راهکاری ترکیبی برای بهبود خاصیت کشسانی در محیط رایانش ابری

مصطفی قبائی آرانی، استادیار

گروه مهندسی کامپیوتر، واحد قم، دانشگاه آزاد اسلامی، قم، ایران– m.ghobaei@qom-iau.ac.ir

چکیده: کشسانی، به‌عنوان یک از مهم‌ترین ویژگی‌هایی محسوب می‌شود که فناوری رایانش ابری را از دیگر فناوری‌های رایانش توزیعی، متمایز می کند. این ویژگی، از این حقیقت بهره می‌گیرد که فرایند تخصیص دهی منابع، به‌عنوان رویه‌ای محسوب می‌شود که می‌توان آن را به‌صورت پویا اجرا نمود. ارائه راهکاری کارامد برای بهبود خاصیت کشسانی هم برای ارائه‌دهندگان و هم برای کاربران سرویس‌های رایانش ابری مفید و کارآمد واقع خواهد شد. ارائه‌دهندگان خواهند توانست با راهکاری که در این مقاله طراحی، ارزیابی و توسعه داده خواهد شد، خاصیت کشسانی سرویس‌های ابری خود را ارزیابی کرده و آن‌ها را بهبود بخشیده و مزیت کمی یا کیفی خود در رقابت با سایر رقبا را افزایش دهند. در این مقاله، راهکاری ترکیبی برای بهبود خاصیت کشسانی با استفاده مدیریت بافر و مدیریت متمرکز کشسانی ارائه می‌شود. مدیریت بافر وظیفه کنترل صف ورودی درخواست را به عهده دارد و مدیریت کشسانی با استفاده از یادگیری تقویتی، کنترل خاصیت کشسانی سیستم را به عهده دارد. مؤثر بودن راهکار پیشنهادی تحت سه بار کاری واقعی Google Cluster، Yahoo Cluster و Wikipedia ارزیابی شده است. نتایج آزمایشات نشان می‌دهد که راهکار پیشنهادی در مقایسه با دو راهکار CTMC و ControCity موجب کاهش زمان پاسخگویی ۱۵/۲ درصد، و افزایش بهره‌وری به میزان ۱۳/۲ درصد و افزایش خاصیت کشسانی را درحد ۱۹/۸ درصد نشان می‌دهد.

واژه‌های کلیدی: خاصیت کشسانی، مدیریت بافر، یادگیری تقویتی، رایانش ابری.

A Hybrid Approach for Improving Elasticity in the Cloud Computing Environment

M. Ghobaei-Arani, Assistant Professor

Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran, Email: m.ghobaei@qom-iau.ac.ir

Abstract: Elasticity is considered as one of the most important features that distinguishes cloud computing from other distributed system approaches. This feature takes into account the fact that the resource allocation process is considered as a process that can be implemented dynamically. Providing an efficient solution for improving elasticity will be useful for both providers and users of cloud computing services. Using the proposed solution in this paper, providers will be able to evaluate and improve the quality of their services, and increase their qualitative or quantitative advantage in competing with other competitors. In this paper, we present a hybrid solution for improving elasticity through using buffer management and centralized elastic management. Buffer managment controls the input queue of the request and the elastic management by using the reinforcement learning controls the elasticity of the system. Finally, we evaluate the effectiveness of our approach under three real workload traces, namely, Yahoo Cluster, Wikipedia, and Google Cluster workload traces. The experimental results show that the proposed approach reduces the response time by up to 15.2%, and increases the resource utilization by up to 13.2 % and the elasticity by up to 19.8 % compared with the CTMC and ControCity approaches.

Keywords: Elasticity, Buffer Management, Reinforcement Learning, Cloud Computing.

تاریخ ارسال مقاله: ۱۳۹۷/۰۷/۲۲
تاریخ اصلاح مقاله: ۱۳۹۸/۰۳/۱۲
تاریخ پذیرش مقاله: ۱۳۹۸/۱۲/۰۱
نام نویسنده مسئول: مصطفی قبائی آرانی
نشانی نویسنده مسئول: ایران - قم - پردیسان - بلوار دانشگاه - دانشگاه آزاد اسلامی مولفه قم - دانشکده فنی مهندسی.

۱- مقدمه

در سال‌های اخیر، رایانش ابری به‌عنوان یک الگوی رایانشی مبتنی بر اینترنت، توجه جامعه دانشگاهی و صنعت را برای تحویل سرویس‌ها از طریق بستر اینترنت به کاربران فراهم کرده است [۱، ۲]. رایانش ابری یک مدل پرداخت-به ازای-میزان استفاده است که این امکان را برای کاربران فراهم می‌کند تا به مخزنی از منابع محاسباتی، پلت فرم‌های توسعه و برنامه‌های کاربردی به‌صورت سرویس‌های مبتنی بر تقاضا بدون نیاز به خرید یا مالکیت آن‌ها دسترسی پیدا کنند [۳، ۴]. یکی از دلایل محبوبیت رایانش ابری به دلیل خاصیت کشسانی آن می‌باشد که به پلت فرم‌های ابری این امکان را می‌دهد تا بتوانند نوسانات بارهای کاری را به‌طور مؤثری بدون اختلال در اجرای نرمال برنامه‌های کاربردی ابری مدیریت کنند. بر طبق [۵]، خاصیت کشسانی به‌عنوان درجه‌ای از سیستم تعریف می‌شود که سیستم قادر به تطبیق تأمین منابع خود به‌صورت اتوماتیک با تغییرات بار کاری نسبت به زمان می‌باشد به‌طوری‌که در هرلحظه از زمان، میزان منابع در دسترس، با تقاضای فعلی تا حد ممکن، مطابقت و هماهنگی داشته باشد.

خاصیت کشسانی، به‌صورت یک شمشیر دو لبه است: بدین معنا که ازیک‌طرف، خاصیت کشسانی به تأمین‌کنندگان ابری این امکان را می‌دهد که منابعشان را به‌طور پویا و بر اساس تغییرات تقاضاهای کاربران، اخذ و آزادسازی نمایند و از طرفی دیگر، تصمیم‌گیری در مورد میزان درست منابع موردنیاز، کار چندان آسانی نیست. در واقع، تعیین میزان درستی از منابع موردنیاز برای برنامه‌های کاربردی ابری، موضوعی مهم و حیاتی در حوزه مدیریت منابع در محیط رایانش ابری است. اگر خاصیت کشسانی به‌درستی مدیریت نشود، پلت فرم رایانش ابری با مسائل اضافه تأمین و کسر تأمین مواجه خواهد شد. در وضعیت اضافه تأمین، اگرچه به دلیل در اختیار داشتن حداکثر منابع موردنیاز، از تخطی‌های توافقنامه سطح سرویس (SLA[1]) تا حد زیادی جلوگیری خواهد شد، اما به دلیل اضافه تأمین منابع، برای مدت‌زمانی طولانی، منابع تأمین‌کننده به‌شدت کم‌بهره‌ور شده و ظرفیت منابع به هدر خواهند رفت. اما در وضعیت کسر تأمین منابع، اگرچه باعث کاهش هزینه‌ها می‌شود اما، ممکن است تأمین‌کننده، منابع به‌اندازه کافی برای تکمیل نیازهای مشتریان را در لحظات اوج بار کاری در اختیار نداشته باشد و باعث ایجاد تخطی از توافقنامه سطح سرویس شود، که این تأثیر منفی بر کیفیت سرویس کاربران گذاشته و به دنبال آن باعث از دست رفتن کاربران و درآمد حاصل از آن‌ها خواهد شد [۶، ۷].

با اینکه خاصیت کشسانی از ویژگی‌های کلیدی پلت فرم‌های رایانش ابری تلقی می‌گردد، لیکن یکی از مسائل مهم و چالش برانگیز در این زمینه، فقدان یک راهکار مناسب برای کنترل و مدیریت خاصیت کشسانی می‌باشد. بنابراین، ارائه یک راه‌حل کارامد، نقش مهمی در مدیریت خودکار منابع برای تضمین خاصیت کشسانی و مقابله با مسائل اضافه تأمین و کسر تأمین منابع دارد. در این مقاله، راهکاری الهام گرفته از معماری استاندارد سه لایه رایانش ابری شامل سه لایه برنامه کاربری، میان‌افزار و زیرساخت برای کنترل کشسانی در پلت فرم‌های ابری ارائه می‌شود. در راهکار پیشنهادی، دو مؤلفه مهم تحت عنوان مدیریت بافر در لایه برنامه کاربردی و مدیریت کشسانی در لایه میان‌افزار وجود دارد. مدیریت بافر، وظیفه کنترل صف ورودی درخواست‌های کاربران را به عهده دارد و مدیریت کشسانی با استفاده از یادگیری تقویتی [۸] اقدام به کنترل خاصیت کشسانی پلت فرم‌های ابری می‌نماید.

دستاوردهای اصلی این پژوهش به شرح زیر می‌باشد:

- ارائه یک راهکار جدید برای مدیریت خاصیت کشسانی با استفاده از مدیریت بافر پیشرفته و یادگیری تقویتی در محیط رایانش ابری
- افزایش بهره‌وری، کاهش زمان پاسخگویی و افزایش خاصیت کشسانی با استفاده از رویکرد پیشنهادی.
- ارزیابی راهکار پیشنهادی با استفاده از سه نوع بار کاری واقعی Google Cluster، Yahoo Cluster و Wikipedia

ادامه این مقاله به‌صورت زیر سازمان‌دهی شده است: در بخش دوم، کارهای مربوطه در حوزه مدیریت خاصیت کشسانی مورد بررسی قرار می‌گیرند. راهکار پیشنهادی با استفاده از مدیریت بافر و رتبه‌بندی مدیریت کشسانی با استفاده از یادگیری تقویتی در بخش سوم ارائه می گردد. در بخش چهارم، با توجه به شبیه‌سازی نتایج راهکار پیشنهادی تولید می‌شود و نتایج حاصل‌شده را مورد ارزیابی قرار داده و با راهکارهای مرجع مقایسه می‌شود. نهایتاً، در بخش پنجم نتیجه‌گیری و پیشنهادها مطرح می‌گردد.

۲- کارهای مربوطه

در این بخش، به بررسی تحقیقات مرتبط انجام‌شده در زمینه مدیریت خاصیت کشسانی می‌پردازیم.

در [۹]، معضل فقدان شاخص مناسب جهت کمی سازی خاصیت کشسانی رایانش ابری مورد بررسی قرار گرفته است. نویسنده با تکیه بر چهار متغیر هزینه، دقت، زمان و مقیاس‌پذیری، یک ابزار محک و آنالیز برای خاصیت کشسانی سرویس‌های ابری ارائه داده که خاصیت کشسانی یک سرویس ابری به‌عنوان توانایی سیستم برای مقیاس بندی انطباقی منابع در پاسخ به کاربر نهایی با انجام کیفیت خدمات در توافقنامه سطح خدمات با بهترین دقت، زمان و هزینه توصیف می‌شود.

در [۱۰]، کیکن لی و همکاران به دو مسئله مهم خاصیت کشسانی رایانش ابری پرداختند. نخست، نیاز به شاخص کمی و قابل‌اندازه‌گیری، مشاهده و محاسبه خاصیت کشسانی پلت فرم‌های رایانش ابری را بررسی کرده و سپس، راهکاری با استفاده از زنجیره‌های مارکوف پیوسته (CTMC[2]) در زمان برای مدل‌سازی، کمی سازی، آنالیز، پیش‌بینی، مقایسه و بهبود عملکرد و هزینه یک پلتفرم رایانش ابری ارائه دادند. با اینکه این پژوهش در راستای ارائه راهکاری برای کمی سازی خاصیت کشسانی رایانش ابری بوده است اما به گفته پژوهشگر، مدل حاصل از این پژوهش فاقد عبارات فرم بسته برای معیارهای عمده‌ای مانند کشش، کارایی و هزینه بوده و استفاده از آن برای تحلیل یک پلتفرم رایانش ابری سخت و دشوار است.

در [۱۱]، نویسندگان، رویکردی مبتنی بر مدیریت بافر برای مدیریت خاصیت کشسانی در محیط پویای ابری ارائه دادند. راهکار آن‌ها، از تکنیک اتوماتای یادگیر برای کنترل خاصیت کشسانی و از تکنیک پیش‌بینی میانگین متحرک برای پیش‌بینی تعداد درخواست‌ها استفاده می‌کند. همچنین، آن‌ها راهکار پیشنهادی خود را تحت بارهای کاری واقعی FIFA, NASA و Clarcknet مورد ارزیابی قرار دادند.

در [۱۲]، نویسندگان، خاصیت کشسانی پلت فرم‌های ابری را از دیدگاه تئوری کنترل مورد بررسی قرار دادند. در حقیقت، آن‌ها تئوری کنترل را به‌عنوان تکنیک پیاده‌سازی و برنامه‌های کاربردی را به‌عنوان حوزه هدف در نظر گرفتند. نتایج بررسی و طبقه‌بندی کردن تحقیقات و چالش‌های مرتبط، به محققین این امکان را می‌دهد که انواع راه‌حل‌های کنترلی بازخوردی به کار گرفته شده برای مدیریت خاصیت کشسانی را درک کنند.

در [۱۳]، روی هان در رساله دکترا خود، مسئله ارائه سرویس‌های کشسان به‌صورت هزینه کارا برای برنامه‌های کاربردی ابری را از دو دیدگاه برنامه‌های کاربردی و الگوریتم‌ها بررسی کرد. در سطح برنامه کاربردی، از معیارهای آگاه از هزینه، برای کشف گلوگاه‌های برنامه‌های کاربردی چندلایه برنامه‌های کاربردی ابری برای مقیاس بندی برنامه‌های کاربردی در راستای برآورده کردن نیازمندی‌های کارایی پرداختند. همچنین در سطح الگوریتم، چارچوبی برای توسعه الگوریتم‌های کشسان برای افزایش دقت پیش‌بینی ارائه دادند.

در [۱۴]، آلبونیپو و همکاران‌اش، راهکاری برای کنترل اتوماتیک خاصیت کشسانی برنامه‌های کاربردی بر اساس نیازمندی‌های متغیر منابع ارائه دادند. راهکار پیشنهادی آن‌ها بر اساس سه وضعیت کشسانی آماده، افزایش و کاهش مقیاس بر اساس شناسایی الگوهای بار کاری می باشد. آن‌ها آزمایشات شان را روی Amazon EC2 انجام دادند و نشان دادند که راهکار پیشنهادی‌شان در مدت زمان کمتری قادر به تأمین منابع با حفظ خاصیت کشسانی در مقایسه با سایر راهکارها می‌باشد.

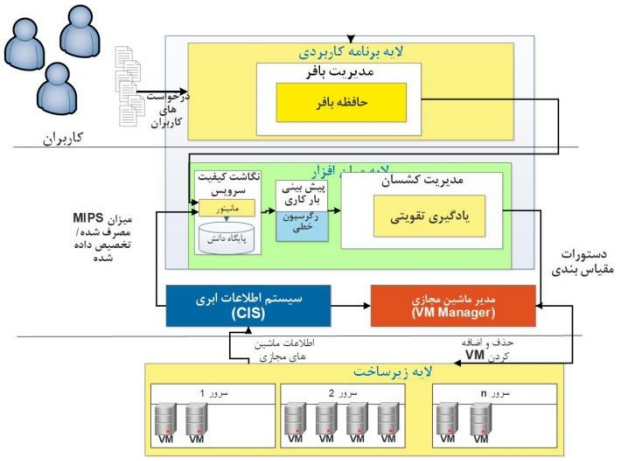
از آنجائی که خاصیت کشسانی سعی در تطبیق میزان تغییرات بارهای کاری با میزان منابع در دسترس در هرلحظه از زمان با تخصیص و باز تخصیص منابع در یک روش اتوماتیک دارد، اولاً، اکثر راهکارهای ارائه شده قبلی بر روی کنترل این خاصیت از طرف منابع و زیرساخت تمرکز داشته‌اند و اصلاً به کنترل خاصیت کشسانی از طرف بار کاری با استفاده از بافر کردن درخواست‌های کاربران توجهی نکرده‌اند، ثانیاً، اکثر راهکاری قبلی صرفاً یکی از تکنیک‌های پیش‌بینی بار کاری یا یادگیری ماشین را برای مدیریت خاصیت کشسانی بکار گرفته‌اند، درحالی‌که، راهکار پیشنهادی ما از ترکیب تکنیک‌های یادگیری ماشین یعنی یادگیری تقویتی و پیش‌بینی بار کاری برای تضمین خاصیت کشسانی استفاده می‌کند.

۳-راهکار پیشنهادی

یک راهکار کشسان برای پلت فرم‌های ابری باید قابلیت عملکردی مناسبی در بارهای کاری سبک، متوسط و سنگین داشته باشد و در صورت افزایش بار کاری بتواند منابع موجود را به‌طور مؤثری در راستای استفاده کاربران مدیریت کند. در ادامه این بخش، چارچوب راهکار پیشنهادی و مؤلفه‌های آن شامل: ساختار مدیریت بافر، مؤلفه نگاشت کیفیت سرویس، مؤلفه پیش‌بینی بار کاری و مدیریت کشسانی با استفاده از یادگیری تقویتی در راهکار پیشنهادی می‌پردازیم.

۳-۱-چارچوب پیشنهادی

شکل ۱ چارچوب راهکار پیشنهادی که الهام گرفته از معماری استاندارد سه لایه‌ای رایانش ابری را نمایش می‌دهد. همان‌طورکه در شکل ۱ مشاهده می‌کنید چارچوب پیشنهادی از سه لایه اصلی شامل: لایه برنامه کاربردی (معادل SaaS)، لایه میان‌افزار (معادل PaaS) و لایه زیرساخت (معادل IaaS) تشکیل شده است. در چارچوب راهکار پیشنهادی، دو مؤلفه مهم تحت عنوان مدیریت بافر در لایه برنامه کاربردی و مدیریت کشسانی در لایه میان‌افزار وجود دارد. در لایه برنامه کاربردی، درخواست‌های تولید شده توسط برنامه‌های کاربردی ابری دریافت شده و سپس در اختیار مدیریت بافر قرار می‌گیرند. مدیریت بافر، وظیفه کنترل صف ورودی درخواست‌ها را به عهده دارد و مدیریت کشسانی در لایه میان‌افزار با استفاده از یادگیری تقویتی اقدام به کنترل خاصیت کشسانی پلت فرم‌های ابری می‌نماید. در راهکار پیشنهادی مدیریت کشسانی دارای بخشی به نام سیستم نگاشت کیفیت سرویس دهی است که وظیفه دریافت اطلاعات ابر شامل میزان MIPS[۳] تخصیص‌داده شده و MIPS مصرفی را به عهده دارد.



شکل ۱: چارچوب راهکار پیشنهادی

با استفاده از اطلاعات بار کاری مرحله بعد و میزان تخطی از شرایط سرویس رخ داده و یادگیری تقویتی برای تأمین صحیح منابع اقدام می کند.

در حقیقت، لایه اصلی برای کنترل خاصیت کشسانی، لایه میان‌افزار می‌باشد که منطبق بر حلقه کنترلی MAPE[۴] شرکت IBM [۱۷-۱۵] شامل چهار فاز اصلی مانیتورینگ (مؤلفه مدیریت بافر و مؤلفه نگاشت کیفیت سرویس)، تحلیل (مؤلفه پیش‌بینی کننده)، برنامه‌ریزی (مؤلفه مدیریت کشسان) و اجرا می‌باشد. برای سهولت، نمادهای مورد استفاده در راهکار پیشنهادی در جدول ۱ نشان داده شده است. در ادامه، هریک از مؤلفه‌های اصلی چارچوب پیشنهادی شامل مؤلفه مدیریت بافر، مؤلفه نگاشت کیفیت سرویس، مؤلفه پیش‌بینی بار کاری و مؤلفه مدیریت کشسانی را با جزئیات مورد بررسی قرار می‌دهیم:

۳-۱-۱-مدیریت بافر

به‌طور مشخص، بخش مدیریت بافر در راهکار پیشنهادی وظیفه دریافت اطلاعات درخواست‌ها را به عهده دارد و بر اساس میزان ترافیک درخواست‌های ورودی، وظیفه اضافه یا حذف کردن حافظه بافر را بر عهده دارد. نهایتاً، درخواست‌های ورودی را بر اساس اولویتشان، دسته‌بندی و ارسال می‌کند. در ادامه این بخش، ابتدا ساختار درخواست‌ها در بافر، ساختار حافظه بافر و نهایتاً عملکرد مدیریت بافر را شرح می‌دهیم.

جدول۱: علامت‌گذاری و تعریف نمادها

تعریف	نماد
زمان جاری سیستم	$CurrentTime$
زمان مجاز پاسخگویی به درخواست i ام	$DeadlineTime_i$
زمان ورود درخواست i ام	$ArrivalTime_i$
اولویت درخواست i ام	$\Pr i_i$
میزان تخطی زمان پاسخگویی	$SLA\_Violation_{Response}$
میزان تخطی هزینه	$SLA\_Violation_{Cost}$
هزینه ابتدایی درخواست	$Cost_{Req}$
هزینه نهایی پاسخگویی به درخواست	$Cost_{Res}$
میزان بار کاری فعلی	$WL_{Current}$
میزان بار کاری پیش‌بینی شده	$WL_{Predict}$
نوع مقیاس بندی	$Scaling\_Type$
میزان مقیاس بندی	$Scaling\_Amount$
تابع حذف درخواست	$Drop$
تابع ارسال درخواست	$Send\_Req$
حد آستانه پایین برای تعداد درخواست‌های صف	$q_{\min}$
زمان زنده ماندن درخواست	TTL
میزان بار کاری پیش‌بینی شده	Y_{t+1}
میزان بار کاری فعلی	X_t
صف i ام (i=1,2,3)	Q_i

۳-۱-۱-۱- ساختار درخواست‌ها در بافر

در این بخش، ساختار درخواست‌های کاربران در راهکار پیشنهادی را شرح می‌دهیم. همان‌طورکه در جدول ۲ نشان داده شده است، هر درخواست کاربر شامل User-ID برای نشان دادن شناسه کاربر، Request-SLA برای نمایش اهداف سطح سرویس(SLO[۵]) مشخص شده توسط کاربر و Request-Resource برای نمایش قابلیت‌های منابع موردنیاز برای اجرای درخواست‌های کاربران می‌باشد.

جدول۲: ساختار درخواست کاربر

User_ID	Request_SLA	Request_Resource

فیلد Request-SLA از دو هدف سطح سرویس تعیین شده توسط کاربر شامل حداکثر هزینه و حد نهایت زمان پاسخگویی مطابق جدول ۳ تشکیل شده است. درصورتی‌که هزینه و زمان پاسخگویی از این حد بالاتر برود، تخطی از شرایط سرویس[۶] رخ خواهد داد.

جدول ۳: فیلدهای Request-SLA

SLO_1	SLO_2
Maximum Cost($)	Maximum Deadline Time(ms)

همچنین، فیلد Request_Resource قابلیت‌های سخت‌افزاری منابع موردنیاز کاربر شامل میزان کارایی پردازنده برحسب MIPS، حافظه اصلی، حافظه ذخیره‌سازی و پهنای باند موردنیاز کاربر برای هر درخواست را به‌صورت جدول ۴ ذخیره می‌کند.

جدول ۴: فیلدهای Request_Resource

Minimum MIPS	Minimum RAM(GB)	Minimum Storage(GB)	Minimum Bandwidth(Mbps)

ساختار این درخواست‌ها و فیلدهای مربوطه‌شان، توسط مؤلفه‌های موجود در راهکار پیشنهادی برای استخراج اطلاعات موردنیاز مورد تحلیل و استفاده قرار خواهند گرفت. به‌عنوان نمونه، واحد مدیریت بافر برای اولویت‌بندی درخواست از فیلد مهلت زمانی موجود در جدول ۲ و مؤلفه نگاشت کیفیت سرویس از فیلد کارایی پردازنده برحسب MIPS موجود در جدول ۲ استفاده می‌کند.

۳-۱-۱-۲-ساختار حافظه بافر

در راهکار پیشنهادی، ساختار حافظه بافر از سه صف تشکیل شده است؛ که هر صف میزان اولویت درخواست‌ها را مشخص می‌کند. جهت مشخص کردن اولویت درخواست‌ها از رابطه (۱) استفاده می‌شود.

$$Pri_i = \alpha \times \frac{1}{(DeadLineTime_i)} + (1-\alpha) \times \frac{1}{(CurrentTime - ArrivalTime_i)} \quad (۱)$$

در رابطه (۱)، $CurrentTime$ زمان جاری سیستم، $ArrivalTime_i$ زمان ورود درخواست و $DeadLineTime_i$ زمان حد مجاز پاسخگویی درخواست i ام می‌باشد. پارامتر α مقدار وزن در نظر گرفته شده برای هر یک از پارامترها است. بدین‌صورت که درخواست‌های ورودی کاربران بر اساس زمان درخواست و اولویت محاسبه شده در رابطه (۱)، در یکی از صف‌ها با اولویت بالا، معمولی و پایین قرار می‌گیرند.

همان‌طورکه در رابطه (۱) مشاهده کردید، اولویت یک درخواست به دو عامل بستگی دارد: مهلت زمانی یا حد مجاز زمان پاسخگویی درخواست($DeadLineTime_i$) و مدت زمان انتظار ($CurrentTime - ArrivalTime_i$). بر اساس رابطه (۱)، هرچه مهلت زمانی یک درخواست کمتر باشد، آن درخواست از اولویت بالاتری برخوردار خواهد و مقدار Pri_i بالاتری خواهد داشت. همچنین، هرچه مدت زمان انتظار یک درخواست بیشتر باشد، بازهم آن درخواست از اولویت بالاتری برخوردار خواهد. بر این اساس، نحوه طبقه‌بندی درخواست‌ها در سه دسته اولویت بالا، معمولی و پایین در الگوریتم ۱ نشان داده شده است.

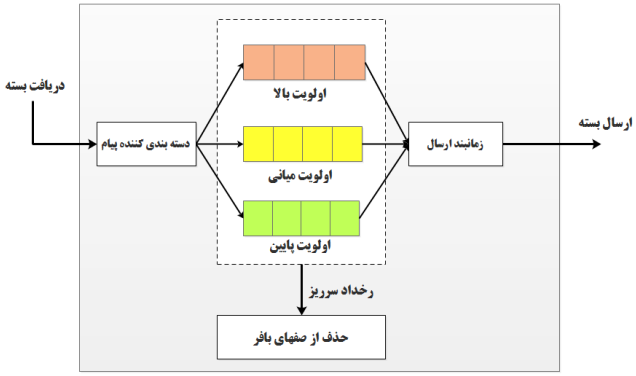
Algorithm 1: Pseudo code for Request Classification ($\mathrm{Re}q_i$)

1: ***Begin***
2: ***Input:*** $\mathrm{Re}q_i, DLT_i, CT, AT_i$ /* Information of Request i*/
3: ***Output:*** $Category_i$ /* Category of Request i */
4: Calculate the value of Priority of Request i using **Eq. (1)**
5: ***If*** (DLT_i & & $(CT - AT_i)$ *is High*) then $Category_i = "Low"$
6: ***Else if*** (DLT_i & & $(CT - AT_i)$ *is Normal*) then
$Category_i = "Middle"$
7: ***Else if*** (DLT_i & & $(CT - AT_i)$ *is Low*) then
$Category_i = "High"$
8: ***Return*** $Category_i$
9: ***End***

تفاوت نحوه اولویت‌بندی درخواست‌ها در راهکار پیشنهادی با دیگر راهکارهای صف اولویت مانند زمان‌بندی صف AWS[۷] [۱۸] در نحوه اولویت‌بندی درخواست‌ها می‌باشد. در زمان‌بندی صف AWS نیز از چندین صف اولویت برای زمان‌بندی درخواست‌ها استفاده می‌شود. به‌عنوان‌مثال، برای درخواست‌های با اولویت بالا از یک صف نمونه ماشین‌های مجازی On-demand برای درخواست‌های با اولویت پایین از یک صف نمونه ماشین‌های مجازی Spot استفاده می‌کند. نکته مهم این است که تعیین اولویت در زمان‌بندی صف AWS به‌صورت ایستا و فقط بر اساس پارامتر زمان ورود یک درخواست انجام می‌شود و زمان مهلت اتمام درخواست در تعیین اولویت نادیده گرفته می‌شود. درحالی‌که در صف اولویت دار پیشنهادی، برای تعیین اولویت درخواست به‌صورت پویا و با در نظر گرفتن هر دو پارامتر زمان ورود و مهلت اتمام انجام می‌شود.

۳-۱-۱-۳-مؤلفه مدیریت بافر

شکل ۲ عملکرد مدیریت بافر را نشان می‌دهد. در این بخش، ابتدا درخواست ورودی توسط الگوریتم دسته‌بندی کننده درخواست در سه صف اولویت بالا، میانی و پایین قرار می‌گیرند. گاهی اوقات ممکن است به هر دلیل سرریز اتفاق بیافتد، بدین معنی که دیگر حافظه بافر فضایی برای اضافه کردن درخواست جدید نداشته باشد، در این حالت باید درخواستی از بافر حذف گردد.



شکل ۲: ساختار مدیریت بافر

حذف درخواست از هر صف به‌صورت جداگانه انجام می‌شود. نکته مهم در حذف درخواست از صف این است که درخواستی از صف حذف می‌شود که دارای بیشترین مقدار زمان زنده ماندن است. بدین معنی که فرسوده‌ترین درخواست که بیشترین فاصله زمانی قرار گرفتن در صف در بین درخواست‌ها را دارد، حذف می‌گردد؛ اما مورد بسیار مهم این است که باید ابتدا درخواست‌های با اولویت پایین، در صورت عدم فضا درخواست‌های متوسط و در صورت عدم وجود فضای حافظه درخواست‌های با اولویت بالا را حذف نمود. بنابراین، در صورت پر بودن فضای حافظه بافر و رسیدن درخواست جدید یکی از این سه حالت زیر پیش خواهد آمد.

حالت اول (درخواستی با اولویت بالا دریافت شود):

در این حالت به‌صورت رابطه (۲) عمل می‌شود. ابتدا درصورتی‌که تعداد درخواست‌ها در صف اولویت پایین، از حد کمینه در نظر گرفته‌شده بیشتر باشد، آن را حذف می‌کنیم. در غیر این صورت اگر تعداد درخواست‌ها در صف اولویت میانی از حد کمینه در نظر گرفته‌شده بیشتر باشد، آن را حذف می‌کنیم و در غیر این صورت حذف از صف اولویت بالا انجام می‌شود.

$$\begin{cases} Drop = \mathrm{Req}\ From\ Q_3\ With\ Least\ TTL & if \quad Count(Q_3) > q_{\min} \\ Drop = \mathrm{Req}\ From\ Q_2\ With\ Least\ TTL & else\ if \quad Count(Q_2) > q_{\min} \\ Drop = \mathrm{Req}\ From\ Q_1\ With\ Least\ TTL & \text{Otherwise} \end{cases} \quad (۲)$$

حالت دوم (درخواستی با اولویت متوسط دریافت شود):

در این حالت به‌صورت رابطه (۳) عمل می‌شود. ابتدا درصورتی‌که تعداد درخواست‌ها در صف اولویت پایین از حد کمینه در نظر گرفته‌شده بیشتر باشد، آن درخواست حذف می‌شود. در غیر این صورت حذف از صف اولویت میانی انجام می‌شود.

$$\begin{cases} Drop = \mathrm{Req}\ From\ Q_3\ With\ Least\ TTL & if \quad Count(Q_3) > q_{\min} \\ Drop = \mathrm{Req}\ From\ Q_2\ With\ Least\ TTL & \text{Otherwise} \end{cases} \quad (۳)$$

حالت سوم (درخواستی با اولویت پایین دریافت شود):

در این حالت به‌صورت رابطه (۴) عمل می‌شود که درخواست تنها از صف اولویت پایین حذف می‌شود.

$$Drop = \mathrm{Req}\ From\ Q_3\ With\ Least\ TTL \quad (۴)$$

هنگامی‌که چندین درخواست در صف‌ها موجود باشند و هرکدام آمادگی ارسال داشته باشند، زمان‌بند بافر ابتدا درخواست‌های صف

اولویت بالا را ارسال می‌کند و پس از اتمام آن به سراغ صف اولویت میانی و درنهایت اولویت پایین خواهد رفت. رابطه (۵) ساختار ارسال درخواست‌ها را با توجه به اولویت آن‌ها نشان می‌دهد.

$$\begin{cases} Send\_Req = Req \in Q_3 & if \;\; Q_3 <> empty \\ Send\_Req = Req \in Q_2 & else\ if \;\; Q_2 <> empty \\ Send\_Req = Req \in Q_1 & \text{Otherwise} \end{cases} \quad (۵)$$

درخواست‌ها به ترتیب در اختیار مؤلفه مدیریت کشسانی قرارگرفته تا این مؤلفه با استفاده از اطلاعات دریافت شده از مؤلفه نگاشت کیفیت سرویس، در مورد تأمین منابع تصمیم‌گیری کند.

پیچیدگی محاسباتی بخش مدیریت بافر به تعداد کل درخواست‌های ورودی در صف‌های اولویت بستگی دارد. یکی از بخش‌ها برای محاسبه پیچیدگی زمانی، محاسبه مقدار اولویت به ازای تعداد درخواست‌ها توسط الگوریتم دسته‌بندی درخواست‌ها بر اساس رابطه (۱) می‌باشد. اگر مدت زمان لازم برای محاسبه اولویت یک درخواست بر اساس رابطه (۱) را یک واحد زمانی و تعداد کل درخواست‌ها را N فرض کنیم، مدت زمان لازم برای محاسبه اولویت تعداد کل درخواست‌ها برابر $O(N)$ خواهد بود. بنابراین، با افزایش تعداد درخواست‌ها، پیچیدگی زمانی راهکار پیشنهادی نیز افزایش خواهد یافت. بخش دیگر، مربوط به محاسبه پیچیدگی زمانی برای حذف درخواست‌ها از صف‌های اولویت در صورت سرریز شدن فضای بافر می‌باشد. بدترین حالت هنگامی است که یک درخواست با اولویت بالا دریافت شود، چرا که در این حالت امکان دارد در بدترین شرایط هر سه صف اولویت پایین، میانی و بالا برای حذف درخواست بررسی شوند. در این صورت مدت زمان لازم برای بررسی حذف یک درخواست برابر با $O(Count(Q_1+Q_2+Q_3))$ خواهد بود.

۳-۱-۲-مؤلفه نگاشت کیفیت سرویس

وظیفه این مؤلفه، ذخیره‌سازی اطلاعات مربوط به میزان بهره‌وری ماشین‌های مجازی است. پس از ثبت این اطلاعات، این مؤلفه میزان تخطی از شرایط سرویس را محاسبه کرده و به اطلاع مؤلفه مدیریت کشسانی می‌رساند. ساختار ذخیره‌سازی اطلاعات در این مؤلفه به‌صورت جدول ۵ می‌باشد.

جدول ۵: ساختار اطلاعات کیفیت سرویس برای یک درخواست

میزان MIPS کل	میزان MIPS استفاده شده	شناسه ماشین

به واسطه دانش در مورد کیفیت سرویس‌دهی، میزان تخطی از شرایط مورد قبول سرویس محاسبه می‌شود. درصورتی‌که زمان پاسخگویی از زمان مجاز پاسخگویی به درخواست بیشتر باشد، از طریق رابطه (۶) میزان تخطی از سرویس را در مورد زمان پاسخگویی محاسبه می‌شود. به همین ترتیب، درصورتی‌که هزینه پاسخگویی به درخواست از هزینه در نظر گرفته‌شده برای پاسخگویی به درخواست بیشتر باشد، از طریق رابطه (۷) میزان تخطی از سرویس در مورد هزینه محاسبه می شود.

$$\begin{aligned} SLA\_Violation_{Response} &= ResponseTime - DeadlineTime \\ SLAV\_Count_{Response} &= Count\left(SLA\_Violation_{Response} > 0\right) \end{aligned} \quad (۶)$$

$$SLA-Violation_{cost} = Cost_{Res} - Cost_{Req} \quad ۷)$$

$$SLAV\_Count_{Cost} = Count\left(SLA-Violation_{Cost} > 0\right) \quad (\text{۷}$$

در رابطه (۶)، $ResponseTime$ زمان پاسخگویی به درخواست و $DeadlineTime$ زمان مجاز پاسخگویی به درخواست است. همچنین، در رابطه (۷)، $Cost_{Req}$ هزینه در نظر گرفته‌شده ابتدایی برای درخواست و $Cost_{Res}$ هزینه نهایی پاسخگویی به درخواست است.

۳-۱-۳-مؤلفه پیش‌بینی بار کاری

این مؤلفه وظیفه پیش‌بینی ترافیک درخواست‌های آتی را بر عهده دارد. به هر میزان پیش‌بینی در این بخش دقیق‌تر باشد، تصمیم‌گیری بخش مدیریت کشسان، کیفیت بالاتری خواهد داشت. در راهکار پیشنهادی از تکنیک مدل پیش‌بینی رگرسیون خطی (LR^) [۱۹] به‌عنوان یک تکنیک سری زمانی استفاده می‌کنیم. عناصر تصادفی مانند نقاط اوج بار کاری در برخی از سری‌های زمانی ممکن است آنقدر قوی باشند که هر گونه نظمی را در سری زمانی از بین ببرند. بنابراین هر گونه تفسیر و تحلیل ذهنی و بصری در خصوص نمودار سری زمانی با مشکل مواجه خواهد شد. در چنین شرایطی ممکن است مجبور باشیم برای رسیدن به تصویر واضح از سری زمانی، آن را هموار کنیم. مدل پیش‌بینی رگرسیون خطی، سبک اصلی مدل‌های اقتصادی است که به ارتباط بین یک متغیر وابسته و یک متغیر مستقل می‌پردازد. فرم کلی مدل رگرسیون خطی برای پیش‌بینی تعداد درخواست‌ها در بازه زمانی بعدی، بر اساس رابطه (۸)، بیان می‌شود:

$$Y_{t+1} = \beta_1 + \beta_2 \times X_t \quad (۸)$$

که t اندیس نمونه مشاهده شده و Y_t متغیر وابسته (معرف بار کاری یا تعداد درخواست‌های کاربران) و X_t متغیر مستقل (معرف مقدار واقعی مهر زمانی لحظه‌ای است که نمونه t گرفته شده است) می‌باشد. همچنین، فرض می‌کنیم که n تعداد کل نمونه‌های مشاهده شده است. بنابراین مقادیر β_1, β_2 با حل معادله رگرسیون خطی بر اساس تکنیک حداقل مربعات، طبق رابطه (۹) و (۱۰) محاسبه می‌شود:

$$\beta_1 = \frac{\sum X_t^2 \sum Y_t - \sum X_t \sum X_t Y_t}{n\sum X_t^2 - (\sum X_t)^2} \quad (۹)$$

$$\beta_2 = \frac{n\sum X_t Y_t - \sum X_t \sum Y_t}{n\sum X_t^2 - (\sum X_t)^2} \quad (۱۰)$$

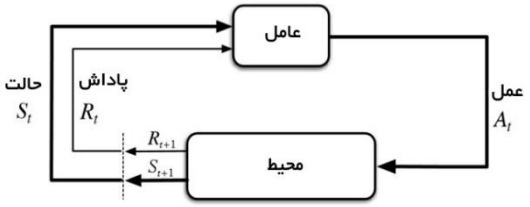
۳-۱-۴-مدیریت کشسانی با استفاده از اتوماتای یادگیر

با دریافت میزان بار کاری آتی پیش‌بینی شده از مؤلفه پیش‌بینی، وظیفه مؤلفه مدیریت کشسان، انتخاب اعمال مقیاس بندی مناسب با استفاده از یادگیری تقویتی [۸] برای تضمین خاصیت کشسانی است. به‌طور مشخص برای اولین بار کاری به هر میزان، مقیاس بندی انجام نخواهد شد، چون خروجی مؤلفه نگاشت کیفیت سرویس، در ابتدا وجود ندارد؛ اما در مراحل بعد، بر اساس میزان پیش‌بینی بار کاری توسط مؤلفه پیش‌بینی، مقیاس بندی انجام خواهد گرفت. یادگیری تقویتی به‌عنوان تصمیم گیر برای انجام عمل مقیاس بندی بر اساس میزان بار کاری فعلی ($\mathrm{WL}_{Current}$ یا X_t) و بار کاری پیش‌بینی شده ($\mathrm{WL}_{Predict}$ یا Y_{t+1}) مطابق رابطه (۱۱)، نوع مقیاس بندی و میزان آن را مشخص می‌کند.

$$Scaling\_Type = \begin{cases} UP & \mathrm{WL}_{Current} < WL_{predict} \\ NO-UP & \mathrm{WL}_{Current} = WL_{predict} \\ DOWN & \mathrm{WL}_{Current} > WL_{predict} \end{cases} \qquad (۱۱)$$
$$Scaling\_Amount = ABS\left(\mathrm{WL}_{Current} - WL_{predict}\right)$$

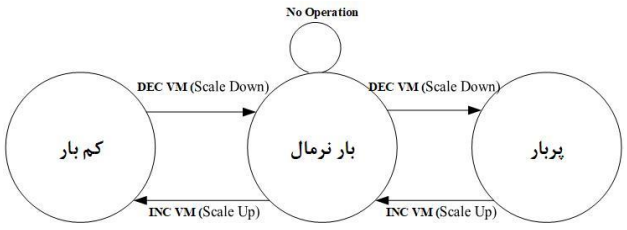
همان‌طورکه اشاره شد، در بخش مدیریت کشسان، یادگیری تقویتی وظیفه تصمیم‌گیری در مورد تصمیمات مقیاس بندی را به عهده دارد. یادگیری تقویتی[۹]، یک فرایند یادگیری است که در آن، عامل یادگیر تقویتی از طریق سعی و خطا با محیط تعامل کرده و یاد می‌گیرد تا عملی بهینه را برای رسیدن به هدف انتخاب کند. در این نوع یادگیری، هیچ‌گونه ناظر خارجی وجود نداشته و عامل به‌تنهایی با محیط تعامل کرده، یاد می‌گیرد، و تجربه کسب می‌کند و پاداشی دریافت می‌کند. در یادگیری تقویتی، عامل به حسگرهایی مجهز شده که می‌تواند ویژگی‌های قطعی محیط را دریافت کند. این ویژگی‌ها، فضای حالت عامل یادگیرنده را تشکیل می‌دهند. سپس در هر بازه زمانی عامل با انجام عملیاتی، محیط را تحت تأثیر قرار می‌دهد. بنابراین عامل ورودی‌های مختلف را در بازه زمانی بعدی با توجه به عمل قبلی دریافت می‌کند. علاوه بر ورودی جدید، عامل یادگیرنده در هر زمان یک سیگنال تقویتی که میزان مطلوبیت عمل قبلی را نشان می‌دهد، دریافت می‌کند که به آن پاداش می‌گویند و با توجه به این که عمل مناسب بوده یا خیر، این مقدار می‌تواند مثبت یا منفی باشد. از یادگیری تقویتی می‌توان برای یافتن پاسخ بهینه بسیاری از مسائل استفاده نمود، ولی این مسائل بایستی در چارچوبی مشخص بیان شوند. بدین منظور از فرایند تصمیم‌گیری مارکوف (MDP[۱۰]) استفاده می‌شود. به مسئله یادگیری تقویتی که در آن خاصیت مارکوف برقرار باشد، فرآیند تصمیم‌گیری مارکوف گویند. اگر فضای حالات و مجموعه اعمال ممکن متناهی باشد، به فرآیند تصمیم‌گیری، مارکوف متناهی گفته می‌شود.

یک مسئله یادگیر تقویتی مدل شده به‌صورت یک فرآیند تصمیم‌گیری مارکوف، شامل مجموعه متناهی از وضعیت‌های S و یک مجموعه اعمال a و یک سیگنال تقویتی r می‌باشد. برای یک مجموعه وضعیت S و یک مجموعه اعمال A، یک تابع پاداش $R: S\times A\rightarrow R$ و یک تابع انتقال وضعیت $T: S\times A\rightarrow \pi(S)$ می‌باشد که $\pi(S)$ تابع توزیع احتمال روی S می‌باشد. هدف یک عامل یادگیر تقویتی، پیدا کردن یک سیاست بهینه ($\pi(S)$) برای انتخاب عملی است که پاداش عامل را در طولانی‌مدت به حداکثر برساند. همان‌طورکه در شکل ۳ نشان داده شده است برای یک سیستم در زمان t با وضعیت $s_t \in S$، عامل یادگیر تقویتی عمل $a_t \in A$ را انتخاب می‌کند. بعد از اینکه عمل a_t به محیط اعمال شد، وضعیت از s_t به s_{t+1} انتقال می‌یابد و پاداش r_{t+1} از محیط برگردانده می‌شود.



شکل ۳: ارتباط بین عامل یادگیر تقویتی و محیط

در این بخش، یک روش مبتنی بر یادگیری تقویتی با استفاده از مدل‌سازی فرایند تصمیم گیر مارکوف (MDP) به‌منظور انتخاب عمل بهینه پیشنهاد می‌دهیم. یک MDP معمولاً با تعدادی حالت و انتقالات بین این حالات نمایش داده می‌شود. در این فاز بر اساس میزان بار کاری پیش‌بینی شده ($\mathrm{WL}_{Predict}$) توسط مؤلفه پیش‌بینی کننده، وضعیت سیستم را به‌صورت یک MDP مطابق شکل ۴ شامل سه وضعیت تعریف می‌کنیم: بار نرمال، کم بار[۱۱]، پر بار[۱۲]. این وضعیت‌ها به ترتیب مشخص می‌کنند که آیا میزان منابع متناسب با میزان درخواست‌ها، تأمین شده‌اند (وضعیت بار نرمال)، منابع بیشتر از حد میزان درخواست‌ها تأمین شده‌اند (وضعیت کم بار) و یا اینکه منابع کم‌تر از حد میزان درخواست‌ها تأمین شده‌اند (وضعیت پر بار). بر اساس رابطه (۱۱)، اگر میزان بار کاری پیش‌بینی شده ($\mathrm{WL}_{Predict}$) کمتر از میزان بار کاری فعلی ($\mathrm{WL}_{Current}$) باشد، وضعیت سیستم به‌صورت کم بار و اگر میزان بار کاری پیش‌بینی شده ($\mathrm{WL}_{Predict}$) بیشتر از میزان بار کاری فعلی ($\mathrm{WL}_{Current}$) باشد، وضعیت سیستم به‌صورت پر بار در نظر گرفته خواهد شد و گرنه وضعیت سیستم به‌صورت بار نرمال خواهد بود.



شکل ۴: MDPپیشنهادی

علاوه بر این، جدول تصمیم‌گیری متناظر با MDP برای انتخاب یکی از اعمال مقیاس بندی (Scale Up, Scale Down, No op) بر اساس وضعیت فعلی سیستم در جدول ۶ نشان داده شده است.

جدول ۶: جدول تصمیم‌گیری متناظر با MDP

	$WL_{Current} > WL_{predict}$	$WL_{Current} \cong WL_{predict}$	$WL_{Current} < WL_{predict}$
وضعیت در لحظه t (State(t))	کم بار	بار نرمال	پر بار
انتخاب عمل در لحظه t+1 Action(t+1)) (	Scale Down(DEC VM)	No-Operation	Scale UP(INC VM)

همچنین، از الگوریتم Q-Learning به‌عنوان نمونه‌ای از استراتژی یادگیری تقویتی به‌منظور انتخاب عمل بهینه استفاده می‌کنیم. الگوریتم Q-Learning توسط تابع Q تعریف می‌شود. Q(s, a) مقدار تابعی است که عامل در لحظه t عمل a_t را در وضعیت فعلی s_t انجام می‌دهد. تابع Q، بعد از بکارگیری عمل به محیط، بر اساس رابطه (۱۲) بروز رسانی می‌شود:

$$Q(s_t, a_t) \leftarrow r(s,a) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \quad (۱۲)$$

در این رابطه، γ فاکتور تخفیف می‌باشد ($0 \le \gamma \le 1$) که یک مقدار نزدیک به ۱ برای γ، وزن بیشتری به پاداش‌های آتی انتساب می‌دهد درحالی‌که مقادیر نزدیک به ۰ فقط آخرین (جدیدترین) پاداش‌ها را در نظر می‌گیرد. مراحل کلی الگوریتم Q-Learning عبارتند از: ۱) مشاهده وضعیت فعلی ۲) تصمیم گرفتن برای انتخاب عمل ۳) انجام عمل و دریافت سیگنال تقویتی برای رسیدن به وضعیت جدید ۴) بروز رسانی تابع Q (یادگیری از تجربیات) و ۶) تکرار مراحل ۱ تا ۴.

۴- ارزیابی کارایی

در این بخش، قصد داریم نتایج پیاده‌سازی راهکار پیشنهادی (BM-RLLR[۱۳]) را بر اساس سه معیار خاصیت کشسانی، زمان پاسخگویی و میزان بهره‌وری مورد ارزیابی قرار می‌دهیم. برای شبیه‌سازی از ابزار [۲۰] CloudSim استفاده می‌کنیم و با دو راهکار [۱۰] CTMC و [۱۱] ControCity مقایسه می‌کنیم. در ادامه، ارزیابی کارایی راهکار پیشنهادی را در سه بخش، تنظیمات شبیه‌سازی، معیارهای کارایی و تحلیل نتایج با جزئیات بیشتر توضیح می‌دهیم.

۴-۱- تنظیمات شبیه‌سازی

برای شبیه‌سازی دقیق و قابل توسعه ابر از ابزار CloudSim استفاده کرده‌ایم. بدین دلیل که ابزار CloudSim به زبان جاوا نوشته شده است، محیط توسعه NetBeans به‌عنوان محیط برنامه‌نویسی جاوا انتخاب شده است. در شبیه سازی روش پیشنهادی از سه نوع بارکاری واقعی شامل سه مجموعه داده [۲۱] Yahoo Cluster و [۲۲] Google Cluster و [۲۳] Wikipedia برای ارزیابی کارایی راهکار پیشنهادی با دو راهکار[۱۰] CTMC و [۱۱] ControCity استفاده می‌کنیم

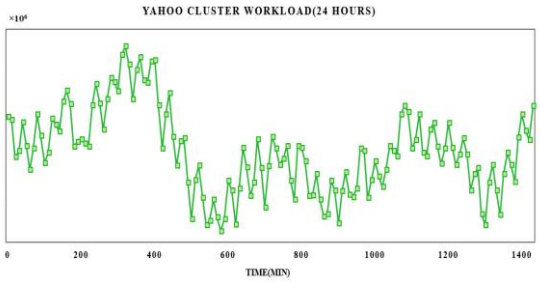
راهکار CTMC [۱۰]، از یک مدل تحلیلی بر اساس زنجیره مارکوف پیوسته برای تخمین تعداد ماشین‌های مجازی موردنیاز برای تنظیم و مدیریت میزان کشسانی پلت فرم‌های ابری استفاده می‌کند.

راهکار ControCity [۱۱]، یک راهکار مبتنی بر مدیریت بافر می باشد که از تکنیک اتوماتای یادگیر برای کنترل خاصیت کشسانی پلت فرم‌های ابری استفاده می‌کند. همچنین این راهکار، برای پیش‌بینی تعداد درخواست‌ها، از تکنیک میانگین متحرک برای اتخاذ تصمیمات مقیاس بندی مناسب و اصلاح سیاست‌های کشسانی در زمان اجرا استفاده می‌کند.

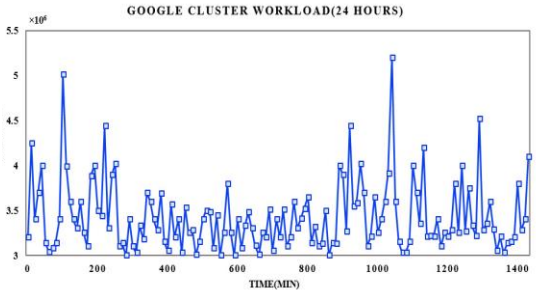
دلیل انتخاب این دو راهکار برای مقایسه با راهکار پیشنهادی بدین علت است که اولاً: هر دو از سیاست پیش کنشی پیروی می‌کنند، ثانیاً هر دو از مقیاس بندی افقی برای تأمین منابع در راستای مدیریت خاصیت کشسانی استفاده می‌کنند.

در شبیه‌سازی راهکار پیشنهادی از سه نوع بار کار واقعی Google Cluster، Yahoo Cluster و Wikipedia استفاده می‌کنیم. ویژگی‌های مختلف این بارهای کارهای، نتایج واقعی‌تر و کاربردی‌تر را برای برنامه‌های کاربردی ابری به همراه خواهد داشت. بار کاری Yahoo Cluster [۲۱] از سیستم‌های تولید شرکت یاهو استخراج شده و شامل یکسری معیارهای سیستمی مانند بهره‌وری پردازنده، بهره‌وری حافظه و دیسک در طول عملیات پایگاه داده PNUTS/Sherpa بر روی یک خوشه هدوپ می‌باشد. همچنین، این بار کاری، مجموعه داده مربوط به آمار دسترسی فایل‌ها مانند اندازه کل فایل، تعداد کل فایل‌ها، تعداد دسترسی فایل، تعداد روزهای بین اولین و آخرین دسترسی و .. را فراهم می‌کند. بار کاریYahoo Cluster در یک بازه زمانی ۱۰ روزه جمع آوری شده و شامل ۳۷،۲۲۵،۸۷۴،۶۳۸ درخواست می‌باشد. بار کاری Google Cluster [۲۲] توسط گوگل ارائه شده و شامل اطلاعات مربوط به سلول روی یک خوشه در حدود ۱۲،۵ هزار ماشین در روز ۲۹ مه ۲۰۱۱ می‌باشد. در این مقاله، بار کاری ClusterData2011_2 به‌عنوان بار کاریGoogle Cluster مورد استفاده قرار گرفته که در آن ۶،۲۵۳،۷۷۱،۴۲۹ درخواست در بازه زمانی ۴ روز وجود دارد. این بار کاری شامل پنج جدول به نام: رویدادهای ماشین، ویژگی‌های ماشین، رویدادهای شغلی، رویدادهای کار، محدودیت‌های کاری و استفاده از منابع کاری می‌باشد. بار کاری Wikipedia [۲۳] شامل یکسری درخواست‌های وب تولید شده از سرورهای ویکی پدیا و شامل ۲۰،۶ میلیارد درخواست‌های HTTP برای بازه زمانی از ۱۹ سپتامبر ۲۰۰۷ تا ۲ ژانویه ۲۰۰۸ می‌باشد. هر درخواست در بار کاری Wikipedia شامل یک شناسه منحصربه‌فرد درخواست، آدرس URL درخواست و مهر زمانی آن درخواست می‌باشد. بار کاریWikipedia شامل ۴،۲۴۷،۵۳۱،۸۸۳ درخواست HTTP متناسب با یک روز هفته می‌باشد. در شبیه‌سازی ۲۴

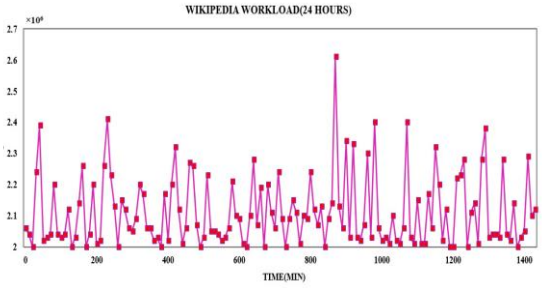
ساعت از ساعات پر درخواست هر مجموعه با سیکل زمانی ۱۰ دقیقه انتخاب شده است. نمایی از داده واقعی انتخابی ۲۴ ساعته در سه بار کاری در شکل‌های ۵ تا ۷ نشان داده شده است.



شکل ۵: نمایی از ۲۴ ساعت از مجموعه داده Yahoo Cluster



شکل ۶: نمایی از ۲۴ ساعت از مجموعه داده Google Cluster



شکل ۷: نمایی از ۲۴ ساعت از مجموعه داده Wikipedia

همچنین، ساختار تمامی مراکز داده مورد استفاده در شبیه‌سازی در جدول ۷ و پارامترهای مورد استفاده در آن‌ها در جدول ۸ نشان شده است.

جدول ۷: ساختار مراکز داده مورداستفاده در شبیه‌سازی

معماری	X86
سیستم‌عامل	Cloud Linux
مدیریت ماشین‌های مجازی	XEN

جدول ۸: پارامترهای مراکز داده مورداستفاده در شبیه‌سازی

پارامتر	مقدار (میلی‌ثانیه)
α	۰/۷
H_thr	٪۸۵
L_thr	٪۱۵

در شبیه‌سازی برای هر مرکز داده، ۵ میزبان فیزیکی تعریف‌شده که به‌صورت تصادفی یکی از سه نوع ماشین زیر در آن‌ها جای می‌گیرد. ماشین‌های مجازی استفاده‌شده در شبیه‌سازی راهکار پیشنهادی، در سه گروه و از هر گروه، سه نمونه انتخاب‌شده است. جدول ۹، مشخصات ماشین‌های مجازی را در هر میزبان نشان می‌دهد.

جدول ۹: مشخصات ماشین‌های مجازی مراکز داده

Machin Name	MIPS	RAM (GB)	Storage	BW	Price ($ per Hour)
Small	2000	۱	5 GB	100Mbps	۰/۲
Medium	4000	۲	20 GB	1Gbps	۰/۷
Large	5000	۴	50 GB	1Gbps	۱/۲

۴-۲- معیارهای کارآیی

معیارهای کارایی مورد استفاده برای ارزیابی عبارتند از : کشسانی، زمان پاسخگویی و میانگین بهره‌وری[۱۴].

• کشسانی

با توجه به این‌که T_m زمان انجام کل شبیه‌سازی خواهد بود، این زمان متشکل از سه زمان وضعیت تأمین نرمال T_n، وضعیت تأمین بیش‌ازحد منابع T_o و وضعیت کمبود تأمین منابع T_u خواهد بود، همان‌طورکه در رابطه (۱۳) نشان داده‌شده است.

$$T_m = T_n + T_o + T_u \quad (۱۳)$$

خاصیت کشسانی، انعکاس وضعیتی است که ابر تحت نوسان حجم کاری تغییر می‌کند و می‌تواند به‌وسیله تصمیم‌گیری، تعداد ماشین‌های مجازی مناسب را در اختیار درخواست‌ها قرار دهد. در واقع کشسانی عبارت است از زمانی است که ابر در وضعیت تأمین نرمال قرار دارد [۱۰]. پس مطابق رابطه (۱۴) کشسانی برابر با نسبت T_n به T_m.

$$E = \frac{T_n}{T_m} = 1 - \left(\frac{T_o}{T_m} + \frac{T_u}{T_m} \right) \quad (۱۴)$$

درصورتی‌که P_n احتمال وضعیت تأمین نرمال، P_o احتمال وضعیت تأمین بیش‌ازحد منابع و P_u احتمال وضعیت کمبود تأمین منابع در نظر گرفته شود، مقادیر این سه از رابطه (۱۵) حاصل می‌گردد.

$$P_n = \frac{T_n}{T_m}, \qquad P_o = \frac{T_o}{T_m}, \qquad P_u = \frac{T_u}{T_m} \quad (۱۵)$$

بنابراین کشسانی از رابطه (۱۶) حاصل می‌گردد.

$$E = P_n = 1 - (P_o + P_u) \quad (۱۶)$$

• زمان پاسخگویی

زمان پا سخ واقعی، تفاوت زمانی دقیق بین زمان درخوا ست کار و زمان تحویل کار انجام شده به کاربر می‌باشد.

• میانگین بهره‌وری

میزان MIPS های موردنیاز تقسیم بر MIPS های در دسترس را بهره‌وری می‌گویند. که طبق رابطه (۱۷) محاسبه می‌شود.

$$Utilization = \frac{Allocated\_MIPS}{Available\_MIPS} \quad (۱۷)$$

۴-۳- تحلیل نتایج

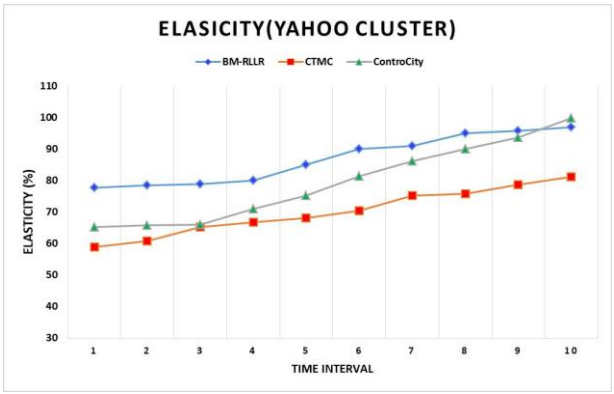
جهت ارزیابی روش پیشنهادی(BM-RLLR)، ۳ سناریو مطابق با جدول (۱۰) برای ارزیابی معیارهای کشسانی، زمان پاسخ گوئی و بهره‌وری ایجاد کرده‌ایم.

جدول (۱۰) سناریوهای مورد ارزیابی

سناریو	بار کاری	هدف
سناریو اول	Yahoo Cluster, Google Cluster, Wikipedia	مقایسه کشسانی
سناریو دوم	Yahoo Cluster, Google Cluster, Wikipedia	مقایسه زمان پاسخگویی
سناریو سوم	Yahoo Cluster, Google Cluster, Wikipedia	مقایسه بهره‌وری

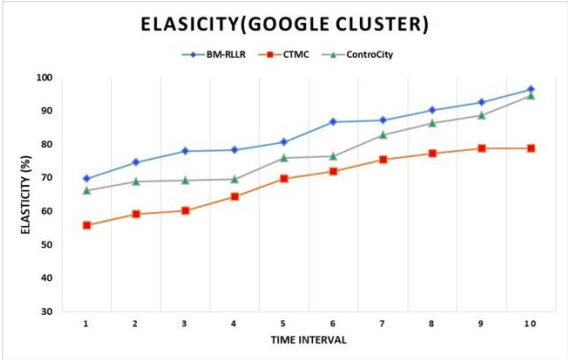
۴-۳-۱- سناریو اول: مقایسه میزان کشسانی

یکی از شاخص‌های مهم در مقایسه عملکرد راهکار تأمین منابع، خاصیت کشسانی است. برای انجام این آزمایش، ده بازه زمانی مختلف از هر بار کاری را انتخاب شد به‌صورتی که تعداد درخواست‌ها از بازه اول تا دهم سیر صعودی داشته باشد. شکل ۸، نتایج شبیه‌سازی معیار خاصیت کشسانی در راهکار BM-RLLR را در مقایسه با دو راهکار CTMC و ControCity را در بار کاری Yahoo Cluster نمایش می‌دهد. با توجه به شکل ۸، به دلیل استفاده از صف اولویت در بخش مدیریت بافر و همچنین اضافه و کم کردن دقیق ماشین مجازی با استفاده از الگوریتم یادگیری تقویتی، راهکار پیشنهادی کارآیی بهتری در زمینه خاصیت کشسانی دارد.

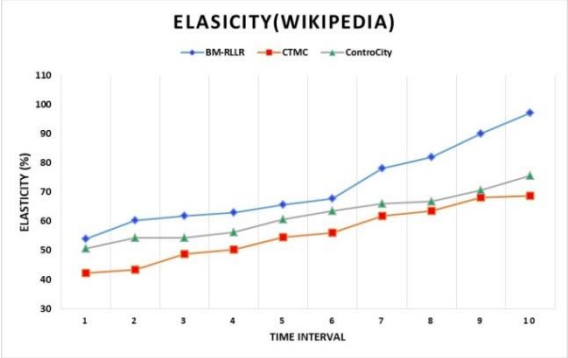


شکل ۸: مقایسه میزان کشسانی در بارکاری Yahoo Cluster

شکل ۹ و ۱۰ آزمایش خاصیت کشسانی در راهکار BM-RLLR را در مقایسه با دو راهکار CTMC و ControCity را در دو بار کاری Google Cluster و Wikipedia نمایش می‌دهد. در هر سه بار کاری میزان کشسانی راهکار پیشنهادی نسبت به دو راهکار CTMC و ControCity بالاتر است.



شکل ۹: مقایسه میزان کشسانی در بارکاری Google Cluster

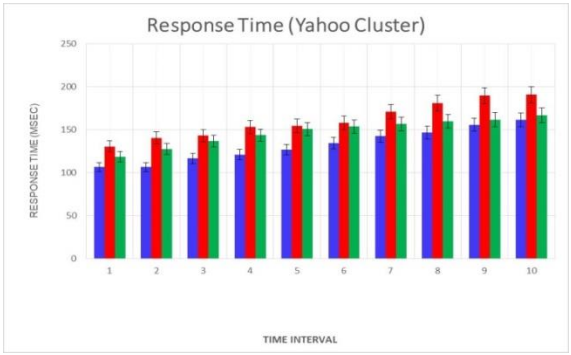


شکل ۱۰: مقایسه میزان کشسانی در بارکاری Wikipedia

میانگین بهبودی میزان کشسانی در راهکار پیشنهادی برای هر سه بار کاری نسبت به دو راهکار CTMC و ControCity به ترتیب ۲۸/۱ درصد و ۱۱/۶۶ درصد است.

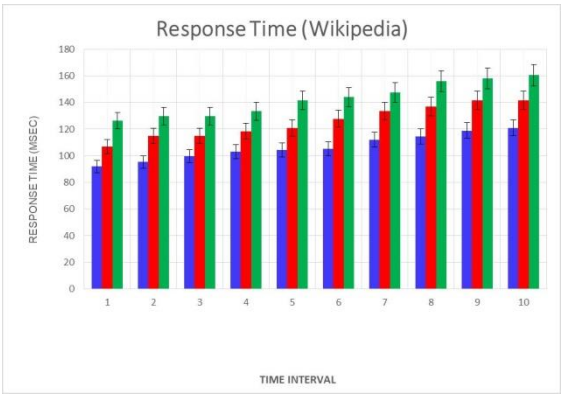
۴-۳-۲- سناریو دوم: مقایسه زمان پاسخگویی

زمان پاسخگویی به‌عنوان یک از مؤثرترین اهداف سطح سرویس، نقش مؤثری در انتخاب ماشین مجازی دارد. درصورتی‌که زمان پاسخگویی موردنظر درخواست (حد مجاز زمان پاسخگویی) برآورده نشود، باید مقیاس بندی صحیح برای بهبود خاصیت کشسانی انجام شود. شکل ۱۱ میانگین زمان پاسخگویی در راهکار BM-RLLR را در مقایسه با دو راهکار CTMC و ControCity را در بار کار Yahoo Cluster نمایش می‌دهد. به دلیل عملکرد مطلوب یادگیری تقویتی به‌عنوان تصمیم گیر سیستم در بخش مدیریت کشسان و مؤلفه نگاشت کیفیت سرویس، تأمین منابع برای درخواست‌ها به‌طور دقیق‌تری انجام می‌گیرد. به‌تناسب، تأمین منبع دقیق‌تر، موجب افزایش سرعت پاسخگویی می‌شود. با توجه به نتایج، راهکار پیشنهادی کارآیی بهتری در مورد زمان پاسخگویی دارد. شکل۱۲میانگین زمان پاسخگویی در راهکار BM-RLLR را در مقایسه با دو راهکار CTMC و ControCity را در بار کار Google Cluster نمایش می‌دهد.



شکل ۱۱: مقایسه میانگین زمان پاسخگویی بارکاری Yahoo Cluster

شکل ۱۲: مقایسه میانگین زمان پاسخگویی بارکاریGoogle Cluster

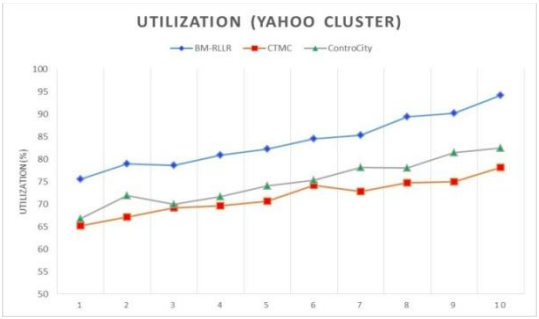


شکل ۱۳: مقایسه میانگین زمان پاسخگویی بارکاری Wikipedia

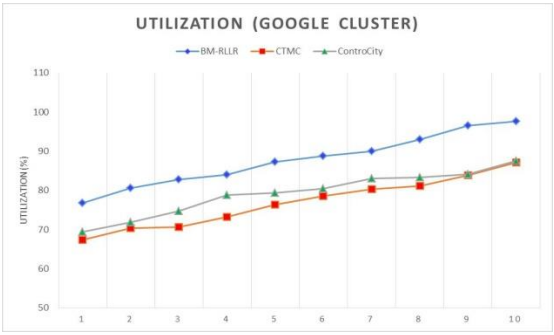
با توجه به شکل ۱۳ میانگین زمان پاسخگویی در بار کاری Google Cluster در راهکار پیشنهادی کمتر از راهکار CTMC و ControCity است. دلیل این امر کارکرد مناسب یادگیری تقویتی و مدیریت مناسب بافر است. در هر سه بار کاری، میانگین زمان پاسخگویی در راهکار پیشنهادی نسبت به دو راهکار CTMC و ControCity کمتر است که این نشان‌دهنده افزایش کیفیت ارائه سرویس در راهکار پیشنهادی است. میانگین زمان پاسخگویی در راهکار پیشنهادی برای هر سه بار کاری نسبت به دو راهکار CTMC و ControCity به ترتیب ۱۷/۳۳درصد و ۱۳/۲ درصد کاهش داشته است.

۴-۳-۳- سناریو سوم: مقایسه میزان بهره‌وری

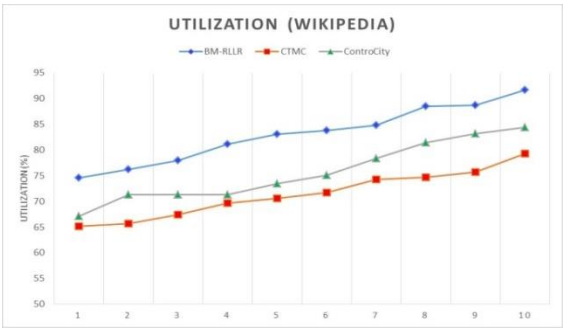
شکل ۱۴ میانگین بهره‌وری در بار کاری Yahoo Cluster را نمایش می‌دهد. استفاده از تکنیک رگرسیون خطی در مولفه پیش‌بینی بار کاری، عامل بسیار مؤثری برای کنترل وضعیت منابع است. به دلیل عملکرد مطلوب برنامه تکنیک رگرسیون خطی و همچنین استفاده از یادگیری تقویتی در بخش مدیریت کشسان، تأمین منابع به نحو مطلوب انجام می‌گیرد. تأمین صحیح منابع موجب ارائه دقیق خدمات به درخواست‌ها می‌شود. به‌طور مشخص به دلیل تأمین صحیح منابع پردازشگر با بهترین وضعیت در اختیار درخواست‌ها قرار می‌گیرد. البته ممکن است در تعداد درخواست بالا میزان بهره‌وری به آستانه یک برسد که دلیل آن تعداد درخواست‌های بالا است که نیاز به در اختیار گرفتن پردازنده دارند.



شکل ۱۴: مقایسه بهره‌وری در بارکاری Yahoo Cluster



شکل ۱۵: مقایسه بهره‌وری در بارکاری Google Cluster



شکل ۱۶: مقایسه بهره‌وری در بارکاری Wikipedia

استفاده از مدیریت بافر تأثیر بسیار زیادی در کنترل مناسب درخواست‌ها و تصمیم‌گیری در مورد کشسانی سیستم ابر شده است. بدین معنی که مولفه مدیریت کشسان، به‌سادگی می‌تواند به ازای رخداد ورود سرویس یا پاسخگویی به آن در مورد اضافه یا کم کردن ماشین مجازی تصمیم‌گیری کند. شکل ۱۵ و ۱۶ میزان بهره‌وری در راهکار BM-RLLR را در مقایسه با دو راهکار CTMC و ControCity را در دو بار کاری Google Cluster و Wikipedia نمایش می‌دهد. در هر سه بار کاری میزان بهره‌وری در راهکار پیشنهادی نسبت به دو راهکار CTMC

و ControCity بالاتر است که این نشان‌دهنده افزایش کیفیت ارائه سرویس در این ساختار است. میانگین بهره‌وری در راهکار پیشنهادی برای هر سه بار کاری نسبت به دو راهکار CTMC و ControCity به ترتیب ۱۵/۳۳ درصد و ۱۱/۱ درصد افزایش داشته است.

۵- نتیجه‌گیری

در این مقاله راهکاری ترکیبی برای بهبود خاصیت کشسانی با استفاده مدیریت بافر و مدیریت کشسانی ارائه شد. در راهکار پیشنهادی، مدیریت بافر وظیفه کنترل صف ورودی درخواست‌ها را به عهده دارد و مدیریت کشسانی با استفاده از یادگیری تقویتی، کنترل خاصیت کشسانی سیستم را به عهده دارد. همچنین، مولفه مدیریت کشسانی از اطلاعات سیستم نگاشت کیفیت سرویس‌دهی برای تأمین صحیح منابع استفاده می‌کند. نتایج ارزیابی بهبود عملکرد راهکار پیشنهادی نسبت به راهکارهای دیگر بر روی سه نوع بار کاری واقعی در زمینه افزایش بهره‌وری، کاهش زمان پاسخگویی و افزایش خاصیت کشسانی را نشان می‌دهد. به‌منظور تکمیل و توسعه این تحقیق، استفاده از شبکه پتری فازی و یا استفاده از تکنیک یادگیری عمیق به‌جای مدل یادگیری تقویتی برای کنترل خاصیت کشسانی پیشنهاد می‌شود.

مراجع

[1] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of internet services and applications* vol.1, no. 1,pp.7-18, 2010.

[۲] شهرام جمالی و سمیرا حورعلی، «موازنه گر نامتمرکز بار در محیط ابر با بهره گیری از سیاست تصمیم‌گیری چند شاخصه»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۶، شماره ۳، صفحه ۱۰۶-۹۶، ۱۳۹۵.

[3] Ghobaei-Arani, Mostafa, Sam Jabbehdari, and Mohammad Ali Pourmina. "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach." *Future Generation Computer Systems* vol. 78, no. 1, pp.191-210, 2018.

[۴] سیمین قاسمی فلاورجانی، محمدعلی نعمت بخش و بهروز شاهقلی قهفرخی، «تخصیص وظایف چند هدفه در واگذاری به ابر سیار»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۶، شماره ۴، صفحه ۲۳۲-۲۱۷، ۱۳۹۵.

[5] Herbst, Nikolas Roman, Samuel Kounev, and Ralf Reussner. "Elasticity in cloud computing: What it is, and what it is not." In *Proceedings of the 10th International Conference on Autonomic Computing ({ICAC} 13)*, pp. 23-27. 2013.

[6] Al-Dhuraibi, Yahya, Fawaz Paraiso, Nabil Djarallah, and Philippe Merle. "Elasticity in cloud computing: state of the art and research challenges." *IEEE Transactions on Services Computing* vol 11, no.1, pp.430-447, 2017.

[7] 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.

[8] Botvinick, Mathew, et al. "Reinforcement learning, fast and slow." *Trends in cognitive sciences* vol.23, no. 5, pp.408-422, 2019.

[9] Beltrán, Marta. "BECloud: A new approach to analyse elasticity enablers of cloud services." *Future Generation Computer Systems* vol.64, no.1, pp.39-49, 2016.

[10] Li, Keqin. "Quantitative modeling and analytical calculation of elasticity in cloud computing." *IEEE Transactions on Cloud Computing*, 2017.

[11] Ghobaei-Arani, M., Souri, A., Baker, T. and Hussien, A., "ControCity: An Autonomous Approach for Controlling Elasticity Using Buffer Management in Cloud Computing Environment." *IEEE Access* vol. 7, no. 1, pp.106912-106924, 2019.

[12] Ullah, Amjad, Jingpeng Li, Yindong Shen, and Amir Hussain. "A control theoretical view of cloud elasticity: taxonomy, survey and challenges." *Cluster Computing* vol 21, no. 4, pp.1735-1764, 2018.

[13] Han, Rui. "Investigations into elasticity in cloud computing." *arXiv preprint arXiv:1511.04651*, 2015.

[14] Albonico, Michel, Jean-Marie Mottu, and Gerson Sunyé. "Controlling the elasticity of web applications on cloud computing." In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 816-819. ACM, 2016.

[15] Computing, Autonomic. "An architectural blueprint for autonomic computing." *IBM White Paper* vol.31, no.1, pp. 1-6, 2006.

[16] Huebscher, Markus C., and Julie A. McCann. "A survey of autonomic computing—degrees, models, and applications." *ACM Computing Surveys (CSUR)* vol. 40, no. 3, p.7, 2008.

[17] Hariri, Salim, Bithika Khargharia, Houping Chen, Jingmei Yang, Yeliang Zhang, Manish Parashar, and Hua Liu. "The autonomic computing paradigm." *Cluster Computing* vol.9, no. 1, pp.5-17, 2006.

[18] https://docs.aws.amazon.com/batch/latest/userguide/job_scheduling.html

[19] Messias, Valter Rogério, Julio Cezar Estrella, Ricardo Ehlers, Marcos José Santana, Regina Carlucci Santana, and Stephan Reiff-Marganiec. "Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure." *Neural Computing and Applications* vol. 27, no. 8, pp. 2383-2406, 2016.

[20] Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and experience vol. 41, no. 1, pp. 23-50, 2011.

[21] https://webscope.sandbox.yahoo.com/catalog.php?datatype=s

[22] Reiss, C., Wilkes, J., Hellerstein, J.L.: Google cluster-usage traces: format ? schema. Google Inc., White Paper, pp. 1–14, 2011

[23] Urdaneta, G., Pierre, G., Van Steen, M.:"Wikipedia workload analysis for decentralized hosting." Comput. Netw. vol. 53, no. 11, pp.1830–1845, 2009.

زیرنویس ها

[1] Service Level Agreement
[2] Continuous-Time Markov Chains
[3] Million Instruction Per Second
[4] Monitor-Analyze-Plan-Execute
[5] Service Level Objective
[6] SLA Violation
[7] Amazon Web Services
[8] Linear Regression
[9] Reinforcement Learning
[10] Markov Decision Process
[11] Under-load
[12] Over-load
[13] Buffer Management-Reinforcement Learning, Linear Regression
[14] Utilization