Journal of Applied And Basic Machine Intelligence Research (ABMIR)

پژوهش‌های نظری و کاربردی هوش ماشینی

سال سوم، شماره اول، بهار و تابستان ۱۴۰۴، صفحات ۱۶۱ - ۱۴۵

مدیریت منابع و حفظ انسجام در سیستم‌های میکروسرویس با رویکرد بهینه‌سازی هوشمند اجماع

مهدی بازرگانی[۱]، نفیسه فارغ زاده[۲]*، فرزانه کبودوند[۳]

[۱] استادیار، دانشکده مهندسی کامپیوتر، دانشگاه آزاد اسلامی، واحد زنجان، زنجان، ایران

[۲] استادیار، دانشکده مهندسی کامپیوتر، دانشگاه آزاد اسلامی، واحد خدابنده، زنجان، ایران

[۳] مربی، دانشکده مهندسی کامپیوتر، دانشگاه آزاد اسلامی، واحد زنجان، زنجان، ایران

مقاله پژوهشی

تاریخ دریافت:
۱۴۰۴/۳/۲۷

تاریخ پذیرش:
۱۴۰۴/۷/۲۰

کلیدواژه‌ها:
پروتکل اجماع Paxos، پایداری توزیع‌شده، مدیریت منابع، هماهنگی وضعیت، معماری میکروسرویس، بهینه‌سازی هوشمند

نویسنده‌مسئول:
fareghzadeh@iau.ac.ir

چکیده

اخیراً معماری‌های میکروسرویس به‌عنوان الگوی غالب طراحی سامانه‌های ابری، مزایای متعدد مانند انعطاف‌پذیری و مقیاس‌پذیری را به ارمغان آورده‌اند، اما در عین حال با چالش‌هایی نظیر تضمین انسجام داده‌ها، مقابله با خرابی گره‌ها و تخصیص بهینه منابع مواجه هستند. در این تحقیق، یک رویکرد اجماع نوین و سازوکار بهینه‌سازی هوشمندی ارائه شده است که با بهره‌گیری از پروتکل اجماع **Paxos**، به ارتقای عملکرد، تاب‌آوری و پایداری بیشتر سامانه‌های میکروسرویس می‌انجامد. رویکرد پیشنهادی با تأکید بر هماهنگی هوشمند در تغییر وضعیت خدمات، کاهش تأخیر و تخصیص پویای منابع، جایگزینی مؤثر برای راهکارهای سنتی مبتنی بر مدل‌های آماری و تکاملی است. با اجرای آزمایش‌های تجربی در محیط‌های ابری پویا و ارزیابی سناریوهای بارکاری متغیر، مشخص شد که الگوریتم پیشنهادی قادر است با حفظ ۹۹/۹٪ سازگاری با نسخه‌های موجود و ۹۵٪ تحمل خطا، به میزان ۲۵٪ بهبود در معیار زمان پاسخ، ۳۰٪ بهبود در توان عملیاتی سیستم و ۲۰٪ بهبود در معیار کارایی سراسری سیستم ایجاد نماید. همچنین، مقایسه با رویکردهای موجود نشان داد که ترکیب الگوریتم‌های اجماع با مکانیزم‌های بهینه‌سازی هوشمند و مدیریت مؤثر منابع، امکان‌پذیری دستیابی به عملکرد پایدار در شرایط بارکاری ناپایدار و متغیر ابری را فراهم می‌آورد. این نتایج نشان‌دهنده ظرفیت بالای دیدگاه پیشنهادی برای استفاده در معماری‌های نوین و پیچیده مبتنی بر میکروسرویس است.

: 10.22034/ABMIR.2025.23292.1135

E-ISSN: 2821-2037

۱- مقدمه

در سال‌های اخیر معماری‌های میکروسرویس[1] به عنوان الگوی غالب و نوین در طراحی سامانه‌های ابری، توانسته‌اند با ارائه مزایایی مانند انعطاف‌پذیری، مقیاس‌پذیری و تفکیک وظایف، توسعه و نگهداری سیستم‌های بزرگ و پیچیده را ساده‌تر کنند. درواقع، اخیراً معماری میکروسرویس به عنوان یک پارادایم کلیدی در توسعه نرم‌افزارهای مقیاس‌پذیر و قابل نگهداری ظهور کرده است. این رویکرد با تقسیم برنامه به اجزای مستقل و کوچک‌تر که هر یک وظیفه خاصی را بر عهده دارند، امکان توسعه تدریجی، استقرار مستقل و مقیاس‌پذیری دینامیک را فراهم می‌کند. معماری میکروسرویس رویکردی نوین در طراحی نرم‌افزار است که در آن یک سیستم بزرگ به مجموعه‌ای از سرویس‌های کوچک، مستقل و خودکفا تقسیم می‌شود. هر میکروسرویس تنها بر یک وظیفه یا حوزه تجاری خاص متمرکز است، پایگاه داده و منطق تجاری خود را دارد و از طریق رابط‌های برنامه‌نویسی کاربردی یا پیام‌رسان‌ها با سایر سرویس‌ها ارتباط برقرار می‌کند. این ساختار امکان توسعه، نگهداری، استقرار و مقیاس‌پذیری مستقل هر سرویس را فراهم می‌کند بدون اینکه بر کل سیستم تأثیر بگذارد [۱]. با این حال، این مزایا در معماری میکروسرویس با چالش‌ها و پیچیدگی‌های قابل‌توجهی نیز همراه هستند، از جمله مهم‌ترین این چالش‌ها حفظ انسجام وضعیت، هماهنگی بین سرویس‌ها و تحمل خطا در زمان خرابی گره‌ها است [۲].

برای غلبه بر چالش‌های معماری میکروسرویس، اخیراً استفاده از الگوریتم‌های اجماع[2] در سیستم‌های توزیع‌شده به‌طور گسترده مورد توجه قرار گرفته است. الگوریتم‌های اجماع در معماری میکروسرویس، ابزارهایی حیاتی برای هماهنگی و توافق بین سرویس‌های توزیع‌شده هستند. این الگوریتم‌ها تضمین می‌کنند که سرویس‌های مستقل در یک سیستم توزیع‌شده، حتی در صورت بروز خطاهای شبکه یا خرابی، بتوانند روی وضعیت یکپارچه سیستم به توافق برسند. الگوریتم Paxos، که یکی از پایه‌ای‌ترین روش‌های اجماع محسوب می‌شود، سازوکاری فراهم می‌کند که گره‌های مختلف بتوانند حتی در صورت وجود خطا، بر سر یک وضعیت مشترک به توافق برسند [۳]. این الگوریتم، پایه بسیاری از سیستم‌های توزیع‌شده صنعتی مانندGoogle Chubby و Azure Cosmos DB بوده است [٤]. در سال‌های اخیر، نسخه‌های بهینه‌شده‌ای از Paxos همچونFlexible Paxos و-Multi Paxosارائه شده‌اند که با کاهش پیچیدگی ارتباطات، تأخیر را کاهش و مقیاس‌پذیری را افزایش داده‌اند. به‌ویژه در محیط‌های میکروسرویس که بار پویا، نیاز به تغییر مقیاس خودکار و کنترل وضعیت توزیع‌شده وجود دارد، این الگوریتم‌ها می‌توانند نقشی کلیدی در حفظ پایداری سیستم ایفا کنند [٦-٥]. اگرچه همچنان نیازمندی هائی مانند سازگاری با نسخه‌های موجود، هوشمندی و مقیاس پذیری در تخصیص و مدیریت منابع در محیط‌های پویای ابری وجود دارد. لذا براساس ضرورت، مقاله حاضر یک دیدگاه جدید و رویکرد نوآورانه برای ترکیب الگوریتم Paxos با مکانیزم‌های مدیریت و تخصیص هوشمند منابع و مقیاس‌گذاری خودکار در سیستم‌های میکروسرویس ابری پویا ارائه می‌دهد. هدف از این دیدگاه پیشنهادی، کاهش تأخیر، تضمین سازگاری بین سرویس‌ها و بهبود تحمل‌پذیری خطا در زیرساخت‌های ابری پویا است. با بهره‌گیری از تحلیل‌های تجربی و شبیه‌سازی‌شده ثابت شد که ترکیب الگوریتم‌های اجماع با چارچوب بهینه‌سازی هوشمند پیشنهادی می‌تواند عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس را به‌طور معناداری ارتقا دهد. لذا، اهداف اصلی پژوهش حاضر به‌صورت زیر قابل بیان هستند.

دیدگاه نوین بهبود تخصیص و مدیریت منابع و حفظ انسجام در سیستم‌های میکروسرویس با رویکرد بهینه سازی هوشمند اجماع.

ترکیب الگوریتم‌های اجماع و بخصوص الگوریتم مبنای Paxos با مکانیزم‌های تخصیص هوشمند منابع و مقیاس‌گذاری خودکار در محیط‌های ابری با بارکاری پویا.

حفظ دسترس پذیری، افزایش قابلیت اطمینان، بهبود کارایی و افزایش استحکام در سیستم‌های میکروسرویس، حتی در مواجهه با خرابی گره‌ها و تغییرات بار کاری پویا.

در ادامه مقاله، ابتدا پژوهش‌های مرتبط مهم انجام‌شده مرور می‌شوند. سپس، با تشریح ابعاد رویکرد پیشنهادی و ارائه جزئیات

[1] Microservice Architecture

[2] Consensus Algorithms

طراحی و الگوریتم‌های به‌کاررفته، مراحل پیاده سازی دیدگاه پیشنهادی مورد بررسی قرار می‌گیرد. در نهایت، نتایج به‌دست‌آمده جمع‌بندی و مقایسه شده و پیشنهادهایی برای توسعه بکارگیری و بهبود عملکرد معماری میکروسرویس ارائه خواهند شد.

۲- تحقیقات مرتبط

اخیراً بهینه‌سازی سیستم‌های میکروسرویس به طور گسترده‌ای مورد مطالعه قرار گرفته است و روش‌های مختلفی برای بهبود عملکرد، مقیاس‌پذیری و قابلیت اطمینان پیشنهاد شده است. 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. جدول (۱) تحقیقات مرتبط و جنبه‌های کلیدی رویکردهای بکارگرفته شده را خلاصه می‌نماید. در ادامه این بخش این رویکردها را با جنبه‌های اصلی دیدگاه پیشنهادی این پژوهش مقایسه می‌نمائیم.

جدول (۱): مقایسه جنبه‌های کلیدی تحقیقات مرتبط

منبع	تکنیک	مزایا	چالش‌ها
[۱]	مدل‌سازی معماری میکروسرویس	ارائه چارچوب مفهومی برای طراحی میکروسرویس	نیاز به اعتبارسنجی تجربی بیشتر
[۲]	تحلیل تأثیر تغییرات در میکروسرویس	شناسایی چالش‌های تغییر در سیستم‌ها	نیاز به توسعه ابزارهای پشتیبان
[۳]	Flexible Paxos	کاهش نیاز به رأی اکثریت در هر مرحله، افزایش کارایی	پیچیدگی پیاده‌سازی و درک الگوریتم
[٤]	تجزیه و تحلیل الگوریتم Paxos	ساده‌سازی درک الگوریتمPaxos	نیاز به پیاده‌سازی عملی برای اعتبارسنجی
[٥]	معماری میکروسرویس پویا	افزایش مقیاس‌پذیری و تحمل خطا	پیچیدگی در مدیریت سرویس‌های پویا
[٦]	چارچوب‌های پردازش جریان در میکروسرویس‌ها	ارزیابی عملکرد در شرایط مختلف بار	محدودیت در تعمیم نتایج به سایر چارچوب‌ها
[۷]	Multi-Paxosبهینه‌شده	بهبود عملکرد سیستم‌های ذخیره‌سازی ابری	عدم انطباق‌پذیری و نیاز به تنظیمات دقیق و ثابت برای محیط‌های مختلف
[۸]	تحلیل سیستم‌های مبتنی برPaxos	شفاف‌سازی مفاهیم پیچیدهPaxos	تمرکز بر جنبه‌های تئوری
[۹]	تکنیک‌های بهینه‌سازی استقرار	مرور رویکردهای موجود	کمبود بررسی‌های تجربی عمیق
[۱۰]	بهبود پروتکل‌های اجماع ناهمزمان	کاهش تأخیر در عملیات خواندن	ناسازگاری با نسخه‌های موجود

بررسی و مرور مقالات فوق نشان می‌دهد که اخیراً تلاش‌های متعددی برای بهبود عملکرد، مقیاس‌پذیری و پایداری در سیستم‌های میکروسرویس از طریق الگوریتم‌های اجماع مانند Paxos صورت گرفته است. با این حال، بسیاری از این مطالعات مانند [۹ و ۸ و ٤] یا بر جنبه‌های تئوری تمرکز داشته‌اند یا در محیط‌های خاصی مورد ارزیابی قرار گرفته‌اند. به‌طور خاص، ترکیب الگوریتم‌های اجماع با مکانیزم‌های تخصیص منابع و مقیاس‌گذاری خودکار در محیط‌های ابری پویا کمتر مورد توجه قرار گرفته است. این شکاف تحقیقاتی نشان‌دهنده نیاز به توسعه رویکردهایی است که بتوانند الگوریتم‌های اجماع را به‌طور مؤثر در معماری‌های میکروسرویس ادغام کرده و به بهبود هماهنگی، پایداری و عملکرد در شرایط متغیر کمک کنند. تلاش درجهت حل این چالش‌ها در حوزه تمرکز پژوهش حاضر قرار می‌گیرد.

در سیستم‌های توزیع‌شده، الگوریتم‌های اجماع نقش بسیار مهمی در دستیابی به توافق بین گره‌ها دارند. این الگوریتم‌ها به گونه‌ای طراحی شده‌اند که حتی در حضور خرابی‌های شبکه، تاخیرها و خطاهای گره‌ها، تمام گره‌ها بتوانند روی یک تصمیم یا مقدار مشترک توافق کنند. در ادامه به بررسی چهار الگوریتم معروف Paxos، Raft، Zab و PBFT[1] پرداخته می‌شود.

[1] Practical Byzantine Fault Tolerance (PBFT)

- الگوریتم Paxos: این الگوریتم که از مهم‌ترین الگوریتم‌های رسمی برای حل مسئله اجماع است، از مؤلفه تحمل خطای خرابی[1] پشتیبانی می‌کند، یعنی قادر است با خرابی یا خاموشی گره‌ها مقابله کند. با این حال، طراحی این الگوریتم پیچیدگی هائی داشت و درک صحیح از نحوه عملکرد آن دشوار بود [۱۱].
- الگوریتم Raft: این الگوریتم دارای سه جریان کار اصلی اصلی شامل: انتخاب راهبر، تکثیر لاگ و مدیریت وضعیت سیستم است. این الگوریتم نیز از CFT پشتیبانی می‌کند و به همین دلیل در بسیاری از سیستم‌های صنعتی مانند Etcd، Consul و CockroachDB استفاده می‌شود [۱۲].
- الگوریتم Zab[2]: الگوریتم اجماعی است که در Apache ZooKeeper استفاده می‌شود. این الگوریتم از یک لیدر مرکزی استفاده می‌کند و براساس CFT طراحی شده است. اهداف اصلی این الگوریتم تضمین ترتیب خطی، اجرای اتمی عملیات و مقاومت در برابر تغییرات راهبر است [۱۳].
- الگوریتم PBFT : این الگوریتم رویکرد اجماعی است که به جای تحمل خطا در گره‌های خراب (CFT)، قادر به مقابله با خطاهای بیزانسی[3] نیز هست، یعنی زمانی که گره‌ها رفتارهای مخرب یا گمراه‌کننده داشته باشند. این الگوریتم در بلاکچین‌های بدون اثبات کار مانند Hyperledger Fabric استفاده می‌شود [۱۴].

جدول (۲): تجزیه و تحلیل الگوریتم‌های اجماع در سیستم‌های توزیع‌شده

الگوریتم	Paxos	Raft	Zab	PBFT
نوع خطا	Crash Fault Tolerance	Crash Fault Tolerance	Crash Fault Tolerance	Byzantine Fault Tolerance
وجود راهبر	دارد (نقش پویا و غیرمستحکم)	دارد (لیدر دائمی و مشخص)	دارد (لیدر مرکزی)	ندارد
سهولت پیاده‌سازی	پیچیده	ساده	متوسط	بسیار پیچیده
سرعت تصمیم‌گیری	متوسط	بالا	بالا	پایین
تحمّل خطا (N فرآیند)	N/2 + 1	N/2 + 1	N/2 + 1	(N-1)/3 + 1
تعداد پیام‌ها در هر دور	متغیر (حداقل ۲ مرحله)	ثابت (heartbeat)	متغیر	$O(n^2)$
کاربرد عمده	سیستم‌های توزیع‌شده عمومی	Etcd, Consul, CockroachDB	Apache ZooKeeper	Hyperledger Fabric
مدیریت لاگ	خیر	بله (لاگ مرتب‌سازی‌شده)	بله (پخش اتمی)	خیر
استحکام در برابر دستکاری	خیر	خیر	خیر	بله
مقیاس‌پذیری	متوسط	بالا	بالا	پایین

براساس جمع‌بندی اطلاعات جدول بالا و با توجه به نیاز روزافزون سیستم‌های توزیع‌شده به یک الگوریتم اجماع قوی، قابل اعتماد و تحمل‌پذیر به خطا، الگوریتم Paxos به عنوان یکی از معروف‌ترین و قدرتمندترین راه‌حل‌ها در این حوزه مطرح است[۱۱]. این الگوریتم به لطف خواص اثبات‌شده‌اش در دستیابی به توافق در حضور خطاهای نوع CFT، در بسیاری از سیستم‌های صنعتی و پژوهشی کاربرد یافته است. با این حال، پیچیدگی بالای آن در درک و پیاده‌سازی باعث شده است که استفاده از آن در عمل با

[1] Crash Fault Tolerance (CFT)
[2] ZooKeeper Atomic Broadcast
[3] Byzantine Faults

چالش‌هایی همراه باشد. اخیراً رویکردهای مختلفی از جمله طراحی مجدد ماژولار، افزودن قابلیت‌های پویا، و ترکیب آن با سایر الگوریتم‌ها مانند Raft به منظور افزایش قابلیت خوانایی و کارایی در محیط‌های واقعی مورد بررسی قرار گرفته است [۱۵]. چالش‌های موجود ما را به سمت بهینه‌سازی و بهبود Paxos سوق داده است، به‌گونه‌ای که بتوانیم از قدرت و قابلیت اطمینان ذاتی آن بهره ببریم، در حالی که موانع موجود در پیاده‌سازی و مدیریت آن را کاهش دهیم.

یک چارچوب مرتبط ارائه شده در تحقیقات قبلی توسط Pinheiro و همکاران، به چالش‌های تخصیص منابع و خودکارسازی مقیاس‌پذیری در محیط‌های میکروسرویس می‌پردازد. این چارچوب از شبکه‌های پتری تصادفی[1]، الگوریتم ژنتیک مرتب‌سازی غیرمغلوب (NSGA-II) و رگرسیون جنگل تصادفی[2] رأی مدل‌سازی و بهینه‌سازی میکروسرویس‌ها استفاده می‌کند. با شناسایی توازن‌های بحرانی بین عملکرد و مصرف منابع، این چارچوب بینش‌های ارزشمندی در مورد میکروسرویس‌ها تحت پیکربندی‌های مختلف ارائه می‌دهد. استفاده از SPNs امکان مدل‌سازی رفتار دینامیکی میکروسرویس‌ها را فراهم می‌کند، در حالی که NSGA-II و RFR بهینه‌سازی و پیش‌بینی عملکرد را تسهیل می‌کنند [۱۶]. اگرچه، در حالی که چارچوب پیشنهادی Pinheiro و همکارانش بر معیارهای عملکردی مانند توان عملیاتی و زمان پاسخ تمرکز دارد، به چالش‌های سازگاری و تحمل خطا نمی‌پردازد.

رویکرد پیشنهادی ما از الگوریتم‌های اجماع Paxos برای پرکردن شکاف تحقیقاتی موجود در تحقیقات قبلی 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.

لازم به ذکر است، نوآوری اصلی این مقاله در ترکیب هوشمند مکانیزم‌های اجماع با انتخاب رهبر پویا، تنظیم پویای حد نصاب[3] و بهینه‌سازی لاگ و براساس توسعه الگوریتم Paxos طراحی شده و با هدف ارتقای کارایی و تحمل خطا است که تاکنون در پژوهش‌های قبلی به صورت یکپارچه و در سیستم‌های میکروسرویس پویا به شکل جامع اجرا نشده است. این رویکرد به صورت یک چارچوب بهینه‌سازی عملکرد و قابلیت اطمینان در محیط‌های ابری همراه با مدیریت منابع ارائه شده که بر اساس نتایج ارزیابی عملیاتی و تجربی اثبات و ارائه شده است. دیدگاه پیشنهادی ضمن حفظ ویژگی‌های کلیدی Paxos، با تلفیق مکانیزم‌های بهینه‌سازی هوشمند منابع و مقیاس‌گذاری خودکار، تغییر وضعیت سریع و پویای خدمات در محیط‌های ابری با بار متغیر را امکان‌پذیر می‌سازد؛ چیزی که در نسخه‌های پیشین بصورت ثابت، محدود و موردی مورد توجه بوده است. این نوآوری در همگرایی چندبعدی اجماع و تخصیص منابع در محیطهای پویا، به عنوان یک افزوده علمی نوین و مؤثر قابل ارائه است. در حالی که چارچوب‌ها و تکنیک‌های بهینه‌سازی موجود بینش‌های ارزشمندی در مورد عملکرد و مقیاس‌پذیری سیستم‌های میکروسرویس ارائه می‌دهند، رویکرد پیشنهادی ما به طور منحصر به فردی این جنبه‌ها را با هدف استحکام پذیری الگوریتم‌های اجماع Paxos ترکیب می‌نماید. این ادغام با هدف ارائه یک راه‌حل جامع برای بهینه‌سازی سیستم‌های میکروسرویس در محیط‌های ابری دینامیک صورت می‌پذیرد. چنین رویکردی علاوه بر دستاوردهای ذکر شده قادر به ایجاد پل ارتباطی بین پژوهش‌های پیشین درجهت مدیریت هوشمند و موثرتر منابع در سیستم‌های میکروسرویس است.

۳- دیدگاه پیشنهادی پژوهش

در این بخش، دیدگاه پیشنهادی پژوهش و الگوریتم بهینه‌سازی را که با بهره گیری از الگوریتم اجماع Paxos به منظور بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس طراحی شده است، ارائه می‌دهیم. رویکرد ما با چارچوبی که در تحقیقات قبلی توضیح داده شده است، مقایسه می‌شود. همانطور که در قسمت تحقیقات مرتبط ذکر شد، رویکردهای پیشنهادی در مقالاتی مانند [۱۵–۱۷ و

[1] Stochastic Petri Nets (SPNs)
[2] Random Forest Regression (RFR)
[3] Quorum Tuning

```
Algorithm OPaMS(nodes, proposal)
Begin
    // Phase 1: Intelligent Leader Election
    function elect_leader(nodes)
        heartbeats ← empty dictionary
        for each node in nodes do
            heartbeats[node] ← node.send_heartbeat()
        end for
        leader ← node with maximum
(heartbeats[node].bandwidth / heartbeats[node].latency)
        return leader
    end function
    // Phase 2: Parallel Acceptance
    function parallel_acceptance(leader, proposal, nodes)
        responses ← empty dictionary
        for each node in nodes do
            responses[node] ←
node.accept_proposal(proposal)
        end for
        accepted_count ← count of responses with value
True
        if accepted_count ≥ floor(length(nodes)/2) + 1 then
            leader.commit(proposal)
            return True
        else
            return False
        end if
    end function
    // Phase 3: Optimized Log Replication
    function optimize_log(log)
        compacted_log ← compact_log(log)
        compressed_log ← compress_log(compacted_log)
        snapshot ← take_snapshot(compressed_log)
        return snapshot
    end function
    // Phase 4: Automatic Quorum Tuning
    function adjust_quorum(nodes, network_status)
        if network_status = "normal" then
            quorum ← floor(length(nodes)/2) + 1
        else if network_status = "partition" then
            quorum ← floor(length(nodes) * 2 / 3)
        end if
        return quorum
    end function
    // Phase 5: Cache-Based Read Optimization and
Reconfiguration
    function read_optimization(cache, log)
        if cache.is_valid() then
            return cache.read()
        else
            return log.read()
        end if
    end function
    // Main execution process
    leader ← elect_leader(nodes)
    accepted ← parallel_acceptance(leader, proposal,
nodes)
    if accepted then
        snapshot ← optimize_log(log)
        quorum ← adjust_quorum(nodes, network_status)
        result ← read_optimization(cache, log)
        return result
    else
        print("Proposal rejected!")
        return None
    end if
End Algorithm
```

٤] از تکنیک هائی مانند شبکه‌های پتری تصادفی (SPNs)، الگوریتم ژنتیک مرتب‌سازی غیرمسلط (NSGA-II) و رگرسیون جنگل تصادفی (RFR) برای بهینه‌سازی عملکرد استفاده می‌کنند و بیشتر بر جنبه‌های تئوری بهینه سازی تمرکز دارند، لذا دارای محدودیت‌های اجرائی در محیط پیاده سازی می‌باشند. الگوریتم پیشنهادی ما مکانیزم‌های پویا و هوشمند اجماع را برای اطمینان از سازگاری و تحمل خطا در حین بهینه‌سازی تخصیص منابع و مقیاس‌بندی بصورت مؤثر ادغام می‌کند. اساساً برای بهینه‌سازی در الگوریتم OPaMS[1] که برای سیستم‌های میکروسرویس و معماری ابری پویا طراحی شده است، باید به دو جنبه اصلی توجه کنیم [۱۸–۱۹].

- بهبود عملکرد: کاهش زمان اجرا و تعداد پیام‌ها.
- افزایش قابلیت اطمینان: مقابله با خرابی و تقسیم شبکه.

لازم به توضیح است، دیدگاه پیشنهادی دارای پنج فاز اجرایی است که هر فاز از تعدادی جریان کاری مرتبط تشکیل شده است. فازها و جریان‌های کاری مرتبط بصورت پیوسته و درقالب یک الگوریتم بهینه سازی هوشمند اجرا می‌گردند. در فاز اول انتخاب لیدر مناسب بر اساس وضعیت فعلی شبکه و گره‌ها بصورت هوشمند صورت می‌پذیرد. در فاز دوم توسط جریان کاری ثبت موازی پذیرش لیدر با توجه به پارامترهای کارائی سیستم یعنی کاهش زمان اجرا و تأخیرات شبکه صورت می‌پذیرد. در فاز سوم نیز لاگ سیستم، با هدف کاهش حجم لاگ و افزایش سرعت بازیابی، بهینه سازی می‌گردد. سپس در فاز چهارم براساس شرایط فعلی شبکه حدنصاب پویایی فعال می‌شود تا اطمینان از ادامه توافق فراهم شود. نهایتاً نیز در آخرین و پنجمین فاز اجرایی کش بمنظور کاهش نیاز به ارسال پیام‌ها برای خواندن داده‌ها، توسط لاگ سیستم بروزرسانی می‌گردد و منابع سیستمی بصورت پویا پیکربندی مجدد می‌شوند. براساس اهداف دیدگاه پیشنهادی، فازهای اجرایی و کلیات عملکردی الگوریتم بهینه سازی هوشمند پیشنهادی به شرح زیر است:

[1] Optimized Paxos for Microservices (OPaMS)

توصیف فازهای اجرایی الگوریتم هوشمند پیشنهادی OPaMS و عملکرد جریان‌های کاری در هر فاز به شرح زیر می‌باشند:

فاز ۱: انتخاب رهبر هوشمندانه[1]: با هدف انتخاب مناسب‌ترین هدایتگر بر اساس وضعیت فعلی شبکه، بارکاری و گره‌ها.

– هر گره در ابتدا به عنوان یک رهبر کاندید است.

– گره‌ها از طریق ارسال پیام‌های بازخورد Heartbeat به یکدیگر وضعیت فعلی خود را اعلام می‌کنند.

– گره‌هایی که بهترین پهنای باند شبکه و کمترین زمان پاسخ‌دهی دارند، به عنوان رهبر انتخاب می‌شوند [۲۰].

– اگر رهبر قبلی خراب شده باشد، گره‌ها با ارسال پیام‌های Vote تصمیم می‌گیرند که کدام گره به عنوان لیدر جدید انتخاب شود.

به عنوان مثال فرض کنید ۵ گره شامل گره‌های A, B, C, D, E وجود دارد و گره A به عنوان لیدر انتخاب شده است. اگر گره A خراب شود، گره‌ها از طریق ارسال پیام‌های Vote تصمیم می‌گیرند که کدام گره به عنوان لیدر جدید انتخاب می‌شود. انتخاب گره‌ای به عنوان لیدر که بهترین پهنای باند شبکه و کمترین تأخیر را دارد، باعث می‌گردد تصمیم‌گیری‌ها سریع‌تر و با کیفیت بهتری انجام شوند. این روش به کاهش تأخیر سیستم کمک کرده و بهبود استفاده را از منابع شبکه به عمل می‌آورد، که یک نقطه ضعف معمول در سامانه‌های توزیع‌شده بوده است.

فاز ۲: ثبت موازی[2]: این فاز با هدف کاهش زمان اجرا و تأخیرات شبکه و بهبود پاسخ‌دهی بصورت زیر اجرا می‌گردد:

– رهبر (proposal) را به تمام گره‌ها ارسال می‌کند.

– گره‌ها به صورت موازی پاسخ می‌دهند و تصمیم می‌گیرند که آیا پیشنهاد را قبول می‌کنند یا خیر.

– اگر حداقل N/2+1 گره پیشنهاد را قبول کردند، لیدر تصمیم را به تمام گره‌ها اعلام می‌کند.

در مثال ما رهبر A پیشنهادی را به گره‌های B, C, D, E ارسال می‌کند و گره‌ها به صورت موازی پاسخ می‌دهند. اگر ۳ گره پیشنهاد را قبول کردند، رهبر تصمیم را به گره‌ها اعلام می‌کند. پذیرش موازی پیشنهادها توسط تمام نودها باعث کاهش زمان پاسخ و افزایش توان عملیاتی می‌شود، زیرا منتظر تکمیل مراحل متوالی نیستیم. این رویکرد باعث بهبود مقیاس‌پذیری و افزایش تحمل خطا در شرایط بار کاری بالا می‌شود.

فاز ۳: بهینه سازی تکرار لاگ[3]: با هدف کاهش حجم لاگ و افزایش سرعت بازیابی به شرح زیر اجرا می‌گردد:

– استفاده از Log Compaction برای حذف داده‌های قدیمی از لاگ و کاهش حجم.

– استفاده از الگوریتم‌های فشرده‌سازی مثل Gzip یا Snappy برای کاهش اندازه پیام‌ها [۲۲ و ۲۱].

– ذخیره آخرین وضعیت لاگ[4] به صورت دوره‌ای برای افزایش سرعت بازیابی در صورت خرابی گره.

در مثال موردنظر بعد از ۱۰۰۰ عملیات، لاگ به صورت فشرده‌سازی شده ذخیره می‌شود و اگر گره خراب شود، فقط آخرین Snapshot و تغییرات بعد از آن باید از گره‌های دیگر دریافت شود. اجرای فشرده‌سازی و کمپکت کردن لاگ‌ها و گرفتن Snapshot از وضعیت فعلی، فضای ذخیره‌سازی را کاهش داده و سرعت بازسازی وضعیت پس از خرابی را افزایش می‌دهد. این کار باعث کاهش هزینه‌های نگهداری و بهبود استحکام سامانه می‌گردد.

فاز ٤: تنظیم حد نصاب اتوماتیک[5]: با هدف تعیین پویا و بهینه حدنصاب بر اساس شرایط شبکه، بصورت زیر اجرا می‌گردد:

– در شرایط عادی، N/2+1 گره برای توافق لازم است.

– در صورت تشخیص مشکل یا تقسیم شبکه، تنظیم پویای حدنصاب فعال می‌شود تا اطمینان از ادامه توافق فراهم شود.

در این فاز متناسب با وضعیت شبکه (عادی یا تقسیم شده)، حد نصاب توافق به صورت پویا تنظیم می‌شود که هم در شرایط عادی و هم هنگام بروز مشکلات تقسیم شبکه، سیستم بتواند عملیات اجماع را پایدار و مؤثر ادامه دهد.

فاز ۵: بهینه‌سازی و پیکربندی آنلاین: در این فاز مدیریت پویای منابع مبتنی بر حافظه پنهان[6] با هدف کاهش نیاز به ارسال پیام‌ها برای خواندن داده‌ها به شرح زیر انجام می‌شود:

[1] Intelligent Leader Election
[2] Parallel Acceptance
[3] Optimized Log Replication
[4] Snapshotting
[5] Quorum Tuning
[6] Cache Based Optimization

– استفاده از دستور Read Cache برای ذخیره آخرین وضعیت داده‌ها.
– اگر خواندن داده ضروری باشد، از Cache استفاده می‌شود.
– Cache به صورت دوره‌ای با لاگ به‌روزرسانی می‌شود.
– بازپیکربندی پویا و آنلاین منابع با امکان اضافه/حذف گره‌ها و تقسیم مجدد داده‌ها بدون اختلال در انسجام یا عملکرد سیستم [۲۳].

به عنوان مثال چنانچه گره A بخواهد داده X را بخواند، اگر X در Cache موجود است، از Cache استفاده می‌شود ولی اگر X در Cache وجود ندارد، از لاگ گره‌های دیگر دریافت می‌شود. در این فاز، استفاده هوشمندانه از حافظه Cache برای پاسخ‌گویی به درخواست‌های خواندن، نیاز به ارسال پیام‌ها و دسترسی به لاگ‌ها را کاهش می‌دهد، که منجر به کاهش تأخیر و افزایش کارایی سیستم می‌شود. همچنین، بازپیکربندی پویا منابع بر اساس این اطلاعات باعث حفظ ثبات و هماهنگی در سیستم‌های میکروسرویس پیچیده می‌گردد.

لازم به ذکر است اکثر دیداههای مرتبط ارائه شده در تحقیقات قبلی مانند [۱۵–۱۷ و ٤] بر بهینه‌سازی معیارهای عملکردی مانند توان عملیاتی و زمان پاسخ با استفاده از SPNs، NSGA-II و RFR تمرکز دارند. در حالی که این چارچوب‌ها بینش‌های ارزشمندی در مورد تخصیص منابع و خودکارسازی مقیاس‌پذیری ارائه می‌دهند، به چالش‌های مربوط به سازگاری و تحمل خطا به طور صریح نمی‌پردازند. در مقابل الگوریتم پیشنهادی ما با ادغام مکانیزم‌های اجماع Paxos، اطمینان حاصل می‌کند که سیستم‌های میکروسرویس حتی در مواجهه با خرابی‌های نود و تغییرات پویا، از دسترسی بالا و قابلیت اطمینان برخوردار باشند. با ترکیب بهینه‌سازی عملکرد با پروتکل‌های اجماع، رویکرد ما راه‌حلی چندمنظوره و هوشمند برای بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس در محیط‌های ابری پویا ارائه می‌دهد.

٤– ارزیابی دیدگاه پیشنهادی

بمنظور ارزیابی دیدگاه پیشنهادی و سنجش اثربخشی الگوریتم بهینه‌سازی هوشمند مبتنی بر Paxos، مجموعه‌ای از آزمایش‌های تجربی 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 و مبتنی بر تحقیقات قبلی انجام شده [۲٤–۲٦] 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. 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.

بمنظور ارزیابی، کاربرد مبتنی بر میکروسرویس را در چندین گره توزیع شده پیاده‌سازی کردیم که هر کدام میزبان انواع مختلفی از میکروسرویس‌ها با عملکردهای خاص بودند. برای مدیریت تراکنش‌های توزیع شده و حفظ سازگاری داده‌ها، از الگوریتم اجماع Paxos استفاده کردیم. تنظیمات آزمایشی ما به دقت برای شبیه‌سازی شرایط دنیای واقعی، از جمله تقسیمات شبکه و خرابی گره‌ها طراحی شده است. از پنج سرور به عنوان میزبان‌های فیزیکی استفاده کردیم. تمام میکروسرویس‌ها به عنوان خدمات وب RESTful با استفاده از چارچوب Java Spring توسعه داده شدند. کد نوشته شده به زبان جاوا در دسترس است و زمان‌های اجرای آن با استفاده از Apache JMeter 5.6.3[1] اندازه‌گیری شد. در طول آزمایش‌ها، تنظیمات فنی و تنظیمات زمان اجرا را به طور مداوم حفظ کردیم. مشخصات دقیق تنظیمات آزمایشی در جدول ۳ ارائه شده است.

[1] https://jmeter.apache.org/download_jmeter.cgi

جدول (۳): تنظیمات سرورهای آزمایشی

تنظیمات	نام میزبان	مشخصات سخت افزاری	محیط اجرا	شبکه
تنظیمات‌هاست فیزیکی	هاست فیزیکی ۱ هاست فیزیکی ۲ هاست فیزیکی ۳	Intel Core i7, 8 Cores 16.0 GB RAM	Java v19	100 Mbps LAN

لازم به ذکر است، مهم‌ترین ویژگی‌های عملیاتی و جزئیات پیاده سازی و ارزیابی الگوریتم پیشنهادی به شرح زیر است:

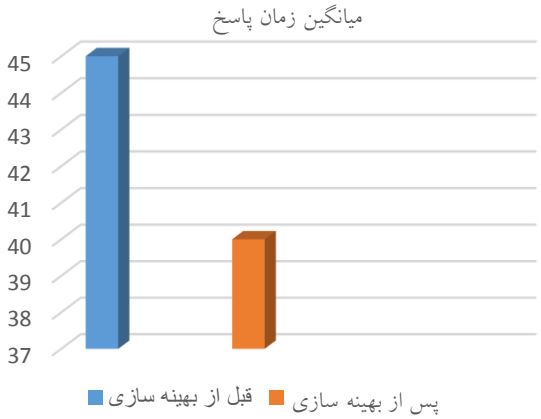
- توزیع تأخیر شبکه: در آزمایش‌ها، برای شبیه‌سازی شرایط مختلف شبکه، تأخیرهای متنوعی در نظر گرفته شده است. این تأخیرها از مدل‌های مختلف مانند تأخیر ثابت و تأخیر متغیر استفاده کرده‌اند. برای شبیه‌سازی تأخیر متغیر، تأخیر شبکه به‌صورت تصادفی در بازه‌های زمانی مشخص (مثلاً ۱۰ میلی‌ثانیه تا ۱۰۰ میلی‌ثانیه) توزیع شده است تا سناریوهای مختلف تأخیر شبکه مانند اختلالات یا کندی ارتباطات در محیط‌های ابری شبیه‌سازی شوند. این تنظیمات تأثیرات تغییرات شبکه بر عملکرد سیستم‌های میکروسرویس را به‌طور دقیق شبیه‌سازی می‌کنند.
- الگوی تولید بار: برای شبیه‌سازی بار کاری، از تولیدکننده بار استفاده شده که درخواست‌های مختلفی را به سیستم ارسال می‌کند. این درخواست‌ها شامل عملیات خواندن و نوشتن در انواع مختلف و با نرخ‌های مختلف بودند. به‌طور خاص، الگوی تولید بار به‌صورت پویا طراحی شده است که در آن بار در زمان‌های مختلف دچار تغییرات می‌شود (مثلاً بارهای اوج و پایین). الگوهای تولید بار تصادفی تولید می‌شدند تا شرایط مختلف کاری را شبیه‌سازی کنند. این فرآیند به‌طور خاص به شبیه‌سازی سناریوهای مختلف مانند بار کاری ثابت، نوسان در بار کاری و تغییرات دینامیکی در حجم درخواست‌ها کمک می‌کند.
- سناریوهای تزریق خطا: برای ارزیابی تحمل خطای سیستم، سناریوهای مختلف تزریق خطا به شرح زیر در شبیه‌سازی‌ها اعمال شدند.
- خرابی گره‌ها: در این سناریوها، گره‌های مختلف در زمان‌های تصادفی از سیستم خارج شدند تا عملکرد سیستم در شرایط خرابی شبیه‌سازی شود.
- تقسیم شبکه (Network Partitioning) :شبیه‌سازی سناریوهای تقسیم شبکه به‌طور عمدی باعث شد تا برخی از گره‌ها ارتباط خود را با بقیه گره‌ها از دست دهند و تأثیر آن بر عملکرد سیستم بررسی شود.
- تزریق تأخیر: برای شبیه‌سازی تغییرات دینامیک در تأخیر شبکه، تأخیر اضافی در ارتباطات گره‌ها تزریق شد تا رفتار سیستم در شرایط مختلف بار و شبکه با تأخیر بالا ارزیابی شود.

تمامی سناریوها در محیط آزمایشی به‌طور سیستماتیک برای ارزیابی عملکرد سیستم میکروسرویس و الگوریتم اجماع Paxos به کار گرفته شده‌اند. این آزمایش‌ها به‌ویژه برای بررسی سازگاری و تحمل خطای سیستم در شرایط واقعی طراحی شده‌اند. در بخش ارزیابی الگوریتم پیشنهادی را به لحاظ میانگین تأثیر با چارچوب‌های ارائه شده در تحقیقات مرتبط [۱۵–۱۷ و ٤] و بخصوص چارچوب مبنا در [۱۵] که از SPNs، NSGA-II و RFR برای بهینه‌سازی عملکرد استفاده می‌کند، مقایسه کردیم. معیارهای عملکردی ارزیابی شده معیارهای بنیادین سنجش و ارزیابی کارکرد سیستم‌های میکروسرویس بوده و شامل موارد زیر می‌باشند [۲٦–۲۷].

- زمان پاسخ: 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.

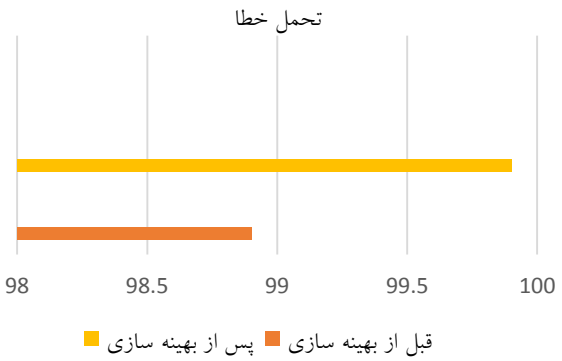
- استفاده از منابع: به عنوان درصد استفاده از منابع CPU، حافظه و شبکه توسط نمونه‌های میکروسرویس اندازه‌گیری شد [۲۸].
- سازگاری: به عنوان درصد تغییرات وضعیت موفق که به طور یکپارچه در تمام نودها اعمال شده‌اند، اندازه‌گیری شد.
- تحمل خطا: به عنوان توانایی سیستم در حفظ عملیات در طول خرابی نودها اندازه‌گیری شد.

شکل (۱) ارزیابی میانگین زمان پاسخ سیستم را نمایش می‌دهد.



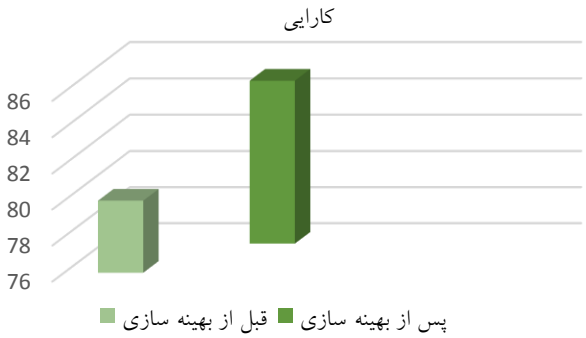
شکل (۱): ارزیابی معیار میانگین زمان پاسخ

همانطور که در نمودار بالا مشهود است میانگین زمان پاسخ سیستم پس از بکارگیری رویکرد هوشمند پیشنهادی از ٤٥ میلی ثانیه به مقدار ٤٠ ثانیه و به میزان ٪۲۵ بهبود یافته است. اگر در بازه‌ای ۱۰۰۰ درخواست ثبت شود و مجموع زمان‌های پاسخ آنها ٤۰۰۰۰ میلی‌ثانیه باشد، میانگین زمان پاسخ برابر ٤۰ میلی ثانیه خواهد بود. کاهش این مقدار نشان‌دهنده بهبود کارایی و پاسخگویی سریع‌تر است. همچنین در ارزیابی دیگر میزان تحمل خطای سیستم را قبل و بعد از بکارگیری الگوریتم پیشنهادی آزمودیم.



شکل (۲): ارزیابی معیار تحمل خطا

در شکل (۲) ارزیابی تحمل خطای سیستم نشان می‌دهد که تحمل خطای سیستم از مقدار ۹۸,۹ به مقدار ۹۹,۹ افزایش یافته است. از ۱۰،۰۰۰ عملیات توزیع‌شده ۹،۸۹۰ مورد موفق بدون خطا بوده، پس از بهینه‌سازی این تعداد به ۹،۹۹۰ عملیات موفق افزایش یافته است، که باعث افزایش تاب‌آوری و پایداری سیستم در مواجهه با خرابی‌های احتمالی می‌شود. شکل (۳) کارائی کلی سیستم پیشنهادی را نمایش می‌دهد.



شکل (۳): ارزیابی کارائی سراسری سیستم

از معادل ۱۰۰ واحد منبع کلی مصرف شده (CPU, حافظه, پهنای باند)، ۸۵ واحد به صورت مفید برای پردازش درخواست‌ها به کار رفته است و مطابق نتایج ارزیابی‌ها کارائی سراسری سیستم پس از بهینه سازی توسط الگوریتم پیشنهادی به مقدار ٪۸۵ افزایش می‌یابد. این یعنی استفاده بهینه‌تر از منابع ابری نسبت به حالت قبل.

همچنین جدول (۳) نتایج کلی ارزیابی دیدگاه پیشنهادی را براساس معیارهای آزمون نمایش می‌دهد.

جدول (٤): نتایج ارزیابی دیدگاه پیشنهادی و مقایسه

معیار	پس از بهینه سازی	قبل از بهینه سازی
میانگین زمان پاسخ	**٤٠ms**	**٤٥ms**
میزان تحمل خطا	٪۹۹/۹	٪۹۸/۹
مقیاس‌پذیری	بالا	متوسط
کارایی	٪۸۵	٪۸۰
هزینه	کم	متوسط

لازم به ذکر است، در ارزیابی آزمایش‌ها با ۳۰ تکرار مستقل انجام شده‌اند تا قابلیت تکرارپذیری و استحکام نتایج تأیید شود. براساس تحلیل نتایج ارزیابی، الگوریتم پیشنهادی مبتنی بر Paxos، کاهش قابل توجهی در تأخیر نسبت به حالت قبل از بهینه سازی به دست آورد. میانگین تأخیر به ٪۲۵ کاهش یافت که کارایی مکانیزم اجماع در اطمینان از تغییرات سریع وضعیت را نشان می‌دهد. توان عملیاتی سیستم با الگوریتم پیشنهادی ٪۳۰ افزایش یافت که نشان‌دهنده مدیریت بهتر بارهای کاری حجیم و تخصیص منابع کارآمد است. همچنین، مدیریت و بهره‌وری استفاده از منابع با بکارگیری الگوریتم پیشنهادی به میزان ٪۲۰ بهبود یافته است. شایان ذکر است، الگوریتم مبتنی بر Paxos نرخ سازگاری بالای ٪۹۹,۹ را حفظ کرد و اطمینان حاصل کرد که تغییرات وضعیت به طور یکپارچه در تمام نودها اعمال می‌شوند. نهایتاً سیستم پیشنهادی تحمل خطای قوی را نشان داد و با نرخ موفقیت ۹۵٪ در حفظ عملیات در طول خرابی نودها عمل کرد. این نتایج بهینه سازی‌های قابل توجهی در عملکرد و قابلیت اطمینان دیدگاه پیشنهادی و الگوریتم بهینه‌سازی مبتنی بر Paxos را نشان می‌دهند. کاهش تأخیر و افزایش توان عملیاتی نشان می‌دهند که الگوریتم می‌تواند بارهای بالا را به طور کارآمد مدیریت کند، در حالی که بهبودهای استفاده از منابع توانایی آن در بهینه‌سازی استفاده از منابع ابری را نشان می‌دهند. نرخ‌های بالای سازگاری و تحمل خطا نیز استحکام الگوریتم در حفظ عملیات قابل اعتماد در محیط‌های ابری پویا را تأیید می‌کنند. در بخش بعدی درخصوص ارزیابی‌ها بحث نموده و نتایج را با رویکردهای مرتبط مقایسه می‌نمائیم.

۵- بحث و مقایسه

ارزیابی‌ها نشان می‌دهد که الگوریتم بهینه‌سازی مبتنی بر **Paxos** پیشنهادی به طور قابل توجهی عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس را در مقایسه با تکنیک‌های بهینه‌سازی سنتی بهبود می‌بخشد. این بخش به بررسی پیامدهای یافته‌های ما، مزایای رویکرد پیشنهادی و زمینه‌های بالقوه برای تحقیقات آینده می‌پردازد. لازم به ذکر است، در این پژوهش رویکرد پیشنهادی**OPaMS** شامل پنج فاز اجرایی است که هر کدام به‌طور هدفمند بر یک جنبه از عملکرد سیستم تأثیر می‌گذارند. برای ارزیابی سهم هر فاز، تحلیل اجزا به صورت افزایشی[1] در محیط شبیه‌سازی انجام شد. در این روش، هر فاز به‌صورت تدریجی به سیستم اضافه شد و تأثیر آن بر معیارهای کلیدی عملکرد (زمان پاسخ، توان عملیاتی، کارایی، سازگاری و تحمل خطا) اندازه‌گیری گردید. نتایج ارزیابی و تحلیل به‌شرح زیر است:

[1] Ablation Study

جدول (۵): تحلیل سهم هر فاز اجرایی در بهبود عملکرد دیدگاه پیشنهادی

فاز	هدف اصلی	سهم تقریبی در بهبود کلی	توضیح تأثیر
فاز ۱	کاهش تأخیر در شروع فرآیند اجماع	~۵٪ کاهش تأخیر	با انتخاب لیدر بر اساس وضعیت شبکه و بار گره‌ها، تأخیر اولیه در رسیدن به اجماع کاهش یافت
فاز ۲	افزایش سرعت رسیدن به توافق	~۱۲٪ کاهش تأخیر	با اجرای پذیرش موازی به جای سری، تعداد مراحل ارتباطی کاهش یافت و تأخیر پردازش تراکنش‌ها بهبود یافت
فاز ۳	کاهش حجم داده و افزایش سرعت بازیابی	~۸٪ بهبود کارایی	فشرده‌سازی و کامپکشن لاگ، حجم انتقال داده را کاهش داد و زمان بازیابی پس از خرابی را بهبود بخشید
فاز ٤	افزایش تحمل خطا و انعطاف در تقسیم شبکه	~۱۰٪ بهبود تحمل خطا	با تنظیم پویای حدنصاب، سیستم در شرایط تقسیم شبکه همچنان قادر به ادامه عملیات بود
فاز ۵	کاهش نیاز به ارسال پیام و افزایش کارایی	~۱۵٪ بهبود کارایی و ~۱۰٪ کاهش تأخیر	استفاده از کش برای خواندن داده‌های متداول، تعداد تعاملات شبکه را کاهش داد و بار سیستم را توزیع کرد

همان‌طور که مشاهده می‌شود، بیشترین سهم در بهبود عملکرد کلی به فاز ۲ (پذیرش موازی) و فاز ۵ (بهینه‌سازی مبتنی بر حافظه پنهان) تعلق دارد، که به‌طور ترکیبی بیش از ۲۵٪ از کاهش تأخیر و بهبود کارایی را توجیه می‌کنند. این دو فاز به‌ویژه در محیط‌های پویای ابری با بار کاری متغیر، تأثیر چشمگیری داشتند. با این حال، باید تأکید کرد که اثر کلی سیستم جمعی و هم‌افزا است. به‌عنوان مثال، بدون فاز ٤ (تنظیم حدنصاب)، عملکرد فاز ۲ در شرایط خرابی شبکه کاهش می‌یافت. بنابراین، اگرچه فاز ۵ بیشترین بهبود را ایجاد می‌کند، اما همه فازها نقش ضروری در دستیابی به عملکرد پایدار و قابل اعتماد دارند. براساس تحلیل نتایج بدست آمده، مهم‌ترین دستاوردهای دیدگاه ما و الگوریتم هوشمند پیشنهادی شامل موارد زیر است:

- کاهش تأخیر: الگوریتم مبتنی بر Paxos به طور مؤثری تأخیر را با اطمینان از تغییرات حالت سازگار و تخصیص منابع کارآمد کاهش می‌دهد. این امر به ویژه در محیط‌های ابری پویا که زمان پاسخ سریع حیاتی است، مفید است. کاهش تأخیر تجربه کاربری و پاسخگویی سیستم را بهبود می‌بخشد و آن را برای برنامه‌های بلادرنگ مناسب می‌سازد.
- افزایش توان عملیاتی: مکانیزم‌های تخصیص منابع پویا و مقیاس‌پذیری خودکار، توان عملیاتی سیستم را افزایش می‌دهند و به سیستم اجازه می‌دهند بارهای بالاتری را مدیریت کند. این بهبود برای برنامه‌هایی با بارهای کاری نوسانی حیاتی است و اطمینان می‌دهد که سیستم پیشنهادی می‌تواند به طور کارآمد برای پاسخگویی به تقاضا مقیاس پذیر شود. افزایش توان عملیاتی قابلیت الگوریتم را در بهینه‌سازی استفاده از منابع و حفظ عملکرد بالا تحت شرایط مختلف نشان می‌دهد.
- استفاده بهینه از منابع: مکانیزم حلقه بازخورد در دیدگاه پیشنهادی اطمینان می‌دهد که منابع به طور 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 درجایی که بیشترین نیاز را دارند استفاده می‌شوند و کارایی کلی سیستم را بهبود می‌بخشد.
- سازگاری پیشرفته: استفاده از اجماع هوشمند Paxos اطمینان می‌دهد که تغییرات حالت به طور 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.

- 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 حیاتی است. تحمل خطای نشان داده شده توسط الگوریتم پیشنهادی ما اطمینان می‌دهد که سیستم می‌تواند به سرعت از خرابی‌ها بازیابی شود و تداوم و قابلیت اطمینان خدمات را حفظ کند.

رویکردهای پیشنهادی در مقالاتی مانند [۲۹ و ۱۵–۱۷ و ٤] از تکنیک هائی مانند شبکه‌های پتری تصادفی (SPNs)، الگوریتم ژنتیک مرتب‌سازی غیرمسلط (NSGA-II) و رگرسیون جنگل تصادفی (RFR) برای بهینه‌سازی عملکرد استفاده می‌کنند و بیشتر بر جنبه‌های تئوری تحلیل و بهینه سازی تمرکز دارند، لذا دارای پیچیدگی‌های کارکردی و محدودیت‌های اجرائی در محیط پیاده سازی می‌باشند عموماً بر بهینه‌سازی معیارهای عمومی مانند توان عملیاتی و زمان پاسخ با استفاده از SPNs، NSGA-II و RFR تمرکز دارند. درحالی‌که این چارچوب‌ها بینش‌های ارزشمندی در مورد تخصیص منابع و مقیاس‌پذیری خودکار ارائه می‌دهد، به چالش‌های مربوط به سازگاری و تحمل خطا به‌طور صریح نمی‌پردازند. الگوریتم پیشنهادی ما مکانیزم‌های پویا و هوشمند اجماع را برای اطمینان از سازگاری و تحمل خطا در حین بهینه‌سازی تخصیص منابع و مقیاس‌بندی به‌صورت مؤثر ادغام می‌کند. الگوریتم پیشنهادی ما با ادغام مکانیزم‌های هوشمند اجماع Paxos تضمین می‌کند که سیستم‌های میکروسرویس حتی در حضور خرابی‌های نود و تغییرات پویا در بار کاری، از دسترس‌پذیری و قابلیت اطمینان بالایی برخوردار باشند. با ترکیب بهینه‌سازی عملکرد با پروتکل‌های اجماع قوی، رویکرد ما یک راه‌حل جامع و هوشمند برای بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس در محیط‌های ابری پویا ارائه می‌دهد. همچنین، سایر مزایای بهینه‌سازی مبتنی بر دیدگاه پیشنهادی این پژوهش به شرح زیر است:

- مقاومت: پروتکل Paxos یک مکانیزم مقاوم برای دستیابی به اجماع بین نودهای توزیع‌شده 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.
- مقیاس‌پذیری: مکانیزم‌های تخصیص منابع پویا و مقیاس‌پذیری خودکار به سیستم اجازه می‌دهند تا به‌طور کارآمد مقیاس‌پذیر باشد و بارهای کاری متغیر را بدون کاهش عملکرد مدیریت کند.
- 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 یابد. این قابلیت تطبیق، الگوریتم را برای طیف وسیعی از برنامه‌ها و سناریوهای استقرار مناسب می‌سازد و در جهت‌های تحقیق‌های آینده می‌توان به موارد زیر اشاره کرد:
- بهبود تحمل خطا: تحقیقات آینده می‌تواند مکانیزم‌هایی برای بهبود تحمل خطا را بررسی کند، مانند ادغام تکنیک‌های یادگیری ماشین برای پیش‌بینی خرابی و بازیابی پیشگیرانه.
- بهره‌وری انرژی: بررسی راه‌هایی برای بهینه‌سازی مصرف انرژی در سیستم‌های میکروسرویس می‌تواند به استقرارهای ابری پایدارتر و مقرون‌به‌صرفه‌تر منجر شود [۳۰].
- 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.
- استقرارهای متنوع: انجام استقرارهای متنوع و مطالعات موردی بیشتر جهت افزایش اعتبار و اثربخشی دیدگاه پیشنهادی در محیط‌های ابری، اینترنت اشیاء، محاسبات مه و حوزه‌های کاربردی مختلف [۳۲ و ۳۱].

رویکرد پیشنهادی با تأکید بر هماهنگی در تغییر وضعیت خدمات، کاهش تأخیر و تخصیص پویا منابع، جایگزینی مؤثر برای راهکارهای سنتی مبتنی بر مدل‌های آماری و الگوریتم‌های تکاملی معرفی می‌شود. با استفاده از مجموعه‌ای از آزمایش‌های تجربی در محیط‌های شبیه‌سازی شده و سناریوهای بار متغیر، مشخص شد که راهکار ارائه‌شده توانایی بهبود شاخص‌هایی چون نرخ

سازگاری، قابلیت بازیابی و کارایی کلی سیستم را داراست. همچنین، مقایسه با چارچوب‌های موجود نشان داد که ترکیب الگوریتم‌های اجماع با مکانیزم‌های بهینه‌سازی منابع، امکان‌پذیری دستیابی به عملکرد پایدار در شرایط ناپایدار و متغیر ابری را فراهم می‌آورد. این نتایج نشان‌دهنده ظرفیت بالای الگوریتمPaxos برای استفاده در معماری‌های نوین و پیچیده مبتنی بر میکروسرویس است. در پایان ما بر این باوریم که برخی پژوهش‌ها مانند [۳۳-۳۴ و ۲۹ و ۱۰ و ۹] می‌توانند از دستاوردهای رویکرد پیشنهادی پژوهش حاضر درجهت توسعه دیدگاه‌های هوشمند چندمنظوره و جامع بهره گیرند.

٦- نتیجه‌گیری و راهکارهای آتی پژوهش

در این پژوهش، یک رویکرد جدید هوشمند جهت بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس در محیط‌های ابری پویا ارائه شده است. دیدگاه پیشنهادی با ترکیب الگوریتم اجماع Paxosبا مکانیزم‌های بهینه‌سازی منابع و مقیاس‌پذیری خودکار، به دنبال حل چالش‌هایی چون حفظ انسجام وضعیت، کاهش تأخیر، تحمل خطا و بهینه‌سازی استفاده از منابع در سیستم‌های میکروسرویس است.

با توجه به نتایج آزمایش‌های انجام شده، الگوریتم پیشنهادی OPaMS که بر پایه توسعه پروتکل Paxos طراحی شده است، توانسته است کاهش ۲۵٪ در معیار زمان پاسخ، ۳۰٪ افزایش در توان عملیاتی، ۲۰٪ بهبود در کارایی سیستم و حفظ نرخ سازگاری بالایی (۹۹/۹ درصد) را به دست آورد. علاوه بر این، این الگوریتم تحمل خطا بالایی (۹۵٪ درصد) نشان داد که نشان دهنده استحکام بالای آن در شرایط نامطلوب و متغیر است. این نتایج نشان می‌دهد که ترکیب الگوریتم‌های اجماع با مکانیزم‌های بهینه‌سازی هوشمند منابع، می‌تواند به عنوان یک راه‌حل مؤثر و قوی برای بهبود عملکرد و قابلیت اطمینان سیستم‌های میکروسرویس در محیط‌های ابری با بارکاری پویا مورد استفاده قرار گیرد. همچنین، این رویکرد به عنوان یک فرآیند پویا و قابل تنظیم، می‌تواند در شرایط مختلف شبکه و بار کاری به طور خودکار بهترین تصمیم را بگیرد. رویکرد بهینه‌سازی پیشنهادی یک راه‌حل قوی و مقیاس‌پذیر برای برنامه‌های کاربردی مدرن مبتنی بر محیط‌های ابری با بارهای کاری متغیر ارائه می‌دهد. این مطالعه اهمیت ادغام هوشمند پروتکل‌های اجماع در چارچوب‌های بهینه‌سازی عملکرد را برای دستیابی به دسترسی بالا، قابلیت اطمینان و کارایی در معماری‌های میکروسرویس برجسته می‌کند. تحقیقات آینده می‌تواند مکانیزم‌های ترکیبی و قابلیت‌های افزوده برای بهبود تحمل خطا، بهینه‌سازی مصرف انرژی و افزایش امنیت در سیستم‌های میکروسرویس را بررسی کند. انجام استقرارهای متنوع و مطالعات موردی بیشتر، اثربخشی الگوریتم پیشنهادی ما را در محیط‌ها و دامنه‌های کاربردی مختلف ابری تأیید خواهد کرد. با ادامه تحقیقات در این جهات، می‌توان به ایجاد سیستم‌های میکروسرویسی پایدار، قابل اعتماد و کارآمدتر دست یافت که در تمامی محیط‌های ابری، محاسبات مه و حوزه اینترنت اشیاء[1] قابل کاربرد باشند.

References

[1] Peidro, J. E., Muñoz-Escoí, F. D., & Bernabéu-Aubán, J. M. (2024). Modeling microservice architectures. J. Syst. Softw., 213, 112041. https://doi.org/10.1016/j.jss.2024.112041.

[2] Lelovic, L., Huzinga, A., Goulis, G., Kaur, A., Boone, R., Muzrapov, U., ... & Cerny, T. (2024). Change impact analysis in microservice systems: A systematic literature review. Journal of Systems and Software, 112241.

[3] Howard, H., Malkhi, D., & Spiegelman, A. (2016). Flexible paxos: Quorum intersection revisited. arXiv preprint arXiv:1608.06696.

[4] Ahmed, A. (2018). Programming Languages and Systems: 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings. Springer Nature.

[5] Baboi, M., Iftene, A., & Gîfu, D. (2019). Dynamic microservices to create scalable and fault tolerance architecture. Procedia Computer Science, 159, 1035-1044.

[6] Henning, S., & Hasselbring, W. (2024). Benchmarking scalability of stream processing frameworks deployed as microservices in the

[1] Internet of Things (IoT)

cloud. Journal of Systems and Software, 208, 111879.
[7] Lin, W., Sheng, X., & Qi, L. (2019). An Optimized Multi-Paxos Consensus Protocol for Practical Cloud Storage Applications. In Cyberspace Safety and Security: 11th International Symposium, CSS 2019, Guangzhou, China, December 1–3, 2019, Proceedings, Part I 11 (pp. 575-584). Springer International Publishing.
[8] Ailijiang, A., Charapko, A., & Demirbas, M. (2016, August). Consensus in the cloud: Paxos systems demystified. In 2016 25th International Conference on Computer Communication and Networks (ICCCN) (pp. 1-10). IEEE.
[9] Nakarmi, A., Kesharwani, H., Mallick, T., Jhingran, S., & Raj, G. (2024, April). A Comprehensive Study on Optimization Techniques for Microservices Deployment. In 2024 Sixth International Conference on Computational Intelligence and Communication Technologies (CCICT) (pp. 133-140). IEEE.
[10] Dong, H., & Liu, S. (2024). Asynchronous consensus quorum read: Pioneering read optimization for asynchronous consensus protocols. Electronics, 13(3), 481.
[11] Lampport, L. (1998). The part-time parliament. ACM Transactions on Computer Systems, 16(2), 133-169.
[12] Ongaro, D., & Ousterhout, J. (2014). In search of an understandable consensus algorithm. In 2014 USENIX annual technical conference (USENIX ATC 14) (pp. 305-319).
[13] Junqueira, F., Reed, B., and Serafini, M., "ZooKeeper's atomic broadcast and linearizable operations," in Proceedings of the Large Scale Distributed Systems and Middleware Workshop (LADIS'11), 2011.
[14] Kamil, S. N. S., Thomas, N., & Elsanosi, I. (2021, October). Performance evaluation of zookeeper atomic broadcast protocol. In EAI International Conference on Performance Evaluation Methodologies and Tools (pp. 56-71).
[15] Castro, M., & Liskov, B. (1999, February). Practical byzantine fault tolerance. In OsDI (Vol. 99, No. 1999, pp. 173-186).
[16] da Silva Pinheiro, T. F., Pereira, P., Silva, B., & Maciel, P. (2023). A performance modeling framework for microservices-based cloud infrastructures. The Journal of Supercomputing, 79(7), 7762-7803.
[17] Nawab, F., & Sadoghi, M. (2024). Consensus in Data Management: With Use Cases in Edge-Cloud and Blockchain Systems. Proceedings of the VLDB Endowment, 17(12), 4233-4236.
[18] Mwotil, A., Anderson, T., Kanagwa, B., Stavrinos, T., & Bainomugisha, E. (2024). LowPaxos: State Machine Replication for Low Resource Settings. IEEE Acces.
[19] Turkkan, B., Rodrigues, E., Kosar, T., Charapko, A., Ailijiang, A., & Demirbas, M. (2024). How to Evaluate Distributed Coordination Systems?--A Survey and Analysis. arXiv preprint arXiv:2403.09445.
[20] Johnson, R. (2025). Practical Replication Architectures and Protocols: Definitive Reference for Developers and Engineers. HiTeX Press.
[21] Amini, E., Miši, J., & Miši, V. B. (2025). Paxos With Priorities for Blockchain Applications. IEEE Transactions on Network and Service Management.
[22] Ferzo, B., & Zeebaree, S. R. (2024). Distributed Transactions in Cloud Computing: A Review Reliability and Consistency. The Indonesian Journal of Computer Science, 13(3).
[23] Jiang, T., Huang, X., Song, S., Wang, C., & Wang, J. (2024, May). On Tuning Raft for IoT Workload in Apache IoTDB. In 2024 IEEE 40th International Conference on Data Engineering (ICDE) (pp. 5307-5319). IEEE.
[24] Fareghzadeh, N., Seyyedi, M. A., & Mohsenzadeh, M. (2018). Dynamic performance isolation management for cloud computing services. The Journal of Supercomputing, 74, 417-455.
[25] Jiang, T., Huang, X., Song, S., Wang, C., & Wang, J. (2024, May). On Tuning Raft for IoT Workload in Apache IoTDB. In 2024 IEEE 40th International Conference on Data Engineering (ICDE) (pp. 5307-5319). IEEE.
[26] Fareghzadeh, N. (2022). An architecture supervisor scheme toward performance differentiation and optimization in cloud systems. The Journal of Supercomputing, 78(1), 1532-1563.
[27] Charapko, A., Ailijiang, A., & Demirbas, M. (2021, June). Pigpaxos: Devouring the communication bottlenecks in distributed consensus. In Proceedings of the 2021 International Conference on Management of Data (pp. 235-247).
[28] Hao, W., Xu, G., Wang, J., & Wen, Y. (2024, October). PPaxos: An Adaptive Pull-Based Group Consensus Protocol for Edge Networks. In 2024 IEEE 30th International Conference on Parallel and Distributed Systems (ICPADS) (pp. 761-768). IEEE.

[29] Montgomery, D. C. (2001). Design and analysis of experiments, John Wiley & Sons. Inc., New York, 1997, 200-1.
[30] Fareghzadeh, N. (2019). Service Contract-Aware Quality Supervisory Methodology in Cloud Systems. Journal of Quality Engineering and Management, 9(2), 172-185.
[31] Xie, Y., et al. (2017). Characterizing delay variability in cloud data center networks. Proceedings of ACM SIGCOMM, 494–507.
[32] [32] M. H. Mohabbati Hamidi and M. Abbasi, "Efficient mechanism for determining a function for workload distribution in fog computing using classifier systems," The Journal of Theoretical and Applied Machine Intelligence, vol. 2, no. 1, pp. 1-13, 2024, doi: 10.22034/abmir.2024.20271.1030.
[33] [33] M. M. Hosseini and H. Zargari, "Energy-efficient clustering in multi-hop wireless sensor networks using multi-objective particle swarm optimization," The Journal of Theoretical and Applied Machine Intelligence, vol. 2, no. 2, pp. 67-81, 2025, doi: 10.22034/abmir.2025.22425.1077.
[34] [34] A. Doldi, A. Mozidi, and M. B. Gorji, "Designing a cloud computing based human resource optimization model for Tejarat Bank," Information Management Sciences and Technologies, vol. 10, no. 2, pp. 235-259, 2024, doi: 10.22091/stim.2023.9683.1980

Journal of Applied And Basic Machine Intelligence Research (ABMIR)

Applied and Basic Machine Intelligence Research

Vol. 3, No. 1, Spring & Summer 2025, pp. 145 - 161

Intelligent Consensus Optimization Approach Toward Coherent Resource Management in Microservice Systems

Mahdi Bazargani[1], Nafiseh Fareghzadeh[2*], Farzaneh Kabudvand[3]

[1]Assistant Professor, Department of Computer Engineering, Islamic Azad University, Zanjan, Iran
[2] Assistant Professor, Department of Computer Engineering, Islamic Azad University, Zanjan, Iran
[3] Instructor, Department of Computer Engineering, Islamic Azad University, Zanjan, Iran

Article Information

Original Research Paper

Received:
2025 June 17

Accepted:
2025 October 12

Keywords:
Paxos consensus protocol, Distributed persistence, Resource management, State coordination, Microservice architecture, Intelligent optimization, Dynamic cloud

Corresponding Author*:
fareghzadeh@iau.ac.ir

Abstract

Microservice architectures, as the dominant design paradigm for cloud systems, offer several advantages such as flexibility and scalability. However, they face challenges such as ensuring data integrity, dealing with node failures, and optimal resource allocation. In this research, a new approach and intelligent optimization mechanism are proposed that, by utilizing the Paxos consensus protocol, leads to improved performance, resilience, and stability in microservice systems. The proposed approach, emphasizing intelligent coordination in changing service states, reducing latency, and dynamically allocating resources, is an effective alternative to traditional solutions based on statistical models and evolutionary algorithms. Evaluations in dynamic cloud environments with variable workloads present that the proposed approach is capable of providing significant improvements in fundamental metrics such as response time (25%), throughput (30%), efficiency (20%), consistency (99.9%), and fault tolerance (99.8%). Also, comparison with existing frameworks present that combining consensus algorithms with intelligent optimization mechanisms and effective resource management enables achieving stable performance in unstable and variable cloud conditions. These results indicate the high capacity of the proposed approach for utilizing in modern and complex microservice architectures.

: 10.22034/ABMIR.2025.23292.1135

E-ISSN: 2821-2037