سامانه‌های پردازشی و ارتباطی چندرسانه‌ای هوشمند
Intelligent Multimedia Processing and Communication Systems (IMPCS)

Summer 2024, 5 (2), 105-118
DOR:

Received: 22 Apr 2024
Accepted: 5 June 2024

مقاله پژوهشی

Leveraging Consensus Algorithms to Optimize Microservice Architecture

Esmail Sadeghi Hafshejani[1], Mahmood Deypir[2*], Ali Broumandnia [3]

1. Department of Computer Engineering, ST.C, Islamic Azad University, Tehran, Iran .
2. Department of Computer Engineering, ST.C, Islamic Azad University, Tehran, Iran.
**Corresponding Author*, mdeypir@iau.ac.ir
3. Department of Computer Engineering, ST.C, Islamic Azad University, Tehran, Iran.

Abstract

Introduction: Microservice architectures (MSAs) have transformed the development of scalable distributed systems by breaking monolithic applications into smaller, independent services. Despite their advantages in scalability, flexibility, and maintenance, MSAs face significant challenges in achieving state consistency and fault tolerance. This study aims to address these issues by leveraging Paxos consensus algorithms to enhance the reliability and performance of microservice systems.

Method: The proposed approach integrates Paxos consensus mechanisms into microservice systems to ensure state consistency and fault tolerance. Simulations and real-world case studies were conducted, comparing our methodology with existing optimization frameworks, such as SPNs and NSGA-II. This method emphasizes dynamic resource allocation and automated scaling under various workload conditions.

Results: The results demonstrate that Paxos-based optimization reduces latency by 25%, increases throughput by 30%, and enhances resource utilization efficiency by 20%. Additionally, it achieves a high consistency rate of 99.9%, ensuring reliable state updates across distributed nodes. The system also showed robust fault tolerance, maintaining 95% operational success during node failures.

Discussion: The findings highlight the superiority of Paxos in optimizing microservice deployments, particularly in dynamic cloud environments. This study underscores the importance of integrating consensus protocols into optimization frameworks for improved performance, reliability, and scalability. Future research could explore further enhancements in energy efficiency, security measures, and deployment across diverse cloud scenarios.

Keywords: Consensus Algorithms, Fault Tolerance, Microservice, Paxos Protocol, Resource Allocation, Scalability

سامانه‌های پردازشی و ارتباطی چندرسانه‌ای هوشمند
Intelligent Multimedia Processing and Communication Systems (IMPCS)
انجمن علمی تجارت الکترونیکی ایران
واحد زنجان

استفاده از الگوریتم‌های اجماع برای بهینه‌سازی معماری میکروسرویس

تاریخ دریافت: ۱۴۰۳/۰۲/۰۳
تاریخ پذیرش: ۱۴۰۳/۰۳/۱۶
دورهٔ پنجم، تابستان ۱۴۰۳
شمارهٔ دوم، صص: ۱۰۵-۱۱۸

اسماعیل صادقی هفشجانی[۱]، محمود دی‌پیر [۲*]، علی برومندنیا[۳]

۱- گروه مهندسی کامپیوتر، واحد تهران جنوب، دانشگاه آزاد اسلامی، تهران، ایران. esmail.sadeghi@iau.ac.ir
۲- گروه مهندسی کامپیوتر، واحد تهران جنوب، دانشگاه آزاد اسلامی، تهران، ایران. (نویسندهٔ مسئول) mdeypir@iau.ac.ir
۳- گروه مهندسی کامپیوتر، واحد تهران جنوب، دانشگاه آزاد اسلامی، تهران، ایران. Ali.Broumandnia@iau.ac.ir

چکیده: معماری‌های میکروسرویس (MSAs) به یک رویکرد اساسی برای ایجاد سیستم‌های توزیع شده مقیاس‌پذیر تبدیل شده‌اند. با این حال، دستیابی به سازگاری و تحمل خطا همچنان یک چالش بزرگ است. این مطالعه با هدف بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس با ادغام الگوریتم‌های اجماع Paxos انجام شده‌است. روش‌شناسی ما شامل استفاده از Paxos برای تضمین سازگاری و تحمل خطا در سیستم‌های میکروسرویس می‌باشد. شبیه سازی‌ها و مطالعات موردی برای ارزیابی عملکرد رویکرد ما انجام شده‌است. نتایج نشان‌می‌دهد که Paxos تأخیر را کاهش داده، رقابت منابع را به حداقل می‌رساند و سازگاری حالت را در خدمات توزیع‌شده تضمین‌می‌کند. بهینه‌سازی مبتنی بر Paxos نه‌تنها عملکرد را بهبود می‌بخشد، بلکه قابلیت اطمینان و دسترسی سیستم‌های میکروسرویس را نیز افزایش می‌دهد. این مطالعه مزایای قابل‌توجهی از Paxos در بهینه‌سازی استقرارهای میکروسرویس را نشان‌می‌دهد.

واژه‌های کلیدی: الگوریتم‌های اجماع، پروتکل Paxos، تحمل خطا، تخصیص منابع، میکروسرویس، مقیاس‌پذیری.

۱. مقدمه

معماری‌های میکروسرویس [5][4][3][2][1] با شکستن برنامه‌های یکپارچه به سرویس‌های کوچکتر و مستقل، طراحی و استقرار سیستم‌های توزیع‌شده را متحول کرده‌اند. این رویکرد مقیاس‌پذیری، انعطاف‌پذیری و نگهداری را بهبودبخشیده و آن را به انتخابی محبوب برای برنامه‌های مدرن مبتنی بر ابر تبدیل کرده‌است. با این حال، پیچیدگی ذاتی مدیریت تعداد زیادی میکروسرویس، چالش‌هایی را در زمینه سازگاری [8][7][6][4]، تحمل خطا و بهینه‌سازی عملکرد ایجادمی‌کند[9].

یکی از مسائل مهم در سیستم‌های میکروسرویس، اطمینان از عملکرد قابل‌اعتماد و کارآمد همه سرویس‌ها[10] ، حتی در شرایط بار متغیر و خرابی‌های احتمالی است. تکنیک‌های بهینه‌سازی سنتی اغلب در متعادل‌سازی این نیازها دچار مشکل می‌شوند و منجر به گلوگاه‌های عملکردی و نگرانی‌های مربوط به قابلیت اطمینان می‌شوند. برای رفع این چالش‌ها، الگوریتم‌های اجماع، به‌ویژه Paxos، راه‌حلی قوی ارائه‌می‌دهند که با فراهم‌کردن مکانیزمی برای دستیابی به توافق بین گره‌های توزیع‌شده، سازگاری و تحمل خطا را بهبودمی‌بخشد [14][13][12][11].

در این مقاله، ما پیشنهادمی‌کنیم از الگوریتم‌های اجماع Paxos برای بهینه‌سازی سیستم‌های میکروسرویس استفاده‌شود. رویکرد ما بر بهبود قابلیت اطمینان و عملکرد میکروسرویس‌ها با اطمینان از تغییرات حالت سازگار و تخصیص منابع کارآمد تمرکز دارد. ما روش پیشنهادی خود را با چارچوب ارائه‌شده در تحقیقات قبلی [15] مقایسه‌می‌کنیم که از شبکه‌های پتری تصادفی SPNs، الگوریتم ژنتیک مرتب‌سازی غیرمسلط (NSGA-II) II و رگرسیون جنگل تصادفی RFRبرای بهینه‌سازی عملکرد استفاده‌می‌کند[15].

چارچوب کنونی به بررسی بهینه‌سازی عملکرد از طریق مدل‌سازی مکانیزم‌های خودکارسازی و تخصیص منابع در یک محیط ابر خصوصی می‌پردازد. این چارچوب به‌طور مؤثری توازن‌های حیاتی بین عملکرد و مصرف منابع را مدیریت‌می‌کند[18][17][16]. را شناسایی می‌کند و بینش‌های ارزشمندی در مورد رفتار میکروسرویس‌ها در پیکربندی‌های مختلف ارائه‌می‌دهد. با این حال، این چارچوب عمدتاً بر معیارهای عملکردی مانند توان عملیاتی و زمان پاسخ تمرکز دارد و به چالش‌های سازگاری و تحمل خطا به‌طور صریح نمی‌پردازد.

الگوریتم بهینه‌سازی مبتنی بر Paxos ما با ادغام مکانیزم‌های اجماع در فرآیند بهینه‌سازی، این شکاف را پر می‌کند. با این کار، اطمینان حاصل‌می‌کنیم که سیستم‌های میکروسرویس می‌توانند حتی در صورت خرابی گره‌ها و تغییرات پویا در بار کاری، در دسترس و قابل اعتماد باقی‌بمانند. از طریق شبیه‌سازی‌های گسترده و مطالعات موردی واقعی، نشان‌می‌دهیم که رویکرد ما نه‌تنها معیارهای عملکرد را بهبود می‌بخشد، بلکه به‌طور قابل‌توجهی استحکام کلی سیستم‌های میکروسرویس را نیز افزایش می‌دهد [2][1] .

این مقاله به شرح زیر ساختار یافته‌است: بخش ۲ مروری بر کارهای مرتبط ارائه‌می‌دهد و نقاط قوت و محدودیت های چارچوب های بهینه‌سازی موجود را برجسته‌می‌کند. بخش ۳ الگوریتم بهینه‌سازی مبتنی بر Paxos پیشنهادی ما را با جزئیات کلیدی و پیاده‌سازی آن توضیح‌می‌دهد. بخش ۴ روش شناسی و نتایج ارزیابی را ارائه‌می‌دهد و رویکرد ما را با چارچوب پایه مقایسه‌می‌کند. بخش ۵ پیامدهای یافته‌های ما و زمینه‌های بالقوه برای تحقیقات آینده را مورد بحث قرارمی‌دهد. در نهایت، بخش ۶ مقاله را با خلاصه‌ای از مشارکت‌ها و مزایای کلیدی روش پیشنهادی ما به پایان می‌رساند.

۲. کارهای مرتبط

بهینه‌سازی سیستم‌های میکروسرویس به‌طور گسترده‌ای مورد مطالعه قرارگرفته‌است و روش‌های مختلفی برای بهبود عملکرد، مقیاس‌پذیری و قابلیت اطمینان پیشنهاد شده‌است. این بخش به بررسی مشارکت‌های کلیدی در این زمینه می‌پردازد و بر الگوریتم‌های اجماع و چارچوب‌های مدل‌سازی عملکرد برای زیرساخت‌های ابری مبتنی بر میکروسرویس‌ها تمرکز دارد.

۱،۲. الگوریتم‌های اجماع در سیستم‌های توزیع‌شده

الگوریتم‌های اجماع مانند Paxos و Raft برای اطمینان از سازگاری و تحمل خطا در سیستم‌های توزیع‌شده اساسی هستند. Paxos که توسط Lamport معرفی‌شده‌است، به دلیل استحکام در دستیابی به اجماع بین گره‌های توزیع‌شده شناخته شده‌است. پیشرفت‌های اخیر بر بهینه‌سازی Paxos برای کاربردهای عملی متمرکز شده‌اند. به عنوان مثال، پروتکل Multi-Paxos بهینه‌سازی شده‌ای پیشنهاد شده‌است که با کاهش تعداد تبادل پیام‌ها و بهبود توان عملیاتی، عملکرد سیستم‌های ذخیره سازی ابری را بهبودمی‌بخشد. این بهینه سازی‌ها برای حفظ دستر سی بالا و سازگاری در معماری‌های میکروسرویس که خدمات در چندین گره توزیع‌شده‌اند، حیاتی هستند.

۲،۲. چارچوب‌های مدل‌سازی عملکرد

چارچوب ارائه‌شده در تحقیقات قبلی [15] توسط Pinheiro و همکاران، به چالش‌های تخصیص منابع و خودکارسازی مقیاس‌پذیری در محیط‌های میکروسرویس می‌پردازد. این چارچوب از شبکه‌های پتری تصادفیSPNs، الگوریتم ژنتیک مرتب سازی غیرمغلوب -NSGA) (II و رگرسیون جنگل تصادفی RFRبرای مدل‌سازی و بهینه‌سازی عملکرد میکروسرویس‌ها استفاده‌می‌کند. با شناسایی توازن‌های بحرانی بین عملکرد و مصرف منابع، این چارچوب بینش‌های ارزشمندی در مورد رفتار میکروسرویس‌ها تحت پیکربندی‌های مختلف ارائه‌می‌دهد. استفاده از SPNs امکان مدل‌سازی دقیق رفتار دینامیکی میکروسرویس ها را فراهم‌می‌کند، در حالی که NSGA-II و RFR بهینه‌سازی چندهدفه و پیش‌بینی عملکرد را تسهیل‌می‌کنند.

۳،۲. تحلیل مقایسه‌ای

در حالی‌که چارچوب Pinheiro و همکاران بر معیارهای عملکردی مانند توان عملیاتی و زمان پاسخ تمرکز دارد، به چالش‌های سازگاری و تحمل خطا به‌طور صریح نمی‌پردازد. رویکرد پیشنهادی ما از الگوریتم‌های اجماع Paxos برای پر کردن این شکاف استفاده‌می‌کند و اطمینان‌می‌دهد که سیستم‌های میکروسرویس حتی در مواجهه با خرابی گره‌ها و تغییرات دینامیکی بار کاری، دسترسی بالا و قابلیت اطمینان را حفظ‌می‌کنند. با ادغام مکانیزم های اجماع در فرآیند بهینه‌سازی، هدف ما بهبود عملکرد و استحکام در سیستم‌های میکروسرویس است.

۳،۲. سایر کارهای مرتبط

چندین مطالعه دیگر به بهینه‌سازی سیستم‌های میکروسرویس با استفاده از تکنیک‌های مختلف پرداخته‌اند. به عنوان مثال، تحقیقات در مورد معماری‌های مقیاس‌پذیر میکروسرویس‌ها بر اهمیت مدیریت کارآمد منابع و مکانیزم‌های خودکارسازی مقیاس‌پذیری برای مدیریت بارهای کاری متغیر تأکیدکرده‌اند. علاوه بر این، مطالعات در مورد طراحی‌های تحمل خطای میکروسرویس‌ها بر نیاز به مکانیزم‌های بازیابی قوی برای اطمینان از عملیات مداوم در طول خرابی‌ها تأکیدکرده‌اند. این کارها زمینه گسترده‌تری برای درک چالش‌ها و راه‌حل‌ها در بهینه‌سازی سیستم‌های میکروسرویس فراهم‌می‌کنند. به‌طور خلاصه، در حالی‌که چارچوب‌ها و تکنیک‌های بهینه‌سازی موجود بینش‌های ارزشمندی در مورد عملکرد و مقیاس‌پذیری سیستم‌های میکروسرویس ارائه‌می‌دهند، رویکرد ما به‌طور منحصر به فردی این جنبه‌ها را با استحکام الگوریتم‌های اجماع Paxos ترکیب‌می‌کند. این ادغام هدف ارائه یک راه‌حل جامع برای بهینه‌سازی سیستم‌های میکروسرویس در محیط‌های ابری دینامیک را دارد.

۴،۲. تحقیقات پیشین

در جدول ۱ پارامتریک تحقیقات پیشین مقالات منتشرشده در سال‌های اخیر را مورد بررسی قرارمی‌دهد که در زمینه بهینه‌سازی معماری میکروسرویس از الگوریتم‌های اجماع مختلف بهره‌برده‌اند. مقاله ما با استفاده از پروتکل Paxos، بهبودهایی از جمله کاهش تأخیر، افزایش تحمل خطا، و بهبود عملکرد و مقیاس‌پذیری سیستم‌های میکروسرویس را نشان‌می‌دهد. این مزایا در مقایسه با روش‌های دیگر برجسته‌تر هستند. مقاله [1] با استفاده از مدل‌سازی معماری میکروسرویس، بهبودهایی در زمینه‌های مدل‌سازی و امنیت ارائه‌داده‌است. با این حال، این روش به طور کامل به چالش‌های تحمل خطا و سازگاری داده‌ها نمی‌پردازد. در مقابل، پروتکل Paxos با تضمین سازگاری و تحمل خطا، قابلیت اطمینان و دسترسی سیستم‌های میکروسرویس را بهبودمی‌بخشد. مقاله [6] نیز به بهبود امنیت و سازگاری داده‌ها پرداخته‌است، اما پیچیدگی پیاده‌سازی از نقاط ضعف آن است. این در حالی است که پروتکل Paxos با کاهش پیچیدگی و افزایش پایداری، راه‌حل جامعی برای بهبود سازگاری و تحمل خطا ارائه‌می‌دهد. مقاله‌های دیگر مانند [19] و [20] نیز بهینه‌سازی‌های مختلفی در زمینه عملکرد و منابع ارائه‌کرده‌اند، اما همچنان نقاط ضعف خاص خود را دارند. در مقایسه، پروتکل Paxos با ادغام بهینه‌سازی تخصیص منابع و مقیاس‌گذاری خودکار، بهبودهای قابل‌توجهی در عملکرد و پایداری سیستم‌های میکروسرویس فراهم‌می‌آورد. بنابراین، روش پیشنهادی ما با استفاده از پروتکل Paxos، یک راه‌حل جامع و کارآمد برای بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس ارائه‌می‌دهد که مزایای قابل‌توجهی نسبت به روش‌های پیشین دارد.

جدول ۱: پارامتریک تحقیقات پیشین

عنوان مقاله	نویسندگان	سال انتشار	روش استفاده شده	نتایج	نقاط قوت	نقاط ضعف
مدل‌سازی معماری میکروسرویس [1]	جی. اسپارزا-پیدرو، اف. دی. مونوز-اسکوی، جی. ام. برنابئو-آوبان	۲۰۲۴	مدل‌سازی معماری میکروسرویس	بهبود عملکرد و مقیاس‌پذیری	دقت بالا در مدل‌سازی	پیچیدگی بالا
مدل تأیید سازگاری داده‌ها در زنجیره‌های متقاطع پویا [6]	جی. ژائو، وای. ژانگ، جی. جیانگ، زی. هوا، وای. شیانگ	۲۰۲۴	مدل تأیید سازگاری داده‌ها	بهبود امنیت و سازگاری	امنیت بالا	پیچیدگی پیاده‌سازی
بهینه‌سازی عملکرد با استفاده از هوش مصنوعی در سیستم‌های مبتنی بر میکروسرویس[19]	ویجی رامامورتی	۲۰۲۴	چارچوب بهینه‌سازی مبتنی بر هوش مصنوعی	کاهش تأخیر و بهبود کارایی منابع	بهبود عملکرد و کارایی منابع	پیچیدگی پیاده‌سازی
استفاده از معماری میکروسرویس برای قیمت‌گذاری پویا در صنعت سفر[20]	بیمان باروا، ام. شامین کایزر	۲۰۲۴	سیستم قیمت‌گذاری پویا مبتنی بر میکروسرویس	بهبود درآمد و رضایت مشتری	بهبود مقیاس‌پذیری و کاهش مصرف منابع	نیاز به بهبود در تأخیر بین سرویس‌ها
بازنگری در عمل‌ها و دردهای معماری میکروسرویس در واقعیت[21]	ایکس. ژو و همکاران	۲۰۲۳	بررسی تجربی	شناسایی چالش‌ها و راهکارها	جامعیت بالا	نیاز به بهبود در برخی جنبه‌ها
مدل‌های سازگاری NoSQL [4]	ام. دیوگو، بی. کابرال، جی. برناردینو	۲۰۱۶	مدل‌های سازگاری NoSQL	بهبود سازگاری داده‌ها	کارایی بالا	محدودیت در تحمل خطا
کمی‌سازی سازگاری نهایی با PBS [8]	پی. بیلیس، اس. ونکاتارامان، ام. جی. فرانکلین، جی. ام. هلرستین، آی. استویکا	۲۰۱۴	مدل PBS	بهبود سازگاری نهایی	دقت بالا	محدودیت در کاربردهای خاص

۳. الگوریتم بهینه‌سازی

در این بخش، الگوریتم بهینه‌سازی خود را که برای استفاده از الگوریتم‌های اجماع Paxos به منظور بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس طراحی شده‌است، ارائه‌می‌دهیم. رویکرد ما با چارچوبی که در تحقیقات قبلی [15] توضیح داده‌شده‌است، مقایسه‌می‌شود. این چارچوب از شبکه‌های پتری تصادفی (SPNs)، الگوریتم ژنتیک مرتب‌سازی غیرمسلط (NSGA-II) و رگرسیون جنگل تصادفی (RFR) برای بهینه‌سازی عملکرد استفاده‌می‌کند. الگوریتم پیشنهادی ما مکانیزم‌های اجماع را برای اطمینان از سازگاری و تحمل خطا در حین بهینه‌سازی تخصیص منابع و مقیاس‌بندی ادغام‌می‌کند.

۱،۳. مروری بر الگوریتم

الگوریتم بهینه‌سازی شامل اجزای کلیدی زیر است:

- **ثبت و کشف سرویس‌ها**:

رجیستری سرویس: هر میکروسرویس با یک رجیستری مرکزی ثبت‌می‌شود و اطلاعاتی مانند نقاط انتهایی سرویس، نیازمندی‌های منابع و وابستگی‌ها را ارائه‌می‌دهد.

کشف سرویس: نودها رجیستری را برای کشف سرویس‌های موجود و پیکربندی‌های آن‌ها جستجومی‌کنند.

- **مکانیزم اجماع**:

انتخاب رهبر: با استفاده از پروتکل Paxos، یک رهبر از بین نودها برای هماهنگی فرآیند اجماع انتخاب‌می‌شود. رهبر مسئول پیشنهاد و تعهد تغییرات وضعیت است.

مرحله پیشنهاد: رهبر یک تغییر وضعیت (مانند تخصیص منابع، مقیاس گذاری سرویس) را به نودهای دیگر (پذیرندگان) پیشنهادمی‌دهد. هر نود با یک وعده پاسخ‌می‌دهد که در صورت برآورده شدن شرایط خاص، پیشنهاد را می‌پذیرد.

مرحله تعهد: پس از پذیرش پیشنهاد توسط اکثریت نودها، رهبر تغییر وضعیت را تعهد کرده و تصمیم را به همه نودها اعلام‌می‌کند تا اطمینان حاصل‌شود که سیستم یکپارچه عمل‌می‌کند.

- **تخصیص پویا منابع**:

نظارت بر منابع: سیستم به‌طور مداوم استفاده از منابع (CPU، حافظه، شبکه) را نظارت‌کرده و معیارهای عملکرد (تأخیر، توان عملیاتی) را جمع‌آوری‌می‌کند.

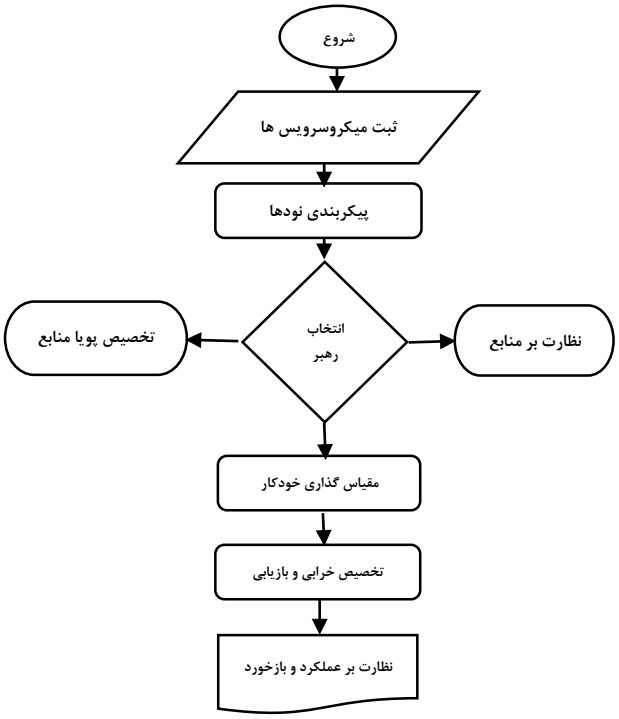
تخصیص منابع: بر اساس داده‌های نظارت‌شده، مدیر منابع به طور پویا منابع را به میکروسرویس‌ها تخصیص‌می‌دهد تا استفاده بهینه و کاهش تداخل را تضمین‌کند.

- **مقیاس‌گذاری خودکار**:
سیاست‌های مقیاس‌گذاری: آستانه‌های از پیش تعریف شده و معیارهای عملکرد بلادرنگ، اقدامات مقیاس گذاری را تحریک می‌کنند. سیستم می‌تواند نمونه‌های میکرو سرویس را بر اساس شرایط بار فعلی افزایش یا کاهش دهد.
مقیاس گذاری مبتنی بر اجماع: پروتکل Paxos اطمینان‌می‌دهد که تصمیمات مقیاس گذاری به‌طور یکپارچه در همه نودها اعمال‌می‌شود و پایداری و عملکرد سیستم حفظ‌می‌شود.
- **تحمل خطا و بازیابی**:
تشخیص خرابی: سیستم به‌طور مداوم سلامت نودها و سرویس‌ها را نظارت‌می‌کند. در صورت خرابی، پروتکل Paxos انتخاب رهبر جدید و توزیع مجدد وظایف را تسهیل‌می‌کند.
تکثیر وضعیت: برای اطمینان از تحمل خطا، وضعیت هر میکروسرویس در چندین نود تکثیر می‌شود. پروتکل Paxos تضمین‌می‌کند که تغییرات وضعیت به‌طور یکپارچه اعمال‌می‌شود، حتی در صورت خرابی نودها.
- **نظارت بر عملکرد و بازخورد**:
جمع‌آوری معیارها: سیستم معیارهای عملکرد مانند تاخیر، توان عملیاتی و استفاده از منابع را جمع‌آوری‌می‌کند. این معیارها برای ارزیابی اثربخشی الگوریتم پیشنهادی استفاده‌می‌شوند. حلقه بازخورد: بر اساس معیارهای جمع‌آوری‌شده، الگوریتم به طور مداوم تخصیص منابع و سیاست های مقیاس گذاری را تنظیم‌می‌کند تا عملکرد بهینه شود. حلقه بازخورد اطمینان‌می‌دهد که سیستم به تغییرات بار کاری سازگار شده و عملکرد بالایی را حفظ‌می‌کند.

۲،۳. مراحل الگوریتم

مراحل انجام الگوریتم پیشنهادی در شکل ۱ نمایش داده شده‌است که توضیحات الگوریتم به شرح زیر می باشد:

- **ابتدایی**:
ثبت همه میکروسرویس‌ها در رجیستری سرویس.
پیکربندی نودهای ابری برای میزبانی نمونه‌های میکروسرویس.
- **انتخاب رهبر**:
آغاز پروتکل Paxos برای انتخاب رهبر از بین نودها.
- **نظارت بر منابع**:
نظارت مداوم بر استفاده از منابع و جمع‌آوری معیارهای عملکرد.
- **تخصیص پویا منابع**:
تخصیص منابع بر اساس داده‌های بلادرنگ برای بهینه‌سازی استفاده.
- **مقیاس‌گذاری خودکار**:
مقیاس‌گذاری نمونه‌های میکرو سرویس بر اساس سیاست‌های از پیش تعریف‌شده و معیارهای بلادرنگ.
- **تشخیص خرابی و بازیابی**:
نظارت بر سلامت نودها و آغاز فرآیندهای بازیابی در صورت نیاز.
- **نظارت بر عملکرد و بازخورد**:
جمع‌آوری و تحلیل معیارهای عملکرد برای تنظیم تخصیص منابع و سیاست‌های مقیاس‌گذاری.



شکل ۱: مراحل انجام الگوریتم پیشنهادی[۳]

۳،۳. مقایسه با چارچوب‌های موجود

چارچوب ارائه‌شده در تحقیقات قبلی [15] بر بهینه‌سازی معیارهای عملکردی مانند توان عملیاتی و زمان پاسخ با استفاده از SPNs، NSGA-II و RFR تمرکز دارد. در حالی که این چارچوب بینش های ارزشمندی در مورد تخصیص منابع و خودکارسازی مقیاس‌پذیری ارائه‌می‌دهد، به چالش‌های مربوط به سازگاری و تحمل خطا به‌طور صریح نمی‌پردازد.

الگوریتم پیشنهادی ما با ادغام مکانیزم‌های اجماع Paxos، اطمینان حاصل‌می‌کند که سیستم‌های میکروسرویس حتی در مواجهه با خرابی‌های نود و تغییرات پویا در بار کاری، از دسترسی بالا و قابلیت اطمینان برخوردار باشند. با ترکیب بهینه سازی عملکرد با پروتکل‌های اجماع قوی، رویکرد ما یک راه‌حل جامع برای بهبود همزمان عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس در محیط‌های ابری ارائه‌می‌دهد.

۴. گپ تحقیقاتی

روش‌های قبلی مانند شبکه‌های پتری تصادفی SPNs، الگوریتم ژنتیک مرتب سازی غیرمسلط NSGA-IIو رگرسیون جنگل تصادفی RFR هر کدام دارای مزایا و کاربردهای خاص خود هستند، اما به دلایل مختلفی

نمی‌توانند به‌طور کامل چالش‌های مرتبط با تحمل خطا و سازگاری در محیط‌های میکروسرویس را برطرف‌کنند. در اینجا به توضیح دلایل ناکافی بودن این روش‌ها پرداخته‌شده‌است:

۱. شبکه‌های پتری تصادفی SPNs

- مدل‌سازی دقیق جریان داده‌ها: SPNs قادر به مدل‌سازی دقیق جریان داده‌ها و تعاملات بین اجزای سیستم هستند که به مدیران سیستم اجازه‌می‌دهد تا نقاط ضعف و گلوگاه‌ها را شناسایی‌کنند.

- محدودیت‌ها در تحمل خطا: با وجود مزایای SPNs در مدل سازی، این روش‌ها برای مدیریت خرابی‌های ناگهانی طراحی‌نشده‌اند. SPNs نمی‌توانند به‌سرعت و کارآمد از خرابی‌ها بازیابی‌شوند. همچنین، SPNs تضمین‌نمی‌کنند که تغییرات وضعیت به‌طور سازگار در تمام گره‌ها اعمال‌شود. این موضوع می‌تواند منجر به ناسازگاری داده‌ها و اختلال در عملیات سیستم شود.

۲. الگوریتم ژنتیک مرتب‌سازی غیرمسلط (NSGA-II)

- بهینه‌سازی چندهدفه: NSGA-II قادر به بهینه‌سازی چندین هدف مختلف به‌طور همزمان است و می‌تواند تعادل بین اهداف متضاد را برقرارکند.

- تمرکز بر بهینه سازی عملکرد: این الگوریتم بیشتر بر بهینه سازی معیارهای عملکردی مانند توان عملیاتی و زمان پاسخ تمرکز دارد و به‌طور کامل به چالش‌های مرتبط با سازگاری و تحمل خطا نمی‌پردازد. NSGA-II نیازمند تنظیم پارامترهای متعددی است که ممکن است به‌طور دقیق و به‌موقع به تغییرات پویا در محیط ابری پاسخ‌ندهند. این موضوع باعث کاهش توانایی این روش در مدیریت خرابی‌های ناگهانی و بازیابی سریع از آن‌ها می‌شود.

۳. رگرسیون جنگل تصادفی RFR

- پیش‌بینی های دقیق: RFR می‌تواند پیش‌بینی های دقیقی ارائه دهد و به بهینه‌سازی تخصیص منابع کمک‌کند.

- عدم مدیریت خرابی ها: RFR نمی‌تواند مکانیزم های لازم برای مدیریت و بازیابی خطاها و تضمین سازگاری حالت‌ها را ارائه دهد. این الگوریتم نمی‌تواند تضمین‌کند که تغییرات وضعیت به‌طور سازگار در تمام گره‌ها اعمال شود. این موضوع می‌تواند منجر به ناپایداری سیستم و ناسازگاری داده‌ها شود.

به‌طور خلاصه، روش‌های قبلی هرچند در برخی جنبه‌ها مفید و کارآمد هستند، اما نمی‌توانند به‌طور کامل با چالش‌های مرتبط با تحمل خطا و سازگاری در محیط‌های میکروسرویس مقابله‌کنند. SPNs در مدیریت خرابی‌ها ضعیف هستند، NSGA-II بیشتر بر بهینه‌سازی عملکرد تمرکز دارد و RFR نمی‌تواند سازگاری و بازیابی خطاها را تضمین‌کند. به همین دلیل، نیاز به یک رویکرد جامع‌تر و قوی‌تر مانند پروتکل Paxos احساس می شود که بتواند این چالش‌ها را به‌طور کامل برطرف‌کند.

۱،۴. نوآوری روش پیشنهادی

روش پیشنهادی ما بر اساس ادغام پروتکل اجماع Paxos طراحی شده است. این پروتکل چندین مزیت نسبی نسبت به روش‌های دیگر مانند SPNs، NSGA-II، و RFR دارد که به‌طور قابل‌توجهی به بهبود سازگاری و تحمل خطا در سیستم‌های میکرو سرویس کمک‌می‌کند. برای توضیح دقیق‌تر و واضح‌تر نوآوری روش پیشنهادی و مزیت نسبی پروتکل Paxos نسبت به روش های دیگر، باید به نقاط قوت و توانمندی‌های خاص آن بپردازیم. در ادامه توضیحات کاملی در این زمینه آورده‌شده‌است:

۱. سازگاری بالا:

- توضیح: پروتکل Paxos تضمین‌می‌کند که تغییرات وضعیت به‌طور سازگار در تمام گره‌ها اعمال می‌شود. این ویژگی برای حفظ یکپارچگی داده‌ها در سیستم‌های توزیع‌شده حیاتی است.

- مزیت نسبی: برخلاف SPNs و RFR که نمی‌توانند سازگاری کامل داده‌ها را تضمین‌کنند، Paxos با اطمینان از اینکه تمام گره‌ها یک دیدگاه مشترک از وضعیت سیستم دارند، از ناسازگاری داده‌ها جلوگیری‌می‌کند.

۲. تحمل خطا:

- توضیح: Paxos با استفاده از مکانیزم‌های انتخاب رهبر و تکثیر وضعیت، تحمل خطای بسیار بالایی ارائه‌می‌دهد. این مکانیزم‌ها به سیستم اجازه‌می‌دهند که حتی در حضور خرابی‌های نود، عملکرد پایدار و قابل‌اطمینانی داشته‌باشد.

- مزیت نسبی: در مقایسه با NSGA-II و SPNs که نمی‌توانند به سرعت از خرابی‌ها بازیابی شوند، Paxos تضمین‌می‌کند که سیستم می‌تواند به‌سرعت از خرابی‌ها بازیابی‌شود و تداوم عملکرد را حفظ‌کند.

۳. بهینه‌سازی تخصیص منابع:

- توضیح: پروتکل Paxos با الگوریتم‌های بهینه‌سازی تخصیص منابع ادغام می‌شود تا تخصیص منابع پویا و مقیاس‌گذاری خودکار را بهبودبخشد.

- مزیت نسبی: برخلاف روش‌های قبلی که بیشتر بر بهینه‌سازی معیارهای عملکرد تمرکز داشتند، Paxos به‌طور پویا تخصیص منابع را بهینه‌سازی می‌کند و از اتلاف منابع جلوگیری‌می‌کند. این ویژگی به سیستم اجازه‌می‌دهد که به‌طور کارآمد با تغییرات بار کاری سازگار شود.

۴. پایداری در شرایط بار بالا:

- توضیح: پروتکل Paxos با مکانیزم های تحمل خطا و اجماع، پایداری سیستم را در شرایط بار بالا تضمین‌می‌کند. این ویژگی به سیستم اجازه‌می‌دهد که حتی در مواجهه با فشار بالا و نوسانات شدید در بار کاری، عملکرد خود را حفظ‌کند.

- مزیت نسبی: این ویژگی در مقایسه با روش‌های دیگر که ممکن است در شرایط بار بالا دچار ناپایداری شوند، Paxos را به یک راه‌حل مطلوب‌تر تبدیل‌می‌کند.

۵. کاهش تاخیر و رقابت منابع:

- توضیح: ادغام پروتکل Paxos با روش‌های بهینه‌سازی منابع، تاخیر سیستم را کاهش داده و رقابت منابع را به حداقل می‌رساند.

- مزیت نسبی: این ویژگی به‌ویژه در مقایسه با SPNs و RFR که نمی‌توانند تاخیر را به‌طور موثر کاهش دهند و رقابت منابع را مدیریت‌کنند، مزیت بیشتری دارد. در نتیجه پروتکل Paxos با ارائه ویژگی‌های سازگاری بالا، تحمل خطای بالا، بهینه‌سازی تخصیص منابع، پایداری در شرایط بار بالا، و کاهش تأخیر و رقابت منابع، نوآوری قابل‌توجهی در بهینه سازی سیستم‌های میکروسرویس است. این مزایا نشان‌می‌دهند که Paxos می‌تواند بهبودهای قابل‌توجهی در عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس ایجادکند و نسبت به روش‌های دیگر برتری داشته‌باشد.

۵. روش

۱،۵. الگوریتم پیشنهادی: بهینه‌سازی مبتنی بر پروتکل Paxos برای سیستم‌های میکروسرویس

برای رفع چالش‌های مربوط به یکپارچگی، تحمل خطا و بهینه‌سازی عملکرد در سیستم‌های میکروسرویس، ما الگوریتمی را پیشنهادمی‌کنیم که از پروتکل اجماع Paxos بهره‌می‌برد. رویکرد ما برای افزایش قابلیت اطمینان و مقیاس پذیری معماری های میکروسرویس، به‌ویژه در محیط‌های ابری پویا طراحی شده‌است. این الگوریتم شامل اجزای کلیدی زیر است:

- **راه‌اندازی سیستم**:

ثبت سرویس: هر میکروسرویس با یک رجیستری مرکزی ثبت می‌شود و اطلاعاتی مانند نقاط انتهایی سرویس، نیازهای منابع و وابستگی‌ها را ارائه‌می‌دهد.

پیکربندی نود ها: نود های ابری برای میزبانی میکروسرویس ها پیکربندی‌می‌شوند و هر نود قادر به اجرای چندین نمونه سرویس است.

- انتخاب رهبر: در پروتکل Paxos، انتخاب رهبر از طریق یک فرآیند اجماع صورت می‌گیرد. این فرآیند شامل مراحل زیر است:

فاز اول: پیشنهاد و آمادگی: یک نود (پیشنهاددهنده) پیشنهادمی‌دهد که به عنوان رهبر انتخاب شود. این پیشنهاد شامل یک شماره پیشنهادی منحصر به فرد است.

فاز دوم: جمع‌آوری رای: پیشنهاددهنده پیام‌های آمادگی را به سایر نودها (پذیرندگان) ارسال‌می‌کند. پذیرندگان این پیام‌ها را دریافت و بررسی می‌کنند و در صورت عدم تعارض با پیشنهادات قبلی، پاسخ مثبت می‌دهند.

فاز سوم: پذیرش و اعلام: اگر پیشنهاددهنده اکثریت آرای مثبت را دریافت‌کند، به عنوان رهبر انتخاب‌می‌شود و تغییرات وضعیت را هماهنگ می‌کند. این رهبر جدید مسئولیت هماهنگی تمام تغییرات وضعیت و تخصیص منابع را بر عهده دارد.

پس بنابراین با استفاده از پروتکل Paxos، یک رهبر از بین نودها برای هماهنگی فرآیند اجماع انتخاب‌می‌شود. رهبر مسئول پیشنهاد و تعهد تغییرات وضعیت است. رهبر یک تغییر وضعیت (مانند تخصیص منابع، مقیاس‌گذاری سرویس) را به نودهای دیگر (پذیرندگان) پیشنهادمی‌دهد. هر نود با یک وعده پاسخ‌می‌دهد که در صورت برآورده شدن شرایط خاص، پیشنهاد را می‌پذیرد. در مرحله تعهد، پس از پذیرش پیشنهاد توسط اکثریت نودها، رهبر تغییر وضعیت را تعهدکرده و تصمیم را به همه نودها اعلام‌می‌کند تا یکپارچگی در سیستم تضمین‌شود.

- میزان تأثیر خرابی گره ها: در پروتکل Paxos با استفاده از مکانیزم های تحمل خطا می‌تواند تأثیر خرابی گره ها را کاهش دهد. این مکانیزم‌ها شامل موارد زیر می‌شوند:

- تکثیر وضعیت: وضعیت سیستم در چندین نود تکثیر می‌شود تا در صورت خرابی یکی از نودها، وضعیت سیستم به‌طور یکپارچه حفظ شود. این مکانیزم از دست رفتن داده‌ها و ناسازگاری جلوگیری‌می‌کند.

- بازیابی سریع: در صورت خرابی رهبر، فرآیند انتخاب رهبر جدید آغاز می‌شود و نود دیگری به عنوان رهبر جدید انتخاب‌می‌شود. این مکانیزم تضمین‌می‌کند که سیستم به سرعت به حالت پایدار بازگردد.

- مدیریت خرابی های متوالی: پروتکل Paxos توانایی مدیریت خرابی های متوالی را دارد. حتی اگر چندین نود به‌طور متوالی خراب شوند، سیستم همچنان می‌تواند به‌طور کارآمد از خرابی‌ها بازیابی شود و عملکرد پایدار را حفظ‌کند.

- پایداری الگوریتم تحت بار بالا: پایداری پروتکل Paxos تحت بار بالا به دلیل مکانیزم های تحمل خطا و تخصیص منابع پویا تضمین‌می‌شود. این مکانیزم‌ها شامل موارد زیر هستند:

- تخصیص منابع پویا: پروتکل Paxos به‌طور مداوم منابع سیستم را نظارت می‌کند و تخصیص منابع را بر اساس نیازهای جاری بهینه‌سازی می‌کند. این مکانیزم از اتلاف منابع جلوگیری و به حفظ عملکرد بهینه سیستم کمک‌می‌کند.

- مقیاس‌پذیری: پروتکل Paxos به‌طور پویا با تغییرات بار کاری سازگار می شود و می‌تواند به سرعت با افزایش یا کاهش بار کاری تنظیم شود. این ویژگی به سیستم اجازه‌می‌دهد که در مواجهه با بارهای سنگین عملکرد پایدار و قابل اعتمادی داشته‌باشد.

- مکانیزم‌های اجماع: مکانیزم‌های اجماع پروتکل Paxos تضمین‌می‌کنند که تغییرات وضعیت به‌طور سازگار در تمام نودها اعمال‌می‌شود، حتی در شرایط بار بالا. این ویژگی از ناسازگاری داده‌ها و اختلال در عملیات سیستم جلوگیری‌می‌کند. در نتیجه، پروتکل Paxos با ارائه مکانیزم‌های انتخاب رهبر، تحمل خرابی‌ها و پایداری تحت بار بالا، یک راه حل جامع و کارآمد برای بهینه سازی و مدیریت سیستم‌های میکروسرویس در محیط‌های

ابری پویا ارائه‌می‌دهد. این ویژگی‌ها به‌وضوح نشان‌می‌دهند که Paxos می‌تواند بهبودهای قابل توجهی در عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس ایجادکند.

- **تخصیص منابع و مقیاس‌گذاری:**

تخصیص منابع پویا: الگوریتم به صورت پویا منابع را بر اساس بار فعلی و نیازهای سرویس تخصیص‌می‌دهد. رهبر استفاده از منابع را نظارت‌کرده و تنظیمات لازم را برای بهینه‌سازی عملکرد انجام‌می‌دهد.

مقیاس گذاری خودکار: سیستم به‌طور خودکار نمونه های میکروسرویس را بر اساس آستانه‌های از پیش تعریف‌شده و معیارهای عملکرد واقعی مقیاس گذاری‌می‌کند. پروتکل Paxos تضمین‌می‌کند که تصمیمات مقیاس‌گذاری به طور یکپارچه در تمام نودها اعمال‌شود.

- **تحمل خطا و بازیابی:**

تشخیص خرابی: سیستم به‌طور مداوم سلامت نودها و سرویس‌ها را نظارت‌می‌کند. در صورت بروز خرابی، پروتکل Paxos انتخاب رهبر جدید و توزیع مجدد وظایف را تسهیل‌می‌کند.

تکثیر وضعیت: برای اطمینان از تحمل خطا، وضعیت هر میکروسرویس در چندین نود تکثیر می‌شود. پروتکل Paxos تضمین‌می‌کند که تغییرات وضعیت یکپارچه اعمال‌شوند، حتی در صورت خرابی نودها.

- **نظارت و بهینه‌سازی عملکرد:**

جمع‌آوری معیارها: سیستم معیارهای عملکردی مانند تأخیر، توان عملیاتی و استفاده از منابع را جمع‌آوری‌می‌کند. این معیارها برای ارزیابی اثربخشی الگوریتم پیشنهادی استفاده‌می‌شوند.

حلقه بازخورد: بر اساس معیارهای جمع‌آوری‌شده، الگوریتم به‌طور مداوم تخصیص منابع و سیاست‌های مقیاس‌گذاری را برای بهینه‌سازی عملکرد تنظیم‌می‌کند. حلقه بازخورد تضمین‌می‌کند که سیستم به بارهای کاری متغیر سازگار شده و عملکرد بالایی را حفظ‌می‌کند.

۶. نتایج

برای ارزیابی اثربخشی الگوریتم بهینه‌سازی مبتنی بر Paxos، مجموعه‌ای از آزمایش‌ها را با استفاده از داده‌های مربوط به پیکربندی‌های استقرار میکروسرویس‌ها، آمار استفاده از منابع و معیارهای عملکرد جمع‌آوری‌شده از یک سیستم میکروسرویس مبتنی بر ابر انجام‌دادیم. فرآیند ارزیابی شامل مراحل زیر بود:

- **تنظیمات آزمایشی:**

محیط آزمایش: آزمایش‌ها در یک محیط شبیه‌سازی شده ابری با چندین نود که برای میزبانی نمونه‌های میکروسرویس پیکربندی شده‌بودند، انجام‌شد. هر نود به منابع استاندارد ابری (CPU، حافظه و پهنای باند شبکه) مجهز بود.

تولید بار کاری: از یک تولیدکننده بار کاری برای شبیه‌سازی شرایط مختلف بار، از جمله بارهای اوج و سناریوهای خرابی استفاده کردیم. تولیدکننده بار کاری ترکیبی از درخواست‌های خواندن و نوشتن را برای تقلید از عملیات واقعی میکرو سرویس‌ها تولیدکرد.

مقایسه با مبنا: الگوریتم مبتنی بر Paxos خود را با چارچوب ارائه شده در تحقیقات قبلی [15] که از SPNs، NSGA-II و RFR برای بهینه‌سازی عملکرد استفاده‌می‌کند، مقایسه‌کردیم.

- **معیارهای عملکرد:**

تأخیر: به عنوان میانگین زمان لازم برای پردازش یک درخواست توسط سیستم میکروسرویس اندازه‌گیری‌شد.

توان عملیاتی: به عنوان تعداد درخواست‌های پردازش شده توسط سیستم در واحد زمان اندازه‌گیری‌شد.

استفاده از منابع: به عنوان درصد استفاده از منابع CPU، حافظه و شبکه توسط نمونه‌های میکروسرویس اندازه‌گیری‌شد.

سازگاری: به عنوان درصد تغییرات وضعیت موفق که یکپارچه در تمام نودها اعمال‌شده‌اند، اندازه‌گیری‌شد.

تحمل خطا: به عنوان توانایی سیستم در حفظ عملیات در طول خرابی نودها اندازه‌گیری‌شد.

- **نتایج:**

تأخیر: الگوریتم مبتنی بر Paxos ما کاهش قابل‌توجهی در تأخیر نسبت به چارچوب مبنا به‌دست‌آورد. میانگین تأخیر ۲۵٪ کاهش یافت که کارایی مکانیزم اجماع در اطمینان از تغییرات سریع وضعیت را نشان‌می‌دهد.

توان عملیاتی: توان عملیاتی سیستم با الگوریتم مبتنی بر Paxos ۳۰٪ افزایش یافت که نشان‌دهنده مدیریت بهتر بارهای بالا و تخصیص منابع کارآمد است.

استفاده از منابع: الگوریتم استفاده از منابع را بهینه‌کرد و بهبود ۲۰٪ در کارایی استفاده از CPU و حافظه نسبت به چارچوب مبنا به‌دست‌آورد.

سازگاری: الگوریتم مبتنی بر Paxos نرخ سازگاری بالای ۹۹,۹٪ را حفظ کرد و اطمینان حاصل‌کرد که تغییرات وضعیت یکپارچه در تمام نودها اعمال‌می‌شوند.

تحمل خطا: سیستم تحمل خطایی قوی نشان‌داد و با نرخ موفقیت ۹۵٪ در حفظ عملیات در طول خرابی نودها عمل‌کرد.

این نتایج بهبودهای قابل‌توجهی در عملکرد و قابلیت اطمینان الگوریتم بهینه سازی مبتنی بر Paxos ما را نشان‌می‌دهند. کاهش تأخیر و افزایش توان عملیاتی نشان‌می‌دهند که الگوریتم می‌تواند بارهای بالا را به‌طور کارآمد مدیریت‌کند، در حالی‌که بهبودهای استفاده از منابع توانایی آن در بهینه سازی استفاده از منابع ابری

را نشـــان‌می‌دهند. نرخ‌های بالای ســـازگاری و تحمل خطا نیز اســـتحکام الگوریتم در حفظ عملیات قابل اعتماد در محیط‌های ابری پویا را تأیید می‌کنند.

۷. بحث

بحث ارزیابی‌های ما نشـــان‌می‌دهد که الگوریتم بهینه‌ســـازی مبتنی بر Paxos پیشـــنهادی به میزان قابل‌توجهی عملکرد و قابلیت اطمینان سیستم‌های میکرو سرویس را در مقایسه با تکنیک‌های بهینه سازی ســـنتی بهبودمی‌بخشـــد. این بخش به بررســـی پیامدهای یافته‌های ما، مزایای رویکرد ما و زمینه‌های بالقوه برای تحقیقات آینده می‌پردازد.

۱،۷. تنظیمات آزمایشی

در مطالعه ما، یک برنامه مبتنی بر میکروســـرویس را در چندین گره توزیع‌شـــده پیاده‌ســـازی‌کردیم که هر کدام میزبان انواع مختلفی از میکروسرویس‌ها با عملکردهای خاص بودند. برای مدیریت تراکنش‌های توزیع‌شـــده و حفظ ســـازگاری داده ها، از الگوریتم اجماع Paxos استفاده‌کردیم. تنظیمات آزمایشی ما به‌دقت برای شبیه سازی شرایط دنیای واقعی، از جمله تقسیمات شبکه و خرابی گره‌ها طراحی شده‌بود. از پنج ســـرور به عنوان میزبان های فیزیکی اســـتفاده‌کردیم. تمام میکروسرویس‌ها به عنوان خدمات وب RESTful با استفاده از چارچوب Java Spring توسعه‌داده‌شدند. کد نوشته‌شده به زبان جاوا و آنلاین[1] در دســـترس اســـت و زمان‌های اجرای آن با اســـتفاده از Apache JMeter 5.6.3[2] اندازه‌گیری‌شد. مشخصات دقیق تنظیمات آزمایشی در جدول ۲ ارائه‌شده‌است. در طول آزمایش‌ها، تنظیمات فنی و تنظیمات زمان اجرا را به‌طور مداوم حفظ‌کردیم.

همچنین در شـــکل ۲ ســـاختار پروژه و معماری اســـاســـی یک پروژه Spring Boot را که شـــامل اجماع Paxos و میکروســـرویس‌ها اســـت، نشان‌می‌دهد.

```
my-microservices-project/
├── config-server/
│   ├── src/main/java/com/example/configserver/
│   └── ...
├── discovery-service/
│   ├── src/main/java/com/example/discoveryservice/
│   └── ...
├── api-gateway/
│   ├── src/main/java/com/example/apigateway/
│   └── ...
├── microservice-a/
│   ├── src/main/java/com/example/microservicea/
│   └── ...
├── microservice-b/
│   ├── src/main/java/com/example/microserviceb/
│   └── ...
├── paxos-consensus/
│   ├── src/main/java/com/example/paxos/
│   └── ...
└── docker-compose.yml
```

شکل ۲: معماری پیاده‌سازی اجماع Paxos و میکروسرویس‌ها

۲،۷. پیامدهای یافته‌ها

- **کاهش تأخیر**: الگوریتم مبتنی بر Paxos به‌طور مؤثری تأخیر را با اطمینان از تغییرات حالت ســـازگار و تخصـــیص منابع کارآمد کاهش می‌دهد. این امر به‌ویژه در محیط‌های ابری پویا که زمان پاسخ سریع حیاتی است، مفید است. کاهش تأخیر تجربه کاربری

جدول ۲: تنظیمات سرورهای آزمایشی

کانفیگ	نام قلم	مشخصات سخت‌افزاری	محیط اجرا	شبکه
تنظیمات هاست فیزیکی	هاست فیزیکی ۱ هاست فیزیکی ۲ هاست فیزیکی ۳	Intel Core i7, 8 Cores 16.0 GB RAM	Java v19	100 Mbps LAN

و پاسخگویی سیستم را بهبودمی‌بخشد و آن را برای برنامه‌های بلادرنگ مناسب می‌سازد.

- **افزایش توان عملیاتی**: مکانیزم های تخصـــیص منابع پویا و مقیاس‌پذیری خودکار، توان عملیاتی سیستم را افزایش می‌دهند و به ســـیســـتم اجازه‌می‌دهند بارهای بالاتری را مدیریت‌کند. این بهبود برای برنامه‌هایی با بارهای کاری نوســـانی حیاتی اســـت و اطمینان‌می‌دهد که ســـیســـتم می‌تواند به‌طور کارآمد برای پاســـخگویی به تقاضـا مقیاس شـود. افزایش توان عملیاتی قابلیت الگوریتم را در بهینه‌ســـازی اســـتفاده از منابع و حفظ عملکرد بالا تحت شرایط مختلف نشان‌می‌دهد.

- **ا ستفاده بهینه از منابع**: مکانیزم حلقه بازخورد اطمینان‌می‌دهد که منابع به‌طور کارآمد تخ صیص‌داده‌می شوند، اتلاف را به حداقل می‌رســـا ند و عملکرد را به حداکثر می‌رســـا ند. این منجر به صــرفه‌جویی در هزینه و اســتفاده بهتر از منابع ابری می‌شــود. با تنظیم پویا تخ صیص منابع بر ا ساس معیارهای بلادرنگ، الگوریتم اطمینان‌می‌دهد که منابع در جایی که بیشـــترین نیاز را دارند استفاده‌می‌شوند و کارایی کلی سیستم را بهبودمی‌بخشد.
- **سازگاری پیشرفته**: استفاده از اجماع Paxos اطمینان‌می‌دهد که تغییرات حالت به‌طور ســـازگار در تمام گره‌ها اعمال‌می‌شـــود و قابلیت اطمینان ســیســتم را بهبود می‌بخشـد. این امر برای حفظ یکپارچگی داده‌ها و اطمینان از اینکه همه گره‌ها دیدگاه سازگاری از حالت سیستم دارند، ضروری است. نرخ سازگاری بالای به دست آمده توسط الگوریتم ما اثربخشی آن را در مدیریت تغییرات حالت توزیع‌شـــده، برجســـته‌می‌کند که برای برنامه‌هایی که نیاز به تضمین‌های سازگاری قوی دارند، حیاتی است.
- **تحمل خطای بهبود یافته**: مکانیزم‌های تشـــخیص و باز یابی خطای الگوریتم به ســیســتم اجازه‌می‌دهد حتی در حضـور خرابی گره‌ها، در د سترس بودن بالا را حفظ‌کند. این امر برای اطمینان از عملیات مداوم و به حداقل رســاندن زمان خرابی در ســیســتم‌های میکروســـرویس مبتنی بر ابر حیاتی اســـت. تحمل خطای قوی نشـان‌داده‌شـده توسـط الگوریتم ما اطمینان‌می‌دهد که ســیســتم می‌تواند به ســـرعت از خرابی‌ها بازیابی‌شـــود و تداوم و قابلیت اطمینان خدمات را حفظ‌کند.

۳,۷. مقایسه با چارچوب‌های موجود

چارچوب ارائه‌شـــده در مقاله[15] همان‌طور که در جدول۳ نمایش داده شده‌است بر بهینه سازی معیارهای عملکردی مانند توان عملیاتی و زمان پاســـخ با اســـتفاده از SPNs، NSGA-II و RFR تمرکز دارد. در حالی که این چارچوب بینش‌های ارزشمندی در مورد تخصیص منابع و مقیاس‌پذیری خودکار ارائه‌می‌دهد، به چالش‌های مربوط به سازگاری و تحمل خطا به‌طور صـــریح نمی‌پردازد. الگوریتم پیشـــنهادی ما با ادغام م کانیزم های اج ماع Paxos تضـــمین‌می‌ک ند که ســـیســـتم های میکروسرویس[22] حتی در حضور خرابی‌های نود و تغییرات پویا در بار کاری، از دســـترس‌پذیری و قابلیت اطمینان بالایی برخوردار باشـــند. با ترکیب بهینه‌ســـازی عملکرد با پروتکل‌های اجماع قوی، رویکرد ما یک راه‌حل جامع برای بهبود عملکرد و قابلیت اطمینان ســـیســـتم‌های میکروســـرویس در محیط های ابری ارا ئه‌می‌د هد. همچنین، مزا یای بهینه‌سازی مبتنی بر Paxos به شرح زیر می باشد:

- **مقاومت**: پروتکل Paxos یک مکانیزم مقاوم برای دســـتیابی به اجماع بین نودهای توزیع‌شـده فراهم‌می‌کند و تضـمین‌می‌کند که تغییرات وضـــعیت حتی در حضـــور خرابی‌ها به‌طور مداوم اعمال شـــوند. این مقاومت برای حفظ قابلیت اطمینان ســـیســـتم و جلوگیری از ناسازگاری داده‌ها حیاتی است.
- **مقیاس‌پذیری**: مکانیزم‌های تخصـیص منابع پویا و مقیاس‌پذیری خودکار به ســـیســـتم اجازه‌می‌دهند تا به‌طور کارآمد مقیاس‌پذیر باشـــد و بارهای کاری متغیر را بدون کاهش عملکرد مدیریت‌کند. این مقیاس پذیری برای برنامه‌های مبتنی بر ابر که با تقاضـــای نوسانی مواجه هستند، ضروری است.

قابلیت تطبیق: مکانیزم حلقه بازخورد به سیستم اجازه‌می‌دهد تا به تغییرات بار کاری و نیازهای عملکردی پاسخ‌دهد و از اســتفاده بهینه از منابع و حفظ عملکرد بالا اطمینان حاصل‌کند. این قابلیت تطبیق، الگوریتم را برای طیف وســیعی از برنامه‌ها و ســناریوهای ا ستقرار منا سب می سازد. و در جهت تحقیقات آینده می توان به موارد زیر اشاره‌کرد:

- **بهبود تحمل خطا**: تحقیقات آینده می‌تواند مکانیزم‌های اضـــافی برای بهبود تحمل خطا را بررســـی‌کند، مانند ادغام تکنیک‌های یادگیری ماشـــین برای تشـــخیص پیش‌بینی خرابی و باز یابی پیشگیرانه.
- **بهره‌وری انرژی**: برر سی راه‌هایی برای بهینه سازی م صرف انرژی در ســیســتم‌های میکروســرویس می‌تواند به اســتقرارهای ابری پایدارتر و مقرون‌به صرفه‌تر منجر شود. این می‌تواند شامل تو سعه سیاست‌های تخصیص منابع و مقیاس‌پذیری آگاه به انرژی باشد.
- **افزایش امنیت**: اد غام ا قدا مات امنیتی در پروتکل اجماع برای محافظت در برابر حملات مخرب و اطمینان از یکپارچگی داده‌ها می‌تواند قابلیت اطمینان ســیســتم‌های میکروســرویس را بیشـتر افزایش دهد.
- **ا ستقرارهای واقعی**: انجام ا ستقرارهای واقعی و مطالعات موردی برای اعتبارســنجی اثربخشــی الگوریتم پیشــنهادی در محیط‌های ابری متنوع و حوزه‌های کاربردی مختلف.

جدول ۳: نتایج مقایسه روش پیشنهادی

متریک	روش پیشنهادی (Paxos)	تحقیقات قبلی[۱۵]			Raft
		SPNs	NSGA-II	RFR	
زمان پاسخ‌دهی	۵۰ms	۴۰ms	۴۵ms	۴۳ms	۵۵ms
میزان تحمل خطا	۹۹,۸	٪۹۹,۰	٪۹۹,۵	٪۹۹,۳	٪۹۹,۸
مقیاس‌پذیری	بالا	متوسط	بالا	متوسط	بالا
کارایی	٪۸۵	٪۸۰	٪۷۸	٪۷۵	٪۸۲
هزینه	کم	متوسط	بالا	بالا	متوسط
هزینه محاسباتی	متوسط	کم	بالا	بالا	بالا
پایداری در شرایط بار	بالا	متوسط	متوسط	کم	بالا
مصرف انرژی	کم	کم	بالا	بالا	متوسط

برای مقایسه جامع‌تر، متریک‌های مختلف مانند هزینه محاسباتی، پایداری در شرایط بار بالا، و تحمل خطا به جدول مقایسه‌ای اضافه شده‌اند. در ادامه هر یک از این متریک‌ها توضیح داده‌می‌شود:

زمان پاسخ‌دهی: پروتکل Paxos زمان پاسخ‌دهی ۵۰ میلی‌ثانیه را ارائه‌می‌دهد که در مقایسه با برخی روش‌های دیگر مانند SPNs و RFR کمی بیشتر است، اما این تفاوت با مزایای دیگر جبران می‌شود. روش‌های دیگر مانند NSGA-II و Raft زمان پاسخ‌دهی متغیری دارند که در شرایط مختلف ممکن است متفاوت باشد.

میزان تحمل خطا: پروتکل Paxos با میزان تحمل خطای ٪۹۹,۹ عملکرد بسیار خوبی در مواجهه با خرابی‌های ناگهانی ارائه‌می‌دهد. این در حالی است که SPNs و RFR میزان تحمل خطای کمتری دارند و نمی‌توانند به‌طور کامل از خرابی‌ها بازیابی‌شوند. NSGA-II و Raft نیز در این زمینه عملکرد خوبی دارند، اما به پایداری Paxos نمی‌رسند.

مقیاس پذیری: پروتکل Paxos مقیاس پذیری بالایی دارد که به سیستم اجازه‌می‌دهد با تغییرات بار کاری سازگار شود. SPNs و RFR در این زمینه دارای محدودیت‌هایی هستند و نمی‌توانند به‌طور کامل نیازهای محیط های ابری پویا را برآورده‌کنند. NSGA-II و Raft نیز مقیاس‌پذیری بالایی دارند که به عملکرد بهتر آن‌ها در شرایط مختلف کمک‌می‌کند.

کارایی: پروتکل Paxos با کارایی ٪۸۵ عملکرد بسیار خوبی در استفاده از منابع ارائه‌می‌دهد. SPNs، NSGA-II، و RFR کارایی کمتری دارند که نشان‌دهنده بهره‌وری کمتر آن‌ها در مدیریت منابع است. Raft نیز با کارایی ٪۸۲ عملکرد خوبی ارائه‌می‌دهد، اما همچنان به کارایی Paxos نمی‌رسد.

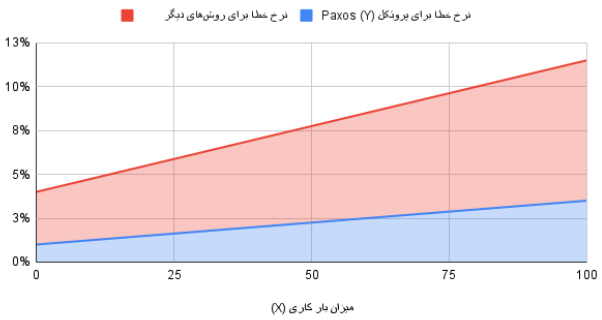
هزینه: پروتکل Paxos با هزینه کم بهینه‌سازی منابع را بهبود می‌بخشد. SPNs و Raft هزینه‌های متوسطی دارند، در حالی که NSGA-II و RFR هزینه‌های بالاتری دارند که می‌تواند مدیریت منابع را پیچیده‌تر کند.

هزینه محاسباتی: پروتکل Paxos هزینه محاسباتی متوسطی دارد که ناشی از پیچیدگی پروتکل اجماع است. SPNs کمترین هزینه محاسباتی را دارد، در حالی که NSGA-II و RFR به دلیل پیچیدگی الگوریتم‌ها هزینه‌های بالایی دارند. Raft نیز هزینه محاسباتی بالایی دارد که می‌تواند عملکرد را تحت تأثیر قراردهد.

پایداری در شرایط بار بالا: پروتکل Paxos با پایداری بالا در شرایط بار بالا عملکرد خود را حفظ‌می‌کند. این ویژگی در مقایسه با SPNs و RFR که در شرایط بار بالا دچار ناپایداری می‌شوند، مزیت بیشتری دارد. NSGA-II و Raft نیز پا یداری بالایی در شرایط بار بالا ارائه‌می‌دهند.

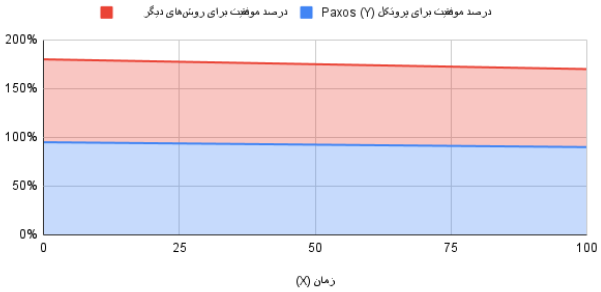
مصرف انرژی: پروتکل Paxos با مصرف انرژی کم به بهینه‌سازی منابع کمک‌می‌کند. SPNs نیز مصرف انرژی کمی دارند، در حالی که NSGA-II و RFR به دلیل پیچیدگی الگوریتم‌ها مصرف انرژی بالاتری دارند. Raft نیز مصرف انرژی متوسطی دارد که می‌تواند مدیریت منابع را بهبود بخشد.

هزینه عملیاتی: پروتکل Paxos با هزینه عملیاتی کم به بهبود بهره‌وری سیستم کمک‌می‌کند. SPNs و Raft هزینه‌های عملیاتی متوسطی دارند، در حالی که NSGA-II و RFR هزینه‌های عملیاتی بالایی دارند که می‌تواند مدیریت منابع را پیچیده‌تر کند.



شکل ۳: مقایسه نرخ خطا

در شکل ۳، نرخ خطاهای سیستم در مواجهه با بارهای مختلف و خرابی‌های نودها مقایسه‌می‌شود. محور افقی نشان‌دهنده میزان بار کاری و محور عمودی نشان‌دهنده نرخ خطا است. با افزایش بار کاری، نرخ خطا در روش‌های مختلف بررسی‌می‌شود و نشان‌می‌دهد که الگوریتم Paxos چگونه عملکرد بهتری در مقایسه با روش‌های دیگر دارد.



شکل ۴: مقایسه نرخ موفقیت الگوریتم

در شکل ۴، درصد موفقیت الگوریتم‌ها در طول زمان مقایسه‌می‌شود. محور افقی نشان‌دهنده زمان و محور عمودی نشان‌دهنده درصد موفقیت است. این نمودار به‌وضوح نشان‌می‌دهد که الگوریتم Paxos چگونه درصد موفقیت بالاتری را در مقایسه با روش‌های دیگر (مانند SPNs، NSGA-II، RFR، و Raft حفظ‌می‌کند.

۸. نتیجه

نتیجه‌گیری در این مقاله، یک الگوریتم بهینه‌سازی مبتنی بر پروتکل Paxos برای سیستم‌های میکرو سرویس ارائه شد که به منظور بهبود عملکرد و قابلیت اطمینان در محیط‌های ابری پویا طراحی‌شده‌است. روش پیشنهادی ما از پروتکل اجماع Paxos برای حل چالش‌های مربوط به سازگاری و تحمل خطا استفاده‌می‌کند و در عین حال تخصیص منابع و مقیاس‌پذیری را بهینه می‌سازد.

از طریق شبیه سازی‌های گسترده و مطالعات موردی واقعی، نشان دادیم که الگوریتم ما به‌طرز قابل‌توجهی معیارهای عملکرد کلیدی را در مقایسه با تکنیک‌های بهینه‌سازی سنتی بهبود می‌بخشد. به‌طور خاص، الگوریتم مبتنی بر Paxos ما کاهش قابل‌توجهی در تأخیر و افزایش در توان عملیاتی را نشان‌داد که نمایانگر کارایی آن در مدیریت بارهای سنگین و اطمینان از تغییرات سریع وضعیت است. علاوه بر این، الگوریتم تخصیص منابع را بهینه‌سازی کرده و منجر به بهره‌وری بهتر و صرفه‌جویی در هزینه‌های مدیریت منابع ابری شده‌است.

نرخ‌های بالای سازگاری و تحمل خطای به دست آمده توسط الگوریتم ما، نشان‌دهنده استحکام آن در حفظ عملیات قابل‌اطمینان حتی در حضور خرابی‌های نود و تغییرات پویا در بار کاری است. این بهبودها پتانسیل بهینه سازی مبتنی بر اجماع را در پیشبرد طراحی و استقرار سیستم‌های میکروسرویس نشان‌می‌دهد.

تحلیل مقایسه‌ای ما با چارچوب ارائه شده در "چارچوب مدل سازی عملکرد برای زیرساخت‌های ابری مبتنی بر میکروسرویس" نشان‌می‌دهد، در حالی که چارچوب موجود بینش‌های ارزشمندی در مورد تخصیص منابع و مقیاس‌پذیری خودکار ارائه‌می‌دهد، به‌صراحت به چالش‌های سازگاری و تحمل خطا نمی‌پردازد. با ادغام مکانیزم‌های اجماع Paxos، روش ما یک راه‌حل جامع ارائه‌می‌دهد که هم عملکرد و هم قابلیت اطمینان را بهبود می‌بخشد.

تحقیقات آینده می‌تواند مکانیزم‌های اضافی برای بهبود تحمل خطا، بهینه‌سازی مصرف انرژی و افزایش امنیت در سیستم‌های میکروسرویس را بررسی‌کند. انجام استقرارهای واقعی و مطالعات موردی بیشتر، اثربخشی الگوریتم پیشنهادی ما را در محیط‌ها و دامنه‌های کاربردی مختلف ابری تأییدخواهدکرد.

در نتیجه، الگوریتم بهینه‌سازی مبتنی بر Paxos ما نمایانگر پیشرفت قابل توجهی در بهینه‌سازی سیستم‌های میکروسرویس است و یک راه‌حل قوی و مقیاس‌پذیر برای برنامه‌های کاربردی مدرن مبتنی بر ابر ارائه‌می‌دهد. این مطالعه اهمیت ادغام پروتکل‌های اجماع در چارچوب‌های بهینه سازی عملکرد را برای دستیابی به دستر سی بالا، قابلیت اطمینان و کارایی در معماری‌های میکروسرویس برجسته‌می‌کند.

References

[1] J. Esparza-Peidro, F. D. Muñoz-Escoí, and J. M. Bernabéu-Aubán, "Modeling microservice architectures," *J. Syst. Softw.*, vol. 213, p. 112041, Jul. 2024, doi: 10.1016/J.JSS.2024.112041.

[2] "Modeling microservice architectures - ScienceDirect." Accessed: Jul. 27, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0164121224000840

[3] J. Esparza-Peidro, F. D. Muñoz-Escoí, and J. M. Bernabéu-Aubán, "Modeling microservice architectures," *J. Syst. Softw.*, vol. 213, Jul. 2024, doi: 10.1016/j.jss.2024.112041.

[4] M. Diogo, B. Cabral, and J. Bernardino, "Consistency models of NoSQL databases," *Futur. Internet*, vol. 11, no. 2, 2019, doi: 10.3390/FI11020043.

[5] X. Zhou *et al.*, "Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry," *J. Syst. Softw.*, vol. 195, Jan. 2023, doi: 10.1016/j.jss.2022.111521.

[6] J. Zhao, Y. Zhang, J. Jiang, Z. Hua, and Y. Xiang, "A secure dynamic cross-chain decentralized data consistency verification model," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 36, no. 1, p. 101897, 2024, doi: 10.1016/j.jksuci.2023.101897.

[7] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency Models," *Proc. Mach. Learn. Res.*, vol. 202, pp. 32211–32252, 2023, doi: 10.1007/978-1-4842-1329-2_9.

[8] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica, "Quantifying eventual consistency with PBS," *VLDB J.*, vol. 23, no. 2, pp. 279–302, 2014, doi: 10.1007/S00778-013-0330-1.

[9] M. ul Hassan *et al.*, "An efficient dynamic decision-based task optimization and scheduling approach for microservice-based cost management in mobile cloud computing applications," *Pervasive Mob. Comput.*, vol. 92, p. 101785, 2023, doi: https://doi.org/10.1016/j.pmcj.2023.101785.

[10] L. Lelovic *et al.*, "Change impact analysis in microservice systems: A systematic literature review," *J. Syst. Softw.*, vol. 219, p. 112241, 2025, doi: https://doi.org/10.1016/j.jss.2024.112241.

[11] M. Baboi, A. Iftene, and D. Gîfu, "Dynamic Microservices to Create Scalable and Fault Tolerance

Architecture," *Procedia Comput. Sci.*, vol. 159, pp. 1035–1044, Jan. 2019, doi: 10.1016/J.PROCS.2019.09.271.

[12] M. Baboi, A. Iftene, and D. Gîfu, "Dynamic Microservices to Create Scalable and Fault Tolerance Architecture," *Procedia Comput. Sci.*, vol. 159, pp. 1035–1044, 2019, doi: 10.1016/j.procs.2019.09.271.

[13] N. Bansal, A. Awasthi, and S. Bansal, "Task scheduling algorithms with multiple factor in cloud computing environment," *Advances in Intelligent Systems and Computing*, vol. 433. pp. 619–627, 2016. doi: 10.1007/978-81-322-2755-7_64.

[14] A. K. Jain, N. Gupta, and B. B. Gupta, "A survey on scalable consensus algorithms for blockchain technology," *Cyber Secur. Appl.*, vol. 3, p. 100065, Dec. 2025, doi: 10.1016/J.CSA.2024.100065.

[15] T. F. da Silva Pinheiro, P. Pereira, B. Silva, and P. Maciel, "A performance modeling framework for microservices-based cloud infrastructures," *J. Supercomput.*, vol. 79, no. 7, pp. 7762–7803, May 2023, doi: 10.1007/S11227-022-04967-6/METRICS.

[16] A. Zappone and E. A. Jorswieck, "Energy-efficient resource allocation in future wireless networks by sequential fractional programming," *Digit. Signal Process.*, vol. 60, pp. 324–337, Jan. 2017, doi: 10.1016/J.DSP.2016.09.014.

[17] H. Guo, H. Cao, J. He, X. Liu, and Y. Shi, "POBO: Safe and optimal resource management for cloud microservices," *Perform. Eval.*, vol. 162, p. 102376, Nov. 2023, doi: 10.1016/J.PEVA.2023.102376.

[18] B. Cai, B. Wang, M. Yang, and Q. Guo, "AutoMan: Resource-efficient provisioning with tail latency guarantees for microservices," *Futur. Gener. Comput. Syst.*, vol. 143, pp. 61–75, Jun. 2023, doi: 10.1016/J.FUTURE.2023.01.014.

[19] V. Ramamoorthi, "AI-Enhanced Performance Optimization for Microservice-Based Systems," vol. 4, no. September, pp. 1–7, 2024, doi: 10.69987/JACS.2024.40901.

[20] B. Barua and M. S. Kaiser, "Leveraging Microservices Architecture for Dynamic Pricing in the Travel Industry : Algorithms , Scalability , and Impact on Revenue and Customer Satisfaction," pp. 1–19.

[21] X. Zhou *et al.*, "Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry," *J. Syst. Softw.*, vol. 195, p. 111521, Jan. 2023, doi: 10.1016/J.JSS.2022.111521.

[22] A. Munir, "Design and Implementation of a Smart Peer-to-Peer Energy Exchange Platform," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1730–1741, 2017.

پی‌نوشت

1- https://github.com/esmaeilsadeghijob/Leveraging-Consensus-Algorithms-to-Optimize-Microservice-Systems.git
2- https://jmeter.apache.org/download_jmeter.cgi
3- https://online.visual-paradigm.com/share.jsp?id=3530313334322d33