components, such as a relational database for durability, an in-memory NoSQL database for achieving high performance and parallel processing in the distributed platform for scalability. The major challenge in this grand design is to orchestrate processes through the related components. In this paper some solutions are presented to overcome the challenges and the grand design was evaluated on a simulated payment system. The simulation results indicate that the proposed grand design will improve the system performance.

Keywords: Consistency, Scalability, Relational Database, NoSQL, Parallel Processing, Interbank Payment

JEL Category: C63, O33, M38

۱. مقدمه

با توجه به استقبال روز افزون جامعه از بانکداری الکترونیکی، حجم تراکنش‌های مالی به شکل محسوس و چشمگیری افزایش یافته است. نظام پرداخت و سوییچ‌های شبکه بین‌بانکی نیز از این قاعده مستثنی نمی‌باشند. به طور مثال، با نگاهی اجمالی می توان به سادگی دید که حجم درخواست‌های سوییچ‌هایی نظیر شتاب و پایا، در سال‌های اخیر رشد نمایی داشتند. همچنین با گرایش جامعه به سمت ایجاد بسترهای نوین پرداخت، در آینده‌ای نزدیک، با پیاده‌سازی استانداردهایی نظیر PSD2، کسب و کارهای نوپا[1] و فین‌تک‌ها آزادی عملی بیشتری در نظام پرداخت پیدا خواهند کرد و به شکل فزاینده‌ای از خدمات پرداخت بهره خواهند برد. در نتیجه نیاز به مقیاس‌پذیری[2] در سوییچ‌های پرداخت برای افزایش کارایی سامانه‌ها امری اجتناب‌ناپذیر می باشد. زمانی که از کارایی در این حوزه سخن به میان می آید با توجه به اینکه، زمان پاسخگویی نیز محدود می باشد، انتظار می رود حجم تراکنش‌های بیشتری نسبت به گذشته در بازه زمانی کوتاهتر، با اطمینان بالا[3] تسویه شوند.

سامانه‌های نظام پرداخت عمدتا شامل تراکنش‌هایی می باشند که جهت انتقال وجه بین بانکی از حساب مبدا به حساب مقصد مورد استفاده قرار می گیرند. این سامانه‌ها دارای خصوصیت‌های مختلفی نظیر برخط و برون خط، پردازش‌های دسته‌ای و انفرادی، انتقال وجه بین بانک[4] و بین مشتری[5] و غیره می باشند. لذا این تراکنش‌ها از بانک مبدا به سوییچ مرکزی ارسال شده، پردازش‌های لازم برروی آن انجام شده و اطلاعات نهایی در سامانه مربوطه در نظام پرداخت ثبت می‌گردد و در نهایت

[1] Startup
[2] Scalable
[3] Reliability
[4] B2B
[5] C2C

تراکنش‌ها به سمت بانک مقصد هدایت می‌شود. این تراکنش‌های بین بانکی باعث تغییر در حساب بانک‌ها نزد بانک مرکزی می‌شوند که به همین منظور، اغلب سامانه‌های نظام پرداخت، دارای فاز تهاتر[1] می‌باشند و در آن حساب بانک‌ها توسط سامانه تسویه[2] دیگر نظام پرداخت، تحت تاثیر قرار می‌گیرد. درکنار تنوع سطح سرویس هر یک از این سامانه‌ها و رسانه‌های درگیر آن‌ها، نظیر Message Queue و فایل، و گرایش رو به افزون جامعه جهت بهره‌برداری از این سامانه‌ها، محدودیت زمانی جهت پردازش تراکنش‌ها و آماده سازی فایل تهاتر، امری اجتناب ناپذیر می‌باشد. لذا، نیاز به افزایش کارایی سامانه‌های مذکور روز به روز بیشتر احساس می گردد.

برای افزایش کارایی، استفاده از معماری‌های توزیع‌شده[3] و بکارگیری فناوری‌های NoSQL، در تولید سامانه‌های مقیاس‌پذیر نقش شایانی خواهند داشت. لذا با این معماری مقیاس‌پذیری افقی و عمودی[4] به سادگی امکان‌پذیر خواهد بود. در عین حال ضعف این دسته از طراحی‌ها، ناپایداری داده‌ها در شرایط بار بالا و از دست دادن بخشی از داده در زمان بروز اختلال در حین اجرای سامانه می‌باشد. چرا که در سوییچ‌های پرداخت، خطا در پردازش تراکنش‌ها باید در هر شرایط، صفر باشد.

برای تمامی پایگاه‌های داده جهت اطمینان‌پذیری سامانه، دو رویکرد مجزا شامل اسید[5] و باز[6]، وجود دارد.[1] رویکرد اسید، بیش از ۳۰ سال است که در حوزه پایگاه‌های داده‌ای رابطه‌ای[7] مورد استفاده قرار گرفته که می توان با افزایش تجهیزات سخت افزاری سرورهای مربوطه، ظرفیت پردازش سیستم توسعه داد[8]. در حالی که رویکرد باز، در حدود ۱۰ سال و با گسترش شبکه های اجتماعی، داده‌های بزرگ[9] و NoSQL ، در دنیای مدیریت داده مطرح شده است. ایده اصلی باز برای افزایش ظرفیت سامانه، توزیع داده‌ها در سرورهای متعدد می باشد[10]. پایگاه‌های داده‌ای مبتنی بر رویکرد باز عموما از نوع بدون ارتباط[11] و با سطح در دسترس بودن بالا می باشند.

در این مقاله سعی شده است یک راهکار جامع برای تولید سامانه‌های مقیاس بزرگ[12] ارائه گردد که قابلیت‌های ثبات[13] و دوام[14] را در کنار مقیاس‌پذیری به صورت هم‌زمان دارا باشد. این خصوصیت باعث می شود، سامانه در عین حالی‌که تعداد تراکنش‌های بیشتری را در لحظه می پذیرد، با وضعیت باثبات خود، قابلیت غلبه بر خطاهای غیر منتظره[15] را نیز دارا باشد. همچنین در این طراحی قابلیت بازیابی سامانه در شرایط غیرمترقبه، در حداقل زمان ممکن حائز اهمیت می باشد. این طراحی با بهره‌گیری از فن‌آوری‌های مبتنی بر اسید و باز یک راهکار جامع برای هماهنگ‌سازی بین آن‌ها ارائه خواهد داد.

در ادامه، مقاله در قسمت ادبیات موضوع اشاره‌ای اجمالی به دستاوردهایی که تا کنون در راستای مقیاس‌پذیری حاصل گردیده، دارد. در قسمت روش تحقیق ابتدا یک طراحی جامع برای مسئله اشاره شده در بالا برای سامانه‌های نظام پرداخت

[1] Cut-Off
[2] Real-Time Gross Settlement(RTGS)
[3] Distributed
[4] Horizontal scaling/Vertical Scaling(Scale out/ Scale up)
[5] Atomicity, Consistency, Isolation, Durability(ACID)
[6] Basically Available, Soft State, Eventual Consistent(BASE)
[7] Relational Databases
[8] Scale-up
[9] Big Data
[10] Scale-out
[11] Non-relational
[12] Large Scale
[13] Consistency
[14] Durability
[15] Failover

ارائه می گردد و سپس نشان خواهیم داد که این طراحی چگونه در هریک از سامانه‌ها منطبق می گردد. در انتها در قسمت یافته‌ها و نتایج به ارزیابی طراحی جدید در مقایسه با طراحی های سنتی پرداخته و نشان خواهیم داد که با این طراحی جامع کارایی افزایش می‌یابد.

۲. ادبیات موضوع

در حوزه طراحی معماری یک سیستم مقیاس‌بزرگ، بر اساس تئوری CAP، هر سیستم داده‌ای با سه چالش اساسی ثبات و دوام داده، دردسترس بودن[۱] و تقسیم پذیری داده[۲] سروکار دارد.[2] بر طبق این تئوری در پردازش‌های توزیع شده، دو خصوصیت امنیت داده‌ای[۳] و فعال بودن[۴] سیستم را نمی‌توان به طور همزمان داشت. امنیت داده‌ای در اینجا به این معنی است که هر پاسخی که برای مشتری ارسال می‌شود، صحیح باشد، حتی اگر اختلالی در حین اجرای سامانه رخ دهد. به عبارت دیگر سامانه همواره در وضعیت با ثبات قرار گیرد. از طرف دیگر، فعال بودن سیستم به این معنی است که هر درخواست از سمت مشتری، پاسخ داده شود که در تئوری CAP با مفهوم در دسترس بودن مشخص شده است. در تئوری CAP این حقیقت بیان می‌شود که سیستم‌های داده‌ای توزیع شده تنها می‌توانند به طور همزمان دو چالش از چالش‌های ثبات، دردسترس بودن و تقسیم پذیری داده را تضمین کنند.

پیاده سازی رویکرد اسید باعث کندی در سامانه‌ها می گردد و قابلیت موازی سازی تراکنش‌ها را در محیط پردازشی توزیع شده کاهش می دهد، در حالی که در رویکرد باز می‌توان پردازش‌ها را موازی انجام داد.[3] از سوی دیگر، اسید اعتبار داده را در هر لحظه تضمین می‌کند، که این موضوع در سامانه‌های نظام پرداخت از اهمیت ویژه‌ای برخوردار می‌باشد. چرا که تراکنش‌های نظام‌های پرداخت عموما دارای بار مالی بوده و تحت هیچ شرایط غیر مترقبه‌ای، نظیر قطعی ارتباط سرورها و از کار افتادن آن‌ها، نباید مغایرتی در تراکنش‌ها ایجاد شود. لذا در سیستم‌های نظام پرداخت، لزوم طراحی سامانه به گونه‌ای که قابلیت‌های ثبات و دوام داده و کنترل سطح موازی سازی پردازش‌ها جهت افزایش کارایی را به طور همزمان داشته باشد، همواره احساس می‌شود.

تخصیص منابع به سیستم‌های کامپیوتری جهت مقیاس‌پذیر نمودن سیستم در راستای افزایش ظرفیت، تازگی ندارد.[4] در دنیای امروز، حجم تراکنش‌ها با در نظر گرفتن محدودیت زمانی برای پاسخگویی به درخواست‌های ارسال شده به سامانه‌ها، افزایش چشمگیری یافته است که سیستم‌های حوزه نظام پرداخت از این قاعده مستثنی نمی‌باشند. یکی از دغدغه‌های اصلی برای طراحی و پیاده سازی این دست سامانه‌ها، افزایش حجم پردازش در یک واحد زمانی (غالبا ثانیه)[۵] می‌باشد. از این رو تمایل به استفاده از قابلیت‌های پردازش موازی در طراحی‌ها، امری اجتناب ناپذیر می‌باشد[5].

در سیستم‌های کامپیوتری توزیع‌شده، پردازش‌ها بر سر دستیابی منابع مشترک با هم در رقابتند و سرویس ارائه‌کنندگان، منابع خود را بین پردازش‌های موازی براساس پارامترهای تعیین شده، توزیع می‌کنند [6]. تقسیم پردازش‌های سنگین به

[1] Availability
[2] Partition Tolerance
[3] Safety
[4] Liveness
[5] Transaction Per Second(TPS)

پردازش‌های کوچکتر و توزیع آن‌ها بین پردازشگرهای متعدد، ایده‌ایست که در دانشگاه‌ها و آزمایشگاه‌ها اغلب با عنوان محاسبات در بستر شبکه[1] برای محاسبات سنگین ریاضی، به چالش کشیده شده است. اما در سیستم‌های بزرگ تجاری[2] به دلیل نیاز به حافظه توزیع شده و سربارهای آن، کمتر از این راهکار استفاده شده است. به این ترتیب با ترکیبی از این راهکارها و رفع چالش‌های مذکور، می توان در سیستم‌های نظام پرداخت، از امکانات پردازش‌های موازی در سیستم‌های توزیع‌شده برای کنترل مناسب توزیع تسک‌ها[3] بر روی سرورهای متعدد براساس ظرفیت پردازشی آن‌ها بهره جست و قابلیت و کارایی سامانه را برای پردازش حجم بیشتری از داده افزایش داد. در این مقاله سعی می‌کنیم به چالش‌های این طراحی بپردازیم.

۳. روش تحقیق

همانطور که در قسمت ۲ اشاره شده است، چالش‌های اصلی که برای طراحی سامانه‌های نظام پرداخت با آن‌ها روبه‌رو هستیم، عبارتند از :

۱- ثبات و دوام داده در کنار در دسترس بودن بلادرنگ آن‌ها

۲- سرعت و کارایی مناسب سامانه

برطرف کردن این چالش‌ها در طراحی، منتج به داشتن سیستمی پایدار با کارایی قابل قبولی خواهد شد، که می‌تواند به عنوان یک سامانه جامع بانکی قابل اعتماد باشد. در ادامه به ارائه راه‌حل‌هایی برای حل این چالش‌ها خواهیم پرداخت.

۳,۱. چالش ثبات ودوام داده در کنار در دسترس بودن بلادرنگ آن‌ها

در این قسمت چالش اول یعنی ثبات و دوام داده در کنار در دسترس بودن بلادرنگ آن‌ها، را مورد بررسی قرار داده و در ادامه راهکارهایی برای آن ارائه می گردد.

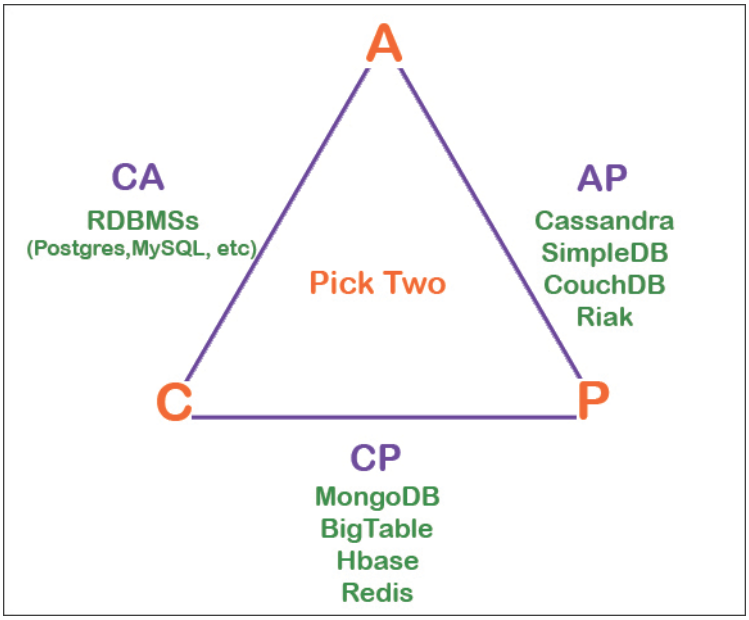
۳,۱,۱. معرفی برخی از پایگاه‌های داده از دو نوع رابطه‌ای و NoSQL

در تئوری CAP همانطور که در شکل ۱ نشان داده شد هر سیستم داده می‌تواند ۳ خاصیت پایه ای را دارا باشد[2]:

- Availability (A): تمامی درخواست‌های داده دارای پاسخ باشند هر چند که این پاسخ معتبر نباشد.
- Consistency (C): سیستم داده‌ای همواره در وضعیت با ثبات و دوام قرار داشته باشد.
- Partition Tolerance (P): داده‌ها در پارتیشن‌های مختلف نگهداری شده و با از بین رفتن یک پارتیشن تنها بخشی از داده در دسترس نبوده و کل سیستم داده‌ای مختل نخواهد شد.

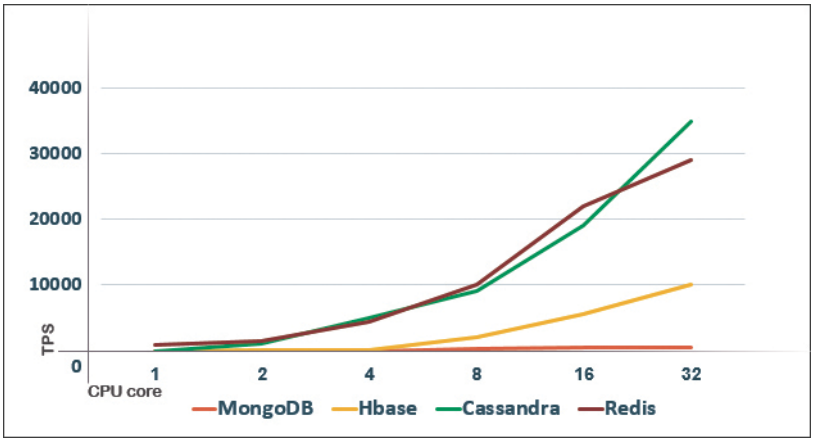
بر اساس این تئوری هر سیستم داده‌ای می‌تواند همزمان تنها دو خاصیت از خاصیت‌های بالا را پوشش دهد. با توجه به ماهیت سامانه‌های پرداخت و بار مالی آن‌ها، باید طرحی ارائه شود که در اولویت اول خاصیت‌های ثبات و دوام و در دسترس بودن را دارا باشد(ضلع CA) و از طرفی در اولویت بعدی باید قابلیت سرعت و ثبات نسبی را درنظر گیرد(ضلع CP).

[1] Grid Computing
[2] Enterprise
[3] Task



شکل ۱- دسته‌بندی سیستم‌های داده‌ای براساس تئوری CAP

پایگاه داده‌های از نوع رابطه‌ای عمدتا در ضلع CA قرار داشته و NoSQL ها در اضلاع CP و AP قرار دارند. برای انتخاب از میان پایگاه‌های داده‌ای NoSQL ، ۴ نوع از آنها انتخاب و معیار[1]های آن‌ها در یک حجم کاری[2] ثابت مورد بررسی قرار گرفت. این حجم کاری شامل سناریوی خواندن و بروزرسانی به طور مساوی[3] می‌باشد.[7] نتایج حاصل از اجرای این سناریو در شکل ۲ قابل مشاهده می‌باشد.



شکل ۲-مقایسه پایگاه‌های داده‌ای NoSQL

[1] Benchmark
[2] Workload
[3] Read/Update (50/50)

مطابق نمودار ۱، دو پایگاه داده Redis و Cassandra نسبت به دو تای دیگر از سرعت بالاتری برخوردارند و از آنجایی که Redis بر روی ضلع CP و Cassandra بر روی ضلع AP قرار دارد. لذا طبق اولویت های اشاره شده، Redis انتخاب مناسب‌تری می‌باشد. Redis یک پایگاه داده داخل حافظه[1] است که با زبان ANSI C پیاده سازی شده است.[8] به همین دلیل، این پایگاه داده مدیریت مناسبی بر روی حافظه دارد، به‌گونه ای که برای نگهداری میلیون‌ها رکورد 100KB تنها به 500GB حافظه نیاز دارد،که اغلب در دسترس می باشد. از این رو می توان از این پایگاه داده، به عنوان یک Cache مشترک نیز استفاده نمود.

۳,۱,۲. مقایسه پایگاه‌های داده از دو نوع رابطه‌ای و NoSQL

پایگاه‌های داده‌ای از نوع رابطه‌ای مبتنی بر خصوصیات اسیدی می‌باشند و در مقابل NoSQL ها عمدتا از خصوصیات بازی بهره برده‌اند. جدول شماره ۱ مقایسه‌ای از خواص هر یک از این رویکردها را نشان می‌دهد.

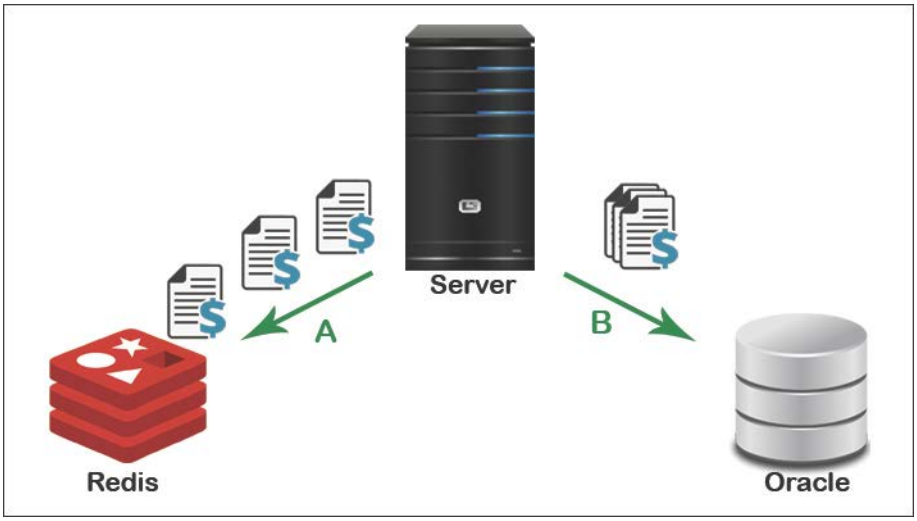
اسید	باز
ثبات قوی	ثبات نسبی/ضعیف
ایزوله بودن تراکنش ها	پاسخ های تقریبی
پشتیبانی از تراکنش های تودرتو	سرعت بالا
در دسترس نبودن در صورت بروز اختلال	همیشه در دسترس
تغییرات در ساختار به راحتی صورت نمی گیرد	راحتی در تغییر در ساختار
تمرکز در Commit	سادگی
محافظه کار/ بد بینانه	جسور/ خوش بینانه

جدول۱– مقایسه خواص رویکردهای اسید و باز

مطابق جدول شماره ۱، برخی از چالش‌های عنوان شده، مانند ثبات و دوام داده، با استفاده از خواص اسیدی در پایگاه‌های داده‌ای رابطه‌ای مرتفع می‌شوند و برخی دیگر، همچون دردسترس بودن و سرعت، با استفاده از خصوصیات بازی در پایگاه‌های داده‌ای NoSQL قابل حل می‌باشند. بنابراین به دنبال طرحی هستیم که بتواند از ترکیب آن‌ها برای مرتفع ساختن چالش-های ۳,۱ به طور همزمان، استفاده نماید. به گونه‌ای که برای نگهداری داده با ثبات و دوام بالا از خاصیت اسیدی و برای

[1] In-Memory

بازیابی و نگهداری سریع داده از خاصیت بازی استفاده نماید. برای این منظور از ترکیب پایگاه داده‌های oracle و Redis استفاده می گردد. به این ترتیب مطابق شکل ۳، داده‌ای که نیاز به نگهداری بلند مدت دارد، به صورت دسته‌ای[1] در Oracle ذخیره می‌شود و داده‌هایی که مراجعات به آن‌ها زیاد می‌باشد، در داخل Redis به عنوان Cache نگهداری خواهند شد.



شکل ۳- انتخاب سیستم داده‌ای جهت نگهداری داده A) تراکنش هایی که باید به صورت تکی مورد پردازش قرار گیرند به Redis ارسال شوند. B) تراکنش هایی که به صورت دسته‌ای مورد پردازش قرار می گیرند، به سمت Oracle هدایت می شوند.

۳،۲. چالش موازی سازی پردازش

همانطور که قبلا اشاره شد، چالش دوم سرعت و کارایی مناسب سامانه می‌باشد که برای این منظور موازی‌سازی امری بدیهی است. برای انتخاب بستر در راستای موازی‌سازی پردازش‌های سامانه‌های نظام پرداخت، با توجه به حساسیت داده‌ها، باید ۴ خصوصیت زیر را در نظر داشت.

۱- مقیاس‌پذیری : بتواند با افزایش سخت افزار در حالت فشار به پردازش حجم بیشتری از تراکنش‌ها در واحد زمان کمک کند

۲- توازن بار[2]: دارای قابلیت تقسیم و توزیع پردازش را در قالب ریز پردازش‌هایی با عنوان تسک متناسب با ظرفیت سیستم باشد

۳- ثبات و دوام پردازش : دارا بودن کنترل مناسب بر روی نحوه اجرا و تکمیل پردازش. در اینجا ثبات پردازش به این معنی است که، پردازش به شکل ناقص انجام نشده باشد.

۴- در دسترس بودن : چنانچه اختلالی در بخشی از سیستم‌های پردازشی رخ دهد، باید سایر سیستم‌های فعال قادر به ادامه عملیات خود باشند

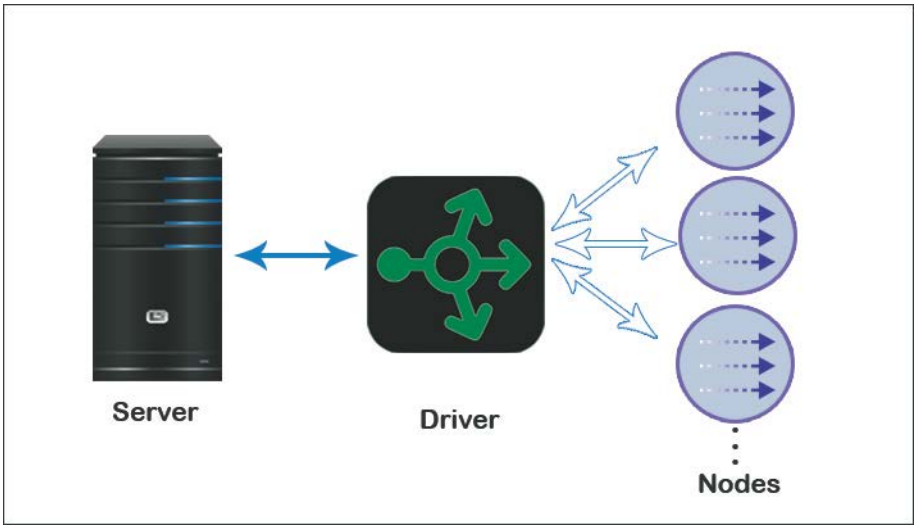
[1] Bulk
[2] Load Balance

با توجه به نیازمندی‌های ذکر شده و بررسی‌های انجام شده، برای امر موازی‌سازی سامانه، بستر JPPF[9][1] انتخاب مناسبی می‌باشد. مطابق شکل ۴ JPPF یک بستر سه مولفه‌ای برای توزیع و موازی‌سازی پردازش‌ها می‌باشد که این سه مولفه عبارتند از:

۱- سرور : این مولفه نقش تقسیم یک پردازش به تسک‌های موازی و قابل اجرا در سیستم‌های مجزا را داراست. این تسک‌ها به صورت دسته‌ای در قالب یک واحد کار[2] آماده ارسال به مولفه بعدی می‌گردد

۲- درایور[3] : توازن بار، کنترل توزیع و تجمیع تسک‌ها در راستای ثبات و دوام پردازش و تا حدی در دسترس بودن، وظیفه اصلی این مولفه می‌باشد. تجمیع تسک‌های یک واحد کار در این بخش صورت می‌گیرد و نتیجه آن برای سرور ارسال می‌گردد.

۳- نود[4] : در عین حال که وظیفه اجرای تسک را بر عهده دارد، خصوصیت مقیاس‌پذیری در این مولفه وجود دارد، به این ترتیب که می‌توان در صورت نیاز، نودها را اضافه و یا کم کرد.



شکل ۴- مولفه‌های اصلی معماری JPPF

۳,۳. طرح جامع معماری

با جمع‌بندی نتایج قسمت‌های قبل، در این قسمت طراحی کلان از معماری سامانه‌هایی که ماهیت پردازش‌های آن‌ها با توجه به تعریفی که در قسمت ۱ ارائه شد، به سامانه‌های نظام پرداخت شباهت دارد، ارائه می‌گردد. در ابتدا فرایندهای اصلی سامانه باید به واحدهای کوچکتری تحت عنوان تسک شکسته شود. به عبارت دیگر، هر تسک یک واحد کار می‌باشد که دارای

[1] Java Parallel Processing Framework
[2] Job
[3] Driver
[4] Node

خصوصیات زیر است:

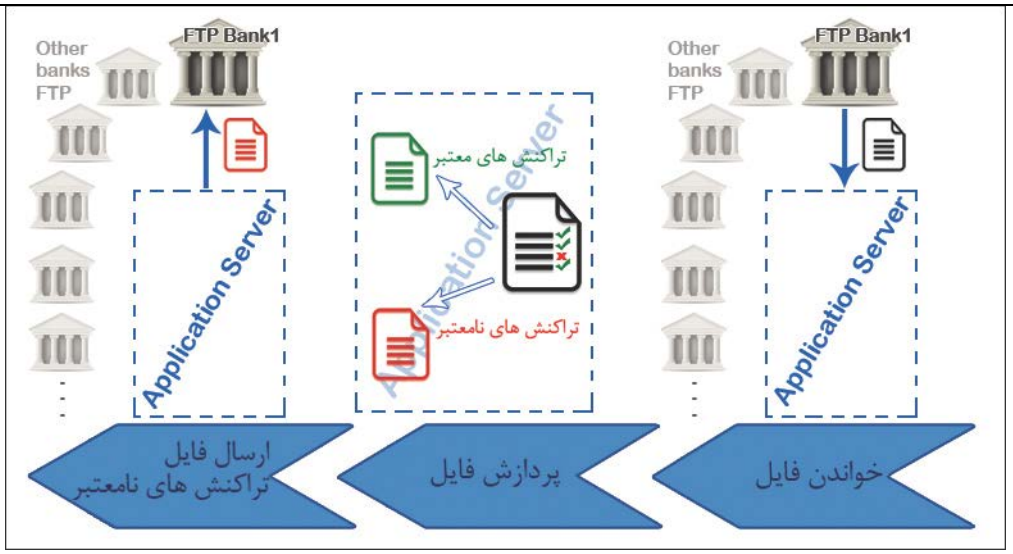
- باید در یک پردازنده بر روی یکی از سرورهای محیط توزیع شده اجرا گردد
- باید قابلیت اجرای موازی در کنار سایر تسک‌های یک فرایند را داشته باشد
- باید به صورت اتمی[1] اجرا گردد. به عبارت دیگر، در صورت بروز خطای سیستمی، کل فرایند برگشت[2] داده شود و برروی پردازنده دیگر دوباره از سرگرفته خواهد شد
- جهت جلوگیری از پیچیدگی طراحی، توصیه می‌شود هر تسک یک تراکنش از پایگاه داده را در اختیار گیرد، با آنکه الزامی در این رابطه وجود ندارد

مجموع تسک‌ها یک واحد کار را تشکیل می‌دهند. به عبارت دیگر، تسک‌ها همان واحدهای موازی در برنامه است که در نتیجه سامانه مقیاس پذیر می‌گردد و با تنظیم پارامترهای مناسب می‌توان به حداکثر بهره‌وری رسید. به طور مثال، در سناریو فرضی که منطبق بر سامانه‌های نظام پرداخت می باشد، در یک چرخه باید فایل‌های حاوی تراکنش‌های بانک‌های عضو همزمان پردازش شده و اعتبار سنجی گردد و تراکنش‌های معتبر برای عملیات تسویه و تهاتر آخر روز نگهداری گردد. بنابراین همانطور که در شکل ۵ نشان داده شد، کل این چرخه را می توان به ۳ واحد کار تقسیم کرد:

۱. خواندن فایل‌های بانک‌های عضو، از FTP های مربوطه
۲. پردازش هر فایل شامل اعتبارسنجی فایل، استخراج تراکنش‌های موجود در آن، جدا سازی تراکنش‌های معتبر و نامعتبر، ذخیره سازی آن‌ها جهت اعلام به بانک فرستنده
۳. ارسال تراکنش‌های نا معتبر به بانک فرستنده به صورت فایل پاسخ و از طریق FTP

بر این اساس هر یک از این واحدهای کار می‌توانند به تسک‌های موازی شکسته شوند، که برروی سرورهای توزیع شده به صورت متوازن بر اساس مشخصات هریک از آن سرورها اجرا می‌شوند. در مرحله اول، به ازای هر بانک یک تسک ایجاد می‌شود که فایل‌ها را از FTP می‌خواند و در جدولی در پایگاه داده ذخیره می‌کند. بنابراین در این مرحله به تعداد بانک‌های عضو، تسک موازی خواهیم داشت. در مرحله ی بعد به ازای هر فایل یک تسک خواهیم داشت که تراکنش‌های مربوط به همان فایل را استخراج کرده و به صورت دسته‌ای در پایگاه داده ذخیره می‌کند. لذا در واحد کاری دوم، به تعداد فایل‌های یک چرخه تسک موازی خواهیم داشت که به طور متوسط بین ۶۰ الی ۱۲۰ فایل خواهد بود. و در انتها، در مرحله سوم مجددا به ازای هر بانک می‌توان تسکی جهت ارسال فایل‌ها از پایگاه داده به درگاه FTP بانک مورد نظر، در نظر گرفت. مجددا سطح موازی-سازی در این مرحله به تعداد بانک‌ها خواهد بود.

[1] Atomic
[2] Rollback

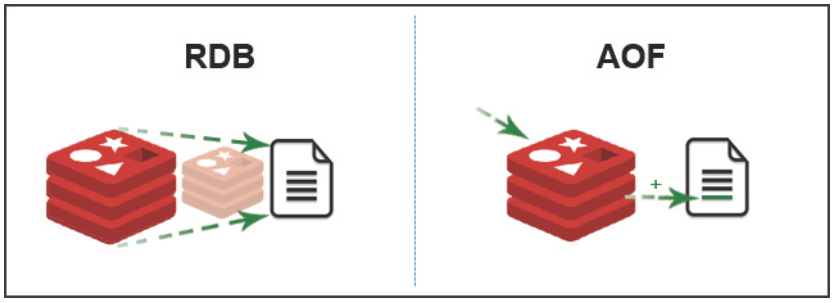


شکل ۵ -مراحل پردازش تراکنش‌های بانکی در سامانه فرضی پرداخت

همانطور که در قسمت ۳,۲ اشاره شد بستر JPPF تضمین می‌کند که حتما همه تسک‌های ایجاد شده در یک واحد کار به صورت کامل اجرا شود و در این رابطه خود را به صورت اتوماتیک بر اساس شرایط سخت افزار محیط اجرا، تطبیق خواهد داد و از این طریق، توازن اجرا صورت می‌پذیرد. به طور مثال، در سروری که پردازنده ۴ هسته‌ای با 64GB حافظه دارد، تعداد تسک‌های موازی اجرا شده در آن به مراتب کمتر از سروری می‌باشد که دارای پردازنده ی ۱۶ هسته‌ای با 128GB حافظه است.

همانطور که در بخش ۳,۱,۲. اشاره شد جهت نگهداری دائمی داده‌ها از پایگاه داده Oracle استفاده می‌شود. به طور مثال، تراکنش‌های تایید شده جهت عملیات تهاتر پایان روز در این قسمت ذخیره می‌گردند. طبق این طراحی دسترسی به این پایگاه داده تنها به صورت دسته‌ای میسر می‌باشد. از سوی دیگر، برای داده‌های پر مراجعه و موردی از Redis به عنوان سرور Cache استفاده می‌گردد. به عنوان مثال، تراکنش‌های اولیه جهت اعتبارسنجی در این قسمت نگهداری می‌شوند. یکی از اعتبارسنجی‌های تراکنش به این صورت می باشد، که شناسه یک تراکنش در کل یک چرخه نباید تکراری باشد. بنابراین شناسه هریک از تراکنش‌های درحال بررسی، در پایگاه داده Redis چک می‌شود و در صورت برقراری این شرایط، آن را به عنوان تراکنش معتبر در ساختار داده‌ای که برای این منظور در Redis تعبیه شده است، ذخیره می‌گردد و در صورت نا معتبر بودن در ساختار داده‌ای دیگری در همان محیط ذخیره می‌شود. در صورتی که سرور Redis دچار اختلال گردد، جهت بازیابی آن باید در حین اجرا، به گونه ای که سربار اضافی به سیستم تحمیل ننماید، یک نسخه پشتیبان از حافظه در دیسک نگهداری شود. در شکل ۶ این مکانیزم نشان داده شده است. نکته‌ای که همواره باید مورد توجه قرار گیرد این است که داده-های موجود در Redis باید به گونه‌ای باشند که در کوتاه مدت پاکسازی[1] گردند. زیرا چنانچه داده‌های ماندگار در Redis نگهداری گردند، حجم فایل پشتیبان آن به صورت فزاینده‌ای رشد می‌کند و باعث کندی در عملیات درج، ویرایش و حذف می‌گردد که به هیچ عنوان بهینه نخواهد بود.

[1] Housekeeping



شکل ۶- انواع پشتیبان‌گیری در Redis

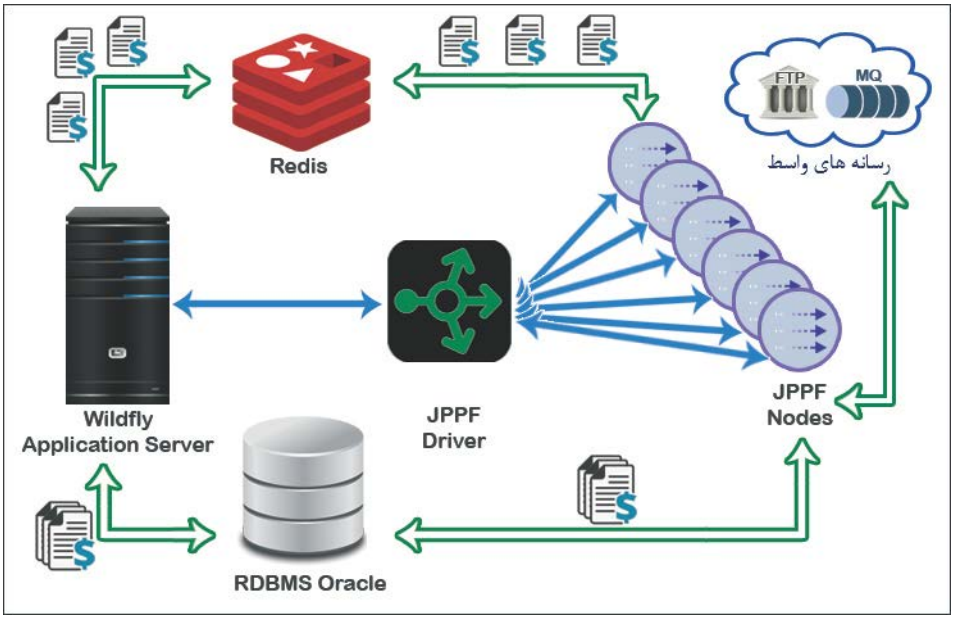
در واقع در Redis دو رویکرد برای پشتیبان گیری[1] وجود دارد. در رویکرد اول با عنوان RDB، سیستم Redis در بازه‌های زمانی که در فایل پیکربندی آن مشخص می‌شود، از کل ساختار حافظه پشتیبان گرفته و آن را به صورت بهینه در فایلی ذخیره می‌کند. خود این این فرایند پشتیبان‌گیری می‌تواند به صورت موازی در کنار سایر فرایندهای درج و حذف اجرا شود، که این حالت از نظر سرعت بالاترین بهره‌وری را داراست. اما ثبات سیستم تضمین نمی شود، چرا که ممکن است قبل از فرایند پشتیبان گیری، در Redis تراکنشی ذخیره شود و در این حین سرور پایین بیاید، در نتیجه از آنجا که تراکنش فقط در حافظه ذخیره شده است، از بین می‌رود. این اتفاق برای تراکنش‌های دارای بار مالی در حوزه نظام پرداخت بسیار حیاتی بوده و در هیچ حالتی نباید مشاهده شود. در رویکرد دوم، تحت عنوان AOF، هر یک از دستورات درج، ویرایش و حذف قبل از ذخیره‌سازی در حافظه به صورت سیاهه‌ای در قالب فایل پشتیبان ذخیره می شود، که البته بهینه نمی باشد. Redis تضمین می دهد که این فایل سیاهه همواره با تراکنش های حافظه هماهنگ باشد، لذا در هنگام بازیابی بعد از خرابی این سیاهه به ترتیب از ابتدا اجرا شده و وضعیت حافظه را به آخرین وضعیت قبل از خرابی بر می‌گرداند. با این که این بازیابی کمی زمان بر است، ولی به دلیل متناوب نبودن آن و اجرا در مواقع غیر مترقبه، قابل پذیرش است. از سوی دیگر، سربار عملیاتی آن در حین اجرا نیز در حدی می‌باشد که چندین برابر ظرفیت پردازش‌های سیستم های پرداخت را پشتیبانی می-نماید.

طرح جامع معماری سیستم‌های پرداخت در شکل ۷ نشان داده شد. این طرح از ۵ مولفه اصلی تشکیل شده است:

- **سرور مرکزی:** جهت پیادسازی سامانه مد نظر و تعریف واحدهای کاری
- **متوازنگر تسک‌ها:** جهت توازن بار در نودها، دسترس‌پذیری بالا[2] و تضمین اجرای تسک‌های یک واحد
- **نود:** جهت اجرای تسک‌ها
- **پایگاه داده مستقر در حافظه:** جهت نگهداری داده‌های با طول عمر کوتاه و با تعداد مراجعات موردی و بالا
- **پایگاه داده رابطه‌ای:** جهت نگهداری داده‌های دائمی و حساس سامانه. بهتر است مراجعه به این پایگاه داده به

[1] Backup
[2] High-Availability

صورت دسته‌ای باشد، به گونه‌ای که هر پرسش و پاسخ با تعداد بالایی از تراکنش‌ها سر و کار داشته باشد



شکل ۷- معماری کلان طرح جامع پیشنهادی

این معماری خصوصیات زیر را که پیشتر به آن‌ها به طور مفصل اشاره شد، تضمین می نماید:

- موازی‌سازی به صورت پارامتریک و تطبیق پذیر با محیط مقصد
- استفاده از حداکثر ظرفیت منابع درگیر
- در دسترس بودن، ثبات و دوام داده‌ها به گونه‌ای که هر مولفه ذخیره‌سازی بخشی از این وظایف را بر عهده دارد
- دسترس‌پذیری بالا از طریق نودها خوشه[1] master/slave
- احیاپذیری سامانه از آخرین وضعیت قبل از خرابی

بدین ترتیب، تمامی اهداف مشخص شده حاصل گردید و برای هریک از آن ها، در طرح جامع راهکار مشخص شد.

۳,۴. تطبیق طرح جامع با سامانه‌های نظام پرداخت

سامانه‌های پرداختی، در حالت کلی به دو دسته تقسیم می‌شوند. سامانه‌هایی که دستورهای پرداخت/برداشت را به صورت دسته‌ای و در چرخه‌های تعریف شده مورد پردازش قرار می‌دهند، مانند پایا و سامانه‌هایی که این دستورها را به صورت موردی پردازش می‌کنند، مانند چکاوک.

به عنوان یک طراحی برای سامانه‌های پردازش دسته‌ای، اساس کار بر مبنای فایل‌های حاوی تراکنش‌های یک بانک می‌باشد.

[1] Cluster

تراکنش‌ها ابتدا جهت اعتبارسنجی در Redis ذخیره شده و به دو دسته تراکنش‌های معتبر و نامعتبر تقسیم می‌شوند. تراکنش‌های معتبر یک فایل به صورت درج دسته‌ای[1]، بدون هیچ پردازش ثانویه دیگری در پایگاه داده رابطه‌ای Oracle ذخیره می‌گردد. در انتهای روز عملیات تهاتر با واکشی مناسب از پایگاه داده Oracle صورت می‌پذیرد و همواره در هر لحظه وضعیت تراکنش‌ها و حساب یک بانک به وسیله سامانه‌های گزارشات و مانیتورینگ قابل مشاهده می‌باشد.

در سامانه‌های پردازش موردی، نیز طراحی به این صورت می‌باشد که هر تراکنش ارسالی از بانک مبدا می‌تواند در لحظه دریافت در Redis ذخیره شده و وضعیت اعتبار آن بررسی گردد و در صورت معتبر بودن، به بانک مقصد ارسال می‌گردد. از سوی دیگر، باید به ازای هر یک از بانک‌های درگیر درسامانه، تسک‌هایی در قالب یک واحد کار در فرایندی مجزا در بازه‌های زمانی مشخص ایجاد شود که تراکنش‌های معتبر را از Redis برداشته و در پایگاه داده Oracle ذخیره نماید، تا جهت عملیات نهاتر آخر روز و سامانه های گزارشات و مانیتورینگ، به صورت دسته ای مورد استفاده قرار گیرد.

از آنجا که پایگاه داده‌های Redis و Oracle طبق راهکارهایی که در قسمت ۳،۳ اشاره شد، همواره در وضعیت با ثبات قرار دارند، بنابراین همیشه ثبات سیستم تضمین می‌شود.

۴. یافته‌ها و نتایج

جهت ارزیابی طرح جامع ارائه شده، سه حجم کاری مختلف در نظر گرفته شده است. این سناریوها با دو طراحی مورد آزمون قرار گرفته‌اند. طرح اول، طراحی مستقل[2] مبتنی بر پایگاه داده رابطه‌ای (Oracle) می‌باشد. طرح دوم، طرح جامع ارائه شده در بخش ۳،۳ می‌باشد، که در آن علاوه بر موازی‌سازی پردازش‌ها در بستر JPPF از سیستم داده‌ای Redis نیز در کنار Oracle برای نگهداری داده‌ها استفاده شده است.

در طراحی اول، ذخیره، واکشی و بروزرسانی داده‌ها در Oracle هم به صورت موردی و هم به صورت دسته‌ای صورت می‌گیرد، در حالی که در طرح جامع پیشنهادی در این مطالعه، واکشی و ذخیره‌های موردی در سرور داخل حافظه‌ای Redis و موارد دسته‌ای در پایگاه داده Oracle رخ می‌دهد.

برای ارزیابی طرح پیشنهادی، بستری با مشخصات زیر فراهم گردید:

۱- یک سرور با 16 Core و 64GB حافظه برای نصب سرور و درایور

۲- ۴ سرور با مشخصات 16 Core و 128GB حافظه برای استقرار نودها

۳- یک سرور با 16 Core و 128GB حافظه به عنوان سرور Oracle

۴- یک سرور با 4 Core و 256GB حافظه به همراه Redis به عنوان سرور Cache

نتایج مقایسه این دو طرح برای حجم کارهای در نظر گرفته شده با تعداد 2,500,000 تراکنش در سقف زمانی ۱ ساعت، در جدول شماره ۲ قابل مشاهده می‌باشد این حجم کارها عبارتند از:

۱- ذخیره در داخل پایگاه داده

۲- واکشی موردی از پایگاه داده برای اعتبارسنجی و بروزرسانی داده

[1] Bulk-Insert
[2] Standalone

۳- واکشی از داده‌های معتبر دائمی

سناریو	طرح اولیه (TPS)	طرح جامع (TPS)
حجم کاری ۱	۷۲۰	۸۴۹۶
حجم کاری ۲	ناتمام در بازه زمانی ۱ ساعته	۲۱۹۴
حجم کاری ۳	ناتمام در بازه زمانی ۱ ساعته	۱۵۵۴۳

جدول۲– مقایسه طراحی مستقل مبتنی بر پایگاه داده رابطه‌ای و طرح جامع پیشنهادی

حجم کارهای ارائه شده، عملیاتی هستند که از سامانه های نظام پرداخت اقتباس شده اند. طی مطالعه‌ای که در بانک مرکزی صورت پذیرفته است، حجم تراکنش‌های سامانه‌های نظام پرداخت در آینده‌ای نزدیک، ماهانه به طور متوسط به بیش از ۲۵۰ میلیون خواهد رسید که ۶۰ درصد از آنها در روزهای پر بار ماه(روزهای پنجم، دهم، پانزدهم و ...) توزیع می شوند. به این ترتیب در هر روز پرفشار در حدود ۲۵ میلیون تراکنش وجود خواهد داشت که در بدترین حالت باید در بازه زمانی ۱ ساعته مورد پردازش قرار گیرند. بنابراین TPS مطلوب برای سامانه‌ها در حدود ۷۰۰۰ بود.

به این ترتیب، مطابق نتایج جدول شماره۲، می توان افزایش محسوس کارایی و سرعت را با به کارگیری Redis به عنوان یک سرور Cache در کنار Oracle و بستر موازی سازی JPPF، مشاهده کرد. نکته حائز اهمیت در این طراحی این است که با افزایش سرورهای سخت‌افزاری درگیر در هر لحظه (مقیاس پذیری افقی)، همواره می توان ظرفیت پردازشی سامانه را به بیش از مقادیر مذکور رساند. در این حالت، در کنار نگهداری با ثبات داده مورد نیاز برای گزارشگیری، می‌توان از سرعت بالای پردازش برای دستیابی به TPS مطلوب اشاره شده، نیز بهره‌مند شد.

۵.جمع بندی

در این مطالعه نشان دادیم که در تراکنش‌های سیستم‌های مقیاس بزرگ سامانه‌های پرداخت، عمدتا به دلیل بار مالی آن‌ها، باید داده‌ها با ثبات و با دوام نگهداری گردند و در عین حال میزان دسترسی به آن‌ها بالا باشد. در کنار این خصایص باید سیستم در پردازش تراکنش‌ها با سرعت بالایی عمل نماید تا بتواند پاسخگوی نیاز روزافزون جامعه به این سیستم‌ها باشد.

در ادامه نشان دادیم که پاسخگویی لحظه‌ای سیستم داده‌ای به گونه‌ای که داده‌های معتبر را همواره در اختیار برنامه کاربردی مقیاس‌پذیر قرار دهد، در تناقض با فاکتور کارایی و سرعت سامانه می‌باشد. به عبارت دیگر سیستم داده‌ای برای پاسخگویی سریع به درخواست‌ها باید اعتبار داده‌ها را قربانی نماید و در غیر این صورت به دلیل منتظر نگهداشتن درخواست‌ها، جهت ارائه نسخه مورد تایید داده مورد نظر، سرعت پردازش افت شدیدی پیدا خواهد کرد. لذا در این مطالعه با توجه به خواص سیستم-های داده‌ای رابطه‌ای و NoSQL به همراه توسعه سامانه در بستری مقیاس‌پذیر برای پردازش‌های موازی، طرحی جامع برای این گونه سامانه‌ها ارائه گردید، که با تقسیم وظایف به مولفه‌های مختلف آن، سامانه فاکتورهای ثبات و دوام داده در کنار

مقیاس‌پذیری و سرعت پاسخگویی را دارا می‌باشد. جهت غلبه بر شکست، در این طراحی تمهیداتی در نظر گرفته شد تا در مواقع خرابی سرورها و قطعی احتمالی، پس از رفع خطا، سیستم همواره بتواند خود را با آخرین وضعیت قبل از خطا بازیابی نماید.

در انتها، در بخش یافته‌ها و نتایج نشان دادیم با اعمال این طراحی، سرعت پردازش در واحد ثانیه افزایش چشمگیری داشته و این سرعت همواره با توجه به نیاز جامعه از طریق مقیاس پذیری افقی و افزودن سرورهای جدید قابل تطبیق می باشد. در عین حال، ثابت شد که سربار حاصل از ثبات و دوام داده تاثیر چندانی برروی کارایی آن نخواهد داشت. به عبارت دیگر، خطای گم شدن تراکنش‌های مالی در وضعیت‌های مختلف صفر می‌باشد و در عین حال سامانه با حداکثر سرعت و توان پردازشی سرورهای درگیر به پاسخگویی تراکنش‌ها می‌پردازد.

تعمیم‌پذیری این طراحی برای سایر سامانه‌های مقیاس بزرگ بانکی نظیر Core-Banking و زیر سیستم‌های آن را نیز می‌تواند به عنوان یک مطالعه جهت کارهای آتی در نظر گرفت. چرا که قطعا این سامانه‌ها دارای ماهیت‌های متفاوتی بوده و چالش‌های جدیدی در پیش رو خواهند داشت. لذا در این طرح جهت تطبیق با سامانه مد نظر و رفع چالش‌های آن، باید تمهیداتی در نظر گرفته شود.

منابع

[1] Banothu, N., Bhukya, S. and Sharma, K.V., 2016, March. Big-data: Acid versus base for database transactions. In Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on (pp. 3704-3709). IEEE.

[2] Gilbert, S. and Lynch, N., 2012. Perspectives on the CAP Theorem. Computer, 45(2), pp.30-36.

[3] Frank, L., 2011, April. Countermeasures against consistency anomalies in distributed integrated databases with relaxed ACID properties. In Innovations in Information Technology (IIT), 2011 International Conference on (pp. 266-270). IEEE.

[4] Danak, A. and Mannor, S., 2010, June. Resource allocation with supply adjustment in distributed computing systems. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on* (pp. 498-506). IEEE.

[5] Júnior, A.M.G., Sato, L.M. and Massetto, F.I., 2012, December. A parallel application programming and processing environment proposal for grid computing. In *Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on* (pp. 154-161). IEEE.

[6] Siddiqui, M., Villazón, A. and Fahringer, T., 2006, November. Grid capacity planning with negotiation-based advance reservation for optimized QoS. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing* (p. 103). ACM.

[7] https://research.yahoo.com/news/yahoo-cloud-serving-benchmark

[8] https://redis.io/

[9] http://www.jppf.org/