

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<sup>1</sup>, Mahmood Deypir<sup>2\*</sup>, Ali Broumandnia<sup>3</sup>

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@iaau.ac.ir](mailto:mdeypir@iaau.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

## استفاده از الگوریتم‌های اجماع برای بهینه‌سازی معماری میکروسرویس

دوره پنجم، تابستان ۱۴۰۳  
شماره دوم، صص: ۱۰۵-۱۱۸

تاریخ دریافت: ۱۴۰۳/۰۲/۰۳  
تاریخ پذیرش: ۱۴۰۳/۰۳/۱۶

اسماعیل صادقی هفشجانی<sup>۱</sup>، محمود دی‌پیر<sup>۲\*</sup>، علی برومندنی<sup>۳</sup>

۱- گروه مهندسی کامپیوتر، واحد تهران جنوب، دانشگاه آزاد اسلامی، تهران، ایران. [esmail.sadeghi@iau.ac.ir](mailto:esmail.sadeghi@iau.ac.ir)

۲- گروه مهندسی کامپیوتر، واحد تهران جنوب، دانشگاه آزاد اسلامی، تهران، ایران. (نویسنده مسئول) [mdeypir@iau.ac.ir](mailto:mdeypir@iau.ac.ir)

۳- گروه مهندسی کامپیوتر، واحد تهران جنوب، دانشگاه آزاد اسلامی، تهران، ایران. [Ali.Broumandnia@iau.ac.ir](mailto: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 شده است. روش‌شناسی ما شامل استفاده از Paxos برای تضمین سازگاری و تحمل خطا در سیستم‌های میکروسرویس می‌باشد. شبیه‌سازی‌ها و مطالعات موردی برای ارزیابی عملکرد رویکرد ما انجام شده است. نتایج نشان می‌دهد که Paxos تأخیر را کاهش داده، رقابت منابع را به حداقل می‌رساند و سازگاری حالت را در خدمات توزیع شده تضمین می‌کند. بهینه‌سازی مبتنی بر Paxos نه تنها عملکرد را بهبود می‌بخشد، بلکه قابلیت اطمینان و دسترسی سیستم‌های میکروسرویس را نیز افزایش می‌دهد. این مطالعه مزایای قابل توجهی از Paxos در بهینه‌سازی استقرارهای میکروسرویس را نشان می‌دهد.

**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 بهبود بخشیده 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 ایجاد می‌کند [9].

یکی از مسائل مهم در سیستم‌های میکروسرویس، اطمینان از عملکرد قابل اعتماد 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 ارائه می‌دهند که با فراهم کردن مکانیزمی برای دستیابی به توافق بین گره‌های توزیع‌شده، سازگاری و تحمل خطا را بهبود می‌بخشد [14][13][12][11].

در این مقاله، ما پیشنهاد می‌کنیم از الگوریتم‌های اجماع Paxos برای بهینه‌سازی سیستم‌های میکروسرویس استفاده شود. 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] مقایسه می‌کنیم که از شبکه‌های پتری تصادفی SPNs، الگوریتم ژنتیک مرتب‌سازی غیرمسلط (NSGA-II) و رگرسیون جنگل تصادفی RFR برای بهینه‌سازی عملکرد استفاده می‌کند [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 مدیریت می‌کند [18][17][16]. با شناسایی می‌کند و بینش‌های ارزشمندی در مورد رفتار میکروسرویس‌ها در پیکربندی‌های مختلف ارائه می‌دهد. 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 حاصل می‌کنیم که سیستم‌های میکروسرویس می‌توانند حتی در صورت خرابی گره‌ها و تغییرات پویا در بار کاری، در دسترس و قابل اعتماد باقی‌مانند. از طریق شبیه‌سازی‌های گسترده و مطالعات موردی واقعی، نشان می‌دهیم که رویکرد ما نه تنها معیارهای عملکرد را بهبود می‌بخشد، بلکه به‌طور قابل توجهی استحکام کلی سیستم‌های میکروسرویس را نیز افزایش می‌دهد [2][1].

این مقاله به شرح زیر ساختار یافته‌است: بخش ۲ مروری بر کارهای مرتبط ارائه می‌دهد و نقاط قوت و محدودیت‌های چارچوب‌های بهینه‌سازی موجود را برجسته می‌کند. بخش ۳ الگوریتم بهینه‌سازی مبتنی بر Paxos پیشنهادی ما را با جزئیات کلیدی و پیاده‌سازی آن توضیح می‌دهد. بخش ۴ روش‌شناسی و نتایج ارزیابی را ارائه می‌دهد و رویکرد ما را با چارچوب پایه مقایسه می‌کند. بخش ۵ پیامدهای یافته‌های ما و زمینه‌های بالقوه برای تحقیقات آینده را مورد بحث قرار می‌دهد. 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 قرار گرفته‌است 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.

### ۲.۱. 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 معرفی شده‌است، 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 ذخیره‌سازی ابری را بهبود می‌بخشد. این بهینه‌سازی‌ها برای حفظ دسترسی بالا و سازگاری در معماری‌های میکروسرویس که خدمات در چندین گره توزیع شده‌اند، 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 تصادفی SPNs، الگوریتم ژنتیک مرتب‌سازی غیرمغلوب (NSGA-II) و رگرسیون جنگل تصادفی RFR برای مدل‌سازی و بهینه‌سازی عملکرد میکروسرویس‌ها استفاده می‌کند. با شناسایی توازن‌های بحرانی بین عملکرد و مصرف منابع، این چارچوب بینش‌های ارزشمندی در مورد رفتار میکروسرویس‌ها تحت پیکربندی‌های مختلف ارائه می‌دهد. استفاده از SPNs امکان مدل‌سازی دقیق رفتار دینامیکی میکروسرویس‌ها را فراهم می‌کند، در حالی که NSGA-II و RFR بهینه‌سازی چندهدفه و پیش‌بینی عملکرد را تسهیل می‌کنند.

### ۳.۲. تحلیل مقایسه‌ای

در حالی که 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 استفاده می‌کند و اطمینان می‌دهد که سیستم‌های میکروسرویس حتی در مواجهه با خرابی گره‌ها و تغییرات دینامیکی بار کاری، دسترسی بالا و قابلیت اطمینان را حفظ می‌کنند. با ادغام مکانیزم‌های 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 تأکید کرده‌اند. علاوه بر این، مطالعات در مورد طراحی‌های تحمل خطای میکروسرویس‌ها بر نیاز به مکانیزم‌های بازیابی قوی برای اطمینان از عملیات مداوم در طول خرابی‌ها تأکید کرده‌اند. این کارها زمینه گسترده‌تری برای درک چالش‌ها و راه‌حل‌ها در بهینه‌سازی سیستم‌های میکروسرویس فراهم می‌کنند. به‌طور خلاصه، در حالی که چارچوب‌ها و تکنیک‌های بهینه‌سازی موجود بینش‌های ارزشمندی در مورد عملکرد و مقیاس‌پذیری سیستم‌های میکروسرویس ارائه می‌دهند، رویکرد ما به‌طور منحصربه‌فردی این جنبه‌ها را با استحکام الگوریتم‌های اجماع Paxos ترکیب می‌کند. 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 قرار می‌دهد 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 نشان می‌دهد. این مزایا در مقایسه با روش‌های دیگر برجسته‌تر هستند. مقاله [1] با استفاده از مدل‌سازی معماری میکروسرویس، بهبودهایی در زمینه‌های مدل‌سازی و امنیت ارائه داده‌است. 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 بهبود می‌بخشد. 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 ارائه می‌دهد. مقاله‌های دیگر مانند [19] و [20] نیز بهینه‌سازی‌های مختلفی در زمینه عملکرد و منابع ارائه کرده‌اند، 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 فراهم می‌آورد. بنابراین، روش پیشنهادی ما با استفاده از پروتکل Paxos، یک راه‌حل جامع و کارآمد برای بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس ارائه می‌دهد 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                            |
|----------------------------------------------------------------------------------|---------------------------------------------------------------------------|------------|-------------------------------------------|---------------------------------|-------------------------------------|-------------------------------------|
| مدل سازی معماری میکروسرویس [1]                                                   | جی. اسپارزا-پیدرو، اف. دی. مونوز-اسکوی، جی. ام. برنابئو-آوبان             | ۲۰۲۴       | مدل سازی معماری میکروسرویس                | بهبود عملکرد و مقیاس پذیری      | دقت بالا در مدل سازی                | 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 قیمت گذاری پویا در صنعت سفر [20]               | بیمان باروا، ام. شامین کایزر                                              | ۲۰۲۴       | سیستم قیمت گذاری پویا مبتنی بر میکروسرویس | بهبود درآمد و رضایت مشتری       | بهبود مقیاس پذیری 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، پی. کابرال، جی. برناردینو                                      | ۲۰۱۶       | مدل‌های سازگاری NoSQL                     | بهبود سازگاری داده‌ها           | کارایی بالا                         | محدودیت در تحمل خطا                 |
| کمی سازی سازگاری نهایی با PBS [8]                                                | پی. بلیس، اس. ونکاتارامان، ام. جی. فرانکلین، جی. ام. هارستین، آی. استویکا | ۲۰۱۴       | مدل PBS                                   | بهبود سازگاری نهایی             | دقت بالا                            | محدودیت در کاربردهای خاص            |

### • مکانیزم اجماع:

انتخاب رهبر: با استفاده از پروتکل Paxos، یک رهبر از بین نودها برای هماهنگی فرآیند اجماع انتخاب می‌شود. 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) پیشنهاد می‌دهد. هر نود با یک وعده پاسخ می‌دهد 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 اعلام می‌کند تا اطمینان حاصل شود که سیستم یکپارچه عمل می‌کند.

### • تخصیص پویا منابع:

نظارت بر منابع: سیستم به طور مداوم استفاده از منابع (CPU، حافظه، شبکه) را نظارت کرده و معیارهای عملکرد (تأخیر، توان عملیاتی) را جمع‌آوری می‌کند.

تخصیص منابع: بر اساس داده‌های نظارت شده، مدیر منابع به طور پویا منابع را به میکروسرویس‌ها تخصیص می‌دهد تا استفاده بهینه و کاهش تداخل را تضمین کند.

### 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 شده است، ارائه می‌دهیم. رویکرد ما با چارچوبی که در تحقیقات قبلی [15] توضیح داده شده است، مقایسه می‌شود. این چارچوب از شبکه‌های پتری تصادفی (SPNs)، الگوریتم ژنتیک مرتب سازی غیرمسلط (NSGA-II) و رگرسیون جنگل تصادفی (RFR) برای بهینه‌سازی عملکرد استفاده می‌کند. الگوریتم پیشنهادی ما مکانیزم‌های اجماع را برای اطمینان از سازگاری و تحمل خطا در حین بهینه‌سازی تخصیص منابع و مقیاس بندی ادغام می‌کند.

#### ۱.۳. مروری بر الگوریتم

الگوریتم بهینه‌سازی شامل اجزای کلیدی زیر است:

#### • ثبت و کشف سرویس‌ها:

رجیستری سرویس: هر میکرو سرویس با یک رجیستری مرکزی ثبت می‌شود و اطلاعاتی مانند نقاط انتهایی سرویس، نیازمندی‌های منابع و وابستگی‌ها را ارائه می‌دهد.

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

## • مقیاس‌گذاری خودکار:

سیاست‌های مقیاس‌گذاری: آستانه‌های از پیش تعریف شده و معیارهای عملکرد بلادرنگ، اقدامات مقیاس‌گذاری را تحریک می‌کنند. سیستم می‌تواند نمونه‌های میکروسرویس را بر اساس شرایط بار فعلی افزایش یا کاهش دهد. مقیاس‌گذاری مبتنی بر اجماع: پروتکل Paxos اطمینان می‌دهد که تصمیمات مقیاس‌گذاری به‌طور یکپارچه در همه نودها اعمال می‌شود و پایداری و عملکرد سیستم حفظ می‌شود.

## • تحمل خطا و بازیابی:

تشخیص خرابی: سیستم به‌طور مداوم سلامت نودها و سرویس‌ها را نظارت می‌کند. در صورت خرابی، پروتکل Paxos انتخاب رهبر جدید و توزیع مجدد وظایف را تسهیل می‌کند. تکثیر وضعیت: برای اطمینان از تحمل خطا، وضعیت هر میکروسرویس در چندین نود تکثیر می‌شود. پروتکل Paxos تضمین می‌کند که تغییرات وضعیت به‌طور یکپارچه اعمال می‌شود، حتی در صورت خرابی نودها.

## • نظارت بر عملکرد و بازخورد:

جمع‌آوری معیارها: سیستم معیارهای عملکرد مانند تاخیر، توان عملیاتی و استفاده از منابع را جمع‌آوری می‌کند. این معیارها برای ارزیابی اثربخشی الگوریتم پیشنهادی استفاده می‌شوند. حلقه بازخورد: بر اساس معیارهای جمع‌آوری‌شده، الگوریتم به‌طور مداوم تخصیص منابع و سیاست‌های مقیاس‌گذاری را تنظیم می‌کند تا عملکرد بهینه شود. حلقه بازخورد اطمینان می‌دهد که سیستم به تغییرات بار کاری سازگار شده و عملکرد بالایی را حفظ می‌کند.

## ۲.۳. مراحل الگوریتم

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

### • 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:

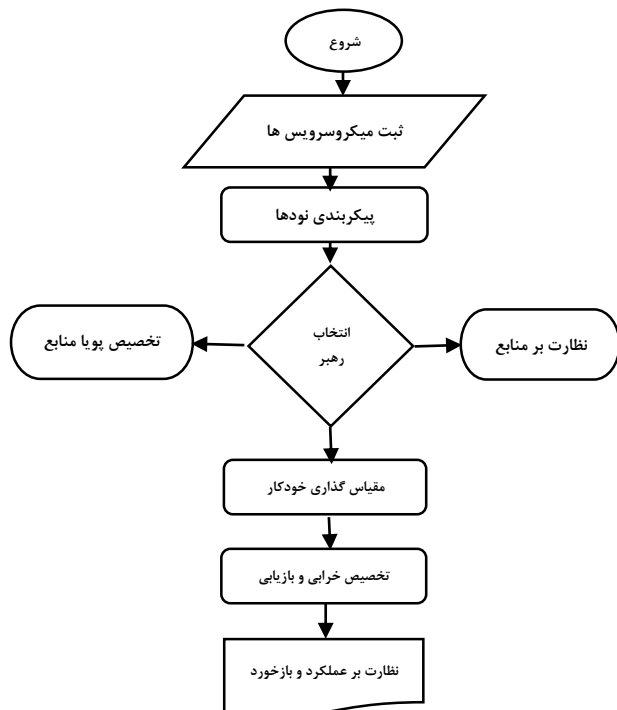
مقیاس‌گذاری نمونه‌های میکروسرویس بر اساس سیاست‌های از پیش تعریف‌شده و معیارهای بلادرنگ.

## • 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<sup>۲</sup>

## 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 بینش‌های ارزشمندی در مورد تخصیص منابع و خودکارسازی مقیاس‌پذیری ارائه می‌دهد، به چالش‌های مربوط به سازگاری و تحمل خطا به‌طور صریح نمی‌پردازد.

الگوریتم پیشنهادی ما با ادغام مکانیزم‌های اجماع Paxos، اطمینان حاصل می‌کند 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 بهینه‌سازی عملکرد با پروتکل‌های اجماع قوی، رویکرد ما یک راه‌حل جامع برای بهبود همزمان عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس در محیط‌های ابری ارائه می‌دهد.

## ۴. گپ تحقیقاتی

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

#### 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 کمک می‌کند. برای توضیح دقیق‌تر و واضح‌تر نوآوری روش پیشنهادی و مزیت نسبی پروتکل Paxos نسبت به روش‌های دیگر، با دید به نقاط قوت و توانمندی‌های خاص آن پردازیم. در ادامه توضیحات کاملی در این زمینه آورده شده است:

##### ۱. سازگاری بالا:

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

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

##### ۲. تحمل خطا:

- توضیح: Paxos با استفاده از مکانیزم‌های انتخاب رهبر و تکثیر وضعیت، تحمل خطای بسیار بالایی ارائه می‌دهد. این مکانیزم‌ها به سیستم اجازه می‌دهند که حتی در حضور خرابی‌های نود، عملکرد پایدار و قابل اطمینانی داشته باشد.

- مزیت نسبی: در مقایسه با NSGA-II و SPNs که نمی‌توانند به سرعت از خرابی‌ها بازیابی شوند، Paxos تضمین می‌کند که سیستم می‌تواند به سرعت از خرابی‌ها بازیابی شود و تداوم عملکرد را حفظ کند.

##### ۳. بهینه‌سازی تخصیص منابع:

- توضیح: پروتکل Paxos با الگوریتم‌های بهینه‌سازی تخصیص منابع ادغام می‌شود تا تخصیص منابع پویا و مقیاس گذاری خودکار را بهبود بخشد.

- 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 جلوگیری می‌کند. این ویژگی به سیستم اجازه می‌دهد که به‌طور کارآمد با تغییرات بار کاری سازگار شود.

##### ۴. پایداری در شرایط بار بالا:

- توضیح: پروتکل Paxos با مکانیزم‌های تحمل خطا و اجماع، پایداری سیستم را در شرایط بار بالا تضمین می‌کند. این ویژگی به سیستم اجازه می‌دهد که حتی در مواجهه با فشار بالا و نوسانات شدید در بار کاری، عملکرد خود را حفظ کند.

- 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 تبدیل می‌کند.

نمی‌توانند به‌طور کامل چالش‌های مرتبط با تحمل خطا و سازگاری در محیط‌های میکروسرویس را برطرف کنند. در اینجا به توضیح دلایل ناکافی بودن این روش‌ها پرداخته شده است:

##### 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 اجازه می‌دهد تا نقاط ضعف و گلوگاه‌ها را شناسایی کنند.

- محدودیت‌ها در تحمل خطا: با وجود مزایای SPNs در مدل‌سازی، این روش‌ها برای مدیریت خرابی‌های ناگهانی طراحی نشده‌اند. SPNs نمی‌توانند به سرعت و کارآمد از خرابی‌ها بازیابی شوند. همچنین، SPNs تضمین نمی‌کنند که تغییرات وضعیت به‌طور سازگار در تمام گره‌ها اعمال شود. 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 برقرار کند.

- تمرکز بر بهینه‌سازی عملکرد: این الگوریتم بیشتر بر بهینه‌سازی 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 پاسخ ندهند. این موضوع باعث کاهش توانایی این روش در مدیریت خرابی‌های ناگهانی و بازیابی سریع از آن‌ها می‌شود.

##### ۳. رگرسیون جنگل تصادفی RFR

- پیش‌بینی‌های دقیق: RFR می‌تواند پیش‌بینی‌های دقیقی ارائه دهد و به بهینه‌سازی تخصیص منابع کمک کند.

- عدم مدیریت خرابی‌ها: RFR نمی‌تواند مکانیزم‌های لازم برای مدیریت و بازیابی خطاها و تضمین سازگاری حالت‌ها را ارائه دهد. این الگوریتم نمی‌تواند تضمین کند 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 مقابله کنند. SPNs در مدیریت خرابی‌ها ضعیف هستند، NSGA-II بیشتر بر بهینه‌سازی عملکرد تمرکز دارد و RFR نمی‌تواند سازگاری و بازیابی خطاها را تضمین کند. به همین دلیل، نیاز به یک رویکرد جامع‌تر و قوی‌تر مانند پروتکل Paxos احساس می‌شود که بتواند این چالش‌ها را به‌طور کامل برطرف کند.

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 مدیریت کنند، 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 قابل توجهی در بهینه‌سازی سیستم‌های میکروسرویس است. این مزایا نشان می‌دهند که Paxos می‌تواند بهبودهای قابل توجهی در عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس ایجاد کند و نسبت به روش‌های دیگر برتری داشته باشد.

## ۵. روش

### ۵.۱. 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 پیشنهاد می‌کنیم که از پروتکل اجماع Paxos بهره‌می‌برد. رویکرد ما برای افزایش قابلیت اطمینان و مقیاس پذیری معماری‌های میکروسرویس، به‌ویژه در محیط‌های ابری پویا طراحی شده است. 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 ارائه می‌دهد.

**پیکربندی نودها:** نودهای ابری برای میزبانی میکروسرویس‌ها پیکربندی می‌شوند 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) پیشنهاد می‌دهد 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) ارسال می‌کند. 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 دریافت کند، به عنوان رهبر انتخاب می‌شود 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 انتخاب می‌شود. رهبر مسئول پیشنهاد و تعهد تغییرات وضعیت است. رهبر یک تغییر وضعیت (مانند تخصیص منابع، مقیاس‌گذاری سرویس) را به نودهای دیگر (پذیرندگان) پیشنهاد می‌دهد. هر نود با یک وعده پاسخ می‌دهد که در صورت برآورده شدن شرایط خاص، پیشنهاد را می‌پذیرد. در مرحله تعهد، پس از پذیرش پیشنهاد توسط اکثریت نودها، رهبر تغییر وضعیت را تعهد کرده و تصمیم را به همه نودها اعلام می‌کند تا یکپارچگی در سیستم تضمین شود.

میزان تأثیر خرابی گره‌ها: در پروتکل Paxos با استفاده از مکانیزم‌های تحمل خطا می‌تواند تأثیر خرابی گره‌ها 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 جلوگیری می‌کند.

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

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

پایداری الگوریتم تحت بار بالا: پایداری پروتکل Paxos تحت بار بالا به دلیل مکانیزم‌های تحمل خطا و تخصیص منابع پویا تضمین می‌شود. 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 کمک می‌کند.

- مقیاس پذیری: پروتکل Paxos به‌طور پویا با تغییرات بار کاری سازگار می‌شود و می‌تواند به سرعت با افزایش یا کاهش بار کاری تنظیم شود. این ویژگی به سیستم اجازه می‌دهد که در مواجهه با بارهای سنگین عملکرد پایدار و قابل اعتمادی داشته باشد.

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

ابری پویا ارائه می‌دهد. این ویژگی‌ها به‌وضوح نشان می‌دهند که Paxos می‌تواند بهبودهای قابل توجهی در عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس ایجاد کند.

#### • تخصیص منابع و مقیاس‌گذاری:

تخصیص منابع پویا: الگوریتم به صورت پویا منابع را بر اساس بار فعلی و نیازهای سرویس تخصیص می‌دهد. رهبر استفاده از منابع را نظارت کرده و تنظیمات لازم را برای بهینه‌سازی عملکرد انجام می‌دهد.

مقیاس‌گذاری خودکار: سیستم به‌طور خودکار نمونه‌های میکروسرویس را بر اساس آستانه‌های از پیش تعریف‌شده و معیارهای عملکرد واقعی مقیاس‌گذاری می‌کند. پروتکل Paxos تضمین می‌کند که تصمیمات مقیاس‌گذاری به‌طور یکپارچه در تمام نودها اعمال شود.

#### • تحمل خطا و بازیابی:

تشخیص خرابی: سیستم به‌طور مداوم سلامت نودها و سرویس‌ها را نظارت می‌کند. در صورت بروز خرابی، پروتکل Paxos انتخاب رهبر جدید و توزیع مجدد وظایف را تسهیل می‌کند. تکثیر وضعیت: برای اطمینان از تحمل خطا، وضعیت هر میکروسرویس در چندین نود تکثیر می‌شود. پروتکل Paxos تضمین می‌کند که تغییرات وضعیت یکپارچه اعمال شوند، حتی در صورت خرابی نودها.

#### • نظارت و بهینه‌سازی عملکرد:

جمع‌آوری معیارها: سیستم معیارهای عملکردی مانند تأخیر، توان عملیاتی و استفاده از منابع را جمع‌آوری می‌کند. این معیارها برای ارزیابی اثربخشی الگوریتم پیشنهادی استفاده می‌شوند. حلقه بازخورد: بر اساس معیارهای جمع‌آوری‌شده، الگوریتم به‌طور مداوم تخصیص منابع و سیاست‌های مقیاس‌گذاری را برای بهینه‌سازی عملکرد تنظیم می‌کند. حلقه بازخورد تضمین می‌کند که سیستم به بارهای کاری متغیر سازگار شده و عملکرد بالایی را حفظ می‌کند.

### 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 جمع‌آوری شده از یک سیستم میکروسرویس مبتنی بر ابر انجام دادیم. 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

شده بودند، انجام شد. 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 میکروسرویس‌ها تولید کرد.

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

#### • معیارهای عملکرد:

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

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

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

سازگاری: به عنوان درصد تغییرات وضعیت موفق که یکپارچه در تمام نودها اعمال شده‌اند، اندازه‌گیری شد.

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

#### • نتایج:

تأخیر: الگوریتم مبتنی بر Paxos ما کاهش قابل توجهی در تأخیر نسبت به چارچوب مینا به دست آورد. میانگین تأخیر ۲۵٪ کاهش یافت که کارایی مکانیزم اجماع در اطمینان از تغییرات سریع وضعیت را نشان می‌دهد.

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 بهینه کرد و بهبود ۲۰٪ در کارایی استفاده از CPU و حافظه نسبت به چارچوب مینا به دست آورد.

سازگاری: الگوریتم مبتنی بر Paxos نرخ سازگاری بالای ۹۹٫۹٪ را حفظ کرد و اطمینان حاصل کرد که تغییرات وضعیت یکپارچه در تمام نودها اعمال می‌شوند.

تحمل خطا: سیستم تحمل خطایی قوی نشان داد و با نرخ موفقیت ۹۵٪ در حفظ عملیات در طول خرابی نودها عمل کرد.

این نتایج بهبودهای قابل توجهی در عملکرد و قابلیت اطمینان الگوریتم بهینه‌سازی مبتنی بر Paxos ما را نشان می‌دهند. کاهش تأخیر و افزایش توان عملیاتی نشان می‌دهند که الگوریتم می‌تواند بارهای بالا را به‌طور کارآمد مدیریت کند، در حالی که بهبودهای استفاده از منابع توانایی آن در بهینه‌سازی استفاده از منابع ابری

```

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 پیاده سازی اجماع Paxos و میکروسرویس ها

## ۲.۷. پیامدهای یافته ها

- **کاهش تأخیر:** الگوریتم مبتنی بر Paxos به طور مؤثری تأخیر را با اطمینان از تغییرات حالت سازگار و تخصیص منابع کارآمد کاهش می دهد. این امر به ویژه در محیط های ابری پویا که زمان پاسخ سریع حیاتی است، مفید است. کاهش تأخیر تجربه کاربری

جدول ۲: تنظیمات سرورهای آزمایشی

| کانفیگ                    | نام قلم                                         | مشخصات سخت-افزاری                        | محیط اجرا | شبکه               |
|---------------------------|-------------------------------------------------|------------------------------------------|-----------|--------------------|
| تنظیمات<br>هاست<br>فیزیکی | هاست فیزیکی ۱<br>هاست فیزیکی ۲<br>هاست فیزیکی ۳ | Intel Core i7, 8<br>Cores 16.0 GB<br>RAM | Java v19  | 100<br>Mbps<br>LAN |

و پاسخگویی سیستم را بهبود می بخشد و آن را برای برنامه های بلادرنگ مناسب می سازد.

- **افزایش توان عملیاتی:** مکانیزم های تخصیص منابع پویا و مقیاس پذیری خودکار، توان عملیاتی سیستم را افزایش می دهند و به سیستم اجازه می دهند بارهای بالاتری را مدیریت کند. این بهبود برای برنامه هایی با بارهای کاری نوسانی حیاتی است و اطمینان می دهد که سیستم می تواند به طور کارآمد برای پاسخگویی به تقاضا مقیاس شود. افزایش توان عملیاتی قابلیت الگوریتم را در بهینه سازی استفاده از منابع و حفظ عملکرد بالا تحت شرایط مختلف نشان می دهد.

را نشان می دهند. نرخ های بالای سازگاری و تحمل خطا نیز استحکام الگوریتم در حفظ عملیات قابل اعتماد در محیط های ابری پویا را تأیید می کنند.

## ۷. بحث

بحث ارزیابی های ما نشان می دهد که الگوریتم بهینه سازی مبتنی بر Paxos پیشنهادی به میزان قابل توجهی عملکرد و قابلیت اطمینان سیستم های میکروسرویس را در مقایسه با تکنیک های بهینه سازی سنتی بهبود می بخشد. این بخش به بررسی پیامدهای یافته های ما، مزایای رویکرد ما و زمینه های بالقوه برای تحقیقات آینده می پردازد.

## ۱.۷. تنظیمات آزمایشی

در مطالعه ما، یک برنامه مبتنی بر میکروسرویس را در چندین گره توزیع شده پیاده سازی کردیم که هر کدام میزبان انواع مختلفی از میکروسرویس ها با عملکردهای خاص بودند. برای مدیریت تراکنش های توزیع شده و حفظ سازگاری داده ها، از الگوریتم اجماع Paxos استفاده کردیم. تنظیمات آزمایشی ما به دقت برای شبیه سازی شرایط دنیای واقعی، از جمله تقسیمات شبکه و خرابی گره ها طراحی شده بود. از پنج سرور به عنوان میزبان های فیزیکی استفاده کردیم. تمام میکروسرویس ها به عنوان خدمات وب RESTful با استفاده از چارچوب Java Spring توسعه داده شدند. کد نوشته شده به زبان جاوا و آنلاین<sup>۱</sup> در دسترس است و زمان های اجرای آن با استفاده از Apache JMeter از 5.6.3 اندازه گیری شد. مشخصات دقیق تنظیمات آزمایشی در جدول ۲ ارائه شده است. در طول آزمایش ها، تنظیمات فنی و تنظیمات زمان اجرا را به طور مداوم حفظ کردیم.

همچنین در شکل ۲ ساختار پروژه و معماری اساسی یک پروژه Spring Boot را که شامل اجماع Paxos و میکروسرویس ها است، نشان می دهد.

- **۱ استفاده بهینه از منابع:** مکانیزم حلقه بازخورد اطمینان می‌دهد که منابع به‌طور کارآمد تخصیص داده می‌شوند، اتلاف را به حداقل می‌رساند و عملکرد را به حداکثر می‌رساند. این منجر به صرفه‌جویی در هزینه و استفاده بهتر از منابع ابری می‌شود. با تنظیم پویا تخصیص منابع بر اساس معیارهای بلادرنگ، الگوریتم اطمینان می‌دهد که منابع در جایی که بیشترین نیاز را دارند استفاده می‌شوند و کارایی کلی سیستم را بهبود می‌بخشد.
- **۲ سازگاری پیشرفته:** استفاده از اجماع Paxos اطمینان می‌دهد که تغییرات حالت به‌طور سازگار در تمام گره‌ها اعمال می‌شود و قابلیت اطمینان سیستم را بهبود می‌بخشد. 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 توزیع شده، برجسته می‌کند که برای برنامه‌هایی که نیاز به تضمین‌های سازگاری قوی دارند، حیاتی است.
- **۳ تحمل خطای بهبود یافته:** مکانیزم‌های تشخیص و بازیابی خطای الگوریتم به سیستم اجازه می‌دهد حتی در حضور خرابی گره‌ها، در دسترس بودن بالا را حفظ کند. این امر برای اطمینان از عملیات مداوم و به حداقل رساندن زمان خرابی در سیستم‌های میکروسرویس مبتنی بر ابر حیاتی است. تحمل خطای قوی نشان داده شده توسط الگوریتم ما اطمینان می‌دهد که سیستم می‌تواند به سرعت از خرابی‌ها بازیابی شود و تداوم و قابلیت اطمینان خدمات را حفظ کند.

### ۳.۷. مقایسه با چارچوب‌های موجود

چارچوب ارائه شده در مقاله [15] همان‌طور که در جدول ۳ نمایش داده شده است بر بهینه سازی معیارهای عملکردی مانند توان عملیاتی و زمان پاسخ با استفاده از SPNs، NSGA-II و RFR تمرکز دارد. در حالی که این چارچوب بینش‌های ارزشمندی در مورد تخصیص منابع و مقیاس پذیری خودکار ارائه می‌دهد، به چالش‌های مربوط به سازگاری و تحمل خطا به‌طور صریح نمی‌پردازد. الگوریتم پیشنهادی ما با ادغام مکانیزم‌های اجماع Paxos تضمین می‌کند که سیستم‌های میکروسرویس [22] حتی در حضور خرابی‌های نود و تغییرات پویا در بار کاری، از دسترس پذیری و قابلیت اطمینان بالایی برخوردار باشند. با ترکیب بهینه سازی عملکرد با پروتکل‌های اجماع قوی، رویکرد ما یک راه حل جامع برای بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس در محیط‌های ابری ارائه می‌دهد. همچنین، مزایای بهینه سازی مبتنی بر Paxos به شرح زیر می باشد:

- **مقاومت:** پروتکل Paxos یک مکانیزم مقاوم برای دستیابی به اجماع بین نودهای توزیع شده فراهم می‌کند و تضمین می‌کند که تغییرات وضعیت حتی در حضور خرابی‌ها به‌طور مداوم اعمال

شوند. این مقاومت برای حفظ قابلیت اطمینان سیستم و جلوگیری از ناسازگاری داده‌ها حیاتی است.

• **مقیاس پذیری:** مکانیزم‌های تخصیص منابع پویا و مقیاس پذیری خودکار به سیستم اجازه می‌دهند تا به‌طور کارآمد مقیاس پذیر باشد و بارهای کاری متغیر را بدون کاهش عملکرد مدیریت کند. این مقیاس پذیری برای برنامه‌های مبتنی بر ابر که با تقاضای نوسانی مواجه هستند، ضروری است.

• **قابلیت تطبیق:** مکانیزم حلقه بازخورد به سیستم اجازه می‌دهد تا به تغییرات بار کاری و نیازهای عملکردی پاسخ دهد و از استفاده بهینه از منابع و حفظ عملکرد بالا اطمینان حاصل کند. این قابلیت تطبیق، الگوریتم را برای طیف وسیعی از برنامه‌ها و سناریوهای استقرار مناسب می‌سازد. و در جهت تحقیقات آینده می‌توان به موارد زیر اشاره کرد:

• **بهبود تحمل خطا:** تحقیقات آینده می‌تواند مکانیزم‌های اضافی برای بهبود تحمل خطا را بررسی کند، مانند ادغام تکنیک‌های یادگیری ماشین برای تشخیص پیش‌بینی خرابی و بازیابی پیشگیرانه.

• **بهره‌وری انرژی:** بررسی راه‌هایی برای بهینه سازی مصرف انرژی در سیستم‌های میکروسرویس می‌تواند به استقرارهای ابری پایدارتر و مقرون به صرفه‌تر منجر شود. این می‌تواند شامل توسعه سیاست‌های تخصیص منابع و مقیاس پذیری آگاه به انرژی باشد.

• **افزایش امنیت:** ادغام اقدامات امنیتی در پروتکل اجماع برای محافظت در برابر حملات مخرب و اطمینان از یکپارچگی داده‌ها می‌تواند قابلیت اطمینان سیستم‌های میکروسرویس را بیشتر افزایش دهد.

• **۱ استقرارهای واقعی:** انجام استقرارهای واقعی و مطالعات موردی برای اعتبارسنجی اثربخشی الگوریتم پیشنهادی در محیط‌های ابری متنوع و حوزه‌های کاربردی مختلف.

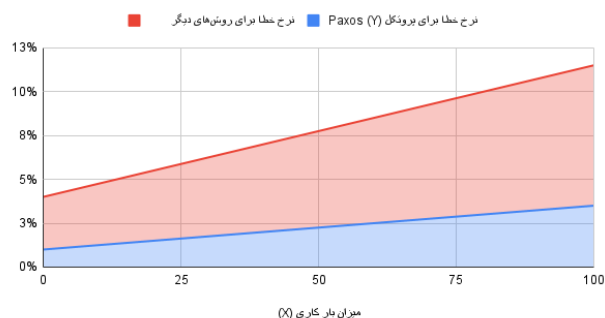
جدول ۳: نتایج مقایسه روش پیشنهادی

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

الگوریتم‌ها هزینه‌های بالایی دارند. Raft نیز هزینه محاسباتی بالایی دارد که می‌تواند عملکرد را تحت تأثیر قرار دهد. پایداری در شرایط بار بالا: پروتکل Paxos با پایداری بالا در شرایط بار بالا عملکرد خود را حفظ می‌کند. این ویژگی در مقایسه با SPNs و RFR که در شرایط بار بالا دچار ناپایداری می‌شوند، مزیت بیشتری دارد. Raft و NSGA-II نیز پایداری بالایی در شرایط بار بالا ارائه می‌دهند.

مصرف انرژی: پروتکل Paxos با مصرف انرژی کم به بهینه‌سازی منابع کمک می‌کند. 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 کمک می‌کند. 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 مقایسه می‌شود. محور افقی نشان‌دهنده میزان بار کاری و محور عمودی نشان‌دهنده نرخ خطا است. با افزایش بار کاری، نرخ خطا در روش‌های مختلف بررسی می‌شود و نشان می‌دهد که الگوریتم Paxos چگونه عملکرد بهتری در مقایسه با روش‌های دیگر دارد.

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 داده می‌شود:

زمان پاسخ‌دهی: پروتکل Paxos زمان پاسخ‌دهی ۵۰ میلی‌ثانیه را ارائه می‌دهد 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 ارائه می‌دهد. این در حالی است که SPNs و RFR میزان تحمل خطای کمتری دارند و نمی‌توانند به‌طور کامل از خرابی‌ها بازبایی شوند. NSGA-II و Raft نیز در این زمینه عملکرد خوبی دارند، اما به پایداری Paxos نمی‌رسند.

مقیاس‌پذیری: پروتکل Paxos مقیاس‌پذیری بالایی دارد که به سیستم اجازه می‌دهد با تغییرات بار کاری سازگار شود. SPNs و RFR در این زمینه دارای محدودیت‌هایی هستند و نمی‌توانند به‌طور کامل نیازهای محیط‌های ابری پویا را برآورده کنند. NSGA-II و Raft نیز مقیاس‌پذیری بالایی دارند که به عملکرد بهتر آن‌ها در شرایط مختلف کمک می‌کند.

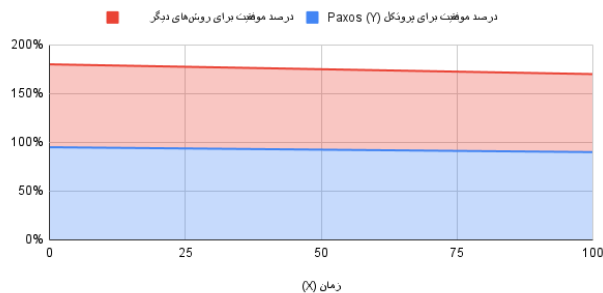
کارایی: پروتکل Paxos با کارایی ۸۵٪ عملکرد بسیار خوبی در استفاده از منابع ارائه می‌دهد. SPNs، NSGA-II، و RFR کارایی کمتری دارند که نشان‌دهنده بهره‌وری کمتر آن‌ها در مدیریت منابع است. Raft نیز با کارایی ۸۲٪ عملکرد خوبی ارائه می‌دهد، اما همچنان به کارایی Paxos نمی‌رسد.

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

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

در نتیجه، الگوریتم بهینه‌سازی مبتنی بر Paxos ما نمایانگر پیشرفت قابل توجهی در بهینه‌سازی سیستم‌های میکروسرویس است و یک راه حل قوی و مقیاس پذیر برای برنامه‌های کاربردی مدرن مبتنی بر ابر ارائه می‌دهد. این مطالعه اهمیت ادغام پروتکل‌های اجماع در چارچوب‌های بهینه سازی عملکرد را برای دستیابی به دسترسی بالا، قابلیت اطمینان و کارایی در معماری‌های میکروسرویس برجسته می‌کند.



شکل ۴: مقایسه نرخ موفقیت الگوریتم

در شکل ۴، درصد موفقیت الگوریتم‌ها در طول زمان مقایسه می‌شود. محور افقی نشان‌دهنده زمان و محور عمودی نشان‌دهنده درصد موفقیت است. این نمودار به وضوح نشان می‌دهد که الگوریتم Paxos چگونه درصد موفقیت بالاتری را در مقایسه با روش‌های دیگر (مانند SPNs، NSGA-II، RFR، و Raft حفظ می‌کند).

## References

- [1] J. Esparza-Peidro, F. D. Muñoz-Escóí, 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-Escóí, 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. Gifu, "Dynamic Microservices to Create Scalable and Fault Tolerance

## ۸. نتیجه

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

از طریق شبیه‌سازی‌های گسترده و مطالعات موردی واقعی، نشان دادیم که الگوریتم ما به طرز قابل توجهی معیارهای عملکرد کلیدی را در مقایسه با تکنیک‌های بهینه‌سازی سنتی بهبود می‌بخشد. به طور خاص، الگوریتم مبتنی بر Paxos ما کاهش قابل توجهی در تأخیر و افزایش در توان عملیاتی را نشان داد 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 شده است.

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

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

- 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. Gifu, “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](https://jmeter.apache.org/download_jmeter.cgi)
- 3- <https://online.visual-paradigm.com/share.jsp?id=3530313334322d33>