نشریه علمی "پدافند الکترونیکی و سایبری"

سال هشتم، شماره ۳، پاییز ۱۳۹۹، ص ۸۱-۶۳

علمی– پژوهشی

مدیریت کشسانی منابع با استفاده از کنترل کننده فازی مبتنی بر تغییرات حد آستانه در محیط رایانش ابری

مصطفی قبائی آرانی[1*]، علی شهیدی نژاد[2]، میترا ترابی[3]

۱– استادیار، ۲– استادیار، ۳– کارشناسی ارشد

۱و۲و۳– گروه مهندسی کامپیوتر، واحد قم، دانشگاه آزاد اسلامی، قم، ایران

(دریافت: ۹۸/۰۶/۲۶، پذیرش: ۹۸/۱۲/۱۲)

چکیده

افزایش محبوبیت و سودآوری رایانش ابری وابسته به تأمین قابلیت‌ها و ویژگی‌های مورد نظر کاربران ابری می‌باشد. خاصیت کشسانی، به‌عنوان یکی از قوی‌ترین ویژگی‌هایی محسوب می‌شود که حوزه رایانش ابری را از دیگر رویکردهای سامانه‌های توزیعی، مجزا می‌کند. رایانش ابری ظرفیت منابع را برای مصرف کننده به‌صورت بی‌نهایت در نظر می‌گیرد و مصرف کننده، می‌تواند منابع را برحسب تقاضا و بر اساس نرخ رقابتی در اختیار بگیرد و میزان منابع را افزایش یا کاهش دهد. اگر چه راه‌حل‌های مختلفی برای مدیریت کشسانی تاکنون توسعه داده شده‌اند، اما کارهای بیشتری نیاز است تا خاصیت کشسانی ابر را به‌صورت کاراتر مدیریت نمایند. در این مقاله، رویکردی برای بهبود خاصیت کشسانی با استفاده از سامانه کنترل فازی مبتنی بر تغییرات حد آستانه برای کاربردهای محاسبات با عملکرد بالا در شبکه‌های ابری ارائه می‌شود. در روش پیشنهادی مدیریت کشسانی بر پایه نظارت و تصمیم‌گیری مستمر انجام می‌شود. نتایج مشخص می‌کند که روش پیشنهادی عملکرد بهتری در مورد زمان پاسخگویی، هزینه و تخطی از شرایط سرویس‌دهی، نسبت به روش‌های پیشین دارد. میانگین زمان پاسخگویی روش پیشنهادی در مقایسه با دو روش مورد مقایسه مقاله به ترتیب ۶/۵٪ و ۹٪، میانگین هزینه به ترتیب ۶٪ و ۱۲٪ و میانگین تخطی از شرایط پذیرش سرویس به ترتیب ۶۸٪ و ۷۷/۵٪ کاهش یافته است.

کلیدواژه‌ها: خاصیت کشسانی، تأمین منابع، سامانه کنترل فازی، حد آستانه، رایانش ابری

۱– مقدمه

سرویس‌های ابری، زیرساخت‌های توزیع شده‌ای می‌باشند که بسترهای ارتباطی و خدماتی را گسترش می‌دهند. مهم‌ترین ویژگی‌های مطلوب رایانش ابری، از نظر کاربران ابری کارایی بالا، هزینه حداقل و اجرای درخواست در زمان معین می‌باشند. ارائه دهندگان سرویس‌های ابری به‌منظور دستیابی به کیفیت مطلوب کاربران و سودآوری بیشتر، ابر را بر اساس ساختارها و فناوری‌های نوظهور، همانند مجازی‌سازی پیاده‌سازی نموده‌اند، تا از این رهگذر پاسخگوی نیاز کاربران باشند و همچنین با بهره‌گیری از مفهوم مجازی‌سازی به‌ویژه مجازی‌سازی منابع، بهترین سرویس را با حداقل هزینه ممکن ارائه دهند. خاصیت کشسانی ابر، یکی از بهترین راه‌های افزایش بهره‌وری و بهبود عملکرد سامانه‌های ابری می‌باشد چرا که بر اساس این ویژگی منابع به مقدار مورد نیاز کاربران فراهم می‌گردد. خاصیت کشسانی درجه‌ای است که در آن سامانه قادر به تطبیق با تغییرات بار کاری از طریق پیش‌بینی و عدم پیش‌بینی منابع به شیوه‌ای خودکار است، به‌طوری که در هر نقطه زمانی، منابع در دسترس، با درخواست اخیر کاربر تا جای ممکن تطبیق می‌یابند. خاصیت کشسانی، از این حقیقت بهره می‌گیرد که فرایند تخصیص‌دهی منابع، به‌عنوان رَویه‌ای محسوب می‌شود که می‌توان آن را به‌صورت پویا (بر اساس تقاضای وارده برای هر یک از سرویس‌ها یا کاربران) اجرا نمود [۱]. ویژگی کشسانی به‌عنوان یک اصل اساسی برای مدل ابری محسوب می‌شود، زیرا نه تنها منابع کافی را برای به اشتراک‌گذاری میان کاربران فراهم می‌سازد، بلکه روال محاسبات مبتنی بر پرداخت هزینه را بر اساس حجم دریافت سرویس مقدور می‌سازد. در حال حاضر، تأمین کشسانی منابع یکی از مهم‌ترین و جذاب‌ترین ویژگی‌های زیرساخت به‌عنوان یک سرویس (IaaS) در سامانه‌های ابری مدرن است.

خاصیت کشسانی ابری، مزایایی برای برنامه‌های کاربردی موازی محسوب می‌شود که شامل تطبیق منابع و فرآیندها در زمان اجرا هست تا کاربران دیگر نگران بهترین انتخاب نباشند و

* رایانامه نویسنده مسئول: m.ghobaei@qom-iau.ac.ir

برای این منظور متداول‌ترین رویکردها یا سیاست‌های کشسانی، کشسانی واکنشی (پس‌کنشی) مبتنی بر آستانه و کشسانی پیشگویانه (پیش‌کنشی) زمان‌بر، استفاده می‌گردد. اما باید به این نکته توجه داشت که هر دو رویکرد حداقل دارای یک مشکل می‌باشند که به تجربه قبلی کاربر، عدم کنترل نقاط اوج بار کاری، کامل شدن پارامترها یا طرح برای زیرساخت خاص و تعیین حجم کار (تراکم کار) مربوط می‌باشد. تعیین مقدار صحیح میزان منابع محاسباتی برای اجرای یک برنامه موازی به‌صورت یک شمشیر دو لبه است که می‌تواند به تأمین منابع کم یا زیاد بیانجامد. عموماً کارهای قبلی، به‌منظور تأمین پویای منابع، شرایط توافق سرویس (SLA)[1] را برای برنامه‌های کاربردی چندلایه مهیا کرده‌اند. اگر چه بسیاری از راه‌حل‌های کشسانی هم اکنون توسعه داده شده‌اند تا مشکلات کشسانی را در ابر کاهش دهند ولی کارهای زیادی نیاز است تا خاصیت کشسانی ابر بهتر مدیریت شود. در این مقاله، رویکردی برای بهبود خاصیت کشسانی با استفاده از سامانه کنترل فازی مبتنی بر تغییرات حد آستانه برای کاربردهای محاسبات با عملکرد بالا در شبکه‌های ابری ارائه می‌شود، طوری که هیچ نیازی به عملیات تعدیل و اصلاح کد منبع برنامه کاربردی، تعاریف مضاعف قوانین و اقدامات مربوط به مقیاس پذیری توسط برنامه‌نویسان و همچنین نیاز به داشتن دانش قبلی نسبت به برنامه کاربردی نباشد. در روش پیشنهادی مدیریت کشسانی بر پایه مدیریت مستمر انجام می‌شود. بدین ترتیب که گره‌ای سخت‌افزاری به‌نام برنامه اصلی وظیفه نظارت بر ماشین‌های مجازی را بر عهده دارد. مدیریت کشسانی بر اساس گزارش‌های دریافت شده از برنامه اصلی و با استفاده از سامانه کنترل فازی مبتنی بر حد آستانه متغیر تصمیم می‌گیرد که گره جدید متشکل از چندین ماشین مجازی به ابر اضافه شود یا گره‌ای که کمتر مورد استفاده قرار می‌گیرد، حذف گردد. بدین ترتیب تأمین منابع با استفاده از این ساختار تصمیم‌گیر انجام می‌شود. هدف نهایی مقاله، ارائه رویکردی برای تأمین منابع است که میزان تخطی از شرایط پذیرش سرویس را کاهش دهد.

سهم علمی این پژوهش به شرح زیر می‌باشد:

- ارائه رویکردی برای بهبود خاصیت کشسانی با استفاده از سامانه کنترل فازی مبتنی بر تغییرات حد آستانه
- تقویت خاصیت کشسانی با استفاده از مدل تأمین منبع بر پایه سامانه کنترل فازی مبتنی بر تغییرات حد آستانه
- افزایش دقت در تأمین منابع (افزایش کیفیت ارائه سرویس و کاهش تعداد منابع)
- کاهش تخطی از شرایط پذیرش سرویس
- کاهش زمان پاسخگویی
- افزایش بهره‌وری

سازمان‌دهی ادامه مقاله به‌صورت زیر می‌باشد:

در بخش دوم این مقاله، تحقیقات انجام شده در زمینه بهبود خاصیت کشسانی بررسی می‌شود. در بخش سوم، روش پیشنهادی برای بهبود خاصیت کشسانی با استفاده از سامانه کنترل فازی مبتنی بر تغییرات حد آستانه ارائه می‌گردد. در بخش چهارم، نتایج عملکرد سامانه پیشنهادی در قالب نتایج شبیه‌سازی مورد ارزیابی قرارگرفته و با روش‌های مرجع مقایسه می‌شود. درنهایت، در بخش آخر نتیجه‌گیری و پیشنهادها مطرح می‌گردد.

۲- مروری بر کارهای انجام شده

هدف اکثر رویکردهای مدیریت کشسانی بهبود بهره‌وری می‌باشد [۱۰-۲]. رویکردهایی هم وجود دارند که به دنبال افزایش ظرفیت منابع محلی [۱۱]، کاهش هزینه‌ها [۱۲] و صرفه‌جویی انرژی [۱۳] هستند. مثال‌هایی از راهکارهایی که به دنبال دسترس‌پذیری هستند شامل مرجع [۱۴] می‌باشد. در مرجع [۱] منافع خدمات دهنده و کیفیت سرویس مشتری به‌صورت هم‌زمان در نظر گرفته می‌شود. راه‌حل‌ها و سیاست‌های کشسانی مختلفی در ابر وجود دارد. این راه‌حل‌ها را می‌توان در چندین گروه بر اساس خاصیت‌های مختلف آن‌ها طبقه‌بندی نمود. همان‌گونه که در منابع [۱۴-۱۲] ثبت شده است راه‌حل‌های کشسانی را می‌توان بر اساس نوع راه‌حل به واکنشی و پیشگویانه تقسیم‌بندی نمود. بر اساس روش‌های پیاده‌سازی این امکان وجود دارد که هر نوع از راه‌حل‌های کشسانی به‌صورت مقیاس پذیری افقی، عمودی و یا مهاجرتی طبقه‌بندی شوند. مقیاس پذیری، توانایی یک سامانه در برابر افزایش بار کاری می باشد به این صورت است که منابع محاسباتی جدید را اضافه می نماید تا کارایی خودش را حفظ نماید، در صورتی که خاصیت کشسانی مربوط می شود به اینکه سامانه چگونه به خوبی منابع مورد نیاز را در هر نقطه از زمان به‌صورت خودکار و بر اساس بار کاری تأمین نماید. مفهوم رایانش خودکار توسط IBM در سال ۲۰۰۰ در سامانه‌های محاسباتی پیچیده به‌منظور کاهش بار کاری مدیر سامانه معرفی شد [۱۵] که شامل چهار هدف اصلی زیر است:

۱- خود پیکربند[2]: توانایی تنظیم خودکار سامانه منطبق با شرایط.

[1] Service Level Agreement

[2] Self-Configuration

۲- خود معالج[1]: قابلیت شناسایی و رفع خودکار خطا که موجب افزایش قابلیت اطمینان سامانه می‌شود.

۳- خود بهینه‌ساز[2]: قابلیت اندازه‌گیری کارایی و بهبود کیفیت خدمات سامانه با استفاده بهینه از منابع.

۴- خود محافظ[3]: آگاهی از تهدیدات و تنظیم خودکار سامانه برای رسیدن به حفاظت از منابع و تأمین امنیت آن‌ها.

سیاست کشسانی، اشاره به تعاملات مورد نیاز به‌منظور انجام اقداماتی است که موجب تضمین خاصیت کشسانی می‌شود. اقدامات کشسانی به‌صورت خودکار انجام می‌شوند. در حالی که مقیاس پذیری می‌تواند به‌صورت دستی و توسط کاربر انجام شود. خاصیت کشسانی، شامل افزایش و کاهش اشتراک منابع انتزاعی برحسب تقاضا می‌باشد، در صورتی که مقیاس پذیری تنها شامل افزایش منابع می‌باشد و شامل چگونگی افزایش منابع بر اساس سطح کارایی مورد نظر نمی‌باشد. مقیاس‌بندی به دو صورت افقی و عمودی انجام می‌شود. در مقیاس‌بندی افقی، ماشین‌های مجازی جدیدی به سامانه یا سکوی محاسباتی اضافه می شوند و متوازن کننده بار، بار کاری را بین همه ماشین‌های مجازی در دسترس، توزیع می‌کند. در مقیاس‌بندی عمودی، منابع بیشتری برای انجام تقاضای برنامه کاربردی، روی همان ماشین مجازی اضافه می‌شود. اکثر سامانه عامل‌ها از تغییرات روی میزان پردازنده یا حافظه موجود، بدون راه اندازی مجدد برای اجرای مقیاس‌بندی عمودی، پشتیبانی نمی‌کنند. به همین دلیل اکثر ارائه‌دهندگان ابری، فقط از مقیاس‌بندی افقی پشتیبانی می‌کنند [۱۶ و ۱۷]. در ادامه راه‌حل‌های کشسانی شرح داده می‌شوند.

۲-۱- راه‌حل کشسانی واکنشی

این راه‌حل بر اساس شرایط تعریف شده در شرایط توافق سرویس (SLA) و بار کاری برنامه کاربردی، اقدام به کشسانی خودکار می‌نماید، راه‌حل واکنشی بر اساس روش Rule-condition-action (فعالیت بر اساس قانون) کار می‌کند. در یک راه‌حل واکنشی صاحب برنامه کاربردی به‌صورت دستی قوانین کشسانی خودکار که مقادیر آستانه را برای تأمین یا بازپس‌گیری منابع مشخص می‌کند را فراهم می‌نماید. یک قانون کشسانی خودکار ترکیبی از شرایط آستانه و عملیات کشسانی می‌باشد یعنی اینکه وقتی هر شرایط اتفاق می‌افتد چه عملیات کشسانی (اضافه یا حذف ماشین مجازی) توسط لایه‌های زیرین پلتفرم ابر اتخاذ شود. مثالی از قانون‌های اقدامات کشسانی خودکار برای تأمین منابع می‌تواند به این صورت باشد که اگر میزان استفاده از پردازنده بیش از ۸۰ درصد شد و این وضعیت به مدت ۵ دقیقه ادامه پیدا کرد، این منبع می‌بایست افزایش داده شود. هر شروطی شامل یک یا بیشتر از معیارهای کشسانی خودکار، به‌عنوان مثال بهره‌وری پردازنده، استفاده از حافظه Ram و تخطی از SLA است که در برابر مقادیر آستانه مقایسه می‌شوند. اکثر ارائه‌دهندگان سرویس ابر معیارهای کشسانی خودکار را از طریق یک سرویس نظارت، فراهم می‌نمایند. به‌عنوان مثال سرویس نظارت Cloud Watch آمازون شامل استفاده از پردازنده، ترافیک شبکه، خواندن و نوشتن دیسک می‌باشد. کنترل کننده کشسانی در لایه‌های پایین پلتفرم ابر به‌طور مداوم این روال را نظارت می‌نماید و هنگامی که یکی از شرایط اتفاق افتد عملیات کشسانی مناسب شروع می‌شود. راه‌حل کشسانی واکنشی در ابر بیشتر استفاده می‌شود و توسط اکثر فراهم کنندگان سرویس ابر ارائه می‌شود مانند مایکروسافت آژور، آمازون و پلتفرم ابر گوگل (GCP). در مراجع [۲۲-۱۸] چندین کار آکادمیک پیشنهاد شده که برخی راه‌حل‌های کشسانی بر پایه قانون را بهبود داده‌اند. برای مثال در مرجع [۲۳] یک الگوریتم بر پایه آستانه انطباقی ارائه شده است که از مدل رگرسیون خطی استفاده می‌نماید. مقادیر آستانه جدید را در هر زمانی که مقادیر کارایی از قبل تعریف شده را نقض کنند، محاسبه می‌نماید. به هر حال راه‌حل مبنای مقادیر آستانه‌ای جدید را فقط بر مبنای رفتار جاری سامانه تنظیم می‌نماید. در مرجع [۲۴] یک رویکرد تأمین منابع ترکیبی برای برنامه‌های چند لایه مبتنی بر ترکیب مفهوم محاسبات خودکار و رویکرد فرآیند سلسله مراتبی تحلیل فازی پیشنهاد شده است. علاوه بر این، یک چارچوب بر اساس حلقه کنترل MAPE-K برای تأمین خودکار برنامه‌های چند لایه در محیط محاسبات ابری ارائه شده است. در مرجع [۲۵] یک مدل کشسانی خودکار در سطح پلتفرم به‌عنوان سرویس (PaaS) برای محاسبات با کارایی بالا (HPC)[4] ارائه شده است رویکرد مشتقه آن شامل ارائه خاصیت کشسانی برای برنامه‌های کاربردی با کارایی بالا بدون دخالت کاربر یا تغییر کد منبع است. در مرجع [۲۶] روش حد آستانه زنده (LT)[5] برای کنترل کشسانی پیشنهاد شده است که مبتنی بر الگوریتم تراکم TCP است و به‌طور خودکار مقدار مرزهای کشسانی را مدیریت می‌کند تا واکنش‌پذیری بهتر در تأمین منابع را افزایش دهد. یک سرویس قطع و اجرا سبک وزن در سطح پلتفرم به‌عنوان سرویس در ابر ارائه شده است که در آن کاربران کاملاً از ویژگی کشسانی آگاهی ندارند، فقط نیاز به کامپایل‌کردن برنامه‌های کاربردی خود با نمونه اولیه Helpar دارد. در مرجع [۲۷] یک سرویس کشسانی ترکیبی برای برنامه‌های کاربردی موازی اصلی- زیردست[6] به‌نام

[1] Self-Healing
[2] Self-Optimization
[3] Self-Protection
[4] High Performance Computing
[5] Live Tresholding
[6] Slave

Helpar (مدل کشسانی ترکیبی برای برنامه‌های کاربردی موازی[1]) توسعه داده شده است. مدل بدون پارامتر Helpar یک معماری کشسانی حلقه بسته را ارائه کرده است که در زمان اجرا مقادیر آستانه‌های پایین و بالا را تطبیق می‌دهد. راه‌حل‌هایی که مسئله کشسانی را به‌صورت واکنشی حل می‌کنند، شامل تأخیر محاسبه می‌باشند به همین دلیل راه‌حل‌های کشسانی پیشگویانه سریع‌تر می‌باشند.

۲-۲- راه‌حل کشسانی پیشگویانه

بر خلاف راه‌حل واکنشی، یک راه‌حل پیشگویانه تلاش می‌کند تا تقاضاهای آینده برای یک برنامه کاربردی را پیش‌بینی نماید تا منابع مناسب را در هنگام افزایش بار کاری در اختیارش قرار دهد، تاکنون تحقیقات مختلفی [۲۸ و ۲۹] در این زمینه انجام شده است. راه‌حل پیشگویانه از روش‌های تحلیل مختلف به‌عنوان مثال تحلیل سری‌های زمانی، روش‌های حل مدل، نظریه صف، نظریه کنترل و یادگیری تقویتی استفاده می‌کند تا مشخص نمایند چه وقت و چگونه منابع را تأمین و یا بازپس گیرد. یکی از سیاست‌های پیشگویانه استفاده از پیش‌بینی کننده بار کاری است تا رفتار بار کاری آینده را بر اساس تاریخچه‌های کاری قبل پیش‌بینی کند و سپس از یک مدل کارایی استفاده نماید تا مقدار منابع مورد نیاز جهت پاسخ‌گویی بار کاری به‌عنوان مثال تعداد منابع مورد نیاز [۳۰] را تصمیم بگیرد. راه‌حل‌های پیشگویانه مختلفی جهت کشسانی خودکار منابع ارائه شده است. برای مثال در مرجع [۳۱] یک سامانه کشسانی خودکار بر اساس یک راه‌حل بر پایه موج ضربه‌ای کوچک [۳۲] ارائه می‌شود تا منابع مورد نیاز یک برنامه کاربردی را برای بارهای کاری مختلف پیش‌بینی نماید. یک تحلیل جامع از راه‌حل کشسانی پیشگویانه در قسمت [۳۳ و ۳۴] ارائه شده است. در مرجع [۳۵] یک الگوریتم پیش‌بینی استفاده از منابع ابری گروه بر اساس نظریه یادگیری ماشینی ارائه شده است که ترکیبی از حالت مدل‌های فن پیش‌بینی است و وزنی برای مدل‌های تشکیل دهنده فردی تعیین شده است. الگوریتم پیشنهادی با ترکیب مقادیر پیش‌بینی همه مدل‌های سازنده بر اساس عملکرد آن‌ها پیش‌بینی می‌کند. راه‌حل کشسانی ترکیبی، هر دو راه‌حل واکنشی و پیشگویانه را ترکیب می‌نماید. برای مثال در مرجع [۳۶] یک راه‌حل که بر پایه راه‌حل ترکیبی ارائه شده است محقق از راه‌حل واکنشی برای کشسانی خارجی و از راه‌حل پیشگویانه برای کشسانی داخلی استفاده می‌نماید. در مرجع [۳۷] سامانه ElasticCloud مورد استفاده برای مدیریت خاصیت کشسانی در ارائه‌دهنده ابر ارائه شده است. این سامانه یک روش ترکیبی را ارائه کرده است که یک کشسانی واکنشی همراه با یک کشسانی پیشگویانه به ترتیب مقیاس بالا و مقیاس پایین منابع در حال اجرا یک برنامه کاربردی وب میزبانی شده در ابر را ترکیب می‌کند. راه‌حل‌های کشسانی پیشگویانه گذشته اغلب پیچیدگی محاسباتی بالایی دارند به همین دلیل در این مقاله، رویکردی برای بهبود خاصیت کشسانی با استفاده از سامانه کنترل فازی مبتنی بر تغییرات حد آستانه برای کاربردهای محاسبات با عملکرد بالا در شبکه‌های ابری ارائه می‌شود.

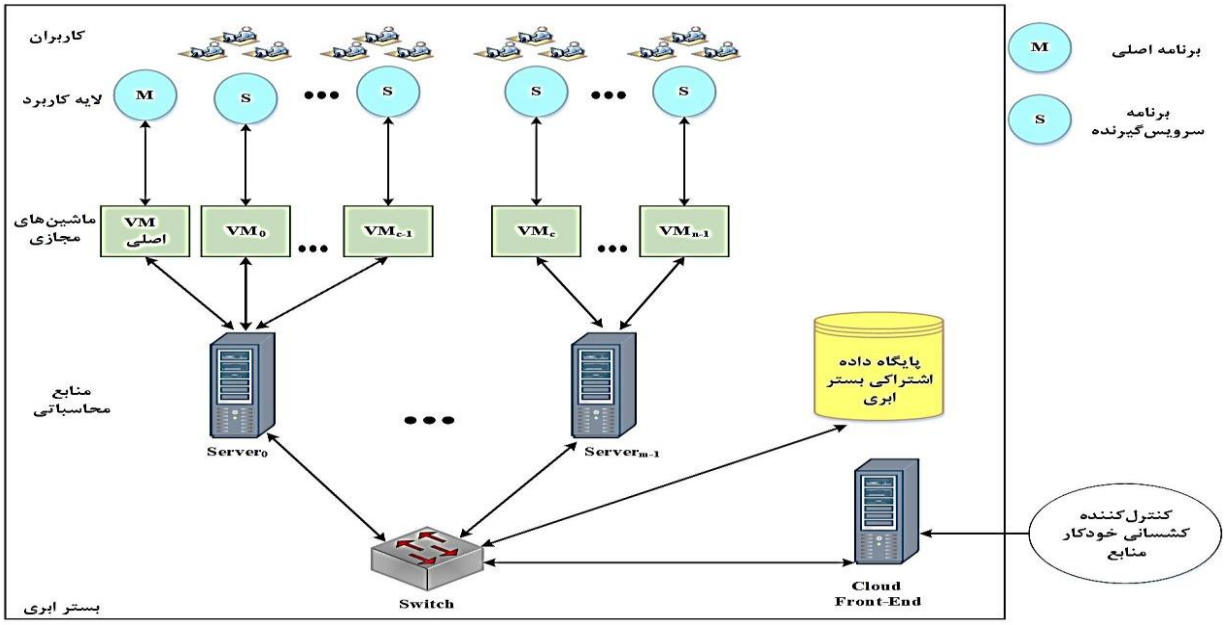
۳- روش پیشنهادی

همان‌طورکه قبلاً اشاره شد، خاصیت کشسانی، توانایی سامانه برای مدیریت و تأمین منابع، هنگامی که سامانه با حجم کاری بالا روبه‌رو است، می‌باشد. البته این را نیز باید اضافه کرد که نحوه مدیریت سامانه بعد از کاهش بار کاری نیز به کشسان بودن آن ارتباط دارد. در روش پیشنهادی، کشسانی خودکار به واسطه افزایش یا کاهش ماشین‌های فیزیکی یا سخت‌افزار جدید اتفاق می‌افتد. ساختار مدیریت و تأمین منابع بر پایه یک شبکه و ارتباط داخلی آن‌ها انجام می‌شود. روش پیشنهادی، کنترل کننده کشسانی خودکار منابع (Autonomic Resource Elasticity Controller)[2] نام دارد و به‌عنوان یکی از اصلی‌ترین اجزاء، وظیفه کنترل منابع سخت‌افزاری را بر عهده دارد. روش پیشنهادی با استفاده از حد آستانه متغیر و سامانه کنترل فازی برای تأمین خودکار منابع و کشسانی ارائه می‌شود. شکل (۱) معماری روش پیشنهادی را نشان می‌دهد. در معماری این رویکرد سه لایه کاربرد، لایه پلتفرم و لایه زیرساخت (منابع محاسباتی) وجود دارد. در معماری پیشنهادی زیرساخت‌ها یا رایانه‌های میزبان[3] به عنوان سرور[4] شناخته می‌شوند. این سرورها شامل پردازنده C هسته‌ای هستند، که هر هسته در اختیار یک ماشین مجازی قرار می‌گیرد. به این ترتیب، هر سرور شامل c ماشین مجازی خدماتی خواهد بود. تعداد سرورها در روش پیشنهادی m در نظر گرفته شده است. در این معماری مؤلفه‌های مؤثر در کنترل کننده کشسانی خودکار منابع و چگونگی نگاشت ماشین‌های مجازی بر روی سرورهای همگن یا میزبان‌ها، نشان داده شده است. کنترل کننده کشسانی خودکار منابع (تأمین منابع) هم به‌صورت یک میزبان مستقل در نظر گرفته شده است. با توجه به اینکه درخواست کاربران در قالب برنامه کاربردی ارائه می‌شود، در این ساختار یک برنامه اصلی (M) جهت مدیریت و مجموعه‌ای از برنامه‌های سرویس گیرنده (S) وجود دارد. با در نظر گرفتن این موضوع که برنامه‌های کاربردی موازی، عموماً درصد بهره‌وری

[1] Hybrid Elasticity Model for Parallel Applications

[2] AREC
[3] Host
[4] Server

بسیار زیاد از پردازنده دارند، بنابراین مدل معماری روش پیشنهادی، از یک فرایند واحد در هر یک از ماشین‌های مجازی و به تعداد c ماشین مجازی نیز جهت هر یک از سرورهای میزبان، بهره می‌گیرد. در این معماری پایگاه داده اشتراکی وجود دارد که تمامی رویدادها، اطلاعات، درخواست‌ها و فرامین در این فضا ثبت می‌شوند. تمامی اجزاء توسط شبکه‌ای داخلی به یکدیگر متصل هستند. تمامی فرامین صادر شده توسط کنترل کننده کشسانی خودکار منابع در پایگاه داده اشتراکی ذخیره می‌شوند. فرامین، عموماً توسط برنامه اصلی پردازش و اجرا می‌شوند. هر نوع ارتباط در این معماری توسط شبکه اتصالات داخلی (به وسیله Switch) و به واسطه دسترسی به پایگاه داده اشتراکی فراهم می‌شود. در معماری پیشنهادی، مؤلفه AREC، با اضافه و کم کردن سرورها خاصیت کشسانی پلتفرم ابری را کنترل می‌کند. بدین صورت که اگر تعداد ماشین‌های مجازی برای پاسخگویی به درخواست‌های کاربران در سرورها کافی نباشد، سروری شامل چندین ماشین مجازی اضافه می‌شود و در حالی که اگر میزان بار کاری ورودی برای یک سرور کمتر از حد تشخیص داده شود، ماشین‌های مجازی آن سرور، به سرورهای مناسب انتقال داده می‌شوند. کنترل کننده کشسانی، توسط ساختار سامانه کنترل فازی وظیفه خود را انجام می‌دهد، بدین ترتیب که واحد ورودی با محاسبه میزان تخطی از شرایط سرویس و دریافت تعداد درخواست، داده‌ها را در اختیار سامانه کنترل کننده فازی قرار می‌دهد، ابتدا حد آستانه برای استفاده در ساختار فازی محاسبه می‌گردد، سپس با استفاده از سامانه فازی، نوع مقیاس‌بندی مشخص می‌شود. پس از تصمیم‌گیری، نوع مقیاس‌بندی برای واحد تعیین میزان کشسانی ارسال می‌گردد تا میزان مقیاس‌بندی مشخص گردد. سپس توسط خروجی نوع و مقدار مقیاس‌بندی در حافظه اشتراکی قرار می‌گیرد.



شکل (۱): معماری روش پیشنهادی AREC.

Algorithm 1: *Main AREC*

```
for i ← 1 to n do
  PUT Req_i to Request_queue
end for
While(Request_queue is not empty)
  foreach req in Request_queue do
    Send_Request(Master_App,req)
    m = Master_App.Check(VM[ ])
    Master_App.Send_State(m,req,Data_Share)
    Elastic_Controller.Check(Data_Share)
    Elastic_Controller.SetCommand(Data_Share)
    C = Master_App.GetCommand(Data_Share)
    Master_App.ExecCommand(C)
  Master_App.SendReply(Data_Share)
 end for
end while
```

شبه کد (۱)، الگوریتم کشسانی خودکار روش پیشنهادی AREC، را نشان می‌دهد. در این الگوریتم عملکرد لایه‌ها، پایگاه داده اشتراکی و کنترل کننده کشسانی خودکار منابع مدیریت می‌شوند. جدول (۱) نیز به معرفی متغیرهای مورد استفاده در الگوریتم‌ها و ساختار روش AREC، در مقاله می‌پردازد. در بخش بعد، برخی از اجزای مدل پیشنهادی کنترل کننده کشسانی خودکار منابع، به جزئیات تشریح می‌شوند.

جدول (۱): متغیرها و تعاریف آن‌ها.

نام	تعریف
N	تعداد درخواست
$Request_i$	درخواست i ام
M	تعداد سرورهای میزبان خدماتی
Req_Cost_i	بودجه در نظرگرفته شده برای درخواست i ام
Req_Rmax_i	زمان حد مجاز برای درخواست i ام
$Request\_queue$	صف درخواست‌ها
C	تعداد VM ها
$VM_k.Cost$	هزینه استفاده از ماشین مجازی k ام
$VM_k.Avail\_MIPS$	میزان MIPS در دسترس ماشین مجازی k ام
$VM_k.Used\_MIPS$	میزان MIPS استفاده شده ماشین مجازی k ام
$VM_k.Allocate\_Datetime_i$	زمان اختصاص ماشین مجازی k ام به درخواستi ام
$VM_k.Free\_Datetime_i$	زمان آزاد شدن ماشین مجازی k ام از درخواستi ام
$VM_k.RequestID$	درخواست مربوط به ماشین مجازی k ام
$Server_i$	سرور i ام
$Shared\ DB\_Cloud\ Platform$	پایگاه داده اشتراکی بستر ابری
$Shared\ DB\_Elasticity\ Controller$	پایگاه داده اشتراکی کنترل کننده کشسانی
$Elastic\_Controller$	کنترل کننده کشسانی
$Slave\_App$	برنامه خدماتی
$Master\_App$	برنامه اصلی
$Cost$	مجموع هزینه
$Utilization$	مجموع بهره‌وری
$Response$	مجموع زمان پاسخگویی

۳-۱- لایه کاربرد

در این لایه دو نوع برنامه کاربردی وجود دارد: برنامه اصلی، برنامه‌های خدماتی. در زیر این برنامه‌های کاربردی، به جزئیات تشریح می‌شوند.

۳-۱-۱- برنامه اصلی

یکی از اصلی‌ترین واحدهای روش پیشنهادی، برنامه اصلی[1] است. در اصل این برنامه وظیفه نظارت و نگاشت وضعیت ماشین‌های مجازی و درخواست‌ها را در پایگاه داده اشتراکی بر عهده دارد. وظیفه دیگر این برنامه اختصاص ماشین‌های مجازی به درخواست کاربران است، بدین‌ترتیب یکی از مهم‌ترین نوع ارتباط‌ها، در شبکه اتصالات داخلی، بین کنترل کننده کشسانی خودکار منابع و برنامه اصلی خواهد بود که توسط پایگاه داده اشتراکی فراهم می‌شود. به دلیل اهمیت این واحد، یک ماشین مجازی به‌صورت مستقل در اختیار این واحد قرار گرفته است. اطلاعاتی که توسط این واحد، برای هر ماشین مجازی در حافظه اشتراکی نگهداری می‌شود در قالب جدول (۲) است. اطلاعات هر ماشین مجازی توسط شماره‌گذاری آن به‌صورت مستقل در پایگاه داده اشتراکی ثبت می‌گردد و هیچ نوع تداخلی در ثبت اطلاعات پیش نمی‌آید. اطلاعات درخواست‌های جدید کاربران نیز توسط این برنامه در حافظه اشتراکی ثبت می‌گردد که ساختار آن در جدول (۳) آمده است. در روش پیشنهادی درخواست کاربر مطابق قوانین SLA ساخته می‌شود. هر SLA از تعدادی هدف یا SLO[2] تشکیل شده است. ساختار درخواست کاربر در این تحقیق شامل میزان هزینه و محدودیت زمان پاسخگویی بر حسب گروه کاربر است. کاربر می‌تواند به‌صورت هم‌زمان چندین درخواست را به فراهم‌کننده ابر ارائه دهد. جدول (۳) ساختار هر درخواست کاربر را نشان می‌دهد. شبه کد (۲)، الگوریتم برنامه اصلی (MasterApplication) را نشان می‌دهد.

[1] Master application

[2] Service Level Objective

جدول (۲): ساختار جدول اطلاعات برنامه اصلی در مورد برنامه‌های خدماتی در پایگاه داده اشتراکی.

شماره ماشین مجازی	آدرس IP	شماره درخواست	تاریخ و ساعت تخصیص ماشین مجازی	تاریخ و ساعت آزاد شدن ماشین مجازی	هزینه	میانگین زمان پاسخگویی	میزان پردازنده کل	میزان پردازنده استفاده شده

جدول (۳): ساختار درخواست‌های جدید کاربر.

IP کاربر	شناسه درخواست	User SLA	
		SLO1	SLO2
		محدودیت زمان پاسخگویی	هزینه

Algorithm 2: *Master Application*

```
        Function Check(VM[] )
         for i ← 1    to    M    do
           for j ← 1    to    C    do
             if Vm_i^j.RequestID == NULL
               Vm_i^j.Free_DateTime = Date( )
                Return Vm_i^j
             end if
            end for
          end for
            Return Null
        end Function
        Function sendstate(k, req, Data_Share)
   k.requestID = req
   k.Allocate_DateTime = Date( )
  Store_Data(k, Data_Share)
         for i ← 1    to    M    do
           for j ← 1    to    C    do
          Store_Data(Vm_i^j, Data_Share)
           end for
         end for
end Function

        Function Getcommand( Data_Share)
   s = Data_share[scale]
   ip = Data_share[ip]
   C.s = s
   C.ip = ip
return C
end Function

        Function Execcommand(C)
  if C.ip ! = Null
    N = getNode(C.ip)
     create VmList(N)
  end if
  if C.s ! = Null
    N = getNode(C.ip)
     Move_Vm(C.ip, Vm[   ])
    SendReply(Data_Share)
  end if
end Function

        Function SendReply( Data_Share)
  Data_share[scale].Confirm = true
end Function
```

در شبه کد (۲)، تابع Check وظیفه نظارت اولیه بر ماشین‌های مجازی را بر عهده دارد. بدین صورت که برای هر ماشین مجازی بررسی می‌شود که این ماشین مجازی به کاربر یا درخواستی اختصاص یافته یا خیر. در صورتی که ماشین درخواستی برای پاسخگویی نداشته باشد، آزاد اعلام می‌شود و به‌عنوان خروجی تابع در نظر گرفته می‌شود. در صورتی که هیچ ماشین مجازی آزادی وجود نداشته باشد این تابع Null بر می‌گرداند که مقدمه‌ای برای عملیات اضافه کردن سرور است.

تابع SendState وضعیت ماشین مجازی را در حافظه اشتراکی ثبت می‌کند. در این تابع تمامی ماشین‌های مجازی یک به یک مرور شده و وضعیت هر کدام در حافظه اشتراکی جهت تصمیمات آتی ثبت می‌گردد.

تابع Getcommand وظیفه دریافت دستور از کنترل کننده کشسانی را بر عهده دارد. این تابع در بازه زمانی مشخص به حافظه اشتراکی مراجعه کرده و اطلاعات قسمت ثبت دستورات را بررسی می‌کند. در صورت وجود دستور جدید به تناسب به آن پاسخ می‌دهد.

تابع Execcommand وظیفه اجرای فرمان در قسمت Getcommand را به عهده دارد.

تابع Sendreply وظیفه پاسخگویی به دستور صادر شده توسط کنترل کننده کشسانی را بر عهده دارد. این تابع برای هر فرمان پاسخ مناسب را در حافظه اشتراکی می‌نویسد تا کنترل کننده کشسانی نتیجه اجرای فرمان را دریافت کند.

۳-۱-۲- برنامه خدماتی

درخواست کاربر در قالب یک برنامه خدماتی[1] بر روی ماشین مجازی عمل می‌کند. بدین صورت که برای هر درخواست دسترسی به منبع یا ماشین مجازی به رابط نیاز دارد، که توسط یک برنامه خدماتی این دسترسی فراهم می‌گردد.

اختصاص ماشین مجازی به یک درخواست و بارگذاری برنامه خدماتی روی آن، توسط برنامه اصلی انجام می‌شود. کنترل وضعیت ماشین مجازی متعلق به هر برنامه خدماتی به عهده برنامه اصلی است.

۳-۲- کنترل کننده کشسانی خودکار منابع

کنترل کننده کشسانی خودکار منابع، وظیفه تأمین منابع و تصمیم‌گیری در مورد منابع را به عهده دارد. به همین دلیل ماژول کنترل کننده کشسانی به‌صورت مداوم به پایگاه داده اشتراکی سرکشی می‌کند و توسط فضای مشترک بین این ماژول و برنامه اصلی، اطلاعات درخواست‌ها را دریافت می‌کند و به واسطه آن تصمیم‌گیری می‌کند. یک سرور مستقل (Cloud Front-End) در اختیار ماژول کنترل کننده کشسانی خودکار قرار دارد. این ماژول دارای دو وظیفه مهم است:

۱- ارسال فرامین به برنامه اصلی.

۲- تصمیم‌گیری در مورد نوع مقیاس بندی.

برنامه اصلی به‌عنوان یک عنصر زیردست یا فرمانبر کنترل کننده کشسانی خودکار منابع عمل می‌کند. پس باید ماژول کنترل کننده کشسانی فرامین مورد نظر را در مرکز داده مشترک قرار دهد تا برنامه اصلی آن‌ها را بخواند و اجرا کند. همچنین برنامه اصلی با اجرای فرمان به کنترل کننده کشسانی پیام تأیید اجرای فرمان را می‌دهد.

جدول (۴) ساختار فرمان‌های صادره و پاسخ‌های وارده درخواست کاربر را نشان می‌دهد.

در عملیات Scale-out پس از ایجاد یک سرور توسط کنترل کننده کشسانی، آدرس IP این سامانه به برنامه اصلی ارسال می‌شود. برنامه اصلی پس از دریافت این آدرس IP، به واسطه این آدرس به سرور دسترسی پیدا می‌کند و ماشین‌های مجازی روی این سرور را در اختیار درخواست‌های جدید قرار می‌دهد.

در عملیات Scale-in ابتدا باید از عدم اختصاص ماشین‌های مجازی سرور مورد نظر مطمئن شد یا به هر نحو آن‌ها را به سروری دیگر انتقال داد.

به همین دلیل ابتدا کنترل کننده کشسانی خودکار منابع، فرمان Scale-in را به‌همراه شماره سرور صادر می‌کند. سپس برنامه اصلی باید ماشین‌های مجازی مربوط به این سرور را به سرورهای دیگر منتقل کند و پیام تأیید را به‌عنوان پاسخ در پایگاه داده اشتراکی قرار دهد تا کنترل کننده کشسانی آن‌را به‌عنوان تأیید دریافت کرده و سرور را حذف کند.

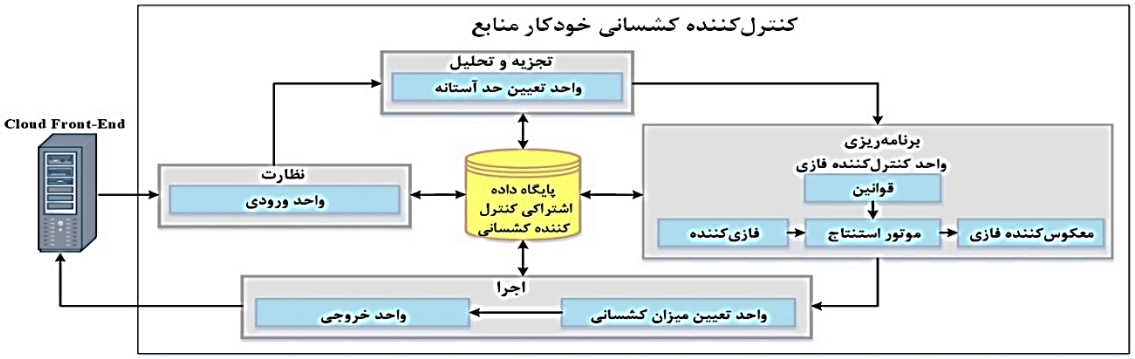
شکل (۲) ساختار کنترل کننده کشسانی خودکار منابع را نشان می‌دهد.

در کنترل کننده کشسانی خودکار، چهار مرحله مهم وجود دارد: مرحله نظارت (واحد ورودی)، مرحله تجزیه و تحلیل (واحد تعیین حد آستانه)، مرحله برنامه‌ریزی (واحد کنترل کننده فازی) و مرحله اجرا (واحد تعیین میزان کشسانی) که در زیر به جزئیات تشریح می‌شوند.

[1] Slave Application

جدول (۴): ساختار فرمان‌های مربوط به کنترل کننده کشسانی.

عملیات	مسیر فرمان	توضیحات
Scale-out	ماژول کنترل کننده کشسانی ← حافظه ← برنامه اصلی	ایجاد یک سرور جدید توسط کنترل کننده کشسانی و معرفی IP آن به برنامه اصلی برای اختصاص ماشین‌های مجازی به برنامه‌های خدماتی
Scale-in	ماژول کنترل کننده کشسانی ← حافظه ← برنامه اصلی	صدور فرمان خاموش کردن سروری خاص و انتظار برای آمادگی برنامه اصلی
پاسخ برنامه به Scale-in	برنامه اصلی← حافظه ← ماژول کنترل کننده کشسانی	اعلام آمادگی برای خاموش کردن یک سرور



شکل (۲): ساختار کنترل کننده کشسانی خودکار منابع.

۳-۲-۱- مرحله نظارت (واحد ورودی)

طبق ساختار سامانه کنترل در این واحد جدولی از داده‌های جاری و خطاهای ناشی از بازخورد سامانه ثبت می‌گردد تا با استفاده از آن حد آستانه سامانه فازی تصحیح شده و تصمیم‌گیری آتی دقیق‌تر شود. در این واحد پس از دریافت اطلاعات از حافظه اشتراکی، میزان تخطی از شرایط پذیرش سرویس در مورد هزینه و زمان پاسخگویی محاسبه می‌شود. زمان پاسخگویی به درخواست i ام از طریق رابطه (۱)، محاسبه می‌شود و میزان هزینه مربوط به درخواست i ام از طریق رابطه (۲)، به‌دست می‌آید.

$$ResponseTime_i = VM_k.Allocate\_Datetime_i - VM_k.Free\_Datetime_i \quad (۱)$$

$$Cost_i = ResponseTime_i \times VM_k.Cost \quad (۲)$$

میزان تخطی از شرایط پذیرش سرویس در مورد زمان پاسخگویی توسط رابطه (۳) محاسبه می‌گردد. به همین ترتیب میزان تخطی از شرایط پذیرش سرویس در مورد هزینه نیز توسط رابطه (۴) محاسبه می‌گردد.

$$SLAViolation_i = ResponseTime_i - Req\_Rmax_i \quad (۳)$$

$$SLAViolation_i = Cost_i - Req\_Cost_i \quad (۴)$$

در ادامه مجموع تخطی از شرایط سرویس با استفاده از رابطه (۵) محاسبه می‌گردد.

$$SLAV = \sum_{i=1}^{N} SLAViolation_i$$

$$SLAViolation_i = \begin{cases} 1 & SLAViolation_i > 0 \\ 0 & SLAViolation_i \le 0 \end{cases} \quad (۵)$$

در ادامه میزان بهره‌وری کل نیز از رابطه (۶) محاسبه می‌شود.

$$Utilization_k = \frac{VM_k.Used\_MIPS}{VM_k.Available\_MIPS}$$

$$Utilization = \frac{\sum_{i=1}^{M}\sum_{j=1}^{C} Utilization_i^j}{M * C} \quad (۶)$$

در رابطه (۶)، $Utilization_i^j$ بهره‌وری ماشین مجازی j ام در سرور i ام است. پس از محاسبه تعداد تخطی و بهره‌وری، اطلاعات بار کاری و تخطی، در جدول سوابق به‌صورت جدول (۵) ثبت می‌گردد.

جدول (۵): جدول سوابق.

مقدار مقیاس‌بندی	نوع مقیاس‌بندی	بهره‌وری	بار کاری	میزان تخطی	زمان

نوع مقیاس‌بندی و مقدار آن در مراحل بعد در جدول سوابق درج می‌گردد. پس از محاسبات بالا جدول سوابق برای واحد کنترل کننده فازی ارسال می‌شود.

۳-۲-۲- مرحله تجزیه و تحلیل (واحد تعیین حد آستانه)

در ابتدای راه‌اندازی سامانه مقدار حداقل و حداکثر برای حد آستانه و مقادیر حد آستانه پایین و بالای تخطی از شرایط سرویس به‌صورت پیش‌فرض مشخص می‌گردد، که در رابطه (۷) نشان داده شده است.

$$\begin{aligned} &SLAV\_Min = 0 \\ &SLAV\_Max = C \\ &E = \frac{C}{20} \\ &SLAV\_TH = \frac{C}{4} + E \\ &SLAV\_TL = \frac{C}{4} - E \end{aligned} \quad (۷)$$

در مشاهدات بعدی سامانه، با توجه به جدول سوابق دریافت شده میزان حد آستانه تخطی از شرایط سرویس تغییر می‌کند. در صورتی که بهره‌وری کمتر از ۳۰ درصد باشد (Utilization<30) رابطه (۸) وظیفه تنظیم حد آستانه پایین و بالا را بر عهده دارد.

$$if\ Utilization < 30 \begin{cases} SLAV\_TH = Min\ (SLAV\_TH + Utilization \times E, SLAV\_Max \\ SLAV\_TL = Max\ (SLAV\_TH - (Utilization \times E)\ , SLAV\_Min) \end{cases} \quad (۸)$$

در صورتی که بهره‌وری بین ۳۰ درصد تا ۷۰ درصد باشد (30<Utilization<70) رابطه (۹) وظیفه تنظیم حد آستانه پایین و بالا را به عهده دارد.

$$if\ \ 30 \le Utilization < 70 \begin{cases} SLAV\_TH = SLAV\_TH \\ SLAV\_TL = SLAV\_TL \end{cases} \quad (۹)$$

و درنهایت در صورتی که بهره‌وری بالاتر از ۷۰ یا مساوی آن درصد باشد (Utilization>70) رابطه (۱۰) وظیفه تنظیم حد آستانه پایین و بالا را به عهده دارد.

$$if\ Utilization \ge 70 \begin{cases} SLAV\_TH = Max\ (SLAV\_TH - Utilization \times C, SLAV\_Min \\ SLAV\_TL = Max\ (C - (Utilization \times C) - E\ , SLAV\_Min) \end{cases} \quad (۱۰)$$

با توجه به رابطه (۱۰)، به هر میزان بهره‌وری کل افزایش یابد سخت‌گیری بیشتری در مورد تخطی از شرایط سرویس انجام گیرد، به همین دلیل در این رابطه در صورت افزایش بهره‌وری، حد آستانه بالا کاهش می‌یابد تا سخت‌گیری بیشتری در مورد تخطی از شرایط سرویس شود. در مورد بار کاری هم باید حد آستانه مشخص گردد. در ابتدای راه‌اندازی سامانه مقدار حداقل و حداکثر برای حد آستانه و مقادیر حد آستانه پایین و بالا مربوط به بار کاری به‌صورت پیش‌فرض مشخص می‌گردد، که در رابطه (۱۱) نشان داده شده است.

$$\begin{aligned} &WL\_Min = 0 \\ &WL\_Max = C \times M \\ &R = \frac{C \times M}{5} \\ &WL\_TH = WL\_Max - R \\ &WL\_TL = WL\_Min + R \end{aligned} \quad (۱۱)$$

در مشاهدات بعدی سامانه، با توجه به جدول سوابق دریافت شده میزان حد آستانه بار کاری تغییر می‌کند. در صورتی که بهره‌وری کمتر از ۳۰ درصد باشد (Utilization<30) رابطه (۱۲) وظیفه تنظیم حد آستانه پایین و بالا را بر عهده دارد.

$$if\ Utilization < 30 \begin{cases} WL\_TH = Min\ (WL\_TH + Utilization \times R, WL\_Max) \\ WL\_TL = Max\ (WL\_TH - (Utilization \times E)\ , WL\_Min) \end{cases} \quad (۱۲)$$

در صورتی که بهره‌وری بین ۳۰ درصد تا ۷۰ درصد باشد (30<Utilization<70) رابطه (۱۳) وظیفه تنظیم حد آستانه پایین و بالا را به عهده دارد.

$$if\ \ 30 \le Utilization < 70 \begin{cases} WL\_TH = WL\_TH \\ WL\_TL = WL\_TL \end{cases} \quad (۱۳)$$

و درنهایت در صورتی که بهره‌وری بالاتر از ۷۰ یا مساوی آن درصد باشد (Utilization>70) رابطه (۱۴) وظیفه تنظیم حد آستانه پایین و بالا را بر عهده دارد.

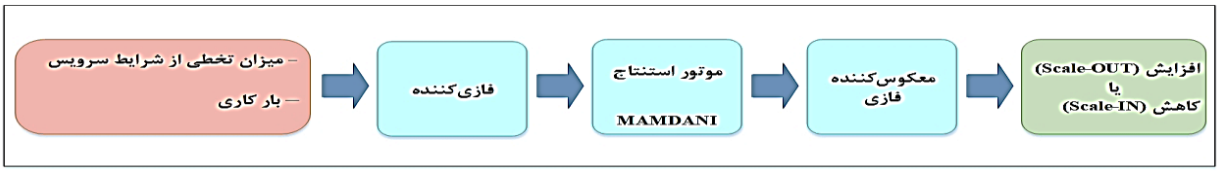
$$if\ Utilization \ge 70 \begin{cases} WL\_TH = Max\ (WL\_TH - (Utilization \times R), WL\_Min) \\ WL\_TL = Max\ (WL\_TL - (Utilization \times R)\ , SLAV\_Min) \end{cases} \quad (۱۴)$$

پس از تعیین حد آستانه دو متغیر فازی (مقدار تخطی از شرایط سرویس و بار کاری)، این حدود به سامانه فازی اعلام می‌گردد تا در مرحله استنتاج استفاده شود.

۳-۲-۳- مرحله برنامه‌ریزی (واحد کنترل کننده فازی)

این واحد پس از دریافت حدود آستانه از واحد تعیین آستانه با استفاده از سامانه فازی در مورد نوع مقیاس‌بندی تصمیم‌گیری می‌کند. با محاسبه میزان حدود آستانه، در این بخش باید در مورد کشسانی تصمیم‌گیری شود. بدین معنی که منطق فازی استفاده شده در این واحد برای نوع مقیاس‌بندی تصمیم‌گیری می‌کند. برای تشخیص نیاز به افزایش یا کاهش سرورها از ساختار فازی استفاده می‌کند. ساختار یک سامانه کنترل فازی کامل از بلوک‌های زیر تشکیل یافته است: فازی‌کننده، موتور

استنباط، معکوس کننده فازی. شکل (۳) ساختار کنترل کننده منطق فازی در روش پیشنهادی را نشان می‌دهد. ماژول فازی کننده مقادیر اولیه ورودی‌های کنترلی را به مقادیر فازی تبدیل می‌کند. در روش پیشنهادی مقدار تخطی از شرایط سرویس و بار کاری به‌عنوان ورودی ساختار کنترل کننده منطق فازی هستند. به واسطه موتور استنباط ممدانی تصمیم‌گیری می‌شود که بر اساس داده‌های ورودی نیاز به افزایش یا کاهش سرور است.



شکل (۳): ساختار جهت‌گیری با استفاده از منطق فازی در روش پیشنهادی.

اجزای واحد کنترل‌کننده فازی شامل متغیرهای ورودی، متغیر خروجی، سامانه استنتاج و تصمیم‌گیر ممدانی، درنهایت نحوه معکوس‌کردن فازی به شرح زیر است:

- **متغیرهای ورودی سامانه فازی**

متغیرهای ورودی در ساختار فازی، دو متغیر مقدار تخطی از شرایط سرویس (SLAV) و بار کاری (Workload) می‌باشند که در زیر به آن‌ها اشاره می‌شود:

۱) میزان تخطی از شرایط سرویس (SLAV)

با توجه به اینکه حدود آستانه تخطی از شرایط سرویس و مقدار کمترین و بیشترین آن در مرحله قبل مشخص شده است، محدوده این متغیر ورودی بین SLAV_Min تا SLAV_Max در نظر گرفته شد.

۲) بار کاری (Workload)

با توجه به اینکه حدود آستانه بار کاری و مقدار کمترین و بیشترین آن در مرحله قبل مشخص شده است، محدوده این متغیر ورودی بین WL_Min تا WL_Max در نظر گرفته شد.

- **متغیرخروجی سامانه فازی**

خروجی سامانه فازی تصمیم‌گیری در مورد افزایش یا کاهش سرورها است.

- **سامانه استنتاج**

در پایگاه قواعد به مجموعه «اگر - آنگاه» فازی گفته می‌شود که قلب سامانه استنتاج فازی را تشکیل می‌دهد. دو روش عمده برای تعیین قواعد فازی وجود دارد: یکی استفاده از دانش خبره و دیگری استفاده از آموزش‌های خودسازمانده، مانند الگوریتم‌های نوین و شبکه عصبی که در اینجا از روش اول برای تعیین قواعد فازی استفاده شده است. ذکر این نکته ضروری است که در روش ممدانی، خروجی به شکل فازی تعریف می‌شود. در این بخش باید یکسری قانون برای تصمیم‌گیری مطرح شود. با توجه به متغیرهای ورودی قوانین موجب تولید یک متغیر خروجی فازی خواهند شد، که درنهایت این امر پس از معکوس کردن فازی عددی برای اولویت خود تولید می‌کند. قوانین مورد استفاده برای محاسبه نوع مقیاس‌بندی در شکل (۴) آمده است.

1. if (SLAV is Low) and (Workload is Low) then (Scale_Type is Remove)(1)
2. if (SLAV is Low) and (Workload is Normal) then (Scale_Type is NoChange)(1)
3. if (SLAV is Low) and (Workload is High) then (Scale_Type is Add)(1)
4. if (SLAV is High) and (Workload is Low) then (Scale_Type is Add)(1)
5. if (SLAV is High) and (Workload is Normal) then (Scale_Type is Add)(1)
6. if (SLAV is High) and (Workload is High) then (Scale_Type is Add)(1)

شکل (۴): ساختار قوانین ممدانی برای استنتاج فازی.

- **معکوس‌کننده فازی (غیر فازی‌کننده)**

وظیفه قسمت غیر فازی کننده، استنتاج است. روش‌های مختلفی برای استنتاج وجود دارد برای مثال ممکن است هر قانونی که صحیح‌تر است به‌عنوان خروجی انتخاب شود. ولی بهتر است که میانگینی از همه قوانین در نظر گرفته شود. به این روش، روش مرکز ثقل نیز گفته می‌شود. خروجی در این روش از رابطه (۱۵) به‌دست می‌آید.

$$Output = \frac{\Sigma x_i \mu(x_i)}{\Sigma \mu(x_i)} \qquad (۱۵)$$

در رابطه فوق، μ(xi) مقدار به‌دست آمده در هر محدوده از خروجی است. سپس برای تصمیم‌گیری در مورد مقیاس‌بندی از رابطه شرطی (۱۶) استفاده می‌شود که بر اساس مقدار Output تصمیم‌گیری می‌کند.

$$Scale\_Type = \begin{cases} Add\ (Scale-OUT) & Output \geq 0.8 \\ NoChange & 0.2 < Output < 0.8 \\ Remove\ (Scale-IN) & Output \leq 0.2 \end{cases} \qquad (۱۶)$$

۳-۲-۴- مرحله اجرا (واحد تعیین میزان کشسانی)

پس از تعیین نوع مقیاس‌بندی باید میزان آن تعیین شود. برای تعیین میزان تغییر مقیاس از جدول سوابق و میزان بار کاری استفاده می‌شود، ابتدا میزان بار کاری آتی توسط الگوریتم ARMA پیش‌بینی می‌شود. مدل ARMA همیشه با دو پارامتر p و q استفاده می‌شود. رابطه (۱۷) نحوه محاسبه بار کاری در مرحله بعد با استفاده از ARMA را نمایش می دهد.

$$W_t = C + \epsilon_t + \sum_{i=1}^{p} \varphi_i W_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \qquad (۱۷)$$

$\theta 1 \ldots \theta q$ پارامترهای اول مدل پیش‌بینی هستند که بر حسب نوع بار کاری تعیین می‌شوند. $\epsilon_t \ldots \epsilon_{t-q}$ مقادیر تصادفی با توزیع نرمال و میانگین صفر هستند. $\phi_1 \ldots \phi_p$ پارامترهای دوم مدل پیش‌بینی هستند که بر حسب نوع بار کاری تعیین می‌شوند. مقدار p و q هم در روش پیشنهادی ۲۰ در نظر گرفته شده است. با توجه به نوع مقیاس پذیری در زمان t ($Scale\_type_t$) و میزان بار کاری، مقدار افزایش و کاهش سرورها به‌صورت زیر محاسبه می‌گردد. ابتدا متغیر A در رابطه (۱۸) بر اساس همانند بودن نوع مقیاس‌بندی در زمان جاری و زمان قبلی تشکیل می‌شود. در ادامه نیز مقدار متغیر B در رابطه (۱۹) بر اساس وضعیت مقیاس‌بندی ایجاد می‌شود.

$$A = \begin{cases} 1 & if\ Scale\_type_t = Scale\_type_{t-1} \\ 0 & Otherwise \end{cases} \qquad (۱۸)$$

$$B = \begin{cases} 1 & if\ Scale\_type_t = OUT \\ 0 & if\ Scale\_type_t = IN \end{cases} \qquad (۱۹)$$

سپس بر اساس A و B، محاسبه نهایی انجام می‌شود. در محاسبات نهایی میزان افزایش U_t و میزان کاهش L_t سرورها است، که از طریق رابطه (۲۰ و ۲۱) به‌دست می‌آید.

$$U_t = \begin{cases} U_{t-1} + \left|W_t - Predict_{W_{t+1}}\right| \\ \quad if\ A = 1\ \ and\ \ B = 1 \\ \left|W_t - Predict_{W_{t+1}}\right| \\ \quad if\ A = 0\ \ and\ \ B = 1 \\ 0 \qquad Otherwise \end{cases} \qquad (۲۰)$$

$$L_t = \begin{cases} L_{t-1} + \left|W_t - Predict_{W_{t+1}}\right| \\ \quad if\ A = 1\ \ and\ \ B = 0 \\ \left|W_t - Predict_{W_{t+1}}\right| \\ \quad if\ A = 0\ \ and\ \ B = 0 \\ 0 \qquad Otherwise \end{cases} \qquad (۲۱)$$

با توجه به مقادیر افزایش U_t و کاهش L_t سرورها، مقادیر سرورها به ترتیب از طریق رابطه (۲۲ و ۲۳) انجام می‌شود.

$$Server\_count = \begin{cases} \left|\frac{U_t}{C}\right| & if\ B = 1 \\ 0 & Otherwise \end{cases} \qquad (۲۲)$$

$$Server\_count = \begin{cases} \left|\frac{L_t}{C}\right| & if\ B = 0 \\ 0 & Otherwise \end{cases} \qquad (۲۳)$$

پس از تعیین نوع مقیاس‌بندی و مقدار پیش‌بینی شده آن، تصمیم برای واحد خروجی ارسال می‌گردد. واحد خروجی وظیفه ذخیره فرمان را در حافظه اشتراکی بر عهده دارد تا برنامه اصلی با دریافت این فرمان فعالیت مربوط به آن را انجام دهد. نکته مهم در حذف یک سرور یا ماشین فیزیکی این است که سروری حذف می‌گردد که تعداد ماشین مجازی کمتری به‌صورت فعال در آن موجود باشد و ماشین‌های مجازی فعال در آن نیز به‌صورت مهاجرت زنده به سرورهای دیگر منتقل می‌شوند.

۴- ارزیابی عملکرد و نتایج

در این بخش نتایج حاصل از پیاده‌سازی روش AREC برای بهبود خاصیت کشسانی با استفاده از سامانه کنترل فازی مبتنی بر حد آستانه مورد بررسی قرار می‌گیرد. محیط مورد استفاده شبیه‌سازی NetBeans است. به‌منظور شبیه‌سازی دقیق و قابل توسعه ابر از ابزار CloudSim استفاده شده است. به دلیل اینکه ابزار CloudSim با زبان جاوا نوشته شده است، محیط توسعه NetBeans به‌عنوان محیط برنامه‌نویسی جاوا انتخاب شده است. در ادامه به بررسی پارامترهای شبیه‌سازی، بار کاری مورد استفاده قرار گرفته در این پژوهش و نتیجه شبیه‌سازی پرداخته می‌شود.

۴-۱- تنظیمات تجربی

ساختار مرکز داده (Data Center) مورد استفاده در شبیه‌سازی روش پیشنهادی در جدول (۶) مشخص شده است. در مرکز داده یک نوع سرور یا میزبان وجود دارد. در جدول (۷) مشخصات میزبان یا ساختار فیزیکی به‌صورت جامع آمده است. به هر میزان سخت‌افزار میزبان قوی‌تر و از سطح بالاتری برخوردار باشد، میزان هزینه دسترسی به منابع مربوط به ماشین مجازی موجود در میزبان افزایش می‌یابد.

جدول (۶): مشخصات مرکز داده.

معماری	X64
سامانه عامل	Cloud Linux
مدیریت ماشین‌های مجازی	XEN
تعداد سرورها (M)	10
تعداد ماشین‌های مجازی هر سرور (C)	15

جدول (۷): مشخصات میزبان.

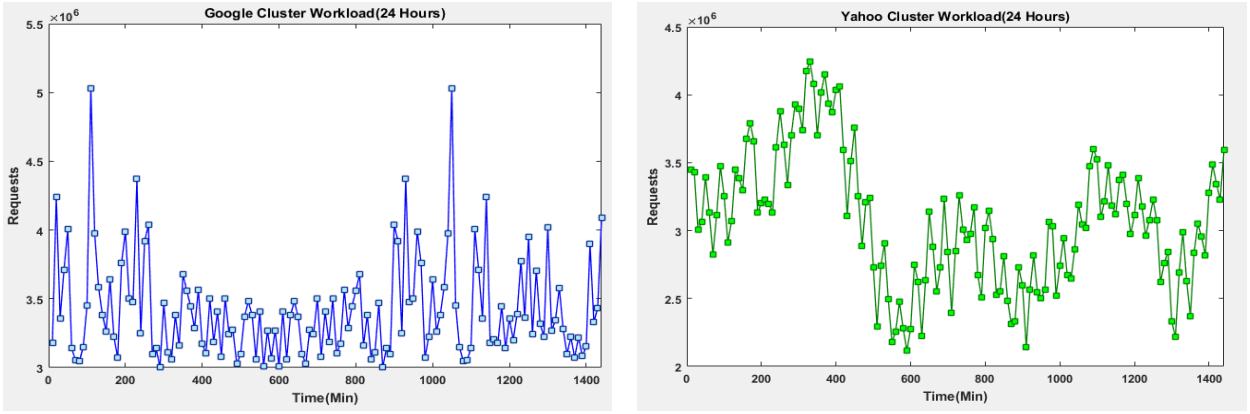
پهنای باند	حافظه اصلی (GB)	فرکانس (MIPS)	تعداد هسته
10 Gbit/s	32	4096	8

۴-۱-۱- بار کاری

در شبیه‌سازی روش پیشنهادی از دو بار کاری واقعی استفاده شده است. توزیع بارهای کاری واقعی شامل توزیع تقاضاهای Google Cluster [۳۸] و Yahoo Cluster [۳۹] است. دلیل استفاده از این بارهای کاری این است که شباهت بسیار زیادی به بارهای کاری معمول مورد پردازش در رایانش ابری دارند که موجب می‌شود نتایج به‌دست آمده در این مقاله واقعی و کاربردی‌تر باشند. شکل (۵) توزیع بارهای کاری واقعی را در ۲۴ ساعت از ساعات پردرخواست برای Google Cluster و Yahoo Cluster نشان می‌دهد. داده‌های Google Cluster شامل ۴ روز بار کاری و ۶۲۵۳۷۷۱۴۲۹ درخواست می‌باشد. داده‌های Yahoo Cluster شامل ده روز بار کاری و ۳۷۲۲۵۸۷۴۶۳۸ درخواست است. در شبیه‌سازی هر ۱۵ دقیقه اطلاعات برداشت شده (۹۶ بازه زمانی) و تعداد تقسیم بر ۱۰۰۰ شده است.

۴-۲- معیارهای کارایی

شاخص‌های ارزیابی کل ساختار عبارتست از: میانگین زمان پاسخگویی (بر طبق رابطه ۱ محاسبه می‌شود)، میانگین هزینه (بر طبق رابطه ۲ محاسبه می‌شود)، میزان تخطی از شرایط پذیرش سرویس (بر طبق روابط ۳ تا ۵ محاسبه می‌شود) و تغییرات حدود آستانه (بر طبق روابط ۷ تا ۱۴ محاسبه می‌شود).



(a) (b)

شکل (۵): توزیع بارهای کاری واقعی در ۲۴ ساعت از ساعات پردرخواست برای Google Cluster و Yahoo Cluster .

۴-۳- نتایج شبیه‌سازی

در این بخش نتایج شبیه‌سازی برای دو بار کاری واقعی بررسی می‌گردد. نتایج روش پیشنهادی AREC با دو الگوریتم AE-AHP[1] و AE-AL[2] مقایسه می‌شوند. علت انتخاب این دو الگوریتم برای مقایسه این است که همانند روش پیشنهادی روی ساختار معماری مشابه عمل می‌کنند، با این تفاوت که الگوریتم AE-AHP [۲۴] در قسمت تأمین منابع در مدیریت کشسانی از روش تحلیل سلسله‌مراتبی استفاده می‌کند و الگوریتم AE-AL [۳۵] در قسمت تأمین منابع در مدیریت کشسانی از روش یادگیری خودکار استفاده می‌کند. جهت ارزیابی روش پیشنهای ۴ سناریو با ساختار جدول (۸) تشکیل شد. در شبیه‌سازی هر سناریو یک معیار مهم ارزیابی در سه الگوریتم AREC، AE-AHP و AE-AL مورد بررسی قرار می‌گیرد.

جدول (۸): سناریوهای مورد ارزیابی.

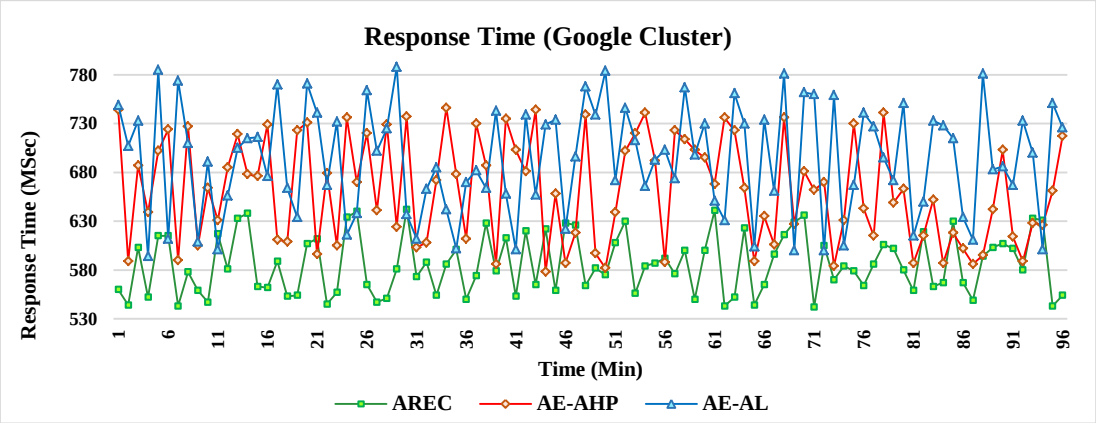
سناریو	هدف
شماره یک	مقایسه میانگین زمان پاسخگویی
شماره دو	مقایسه میانگین هزینه
شماره سه	میزان تخطی از شرایط پذیرش سرویس
شماره چهار	تغییرات حدود آستانه

۴-۳-۱- ارزیابی زمان پاسخگویی

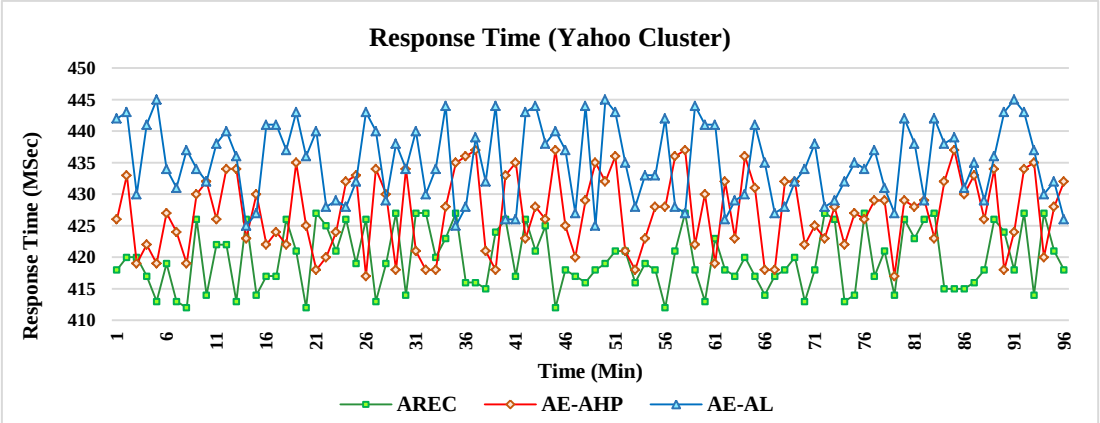
زمان پاسخگویی به‌عنوان یکی از مؤثرترین اهداف شرایط سرویس نقش مؤثری در انتخاب ماشین مجازی دارد. در صورتی که زمان پاسخگویی مورد نظر درخواست که با نام حد مجاز زمان پاسخگویی شناسایی می‌شود، حاصل نشود باید مقیاس‌بندی انجام شود. در این بخش زمان پاسخگویی روش پیشنهادی در دو بار کاری Google Cluster و Yahoo Cluster بررسی می‌شود و با دو الگوریتم AE-AHP و AE-AL مقایسه می‌شود. شکل‌های (۶ و ۷) زمان پاسخگویی در طول شبیه‌سازی دو بار کاری Google

[1] Automatic Elasticity Provisioning Analytic Hierarchy Process
[2] Automatic Elasticity Automata Learning

Cluster و Yahoo Cluster را در سه الگوریتم هدف نمایش می‌دهد. میانگین زمان پاسخگویی روش پیشنهادی AREC در مقایسه با دو الگوریتم AE-AHP و AE-AL به ترتیب ۱۱٪ و ۱۵٪ تحت بار کاری Google Cluster و ۲٪ و ۳٪ تحت بار کاری Yahoo Cluster کاهش یافته است.

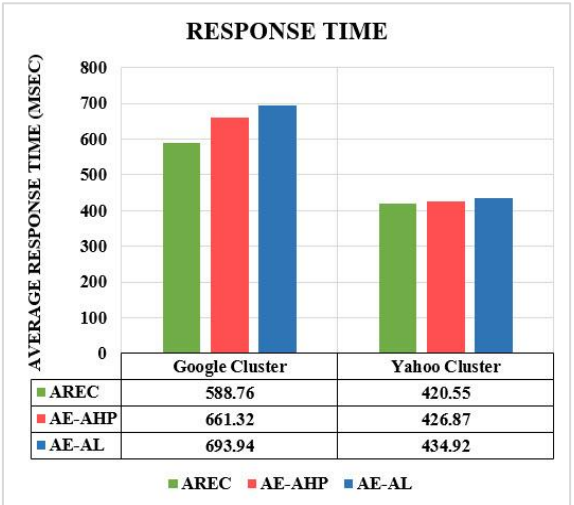


شکل (۶): زمان پاسخگویی در طول شبیه‌سازی ۲۴ ساعت کاری سری Google Cluster.



شکل (۷): زمان پاسخگویی در طول شبیه‌سازی ۲۴ ساعت کاری سری Yahoo Cluster.

به دلیل عملکرد مطلوب سامانه کنترل فازی به‌عنوان تصمیم‌گیر سامانه، همچنین ایجاد ساختار حد آستانه متغیر و متناسب با شرایط تخطی از سرویس برای آن، عمل کشسانی منابع به درستی انجام می‌شود، که این امر موجب تأمین بهینه منابع برای درخواست‌ها شده و به تناسب اختصاص منبع دقیق‌تری صورت‌گرفته و درنهایت سرعت پاسخگویی افزایش می‌یابد. با توجه به نتایج، روش پیشنهادی کارایی بهتری در مورد زمان پاسخگویی دارد.



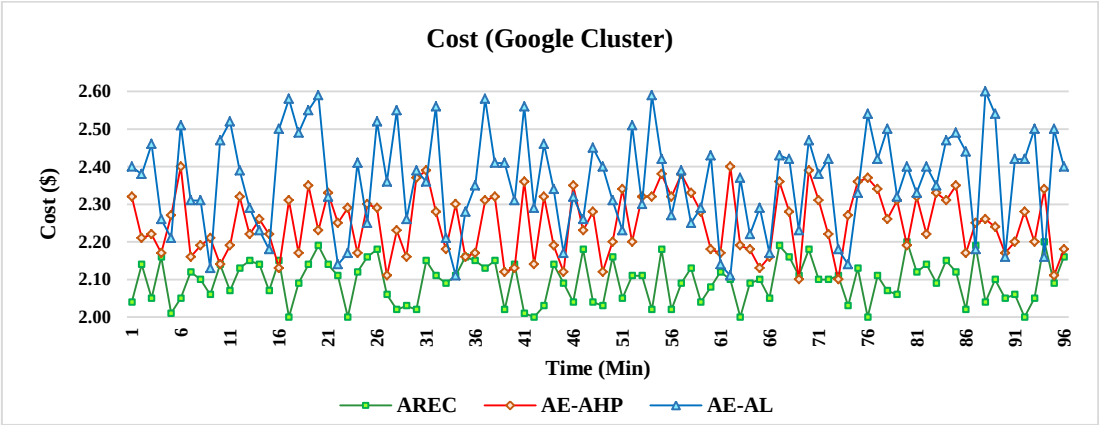
شکل (۸): میانگین زمان پاسخگویی در طول شبیه‌سازی ۲۴ ساعت کاری در دو بار کاری.

۴-۳-۱-۱- ارزیابی میانگین زمان پاسخگویی

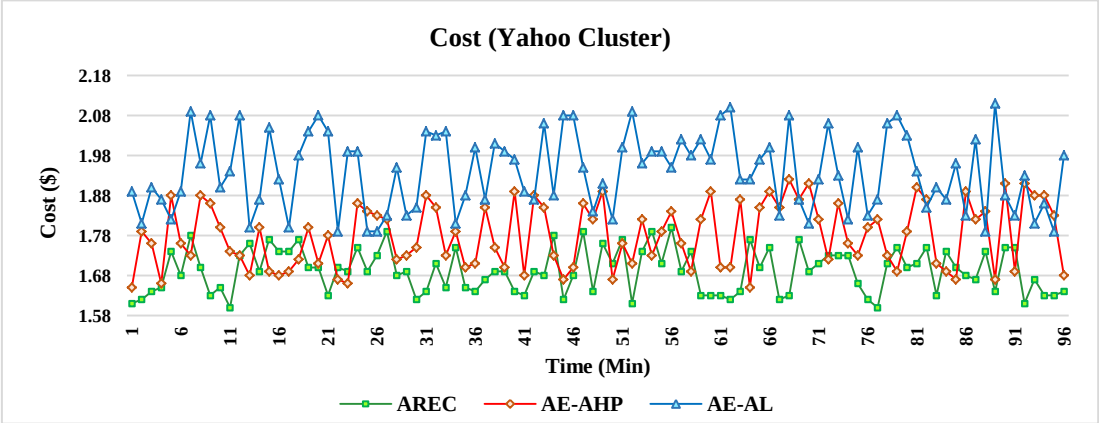
شکل (۸) میانگین زمان پاسخگویی در طول شبیه‌سازی دو بار کاری واقعی را در سه الگوریتم هدف نمایش می‌دهد. در کل میانگین زمان پاسخگویی روش پیشنهادی AREC در مقایسه با دو الگوریتم AE-AHP و AE-AL به ترتیب ۶/۵٪ و ۹٪ کاهش یافته است.

۴-۳-۲- ارزیابی هزینه

اصلی‌ترین شرط سرویس میزان هزینه مورد توافق کاربر است. در صورتی که هزینه سرویس‌دهی افزایش یابد، به‌طور مشخص سود سرویس دهنده کاهش و به همان میزان موجب نارضایتی کاربر از ساختار سرویس‌دهی می‌شود. در این بخش میزان هزینه در بار کاری واقعی در سه روش AREC، AE-AHP و AE-AL بررسی می‌شود. شکل‌های (۹ و ۱۰) میزان هزینه در طول شبیه‌سازی دو بار کاری Google Cluster وYahoo Cluster را در سه الگوریتم هدف نمایش می‌دهد. میزان هزینه روش پیشنهادی AREC در مقایسه با دو الگوریتم AE-AHP و AE-AL به ترتیب ۷٪ و ۱۱٪ تحت بار کاری Google Cluster و ۵٪ و ۱۳٪ تحت بار کاری Yahoo Cluster کاهش یافته است.



شکل (۹): میزان هزینه در طول شبیه‌سازی ۲۴ ساعت کاری سری Google Cluster.

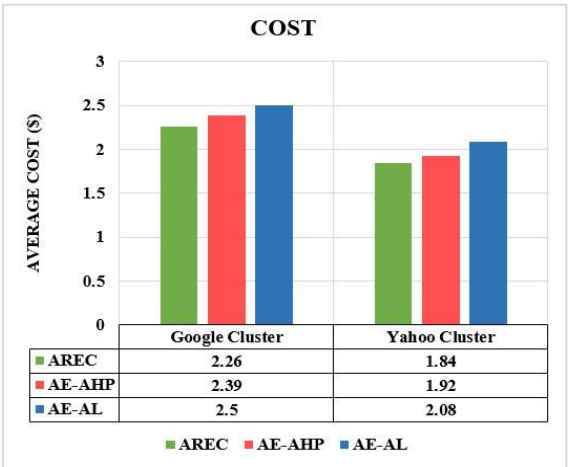


شکل (۱۰): میزان هزینه در طول شبیه‌سازی ۲۴ ساعت کاری سری Yahoo Cluster.

با توجه به نتایج، روش پیشنهادی کارایی بهتری در مورد میزان هزینه دارد. در الگوریتم پیشنهادی به دلیل در نظر گرفتن هزینه به‌عنوان یکی از معیارهای شاخص شرایط پذیرش سرویس و استفاده از تخطی در آن و به نسبت سخت‌گیری تعیین حد آستانه و استفاده از سامانه کنترل فازی، بیشترین دقت را در مورد هزینه تأمین منابع اعمال می‌کند. به همین دلیل دقت بالایی در انتخاب بهترین مقیاس پذیری وجود دارد که هزینه مورد نظر کاربران را برآورده نماید.

۴-۳-۲-۱- ارزیابی میانگین هزینه

شکل (۱۱) میانگین هزینه در طول شبیه‌سازی دو بار کاری واقعی را در سه الگوریتم هدف نمایش می‌دهد.

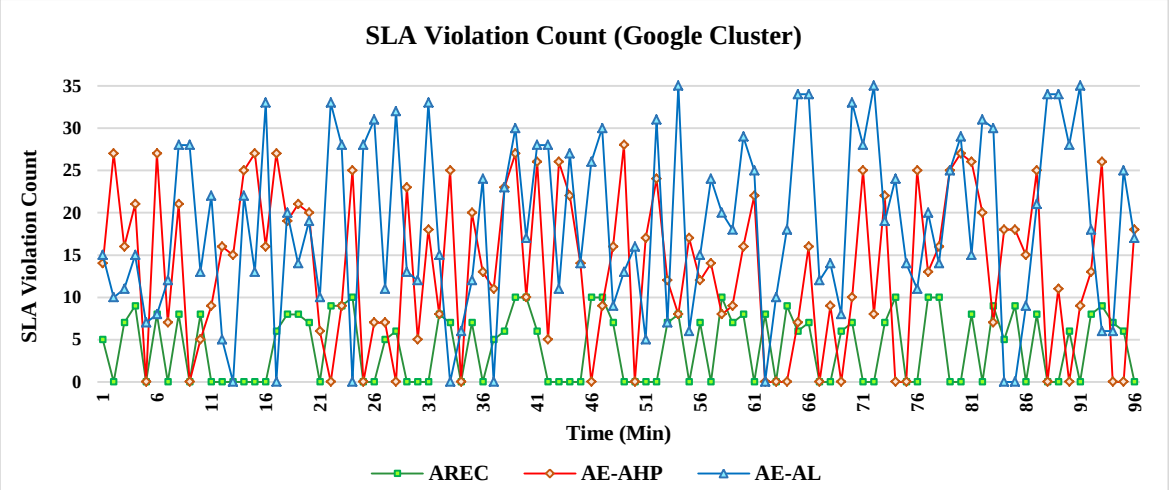


شکل (۱۱): میانگین هزینه در طول شبیه‌سازی ۲۴ ساعت کاری در دو بار کاری.

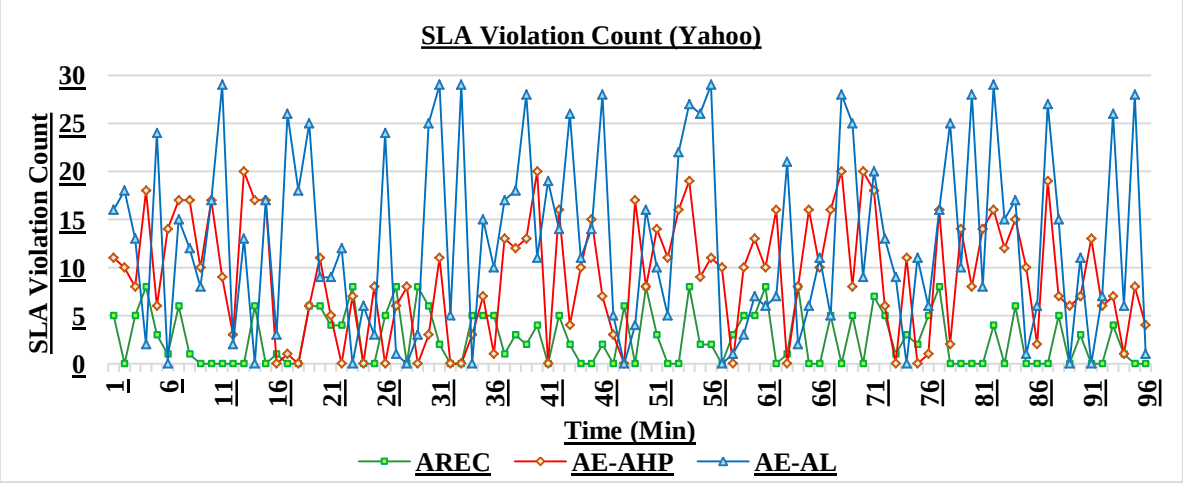
در کل میانگین هزینه روش پیشنهادی AREC در مقایسه با دو الگوریتم AE-AHP و AE-AL به ترتیب ۶٪ و ۱۲٪ کاهش یافته است.

۴-۳-۳- ارزیابی تخطی از شرایط پذیرش سرویس

رخداد تخطی به معنای کاهش کیفیت سرویس‌دهی است و به‌طور مشخص کاهش کیفیت سرویس‌دهی و تخطی از شرایط پذیرش سرویس موجب عدم رضایت کاربر می‌گردد. در این بخش میزان تخطی از شرایط سرویس در بار کاری واقعی در سه روش AREC، AE-AHP و AE-AL ارزیابی می‌شود. روش پیشنهادی در بار کاری Google Cluster وYahoo Cluster بررسی می‌شود و با دو الگوریتم AE-AHP و AE-AL مقایسه می‌شود. شکل‌های (۱۲ و ۱۳) میزان تخطی از شرایط سرویس را در طول شبیه‌سازی دو بار کاری Google Cluster و Yahoo Cluster در سه الگوریتم هدف نمایش می‌دهد. میزان تخطی از شرایط سرویس روش پیشنهادی AREC در مقایسه با دو الگوریتم AE-AHP و AE-AL به ترتیب ۶۹٪ و ۷۸٪ تحت بار کاری Google Cluster و ۶۷٪ و ۷۷٪ تحت بار کاری Yahoo Cluster کاهش یافته است.



شکل (۱۲): میزان تخطی از شرایط سرویس در طول شبیه‌سازی ۲۴ ساعت کاری سری Google Cluster.



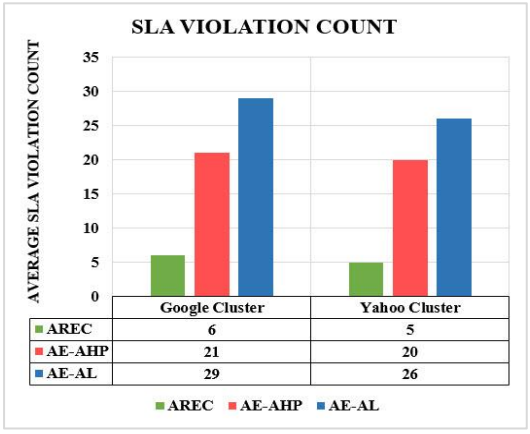
شکل (۱۳): میزان تخطی از شرایط سرویس در طول شبیه‌سازی ۲۴ ساعت کاری سری Yahoo Cluster.

با توجه به استفاده از میزان تخطی از شرایط سرویس در تغییرات حد آستانه و عملکرد درست سامانه کنترل فازی به‌عنوان تصمیم‌گیر سامانه، عمل کشسانی منابع به درستی انجام می‌شود، که این امر موجب تأمین بهینه منابع برای درخواست‌ها شده و به تناسب اختصاص منبع دقیق‌تری صورت گرفته و درنهایت کمترین میزان تخطی در روش پیشنهادی رخ می‌دهد. با توجه به نتایج، روش پیشنهادی کارایی بهتری در مورد تخطی از شرایط سرویس دارد.

۴-۳-۳-۱- ارزیابی میانگین تخطی از شرایط پذیرش سرویس

شکل (۱۴) میانگین تخطی از شرایط پذیرش سرویس در طول

شبیه‌سازی دو بار کاری واقعی را در سه الگوریتم هدف نمایش می‌دهد. در کل میانگین تخطی از شرایط پذیرش سرویس روش پیشنهادی AREC در مقایسه با دو الگوریتم AE-AHP و AE-AL به ترتیب ۶۸٪ و ۷۷/۵٪ کاهش یافته است.

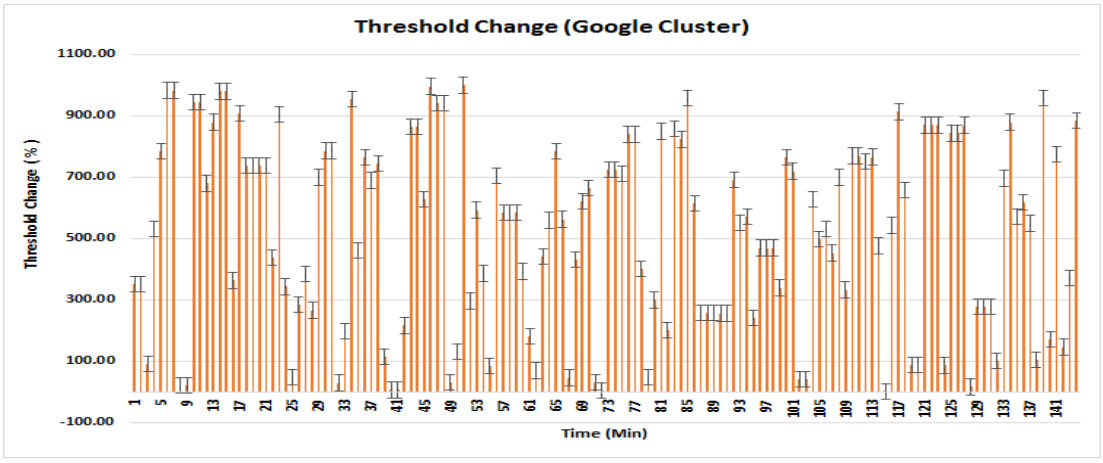


شکل (۱۴): میانگین تخطی از شرایط پذیرش سرویس در طول شبیه‌سازی ۲۴ ساعت کاری در دو بار کاری.

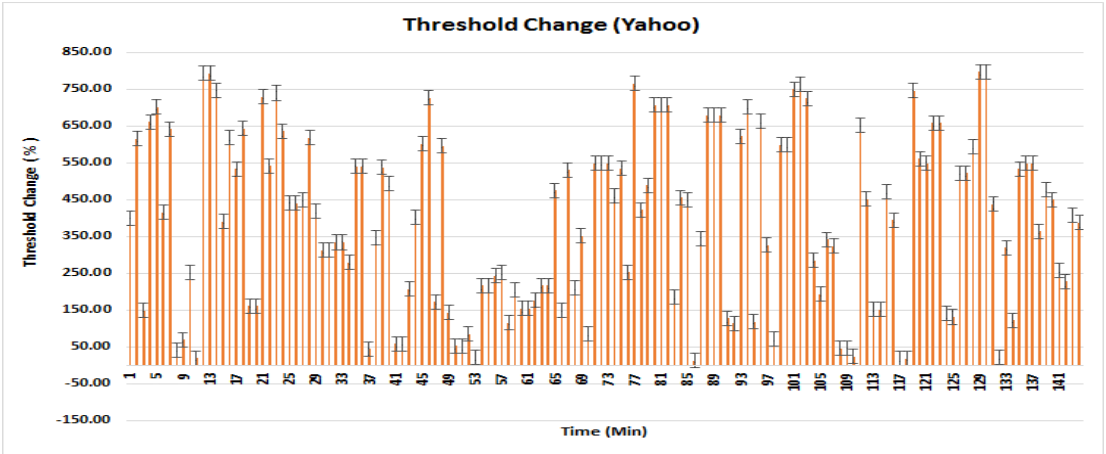
۴-۳-۴- تغییرات حد آستانه

با توجه به اینکه حد آستانه به‌عنوان ویژگی مهم در این پژوهش مطرح شده است، در این بخش میزان تغییرات حد آستانه مشخص می‌گردد. در این بخش میزان تغییرات حد آستانه روش پیشنهادی در بارهای کاری Google Cluster و Yahoo Cluster محاسبه می‌شود.

شکل‌های (۱۵ و ۱۶) میزان تغییرات حد آستانه در طول شبیه‌سازی دو بار کاری Google Cluster و Yahoo Cluster در روش پیشنهادی را نمایش می‌دهد.



شکل (۱۵): تغییرات حد آستانه در طول شبیه‌سازی ۲۴ ساعت کاری سری Google Cluster.



شکل (۱۶): تغییرات حد آستانه در طول شبیه‌سازی ۲۴ ساعت کاری سری Yahoo Cluster.

با توجه به استفاده از میزان تخطی از شرایط سرویس در تغییرات حد آستانه و عملکرد درست سامانه کنترل فازی به‌عنوان تصمیم‌گیر سامانه، عمل کشسانی منابع به‌درستی انجام می‌شود، که این امر موجب تأمین بهینه منابع برای درخواست‌ها شده و به‌تناسب اختصاص منبع دقیق‌تری صورت می‌گیرد.

۵- نتیجه‌گیری

در این مقاله، رویکردی برای بهبود خاصیت کشسانی خودکار با استفاده از سامانه کنترل فازی مبتنی بر تغییرات حد آستانه برای کاربردهای محاسبات با عملکرد بالا در شبکه‌های ابری ارائه شد. در معماری این رویکرد سه لایه کاربرد، لایه پلتفرم و لایه زیرساخت (منابع محاسباتی) وجود دارد. در لایه کاربرد دو نوع برنامه وجود دارد. برنامه خدماتی به‌عنوان رابط کاربر و ماشین مجازی و برنامه اصلی که مدیریت درخواست‌ها، منابع و ثبت وقایع آن‌ها را بر عهده دارد. میزبان‌ها در این ساختار به‌عنوان گره شناخته شده و دارای منابع سخت‌افزاری هستند. ماژول کنترل کننده کشسانی خودکار منابع، مهم‌ترین گره در این ساختار است و وظیفه تأمین خودکار منابع را بر عهده دارد. پایگاه داده اشتراکی محلی برای نگهداری اطلاعات و رویدادهای برنامه‌ها در لایه کاربرد است و همچنین این فضا به‌عنوان واسط کنترل کننده کشسانی خودکار و برنامه اصلی نیز عمل می‌کند. کنترل کننده کشسانی خودکار منابع، برای تعداد درخواست‌های بالا و رخداد تخطی از شرایط سرویس گره‌ای شامل چندین منبع اضافه می‌کند و در صورت زیاد بودن منابع با تجمیع آن‌ها در گره‌ای دیگر، گره کم‌کار را خاموش می‌کند. روش پیشنهادی بر روی داده‌های واقعی آزمایش و ارزیابی شد. نتایج مشخص کرد که روش پیشنهادی عملکرد مطلوبی در مورد هزینه، زمان پاسخگویی و تخطی از شرایط سرویس‌دهی دارد. با توجه به نتایج به‌دست آمده از مقاله حاضر و برای ارزیابی بیشتر نتایج و تکمیل و توسعه این تحقیق، استفاده از روش‌های خوشه‌بندی بار کاری برای شناخت قوی‌تر و تصمیم‌گیری بهتر الگوی بار کاری پیشنهاد می‌گردد تا موجب تأمین منبع با کیفیت بالاتر شود.

۶- مراجع

[1] M. Beltrán, "BECloud: A New Approach to Analyse Elasticity Enablers of Cloud Services," Future Generation Computer Systems, vol. 64, pp. 39-49, 2016.

[2] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan, "Prepare: Predictive Performance Anomaly Prevention for Virtualized Cloud Systems," In 2012 IEEE 32nd Int. Conf. on Distributed Computing Systems, pp. 285-294, 2012.

[3] E. Barrett, E. Howley, and J. Duggan, "Applying Reinforcement Learning Towards Automating Resource Allocation and Application Scalability in the Cloud," Concurrency and Computation: Practice and Experience, vol. 25, pp. 1656-1674, 2013.

[4] L. R. Moore, K. Bean, and T. Ellahi, "Transforming reactive Auto-Scaling Into Proactive Auto-Scaling," In Proc. of the 3rd Int. Workshop on Cloud Data and Platforms, pp. 7-12, 2013.

[5] P. D. Kaur and I. Chana, "A Resource Elasticity Framework for QoS-aware Execution of Cloud Applications," Future Generation Computer Systems, vol. 37, pp. 14-25, 2014.

[6] E. B. Lakew, C. Klein, F. Hernandez-Rodriguez, and E. Elmroth, "Towards Faster Response Time Models for Vertical Elasticity," In 2014 IEEE/ACM 7th Int. Con. on Utility and Cloud Computing, pp. 560-565, 2014.

[7] G. A. Moreno, J. Cámara, D .Garlan, and B. Schmerl, "Proactive Self-adaptation under Uncertainty: A Probabilistic Model Checking Approach," In Proc. of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp. 1-12, 2015.

[8] L. Baresi, S. Guinea, A. Leva, and G. Quattrocchi, "A Discrete-time Feedback Controller for Containerized Cloud Applications," In Proc. of the 2016 24th ACM SIGSOFT Int. Symposium on Foundations of Software Engineering, pp. 217-228, 2016.

[9] G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl, "Efficient Decision-making under Uncertainty for Proactive Self-adaptation," In 2016 IEEE Int. Conf. on Autonomic Computing (ICAC), pp. 147-156, 2016.

[10] F. Paraiso, P. Merle, and L. Seinturier, "Socloud: A Service-Oriented Component-based PaaS for Managing Portability, Provisioning, Elasticity, and High Availability Across Multiple Clouds," Computing, vol. 98, pp. 539-565, 2016.

[11] S. Lehrig, R. Sanders, G. Brataas, M. Cecowski, S. Ivanšek, and J. Polutnik, "CloudStore—towards Scalability, Elasticity, and Efficiency Benchmarking and Analysis in Cloud Computing," Future Generation Computer Systems, vol. 78, pp. 115-126, 2018.

[12] A. Ashraf, B. Byholm, and I. Porres, "CRAMP: Cost-Efficient Resource Allocation for Multiple Web Applications with Proactive Scaling," In 4th IEEE Int. Conf. on Cloud Computing Technology and Science Proc., pp. 581-586, 2012.

[13] A. Beloglazov and R. Buyya, "Adaptive Threshold-based Approach for Energy-efficient consolidation of Virtual Machines in Cloud Data Centers," MGC@ Middleware, vol. 4, 2010.

[14] K. Li, "Quantitative Modeling and Analytical Calculation of Elasticity in Cloud Computing," IEEE Transactions on Cloud Computing, 2017.

[15] M. C. Huebscher and J. A. McCann, "A Survey of Autonomic Computing—degrees, Models, and Applications," ACM Computing Surveys (CSUR), vol. 40, p. 7, 2008.

[16] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A Review of Auto-scaling Techniques for elastic Applications in Cloud Environments," J. of Grid Computing, vol. 12, pp. 559-592, 2014.

[17] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in Cloud Computing: State of the Art and Research Challenges," IEEE Transactions on Services Computing, vol. 11 ,pp. 430-447, 2017.

[18] S. Yi, D. Kondo, and A. Andrzejak, "Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud," In 2010 IEEE 3rd Int. Conf. on Cloud Computing, pp. 236-243, 2010.

[19] S. Wee, "Debunking Real-time Pricing in Cloud Computing," In Proc. of the 2011 11th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing, pp. 585-590, 2011.

[20] A. Kivity, D. Laor, G. Costa, P. Enberg, N. Har'El, D. Marti, and V. Zolotarov, "OSv—optimizing the Operating System for Virtual Machines," In 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14), pp. 61-72, 2014.

[21] Y. Al-Dhuraibi, F. Zalila, N. Djarallah, and P. Merle, "Coordinating Vertical Elasticity of both Containers and Virtual Machines," 8th International Conference on Cloud Computing and Services Science, 2018.

[22] A. Shahidinejad, M. Ghobaei-Arani, and L. Esmaeili, "An Elastic Controller Using Colored Petri Nets in Cloud Computing Environment," Cluster Computing, pp. 1-27, 2019.

[23] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, M. sadjadi, and M. Parashar, "Cloud Federation in a Layered Service Model," Journal of Computer and System Sciences, vol. 78, pp. 1330-1344, 2012.

[24] R. Khorsand, M. Ghobaei- Arani, and M. Ramezanpour, "FAHP Approach for Autonomic Resource Provisioning of Multitier Applications in Cloud Computing Environments," Software: Practice and Experience, vol. 48, pp. 2147-2173, 2018.

[25] R. da Rosa Righi, V. F. Rodrigues, C. A. Da Costa, G. Galante, L. C. E. De Bona, and T. Ferreto, "Autoelastic: Automatic Resource Elasticity for High Performance Applications in the Cloud," IEEE Transactions on Cloud Computing, vol. 4, pp. 6-19, 2015.

[26] R. da Rosa Righi, V. F. Rodrigues, G. Rostirolla, C. A. da Costa, E. Roloff, and P. O. A. Navaux, "A Lightweight Plug-and-play Elasticity Service for Self-organizing Resource Provisioning on Parallel Applications," Future Generation Computer Systems, vol. 78, pp. 176-190, 2018.

[27] V. F. Rodrigues, R. da Rosa Righi, G. Rostirolla, J. L. V. Barbosa, C. A. da Costa, A. M. Alberti, and V.Chang, "Towards Enabling Live Thresholding as Utility to Manage Elastic Master-Slave Applications in the Cloud," Journal of Grid Computing, vol. 15, pp. 535-556, 2017.

[28] T. Bhardwaj and S. C. Sharma, "Fuzzy Logic-based Elasticity Controller for Autonomic Resource Provisioning in Parallel Scientific Applications: A Cloud Computing Perspective," Computers & Electrical Engineering, vol. 70, pp. 1049-1073, 2018.

[29] V. Cardellini, T. G. Grbac, M. Nardelli, N. Tanković, and H.-L. Truong, "QoS-Based Elasticity for Service Chains in Distributed Edge Cloud Environments," In Autonomous Control for a Reliable Internet of Services, ed: Springer, Cham, pp. 182-211, 2018.

[30] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. N. Tantawi, and C. Krintz, "See Spot Run: Using Spot Instances for Mapreduce Workflows," HotCloud, vol. 10, pp. 7-7, 2010.

[31] M. Mattess, C. Vecchiola, and R. Buyya, "Managing Peak Loads by Leasing Cloud Infrastructure Services from a Spot Market," In 2010 IEEE 12th Int. Conf. on High Performance Computing and Communications (HPCC), pp. 180-188, 2010.

[32] A. Andrzejak, D. Kondo, and S. Yi, "Decision Model for Cloud Computing under SLA Constraints," In 2010 IEEE Int. Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 257-266, 2010.

[33] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun, "Benchmarking in the Cloud: What it Should, Can, and Cannot Be," In Technology Conf. on Performance Evaluation and Benchmarking, pp. 173-188, 2012.

[34] S. Islam, K. Lee, A. Fekete, and A. Liu, "How a Consumer Can Measure Elasticity for Cloud Platforms," In Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, pp. 85-96, 2012.

[35] A. A. Rahmanian, M. Ghobaei-Arani, and S. Tofighy, "A Learning Automata-based Ensemble Resource Usage Prediction Algorithm for Cloud Computing Environment," Future Generation Computer Systems, vol. 79, pp. 54-71, 2018.

[36] H.-J. Zimmermann, "Fuzzy Set Theory—and its Applications," Springer Science & Business Media, 2011.

[37] R. Bouabdallah, S. Lajmi, and K. Ghedira, "Use of Reactive and Proactive Elasticity to Adjust Resources Provisioning in the Cloud Provider," In 2016 IEEE 18th Int. Conf. on High Performance Computing and Communications; IEEE 14th Int. Conf. on Smart City; IEEE 2nd Int. Conf. on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1155-1162, 2016.

[38] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google Cluster-usage Traces: Format+ Schema," Google Inc., White Paper, pp. 1-14, 2011.

[39] https://webscope.sandbox.yahoo.com/catalog.php?datatype=s, "Yahoo Cluster".

Journal of Electronical & Cyber Defence
Vol. 8, No. 3, 2020, Serial No. 31

Resource Elasticity Management using Fuzzy Controller Based on Threshold Changes in the Cloud Computing Environment

M. Ghabaei Aarani*, A. Shahidinejad, M. Torabi

*Department of computer engineering, Qom Branch, Islamic Azad University, Qom, Iran

(Received: 17/09/2019, Accepted: 01/03/2020)

ABSTRACT

Increasing the popularity and profitableness of cloud computing is dependent on providing the capabilities and features that the users desire. Elasticity is one of the strongest features that distinguish the cloud computing domain from other distributed system approaches. Cloud computing takes into account an unlimited capacity of the resources for the consumer, and the consumer can take the resources in demand based on competitive rates and increase or decrease the number of resources. There have been many improvements to elasticity management by previous researches. However, further researches are necessary to manage elasticity more efficiently. In this paper, an approach for improving elasticity is presented using the fuzzy control system based on threshold changes for high-performance computing applications in cloud computing. In the proposed approach, elasticity management is based on continuous monitoring and decision making. The results indicate that the proposed approach has a better performance in terms of response time, cost and service level agreement (SLA) violation, compared to previous studies. In comparison with each of the two specified approaches, the response time of the proposed method has decreased by 6.5% and 9%, cost by 6% and 12%, and service level agreement (SLA) violation by 68% and 77.5%, respectively.

Keywords: Elasticity, Resource Provisioning, Fuzzy Control System, Threshold, Cloud Computing

* Corresponding Author Email: m.ghobaei@qom-iau.ac.ir