مدیریت کشسانی در رایانش ابری با استفاده از شبکه پتری رنگی

علی شهیدی‌نژاد[1]، استادیار

۱- گروه مهندسی کامپیوتر، واحد قم، دانشگاه آزاد اسلامی، قم، ایران- a.shahidinejad@qom-iau.ac.ir

چکیده: رایانش ابری، فناوری جدیدی است که روزبه‌روز بر محبوبیت آن افزوده می‌شود، این محبوبیت به دلیل خاصیت کشسانی آن است. به‌عبارت دیگر، رایانش ابری، ظرفیت منابع را برای مصرف‌کننده به‌صورت بینهایت در نظر می‌گیرد و مصرف‌کننده، می‌تواند منابع را برحسب تقاضا و بر اساس نرخ رقابتی در اختیار بگیرد و میزان منابع را افزایش یا کاهش دهد. اگرچه 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 مدلی برای بهبود خاصیت کشسانی با استفاده از شبکه پتری رنگی برای تأمین منابع در شبکه‌های ابری ارائه می‌شود. در مدل پیشنهادی مدیریت کشسانی با استفاده شبکه پتری رنگی و در قالب کنترل صف‌های M/M/N صورت می‌گیرد. بدین ترتیب که به ازای ورود هر درخواست یا ارائه سرویس در صف حرکت افقی و به ازای نیاز به افزایش یا کاهش ماشین مجازی حرکت عمودی در صف وجود دارد. نتایج عملکرد روش پیشنهادی تحت بار کار واقعیGoogle cluster بهبود خاصیت کشسانی، افزایش دقت و افزایش سرعت را در مقایسه با رویکردهای مشابه نشان می‌دهد.

واژه‌های کلیدی: خاصیت کشسانی، شبکه پتری رنگی، رایانش ابری.

Elasticity Management in Cloud Computing Using Colored Petri Net

Ali Shahidinejad[1], Assistant Professor

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

Abstract: Cloud computing is a new technology which its popularity increases every day, a popularity due to its elasticity. On the other words, cloud computing takes into account an unlimited capacity of the resource for the consumer, and the consumer can take resources in demand based on competitive rates and increase or decrease the amount of resources. There have been many improvements to elasticity management by previous researches. However, further reasearches are necessary to manage elasticity more efficiently. In this paper, a model for the elasticity improvement using a colored Petri network is proposed to provide resources in cloud computing. In the proposed model, elasticity management is performed using a colored Petri net in the form of control of the M/M/N queues. In this way, there is a horizontal queue for each request or service in the vertical queue for the need to increase or decrease the virtual machine. The results of the proposed method show an improvment in elasticity, accuracy and speed, compared with the other approaches.

Keywords: Elasticity, Colored Petri Net, Cloud Computing.

تاریخ ارسال مقاله: ۱۳۹۷/۰۶/۰۵
تاریخ اصلاح مقاله: ۱۳۹۷/۰۹/۱۷
تاریخ پذیرش مقاله: ۱۳۹۸/۰۱/۱۸
نام نویسنده مسئول: علی شهیدی‌نژاد
نشانی نویسنده مسئول: ایران - قم - پردیسان - بلوار دانشگاه - دانشگاه آزاد اسلامی واحد قم - دانشکده فنی مهندسی.

۱- مقدمه

رایانش ابری، امکان دسترسی به محدوده وسیعی از منابع مجازی را فراهم می‌آورد که این منابع می‌توانند به‌صورت پویا و با حجم متغیر به کاربران خود سرویس دهند، یعنی هر کاربر به اندازه‌ای که منابع نیاز دارد، از آن منابع استفاده می‌کند که این کار به استفاده بهینه از منابع منجر می‌شود. رایانش ابری، باید به‌صورت خودکار منابع مورد نیاز را به کاربران تخصیص دهد و اگر کاربری مقداری بیشتر از منابع را خواستار باشد، باید آن منابع بدون آن‌که سرویس کاربر با مشکل مواجه شود، به مشتری تخصیص یابد و درصورتی‌که منابع مورد استفاده بیش از حد مورد نیاز باشد، باید منابع اضافی به‌صورت موقت خاموش شده تا در صورت نیاز مجدداً مورد استفاده قرار گیرند [۱]. این مفاهیم در دنیای رایانش ابری تحت عنوان مقیاس‌پذیری شناخته می‌شود.

مقیاس‌پذیری، توانایی یک سیستم در برابر افزایش بار کاری می‌باشد به این صورت است که منابع محاسباتی جدید را اضافه می‌نماید تا کارایی خودش را حفظ نماید، درصورتی‌که خاصیت کشسانی مربوط می‌شود به اینکه سیستم چگونه به‌خوبی منابع مورد نیاز را در هر نقطه از زمان به‌صورت خودکار و بر اساس بار کاری تأمین نماید. سیاست کشسانی، اشاره به تعاملات مورد نیاز به‌منظور انجام اقداماتی است که موجب تضمین خاصیت کشسانی می‌شود. اقدامات کشسانی به‌صورت خودکار انجام می‌شوند. درحالی‌که مقیاس‌پذیری می‌تواند به‌صورت دستی و توسط کاربر انجام شود. خاصیت کشسانی، شامل افزایش و کاهش اشتراک منابع انتزاعی برحسب تقاضا می‌باشد، درصورتی‌که مقیاس‌پذیری تنها شامل افزایش منابع می‌باشد و شامل چگونگی افزایش منابع بر اساس سطح کارایی مورد نظر نمی‌باشد. اگرچه بسیاری از راه‌حل‌های کشسانی تاکنون توسعه داده شده‌اند تا مشکلات کشسانی در ابر را کاهش دهند، کارهای زیادی نیاز است تا خاصیت کشسانی ابر را به‌طور کاراتری مدیریت نمایند.

کارهای گذشته، برای تأمین پویای منابع، شرایط توافق سرویس در برنامه‌های کاربردی چندلایه را ملاک قرار داده‌اند و تمرکز خود را بر روی پردازش‌های دسته‌ای یا فرضیاتی همچون دانش قبلی فازهای کاربردی، بازنویسی کد منبع، رویکرد توقف، پیکربندی مجدد و ادامه کار معطوف نموده‌اند. با استفاده از روش‌های کمی‌سازی ارائه‌شده در گذشته، مقدار دقیق کشسانی پلتفرم رایانش ابری به آسانی، قابل محاسبه نیست. از طرفی دیگر، مدل حاصل از این پژوهش‌ها فاقد عبارات شکل بسته برای معیارهای عمده‌ای مانند کشش، کارایی و هزینه بوده و استفاده از آن برای تحلیل یک پلتفرم رایانش ابری سخت و دشوار است. به همین منظور، در این پژوهش برآنیم که از یک روش مدل‌سازی کمی جدید بر اساس شبکه پتری استفاده نماییم. بدین ترتیب که با استفاده از صف‌های M/M/N نظارت مؤثر بر نحوه ورود درخواست، ارائه سرویس و افزایش و کاهش ماشین مجازی به تناسب مدیریت منابع صورت می‌گیرد. در شبکه‌های پتری، مفهوم حالت به‌طور صریح نشان داده می‌شود و این باعث می‌شود تا در مدل‌سازی یک فرایند، بتوان مراحل مختلف فرایند را به‌وضوح نشان داد. در مدل‌های معرفی‌شده در روش‌های پیشین، تنها عملکردها به‌طور صریح توصیف می‌شوند و حالت‌های بین آن‌ها به‌صورت ضمنی درنظرگرفته می‌شوند. حالت به سطح پیاده‌سازی نزدیک است، درحالی‌که عملکرد به سطح کاربر نزدیک است. از طرفی، رایانش ابری یک سیستم توزیع شده است و وجود هم‌روندی در بین سرویس‌هایی که در این سیستم اجرا می‌شوند، امری رایج است. شبکه پتری با استفاده از نشانه‌ها، که می‌توانند در مسیرهای مختلف اجرا به‌طور موازی حرکت کنند، این مفهوم را بدون هیچ محدودیتی نشان می‌دهد. در بسیاری از ابزارهای مدل‌سازی، برای توصیف هم‌روندی محدودیت‌هایی وجود دارد. با گذشت مدت زمان زیادی که از ارائه شبکه پتری به عنوان یک ابزار مدل‌سازی می‌گذرد، در موارد بی‌شماری از آن استفاده‌شده و کارهای زیادی در زمینه بررسی ویژگی‌های آن و روابط موجود بین آن‌ها انجام شده است و دانش مشترک زیادی در این باره وجود دارد. شبکه پتری بر پایه قوی ریاضی استوار است و از این‌رو روش‌های مختلفی برای تحلیل مدل‌های مبتنی بر آن وجود دارد که می‌توان از این روش‌ها برای اثبات ویژگی‌های مختلف مانند دسترس‌پذیری و بن‌بست استفاده کرد. همچنین با استفاده از این روش‌ها می‌توان برای یک مدل مبتنی بر شبکه پتری، مدل معادل یافت که این امر می‌تواند باعث پی‌بردن به وجود فرایندهای کم‌هزینه‌تر برای انجام یک هدف شود.

سهم علمی این پژوهش به شرح زیر می‌باشد:

- ارائه رویکردی مبتنی بر شبکه‌های پتری رنگی برای مدیریت خاصیت کشسانی.
- تقویت خاصیت کشسانی با استفاده از مدل تأمین منبع بر پایه شبکه پتری رنگی.
- ارزیابی کارایی رویکرد پیشنهادی در زمینه دقت و سرعت تأمین منابع تحت بار کاری واقعی Google Cluster.

سازمان‌دهی ادامه مقاله به‌صورت زیر می‌باشد. در بخش دوم این مقاله، تحقیقات انجام‌شده در زمینه بهبود خاصیت کشسانی بررسی می شود و به پیش‌زمینه‌ای در مورد شبکه‌های پتری رنگی می‌پردازد. در بخش سوم، روش پیشنهادی برای بهبود خاصیت کشسانی با استفاده از شبکه پتری رنگی ارائه می‌گردد. در بخش چهارم، نتایج عملکرد سیستم پیشنهادی در قالب نتایج شبیه‌سازی مورد ارزیابی قرارگرفته و با روش‌های مرجع مقایسه می‌شود. درنهایت، در بخش آخر نتیجه‌گیری و پیشنهادها مطرح می‌گردد.

۲- ادبیات تحقیق

در این بخش، ابتدا، به تحقیقات انجام‌شده در زمینه بهبود خاصیت کشسانی پرداخته می‌شود و سپس، پیش‌زمینه‌ای در مورد شبکه‌های پتری رنگی ارائه می‌شود.

۲-۱- مروری بر رویکردهای مدیریت خاصیت کشسانی

در این بخش، به معرفی کارهای گذشته در زمینه مدیریت کشسانی از زاویه‌های اهداف و پارامترها، سیاست، روش‌ها و نوع مقیاس‌بندی می‌پردازیم. اهداف کشسانی می‌تواند شامل دلایلی مانند بهبود بهره‌وری، افزایش ظرفیت منابع، صرفه‌جویی در انرژی، کاهش هزینه، زمان، دقت، کیفیت سرویس و اطمینان از دسترس‌پذیری باشد. هنگامی‌که ما به اهداف کشسانی نگاه می‌کنیم، دیدگاه‌های مختلفی وجود دارد. ارائه‌دهندگان خدمات زیرساخت به عنوان سرویس سعی در به حداکثر رساندن سود خود با به حداقل رساندن منابع دارند البته با حفظ کیفیت خدمات، ارائه‌دهندگان سکو به عنوان سرویس به دنبال به حداقل رساندن هزینه‌هایی هستند که به فراهم‌کنندگان ابر و مشتریان می‌پردازند. هدف آن‌ها ارائه بهترین خدمات به مشتریان درحالی است که هزینه‌های تمامی عوامل پشته رایانش ابری را کاهش دهند. در نتیجه، برای کاربردهای مختلف، نیاز به ایجاد موازنه در برقراری اهداف مختلف است. 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 [۱۴، ۱۵] می‌باشد. [۱۶] منافع خدمات‌دهنده و کیفیت سرویس مشتری را به‌صورت همزمان درنظر می‌گیرد.

سیاست‌های خودکار کشسانی به دو دسته واکنشی و پیشگویانه تقسیم‌بندی می‌شوند. سیاست‌های واکنشی به این معنی است که سیستم، اقدامات کشسانی را بر اساس آستانه‌های خاص یا قوانین انجام می‌دهد. تعداد زیادی از کارهای تحقیقاتی [۴-۲، ۲۲-۱۷] و همچنین اکثر سیستم‌های ابری مانند آمازون EC2، Scalr و Rightscale از سیاست واکنشی استفاده می‌کنند. سیاست‌های واکنشی خود به دو دسته سطح آستانه ایستا و سطح آستانه پویا تقسیم‌بندی می‌شوند.

- سطح آستانه ایستا: سیاست‌های ایستا، اقدامات کشسانی شامل کاهش یا افزایش منابع را با برقراری شرایط از پیش تعریف‌شده اتخاذ می‌کنند. این سیاست به سطح آستانه‌های از پیش تعریف شده، ملزومات SLA و شرایطی که بر اساس اندازه‌گیری میزان استفاده از حافظه، میزان استفاده از پردازنده، زمان پاسخ و غیره شکل گرفته‌اند، بستگی دارد. برای مثال، می‌توان شرایط را این‌گونه تعریف کرد 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. سرویس EC2 آمازون و تحقیقات [۲، ۱۷] از این مکانیزم بهره می‌گیرند.
- سطح آستانه پویا: سطح‌های آستانه قبلی ثابت بودند، درحالی‌که در سیاست‌های سطح آستانه پویا، سطح‌های آستانه بر اساس شرایط کاربرد به‌صورت پویا تغییر می‌یابند. از این سیاست‌ها به‌عنوان سطح آستانه تطبیقی نیز یاد می‌شود. کارهای [۱۸، ۲۲] از تکنیک سطح آستانه تطبیقی استفاده می‌کنند.

سیاست‌های پیشگویانه، از تکنیک‌های پیش‌بینی برای نیازهای آینده استفاده می‌کند و واکنش‌ها را بر اساس این پیش‌بینی‌ها انجام می‌دهد. تاکنون تحقیقات مختلفی [۱۱-۵، ۱۶-۱۴، ۲۵-۲۳] در این زمینه انجام شده است که در ادامه به معرفی آن‌ها می‌پردازیم.

- تحلیل سری‌های زمانی: سری زمانی مسئول پیش‌بینی بار کاری و میزان به‌کارگیری منابع در آینده می‌باشد. بعد از این پیش‌بینی، کنترل‌کننده کشسانی، تصمیمات لازم جهت ایجاد این خاصیت را اجرا می‌کند. برای دستیابی به این هدف، تکنیک‌های میانگین حرکت [۲۶]، رگرسیون خودکار [۲۷]، آرما [۲۸]، هالت وینتر [۲۳]، یادگیری ماشین [۵، ۶] و الگو [۲۴] تاکنون استفاده شده‌اند.
- تکنیک‌های حل مدل: این رویکرد مبتنی بر بررسی مدل احتمالاتی یا چارچوب‌های مدل‌سازی ریاضی برای مطالعه رفتارهای مختلف سیستم و پیش‌بینی حالت‌های آینده می‌باشد. فرآیندهای تصمیم‌گیری بر اساس مدل مارکوف [۲۹] و اتوماتای زمان‌بندی احتمالی [۱۴]، نمونه‌هایی از این راه‌حل می‌باشند.
- یادگیری تقویتی: یادگیری تقویتی یک رویکرد محاسباتی مبتنی بر یادگیری از طریق تراکنش‌های بین عامل و سیستم یا محیط می‌باشد. به‌عنوان‌مثال سیستم معرفی شده در [۳۰] از سیاست یادگیری تقویتی استفاده کرده‌اند. لازم به ذکر است که می‌توان از یادگیری تقویتی در سیاست واکنشی نیز استفاده نمود [۴].
- نظریه کنترل: نظریه کنترل توابع سیستم را در حالت واکنشی کنترل می‌کند [۲۰، ۲۱]. اگرچه در برخی موارد در حالت پیش‌گویانه هم از آن استفاده شده است [۱۰، ۲۵].
- نظریه صف: نظریه صف‌بندی برای سیستم‌هایی با ماهیت ثابت در نظر گرفته می شوند. مطالعات گذ شته [۹، ۱۹] از نظریه صف در حالت پیش‌گویانه ا ستفاده کرده‌اند. لازم به ذکر ا ست که می‌توان از نظریه صف در سیاست واکنشی نیز استفاده نمود.

فرآیند تطبیق میزان منابع مورد نیاز به تقاضاهای یک برنامه کاربردی، مقیاس‌بندی نام دارد که می‌تواند بسیار چالش‌برانگیز باشد. مقیاس‌بندی به دو صورت افقی و عمودی انجام می‌شود. در مقیاس‌بندی افقی، ماشین‌های مجازی جدیدی به سیستم یا سکوی محاسباتی اضافه می‌شوند و متوازن‌کننده 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 سیستم‌عامل‌ها از تغییرات روی میزان پردازنده یا حافظه موجود، بدون راه‌اندازی مجدد برای اجرای مقیاس‌بندی عمودی، پشتیبانی نمی‌کنند. به همین دلیل اکثر ارائه‌دهندگان ابری، فقط از مقیاس‌بندی افقی پشتیبانی می‌کنند [۳۱]. در این پژوهش، از شبکه پتری رنگی برای مدیریت خاصیت کشسانی پلتفرم‌های ابری با ترکیب مقیاس‌بندی افقی و عمودی استفاده می‌شود. رایانش ابری، یک سیستم توزیع‌شده است و وجود هم‌روندی در بین سرویس‌هایی که در این سیستم اجرا می‌شوند،

امری رایج است. شبکه پتری با استفاده از نشانه‌ها، که می‌توانند در مسیرهای مختلف اجرا به طور موازی حرکت کنند، مفهوم هم‌روندی را بدون هیچ محدودیتی نشان می‌دهد. درحالی‌که در بسیاری از دیگر ابزارهای مدل‌سازی، برای توصیف هم‌روندی محدودیت‌هایی وجود دارد.

۲-۲- شبکه‌های پتری رنگی

نظریه شبکه‌های پتری توسط کارل آدام پتری در سال ۱۹۶۲ در رساله دکتری ایشان ارائه شد. او از شبکه‌های پتری برای نمایش ارتباطات علت و معلول استفاده نمود. شبکه‌های پتری یک ابزار توصیف بصری و مبتنی بر اصول ریاضی است که به‌صورت یک گراف متصل و جهت‌دار نشان داده می‌شود. معمولاً، شبکه‌های پتری را از دو دیدگاه ساختاری و رفتاری مورد بررسی قرار می‌دهند [۳۲]. در دیدگاه ساختاری، اجزا و ساختار یک شبکه پتری و مفاهیم مقدماتی مورد بحث قرار می‌گیرند. در دیدگاه رفتاری، نحوه نشانه‌گذاری و اجرای شبکه‌های پتری توسط نشانه‌ها و قواعد اجرا و همچنین ویژگی‌های رفتاری شبکه‌های پتری مورد بررسی قرار می‌گیرد. تعداد اولیه نشانه‌ها در شبکه با استفاده از تابع M_0 نشان داده می‌شود که نشان‌دهنده تعداد اولیه نشانه‌ها در هر مکان از شبکه است. زمانی‌که یک گذار فعال می‌شود، ممکن است نشانه‌گذاری شبکه تغییر کند. کمان ورودی، به کمانی گفته می‌شود که از مکان به یک گذار وارد می‌شود و نشان‌دهنده شرطی است که باید برآورده شود تا رویداد اتفاق بیافتد. کمان خروجی به کمانی گفته می‌شود که از گذار به مکان وارد می‌شود و نشان‌دهنده نتایج حاصل از وقوع رویداد است. در شبکه‌های پتری، شلیک شدن به‌صورت خودکار و پس از فعال شدن گذار انجام می‌شود. ممکن است که چند گذار فعال شوند، اما در هر زمان تنها یک شلیک انجام شود.

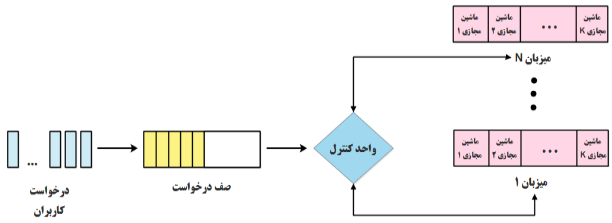
شبکه‌های پتری در طول زمان کامل‌تر شدند و مفاهیم کاربردی‌تر و جدیدتری به مفاهیم مورد استفاده آن افزوده شد. از جمله این مفاهیم، می‌توان به افزوده شدن زمان قطعی، زمان تصادفی و رنگ اشاره نمود. برخی از این مفاهیم منجر به تولید شبکه‌های خاصی شدند که از آن جمله می‌توان به شبکه‌های پتری رنگی اشاره نمود. شبکه‌های پتری رنگی توسط Kurt Jenson به عنوان یک مدل توسعه‌یافته از شبکه‌های پتری کلاسیک معرفی شده است [۳۳]. علاوه بر مکان‌ها، گذارها و نشانه‌ها در این شبکه، مفاهیم رنگ، محافظ و عبارت معرفی می‌شوند. مقادیر داده‌ای در این شبکه‌ها توسط نشانه‌ها حمل می‌شوند. شبکه‌های پتری رنگی ارائه‌دهنده مدل دقیق‌تری از سیستم‌های پردازشی غیرهمگام پیچیده هستند. در این شبکه‌ها، بر خلاف شبکه‌های پتری کلاسیک، مهره‌ها از یکدیگر قابل تمایز هستند. زیرا هریک از مهره‌ها دارای صفاتی به عنوان رنگ هستند. در یک شبکه پتری رنگی، محافظ، یک عبارت بولی است که به یک گذار منتسب می شود و شرایطی برای فعال شدن کمان ورودی ایجاد می‌نماید. هریک از مکان‌ها، کمان‌ها و گذارها می‌توانند بسته به رنگی که دارند، دارای محافظ مخصوص به خود باشند و وقتی حاصل محافظ "درست" باشد، عملیات انجام می‌شود [۳۴].

۳- روش پیشنهادی

همان‌طورکه قبلاً اشاره کردیم، خاصیت کشسانی، درجه‌ای هست که یک سیستم می‌تواند با تغییرات حجم کاری به‌وسیله آماده‌سازی و لغو مجوز برای امکانات به روشی خودمختار تطابق یابد. پس خاصیت کشسانی، توانایی سیستم برای مدیریت و تأمین منابع در هنگامی‌که سیستم با حجم کاری بالا روبرو است، می‌باشد. البته این را نیز باید اضافه کرد که نحوه مدیریت سیستم بعد از کاهش بار کاری نیز به کشسان بودن آن ارتباط دارد. در این بخش، روش پیشنهادی در مدیریت کشسانی با جزئیات تشریح می‌شود. در روش پیشنهادی از سیاست حل مدل که در آن از روش کمی‌سازی با استفاده از شبکه پتری رنگی است، استفاده می کنیم. در ادامه این بخش، به چارچوب روش پیشنهادی و مدل‌سازی روش پیشنهادی می‌پردازیم.

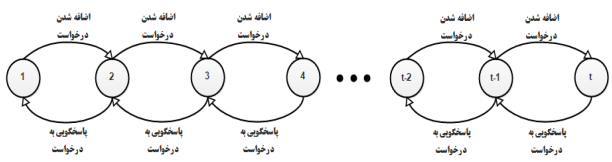
۳-۱- چارچوب پیشنهادی

شکل ۱ چارچوب روش پیشنهادی را نمایش می‌دهد. در این ساختار، درخواست کاربران برای درخواست سرویس در صف ابتدایی درخواست قرار می‌گیرد. سپس، درخواست‌ها در اختیار واحد کنترل قرارگرفته تا پس از کنترل وضعیت میزبان‌ها به‌تناسب سرویس مورد نیاز کاربر در اختیار میزبان مربوطه قرار گیرد. با توجه به این‌که هر ماشین مجازی توانایی پاسخگویی به چندین درخواست را دارد، ساختار هر ماشین مجازی به‌صورت یک صف M/M/1 در نظر گرفته شده است.



شکل ۱: چارچوب روش پیشنهادی

همان‌طورکه در شکل ۲ نشان داده شده است، هر ماشین مجازی توانایی اجرای t درخواست را دارد. پس، صف مربوط به یک ماشین مجازی یک صف M/M/1/t است. نکته مهم در روش پیشنهادی این است که به ازای اضافه شدن درخواست و عدم پاسخگویی، نیاز به اضافه شدن ماشین مجازی خواهد بود یا به تناسب به ازای ارائه سرویس و بیکار ماندن ماشین مجازی نیاز به حذف ماشین مجازی خواهد بود که در ادامه با جزئیات بیشتری توضیح داده خواهد شد.

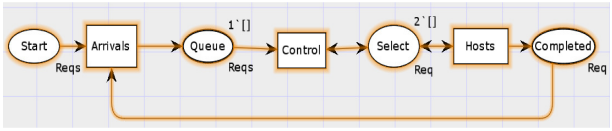


شکل ۲: ساختار هر ماشین مجازی

۳-۲- مدل پیشنهادی

در این بخش، مدل کشسانی خودکار پیشنهادی با نام Auto Elastic-CPN مورد بررسی قرار می‌گیرد. در این مدل سه عنصر مهم وجود دارد. صف اولیه درخواست، واحد کنترل و صف M/M/Z ماشین‌های مجازی. در ادامه، در مورد هرکدام از این عنصرها، به‌صورت دقیق توضیحات ارائه می‌شود. شکل ۳ ساختار شبکه پتری رنگی مدل پیشنهادی را نشان می دهد. با توجه به شکل، ۴ محل و ۳ گذر در شبکه مشخص شده است. ساختار مدل اولیه به‌صورت زیر خواهد بود.

N = {P, T, I, O}
P = {Start, Queue, Select, Completed}
T = {Arrivals, Control, Hosts}
I(Arrivals) = {Start, Completed}, O(Arrivals) = {Queue}
I(Control) = {Queue, Select}, O(Control) = {Select}
I(Hosts) = {Select}, O(Hosts) = {Completed, Select}



شکل ۳: مدل پتری رنگی پیشنهادی

متغیری که در شبکه پتری رنگی وجود دارد Req یا درخواست کاربر است. ساختار متغیر Req در شبکه پتری رنگی به‌صورت زیر تعریف شده است.

```
colset SLA_SLO = record Cost:REAL *
Deadline_Time:REAL;
colset Req_Res = record MIPS:INT * RAM:INT *
Storage:INT * BW:INT * ;
colset Request = record SLA :SLA_SLO * Resource:
Req_Res;
colset Requests = list Request;
var Req: Request;
var Reqs: Requests;
```

در این مدل، سه گذر اصلی وجود دارد که گذر اول دریافت درخواست‌ها و قرار دادن آن‌ها را به عهده دارد. گذر دوم کنترل‌کننده و ارسال‌کننده درخواست‌ها به میزبان‌های متناسب با ساختار درخواست است. گذر سوم تأمین منابع و کشسانی ابر را به عهده دارد.

۳-۲-۱- ساختار درخواست

بر اساس درخواست کاربر، ماشین‌های مجازی در اختیار کاربر قرار می گیرد و دسترسی به منابع وجود خواهد داشت. ساختار درخواست کاربر به‌صورت جدول ۱ است. هر درخواست شامل شرایط پذیرش سرویس در رویکرد پیشنهادی است، درخواست کاربر شامل دو بخش SLA(Service Level Agreement) و ویژگی‌های سخت‌افزاری مورد نیاز است. هر SLA شامل دو SLO(Service Level Objective) است که در جدول ۲ نشان داده شده است. به همین ترتیب ویژگی‌های سخت‌افزاری هم شامل ۴ ویژگی مطابق جدول ۳ می‌شود.

جدول۱: ساختار درخواست کاربر

User_ID	Request_SLA	Request_Resource

در جدول ۱ فیلد User_ID معرف شناسه کاربر است. فیلد Request_SLA معرف شرایط پذیرش سرویس توسط کاربر است که در قالب یک ماتریس به‌صورت جدول ۲ استفاده می‌شود. در این جدول، به ترتیب اهداف حداکثر هزینه و حد نهایت زمان پاسخگویی ثبت می‌گردد.

جدول ۲: قیود SLA درخواستی کاربر

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

فیلد Request_Resource مانند فیلد شرایط پذیرش سرویس، معرف ویژگی‌های منابع مورد نیاز کاربر است که در قالب یک ماتریس به‌صورت جدول ۳ استفاده می‌شود.

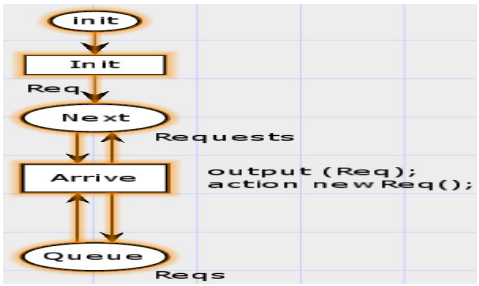
جدول ۳: قیود منابع درخواستی توسط کاربر

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

در جدول ۳، به ترتیب میزان کارایی پردازنده، حافظه RAM، حافظه ذخیره‌سازی و پهنای باند مورد نیاز کاربر برای هر درخواست، ثبت می‌گردد.

۳-۲-۲- صف اولیه درخواست

در صف اولیه ثبت درخواست، ابتدا درخواست‌ها توسط گذر Arrivals به صف تحویل داده می‌شوند. بدین صورت که هر درخواست با ویژگی‌های مورد نیاز کاربر وارد شده، سپس به‌عنوان متغیر جدید موجود در صف معرفی می‌گردد. شکل ۴ ساختار گذر Arrivals جهت ثبت درخواست‌ها در صف را نشان می‌دهد.



شکل ۴: ساختار پتری گذر Arrivals

در ساختار Arrivals سه محل و دو گذار وجود دارد. ساختار زیرشبکه Arrivals، که به‌عنوان یک گذار اصلی در شبکه شکل ۳ استفاده‌شده به‌صورت زیر هست.

N = {P, T, I, O}
P = {Init, Next, Queue}
T = {Init, Arrive}
I(Init) = {Init}, O(Init) = {Next}
I(Arrive) = {Next, Queue}, O(Arrive) = {Queue, Next}

توسط گذر Init درخواست از کاربر دریافت شده و به محل Next می رسد. محل Next به‌عنوان یک انباره اولیه عمل می‌کند و درصورتی‌که گذر Arrive درخواست قبلی را در صف درج کرده باشد توکن (درخواست) بعدی را از Next دریافت کرده و به صف منتقل می‌کند. تابع newreq در گذر Arrive برای ساخت درخواست با ساختار قابل قبول توسط صف است که ساختار آن در قسمت پایین آمده است.

```
fun newReq() = {
let
val c=Req.Cost
val d=Req. Deadline_Time
```

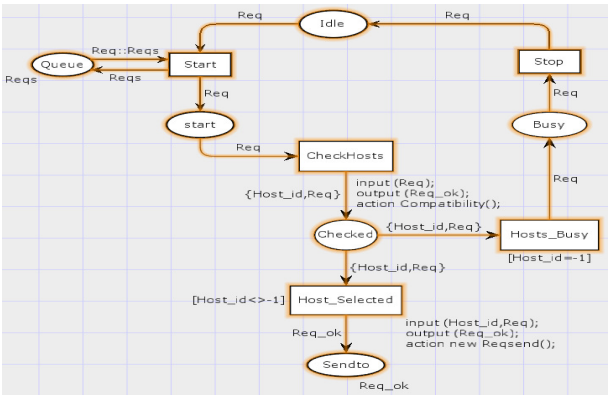
```
val m=Req.MIPS
val r=Req.RAM
val s=Req.Storage
val b=Req.BW
In
  { {c,d} , {m,r,s,b} }
}
```

این تابع ورودی گذر را که Req است را گرفته و تبدیل به متغیر درخواست قابل قبول به صف کرده و در اختیار آن قرار می‌دهد.

۳-۲-۳- واحد کنترل

این واحد، به تناسب درخواست کاربر تصمیم‌گیری می‌کند که درخواست در اختیار کدام نوع میزبان قرار گیرد. بدین معنی که شرایط سخت‌افزاری درخواست کاربر بررسی شده و به نسبت شرایط سخت‌افزاری منبع مورد درخواست کاربر میزبان انتخاب می‌گردد. شکل ۵ ساختار گذر کنترل، جهت کنترل درخواست‌های موجود در صف را نشان می‌دهد. در ساختار کنترل، پنج محل و سه گذار وجود دارد. ساختار زیرشبکه کنترل، که به‌عنوان یک گذار اصلی در شبکه شکل ۳ استفاده‌شده به‌صورت زیر می باشد.

```
N = {P, T, I, O}
P = {Queue, Start, Busy, Idle, Checked, Sendto}
T = {Start,CheckHosts, Host_Selected, Host_Busy, Stop}
I(Start) = {Queue, Idle}, O(Start) = {Queue, Start}
I(CheckHosts) = {start}, O(CheckHosts) = {Checked}
I(Hosts_Selected) = {Checked}, O(Host_Selected) = {Sendto}
I(Hosts_Busy) = {Checked}, O(Host_Selected) = {Busy}
I(Stop) = {Busy}, O(Stop) = {Idle}
```



شکل ۵: ساختار پتری گذر کنترل

در این ساختار، مجدداً متغیر Reqs به‌عنوان فهرستی از درخواست‌ها و متغیر Req به‌عنوان یکی از درخواست‌های لیست وجود دارد. ابتدا، درخواستی از صف درخواست‌ها خوانده‌شده و از گذر Start عبور کرده و به محل start می‌رسد. سپس درخواست به گذر CheckHosts ارسال‌شده و توسط تابع Compatibility میزبان متناسب با ساختار درخواست انتخاب می‌گردد. خروجی این گذر، شناسه میزبان و درخواست است. این خروجی به‌عنوان یک توکن به محل Checked منتقل می‌شود؛ که پس از شلیک این توکن به دو گذر Host_Selected و Hosts_Busy منتقل می‌شود. عبور از این دو گذر بر اساس شرط انجام می‌شود. شرط عبور از این دو گذر انتخاب یا عدم انتخاب Host است که بر مبنای رابطه (۱) صورت می‌گیرد.

$$State = \begin{cases} Busy & if \quad Hos\_id = -1 \\ Selected & if \quad Hos\_id <> -1 \end{cases} \qquad (۱)$$

بدین ترتیب، اگر یک میزبان مناسب جهت اختصاص به درخواست انتخاب شود توکن درخواست به محل Sendto منتقل می‌شود، در غیر این صورت محل انتخابی Busy خواهد بود. درصورتی‌که میزبان مناسب انتخاب شود، متغیری از جنس Req_ok توسط گذر Host_Selected تولید می‌شود و در اختیار محل Sendto برای ارسال به گذر Hosts در شکل ۳ قرار می‌گیرد. در صورت مشغول بودن میزبان‌ها، توکن درخواست با عبور از گذر Hosts_Busy به محل Busy منتقل می‌شود. سپس با عبور از گذر Stop به محل Idle می‌رسد. درنهایت درخواست با گذر از Start مجدد به صف درخواست‌ها بازمی‌گردد.

دو مورد مهم در این مدل متغیر Req_ok و دو تابع Compatibility و Reqsend است که در ادامه در مورد آن‌ها توضیحاتی ارائه می‌شود. ساختار متغیر Reg_ok به صورت زیر است.

```
colset Req_OK = record Host_id:INT * Req: Request;
var Req_ok: Req_OK;
```

تابع Reqsend به صورت زیر تعریف می‌شود.

```
fun Reqsend (Host_id,Req) = {
  { {Host_id,Req} }
}
```

تابع مهم در این بخش تابع یافتن میزبان مناسب برای پاسخگویی به درخواست با نام Compatibility است. این تابع بر اساس شرایط درخواست کاربر ویژگی‌های میزبان‌ها را بررسی می‌کند و میزبانی با شرایط مناسب کاربر معرفی می‌کند. تابع Compatibility به‌صورت زیر تعریف شده است.

```
fun Compatibility(req)={
let
  val count1 = ref Z
in
  while !counter > 0 do (
    let Host=Hosts.take(counter)
    if Host.length<K
     let VM_type= Host.take(1)
                if VM_type.MIPS>=Req.MIPS andalso
                   VM_type.RAM>=Req.RAM andalso
                   VM_type.Storage>=Req. Storage andalso
                   VM_type.BW>=Req.BW
                let sel= counter
     counter := !counter - 1
 )
};
```

۳-۲-۴- میزبان‌ها در ابر

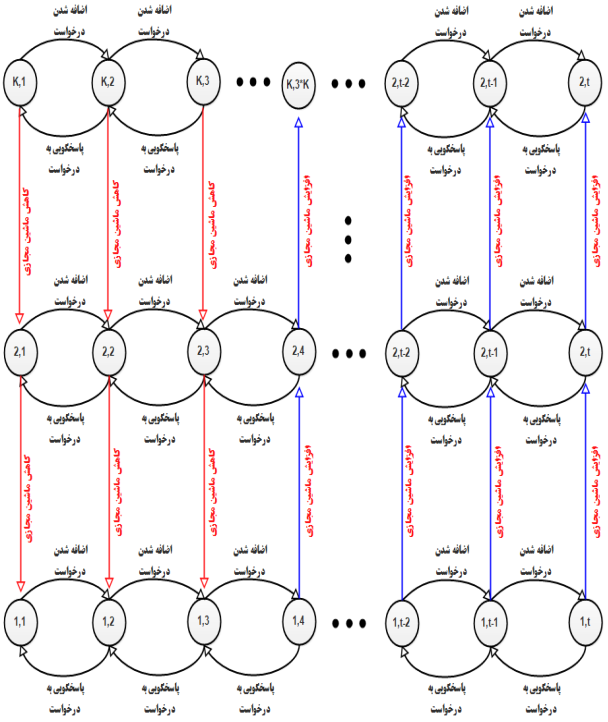
یکی از اصلی‌ترین قسمت‌های مدل پیشنهادی مدل کشسانی در ابر است. در این بخش تعدادی ماشین فیزیکی (Z) وجود دارد که در حالت آماده‌به‌کار هستند و به‌واسطه ورود درخواست ماشین مجازی در آن‌ها شروع به کار می‌کنند. در صورت افزایش تعداد درخواست‌ها ماشین مجازی جدید شروع به کار خواهد کرد. ساختار هر میزبان مطابق شکل ۶ خواهد بود. در این شکل به‌صورت مشخص هر سطر تعیین‌کننده یک ماشین مجازی است. هنگامی‌که تعداد درخواست ورودی در تعداد یک

ماشین مجازی بیشتر از سه درخواست باشد، به‌صورت خودکار یک ماشین مجازی جدید ایجاد می‌شود.

نکته مهم در مورد وضعیت ابر در هر میزبان این است که سه وضعیت تأمین بیش از حد منبع (*Over_P*)، تأمین کمتر از حد منبع (*Under_P*) و تأمین نرمال (*Normal*) از رابطه (۲) حاصل می‌شود.

$$State(i,j)=\begin{cases} Over\_P & if \quad 0\le i\le j \\ Normarl & if \quad i<j\le 3*i \\ Under\_P & if \quad j>3*i \end{cases} \qquad (۲)$$

ثابت‌های پارامتر i بسته به چگونگی مدیریت پلتفرم ابری می‌توانند مقادیر مختلفی داشته باشند. کاربران ابری مختلف و یا برنامه‌های کاربردی ممکن است حد آستانه‌های مختلفی را ترجیح دهند [۱۴]. تکرار فرآیند تأمین منابع می‌تواند با کاهش/ افزایش فاصله بین دو حد آستانه بالا و پایین، افزایش/کاهش یابد. در این مقاله حد آستانه بالا را (3i) و حد آستانه پایین را (*i*) برای تعیین وضعیت‌های اضافه تأمین، کسر تأمین و نرمال در نظر می‌گیریم [۳۵].

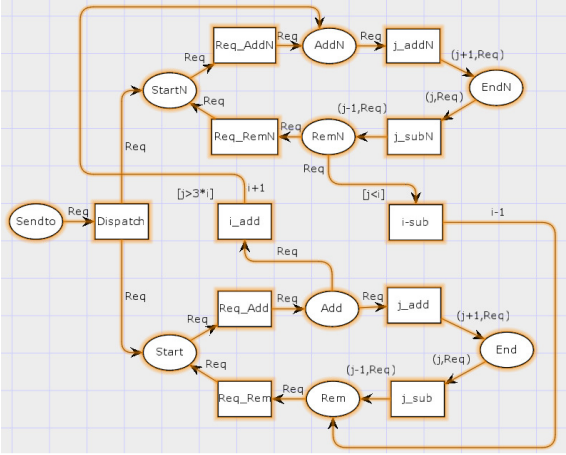


شکل ۶: ساختار هر میزبان در مدل پیشنهادی

شکل ۷ ساختار یک نمونه از گذر میزبان، جهت کنترل درخواست های موجود در صف را نشان می‌دهد. نکته مهم در این ساختار استفاده از ماتریسی از این گذر در ساختار شبکه اصلی است.

در ساختار میزبان، ۹ مکان و ۱۱ گذار وجود دارد. ساختار زیرشبکه میزبان که به عنوان یک گذار اصلی در شبکه استفاده شده به صورت زیر می‌باشد.

N = {P, T, I, O}
P = {Sendto, Start, Add, End, Rem, StartN, AddN, EndN, RemN}
T = {Dispatch, Req_add, j_add, j_sub, Req_Rem, i_add, i_sub, Req_addN, j_addN, j_subN, Req_RemN}
I(Dispatch) = {Sendto} O (Dispatch) = {Start, StartN}
I(Req_Add) = {Start} O (Req_Add) = {Add}
I(j_add) = {Add} O(j_add) = [4]
I(j_sub) = [4] O (j_sub) = {Rem}
O(Req_Rem) = {Start} I(Req_Rem) = {Rem}
O(i_add) = {addN} I (i_add) = {add}
O(i_sub) = {Rem} I(i_sub) = {RemN}
I(Req_AddN) = {startN} O (Req_Add) = {AddN}
I(j_addN) = {AddN} O(j_addN) = {EndN}
I(j_subN) = {EndN} O(j_sub) = {RemN}
O(Req_RemN) = {StartN} I(Req_RemN) = {RemN}



شکل ۷: مدل شبکه پتری زیرشبکه میزبان

۳-۳- مدل‌سازی فرمال مدل پیشنهادی

در این بخش، مدل‌سازی فرمال از روش پیشنهادی ارائه می‌شود. همان‌طورکه قبلاً اشاره شد، خاصیت کشسانی در یک پلتفرم ابری، انعکاس وضعیتی است که پلتفرم ابر، تحت نوسان حجم کاری تغییر می‌کند و می‌تواند به‌وسیله تصمیم‌گیری، تعداد ماشین‌های مجازی را متناسب با میزان درخواست‌ها در اختیار کاربران قرار دهد. به همین دلیل در ابتدا لازم است، مدل‌سازی فرمالی از درخواست‌های سرویس کاربر با استفاده از شبکه پتری ارائه شود.

۳-۳-۱- مدل‌سازی فرمال درخواست کاربر

یک درخواست سرویس کاربر یک شبکه پتری $N=\langle P,T,I,O\rangle$ است:

P: یک مجموعه‌ای از مکان‌هاست (مجموعه‌ای از درخواست‌های سرویس کاربران را نشان می‌دهد).

T: یک مجموعه‌ای از گذارها (مجموعه‌ای از فراخوانی‌های درخواست‌های سرویس کاربران را نشان می‌دهد).

$I: P\times T\rightarrow\{0,1\}, O: T\times P\rightarrow\{0,1\}$.

همچنین، برای یک مکان P و گذار T، در این مقاله، $^*p$ و p^* به عنوان ورودی و خروجی گذارها نشانه‌گذاری می‌شود، مجموعه‌ای از مکان p، $^*t$ و t^* به عنوان ورودی و خروجی مکان‌های مجموعه گذار t نشانه‌گذاری می‌شود.

۳-۳-۲- مدل‌سازی فرمال نشانه‌گذاری

در نظر بگیرید که N یک شبکه پتری باشد، این مقاله، یک سیستم شبکه با یک نشانه‌گذاری که به هر مکان یک مقداری از نشانه‌ها را اختصاص

می‌دهد، تعریف می‌کند. نشانه‌گذاری یک شبکه پتری، یک توزیع فراخوانی بین مجموعه‌ای از درخواست‌های سرویس کاربران را در یک شبکه پتری نشان می‌هد.

۳-۳-۳- مدل‌سازی فرمال شلیک

با توجه به یک سیستم شبکه $S=\langle N,M\rangle$ ما می‌گوییم که یک گذار t در نشانه‌گذاری M اشاره‌شده توسط رابطه (۳) شلیک‌پذیر است.

$$M[t\rangle iff\ \forall p\in {}^*t: M(p)\geq 1 \qquad (۳)$$

تعریف: شلیک یک گذار t در نشانه‌گذاری M، نشانه‌گذاری را به رابطه (۴) تغییر می‌دهد که با $M[t\rangle M'$ نشان داده می‌شود.

$$M' s.t. \forall p: M'(p)=M(p)+(O(t,p)-I(p,t)) \qquad (۴)$$

۳-۳-۴- مدل‌سازی تکثیر/تلفیق

در این مقاله دو راهبرد کشسانی تکثیر/تلفیق (Duplication/Consolidation) را در نظر می‌گیریم. قابلیت کشسانی، توانایی برای تکثیر یا تلفیق شدن نمونه‌های ماشین‌های مجازی برای اجرای درخواست سرویس کاربران می‌باشد. همان‌طورکه قبلاً اشاره شد، به ازای اضافه شدن درخواست و عدم پاسخگویی، نیاز به اضافه شدن ماشین مجازی خواهد بود (راهبرد تکثیر)، از طرفی دیگر، به ازای ارائه سرویس و بیکار ماندن ماشین مجازی، نیاز به حذف ماشین مجازی خواهد بود (راهبرد تلفیق). در ادامه، تعریف فرمالی از دو راهبرد کشسانی تکثیر/ تلفیق ارائه می‌شود.

فرض کنید $S=\langle N,M\rangle$ یک سیستم شبکه و $p\in P$، تکثیر p در S توسط یک مکان جدید $p^c(\notin P)$ به عنوان $D(S,p,p^c)$ ذکر شده است، $S'=\langle N',M'\rangle s.t$ یک سیستم شبکه جدید است که در آن روابط (۵) تا (۸) برقرار است.

$$P'=P\cup\{p^c\} \qquad (۵)$$

$$T'=T\cup T'' \text{ with} \qquad (۶)$$

$$T''=\{t^c \mid t\in({}^*p\cup p^*)\wedge t^c=\eta(t)\}$$

$\eta(t)$ یک کپی جدید از t که در T نیست ایجاد می‌کند.

$$I': P'\times T'\to\{0,1\}. \qquad (۷)$$
$$O': T'\times P'\to\{0,1\}.$$

$$M': P'\to N \text{ with } M'(p')=M(p') \text{ if } p'\neq p^c \text{ and } \qquad (۸)$$
$$0$$

توابع I′ (و همچنین O′) به شرح زیر در رابطه (۹) تعریف می‌شوند.

$$I'(p',t')=\begin{cases} I(p',t') & p'\in P\wedge t'\in T \\ I(p',t) & t\in T\wedge t'\in(T'\setminus T) \\ & \wedge p'\in(P\setminus\{p\}) \\ I(p,t) & t\in T\wedge t'\in(T'\setminus T) \\ & \wedge p'=p^c \\ 0 & otherwise \end{cases} \qquad (۹)$$

O′ را می‌توان با جایگزینی I توسط O بدست آورد.

فرض کنید $S=\langle N,M\rangle$ یک سیستم شبکه و p، p^c دو مکان در N با $p\neq p^c$ باشند. تلفیق p^c در p، به عنوان $C(S,p,p^c)$ ذکر شده است، $S'=\langle N',M'\rangle s.t$ یک سیستم شبکه جدید است که در آن N′ به صورت زیر تعریف می‌شود و نیز M′ به‌صورت رابطه (۱۰) می‌باشد.

N′: شبکه N بعد از حذف مکان p^c و گذار $(p^c)^*\cup^* p^c$ هست.

$$M': P'\to N \text{ with } M'(p)=M(p)+M(p^c) \qquad (۱۰)$$
$$\text{and } M'(p')=M(p') \text{ if } p'\neq p$$

۳-۳-۵- مدل‌سازی کنترل‌کننده کشسانی

برای مدیریت خاصیت کشسانی، باید به‌طور مداوم اطلاعات موجود در صف درخواست‌های کاربران را نظارت و تحلیل کرده و در نهایت، راهبردهای کشسانی تکثیر/تلفیق را برای پیکربندی جدید پلتفرم محاسباتی ابر در راستای تضمین کیفیت سرویس انجام شود. ساختار کنترل‌کننده کشسانی بر اساس عملیات تکثیر/تلفیق بوده و مشخص می کند چه موقع و چگونه گذارهای این عملیات قابل شلیک خواهد بود.

- گذار تکثیر قابل شلیک است، اگر ما بتوانیم متغیر Z را به یک سیستم شبکه ⟨N,M⟩ = S متصل کنیم که یک مکان s وجود داشته باشد و پیش‌فرض Ready_D(Z, s) برآورده شود. شلیک گذار تکثیر یک سیستم شبکه جدید را که منجر به تکثیر از s در S شده است، اضافه می‌کند.
- گذار تلفیق قابل شلیک است اگر ما بتوانیم متغیر Z را به یک سیستم شبکه ⟨N,M⟩ = S متصل کنیم که دو نمونه از همان خدمات وجود دارد، s و s′ و پیش‌فرض Ready_C(Z, s, s′) برآورده شود. شلیک گذار تلفیق یک سیستم شبکه منجر شده از تلفیق s′ در s را اضافه می‌کند.

۴- ارزیابی عملکرد و نتایج

در این بخش، نتایج حاصل از پیاده‌سازی روش Auto Elastic-CPN برای بهبود خاصیت کشسانی با استفاده از مدل شبکه پتری مورد بررسی قرار می‌گیرد. محیط مورد استفاده شبیه‌سازی NetBeans است. به‌منظور شبیه‌سازی دقیق و قابل توسعه ابر از ابزار CloudSim استفاده شده است. در ادامه به بررسی پارامترهای شبیه‌سازی، بار کاری مورد استفاده قرارگرفته در این پژوهش و نتیجه شبیه‌سازی پرداخته می‌شود.

۴-۱- پارامترهای شبیه‌سازی

ساختار تمامی مراکز داده مورد استفاده در شبیه‌سازی در جدول ۴ مشخص شده است.

جدول ۴: ساختار مراکز داده مورد استفاده در شبیه‌سازی

معماری	X64
سیستم‌عامل	Cloud Linux
مدیریت ماشین‌های مجازی	XEN

در شبیه‌سازی برای هر مرکز داده، ۲۷ میزبان فیزیکی تعریف‌شده که به‌صورت تصادفی یکی از هفت نوع ماشین زیر در آن‌ها جای می‌گیرد. ماشین‌های مجازی استفاده‌شده در شبیه‌سازی روش پیشنهادی، در سه گروه و از هر گروه، سه نمونه انتخاب شده است. جدول ۵، مشخصات ماشین‌های مجازی را در هر میزبان نشان می‌دهد.

جدول ۵: مشخصات ماشین‌های مجازی مراکز داده

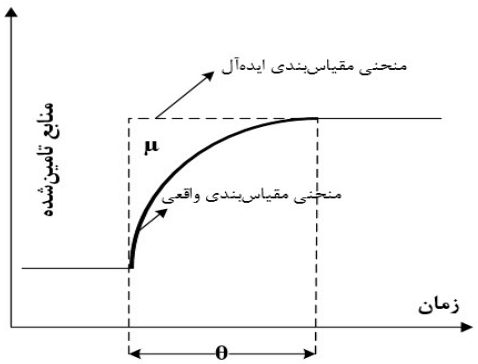
Machin Name	MIPS	RAM(GB)	Storage	BW	Price ($ per Hour)
t2.small	10200	2	1 GB - 16 TB	100Mbps	0.023
m3.Medium	12000	3.75	1×4 GB	1Gbps	0.070
m4.4Xlarge	15000	64	1 GB - 16 TB	1Gbps	0.862
r3.4Xlarge	80000	122	1×320 GB	10Gbps	1.330
m4.10Xlarge	97000	160	1 GB - 16 TB	10Gbps	2.155
d2.4Xlarge	105000	122	12×2000 GB(24 TB)	10Gbps	2.76
m4.16Xlarge	280000	256	1 GB - 16 TB	100Gbps	3.447
r4.16Xlarge	350000	488	1 GB - 16 TB	100Gbps	4.256
d2.8Xlarge	500000	244	24×2000 GB(48 TB)	100Gbps	5.52

در جدول ۵، از ستون‌های RAM، Storage، MIPS، BW و Price برای مقایسه ویژگی‌های ماشین‌های مجازی با قیود کاربر استفاده می‌شود.

۴-۲- پارامترهای ارزیابی

در این بخش پارامترهای ارزیابی مدیریت خاصیت کشسانی استفاده شده در این پژوهش معرفی می‌شود. طبق [۳۶]، پارامترهای مؤثر بر خاصیت کشسانی از دو دیدگاه قابل بحث و ارزیابی هستند: (۱) چگونگی تغییر سریع و یا به موقع وضعیت منابع موجود در یک ابر؟ (۲) با چه دقتی منابع تأمین شوند تا تغییرات بار کاری را پوشش دهند؟ خاصیت کشسانی این امکان را فراهم می‌کند که پیکربندی مجدد در یک زمان بسیار کوتاه توسط مجازی‌سازی ماشین انجام شود.

همان‌طور که در شکل ۸ نشان داده شده است، خاصیت کشسانی با دو پارامتر اندازه‌گیری می‌شود: سرعت و دقت. سرعت توسط زمان پیکربندی مجدد پلتفرم ابری (θ) محاسبه می‌شود، درحالی‌که دقت (μ) بر اساس میزان انحراف از تأمین نرمال (تأمین بیش از حد منبع یا تأمین کمتر از حد منبع) بدست می‌آید [۳۶].



شکل ۸: تصویری از تأمین منابع ابری است، که در آن θ زمان سربار و μ انحراف از تأمین نرمال است

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، همان‌طور که در رابطه (۱۱) نشان داده‌شده است.

$$T_m = T_n + T_o + T_u \quad (۱۱)$$

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 پس مطابق رابطه (۱۲) کشسانی برابر با نسبت T_n به T_m.

$$E = \frac{T_n}{T_m} = 1 - \left(\frac{T_o}{T_m} + \frac{T_u}{T_m} \right) \quad (۱۲)$$

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 (۱۳) حاصل می‌گردد.

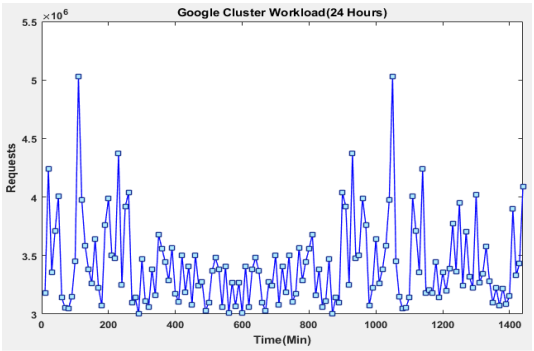
$$P_n = \frac{T_n}{T_m}, \qquad P_o = \frac{T_o}{T_m}, \qquad P_u = \frac{T_u}{T_m} \quad (۱۳)$$

بنابراین کشسانی از رابطه (۱۴) حاصل می‌گردد.

$$E = P_n = 1 - (P_o + P_u) \quad (۱۴)$$

۴-۳- بار کاری

برای ارزیابی رویکرد پیشنهادی، بار کاری واقعی Google Cluster استفاده شده است. شکل ۹ توزیع بارهای کاری واقعی در ۲۴ ساعت از ساعات پر درخواست را برای Google Cluster نشان می‌دهد. این بارهای کاری دارای ویژگی‌هایی هستند که باعث می‌شود نتایج گرفته‌شده در محیط ابری بسیار نزدیک به یک پیاده‌سازی واقعی باشند. بار کاری Google Cluster [۳۷] توسط گوگل منتشر شده است و آن حاوی اطلاعات مربوط به سلول، در حدود ۲۹ روز در ماه مه ۲۰۱۱، روی یک خوشه در حدود 12.5 k ماشین می‌باشد. در این پژوهش، بار کاری ClusterData2011_2 به عنوان بار کاری Google Cluster بکار گرفته شده است که در آن ۶،۲۵۳،۷۷۱،۴۲۹ درخواست 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.



شکل ۹: نمایی از ۲۴ ساعت از مجموعه داده Google Cluster

۴-۴- نتایج شبیه‌سازی

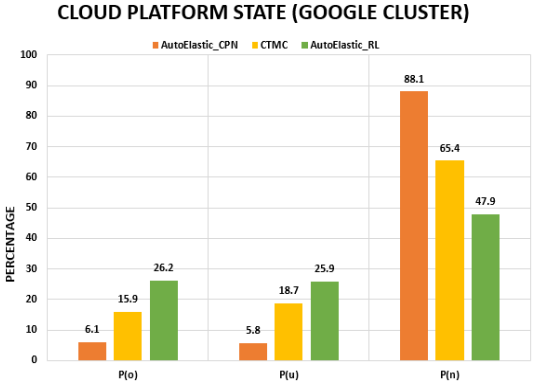
در این بخش نتایج شبیه‌سازی برای نوع بار کاری واقعی Google Cluster بررسی می‌گردد. نتایج روش پیشنهادی کشسانی خودکار با استفاده از

شبکه پتری رنگی (Auto Elastic-CPN) با دو روش کشسانی خودکار با استفاده از زنجیره مارکوف پیوسته در زمان (CTMC) [۲۹] و کشسانی خودکار با استفاده از یادگیری تقویتی Auto Elastic-RL [۳۰] مقایسه می‌شود. روش CTMC همانند روش پیشنهادی روی ساختار معماری مشابه عمل می‌کند با این تفاوت که این روش جهت مدیریت کشسانی از زنجیره مارکوف زمان پیوسته استفاده می‌کند. در این روش از یک مدل تحلیلی بر پایه زنجیره مارکوف مداوم برای برآورد تعداد ماشین‌های مجازی مورد نیاز برای تنظیم مقدار کشسانی منابع یک پلتفرم ابری استفاده می‌شود. روش Auto Elastic-RL نیز در قسمت تأمین منابع در ماژول مدیریت کشسانی از روش یادگیری تقویتی استفاده می‌کند. دلیل انتخاب این رویکردها برای مقایسه این است که اولاً این رویکردها پیشگویانه هستند یعنی سعی می‌کنند تعداد منابع در هر زمان داده‌شده را برای مقابله با نوسانات بار کاری در نظر بگیرند و دوم اینکه این روش‌ها مقیاس افقی (یعنی تکرار) برای اضافه کردن/ حذف نمونه‌های ماشین مجازی از یک پلتفرم ابری به منظور خاصیت کشسانی ارائه می‌دهند.

۴-۴-۱- مقایسه کشسانی

در این آزمایش خاصیت کشسانی در روش Auto Elastic-CPN با دو روش CTMC و Auto Elastic-RL مورد بررسی قرار می‌گیرد. یکی از شاخص‌های مهم در مقایسه عملکرد روش‌های تأمین منابع، خاصیت کشسانی است. برای انجام این آزمایش ده بازه زمانی مختلف از بار کاری Google Cluster انتخاب شد به صورتی که تعداد درخواست‌ها از بازه اول تا دهم سیر صعودی داشته باشد.

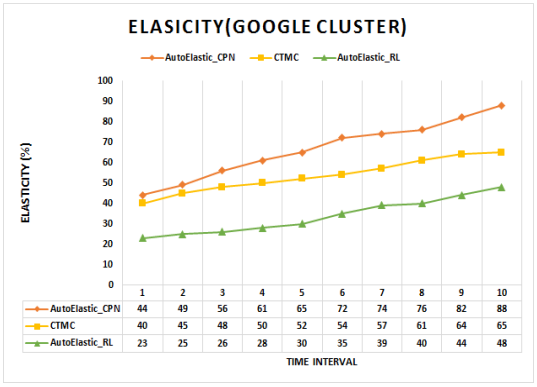
شکل ۱۰ میزان احتمال قرارگیری هر کدام از الگوریتم‌ها در حالت های مختلف نرمال، اضافه تأمین و کسر تأمین (P_n، P_o و P_u) را در بار کاری Google Cluster نشان می‌دهد. واضح است که به هر میزان پلتفرم ابری با درصد احتمال کمتری در دو وضعیت اضافه تأمین و کسر تأمین (P_o و P_u) قرار بگیرد، خاصیت کشسانی بهتری دارد.



شکل ۱۰: میزان درصد قرارگیری هر کدام از الگوریتم‌ها در حالت‌های مختلف در بار کاری Google Cluster

با توجه به نتایج حاصل شده، روش پیشنهادی نسبت به دو روش دیگر با میزان درصد احتمال کمتری در وضعیت اضافه تأمین و کسر تأمین (Po و Pu) قرار می‌گیرد و با درصد احتمال بیشتری هم در وضعیت تأمین نرمال (Pn) است، که نشان دهنده عملکرد مطلوب روش در ویژگی کشسانی است.

شکل ۱۱ آزمایش خاصیت کشسانی در روش Auto Elastic-CPN را در مقایسه با دو روش CTMC و Auto Elastic-RL را در بار کاری Google Cluster نمایش می‌دهد. با توجه به شکل ۱۱، به دلیل ایجاد دو مرحله صف و استفاده از ساختار کنترل‌کننده برای انتخاب میزبان مناسب و اضافه و کم کردن دقیق ماشین مجازی روش پیشنهادی کارایی بهتری در مورد تأمین منبع دارد.

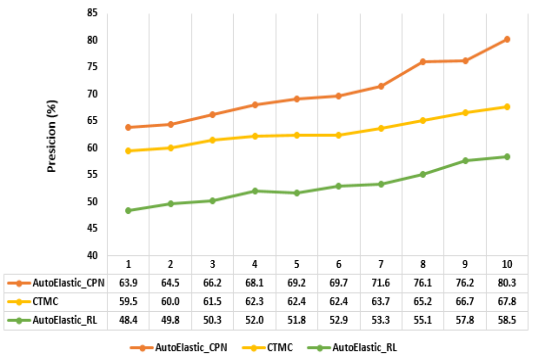


شکل ۱۱: مقایسه میزان کشسانی در بار کاری Google Cluster

خاصیت کشسانی در روش پیشنهادی نسبت به دو روش CTMC و Auto Elastic-RL تحت بار کاری Google Cluster به ترتیب ۱۶ درصد و ۸۳ درصد بهبودی داشته است.

۴-۴-۲- مقایسه سرعت و دقت مقیاس‌بندی

دقت مقیاس‌بندی به عنوان انحراف مطلق از مقدار فعلی منابع اختصاص یافته از تقاضای واقعی منابع تعریف می‌شود. شکل ۱۲ آزمایش دقت در روش Auto Elastic-CPN را در مقایسه با دو روش CTMC و Auto Elastic-RL را در بار کاری Google Cluster نمایش می‌دهد.

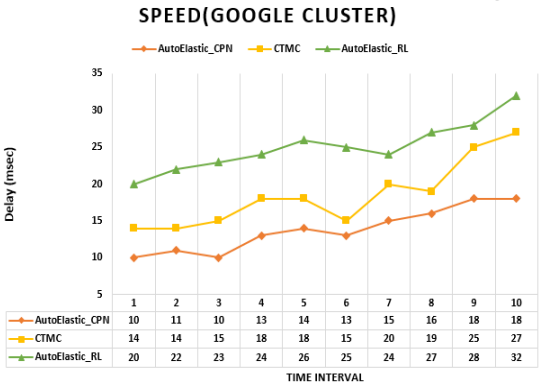


شکل ۱۲: مقایسه میزان دقت مقیاس‌بندی در بار کاری Google Cluster

دقت مقیاس‌بندی در روش پیشنهادی نسبت به دو روش CTMC و Auto Elastic-RL تحت بار کاری Google Cluster به طور میانگین به ترتیب ۱۲ درصد و ۳۳ درصد بهبودی داشته است.

سرعت بر اساس پیکربندی مجدد پلتفرم ابری سنجیده می‌شود. شکل ۱۳ این زمان تأخیر را برای روش Auto Elastic-CPN در مقایسه

با دو روش CTMC و Auto Elastic-RL در بار کاری Google Cluster نمایش می‌دهد.



شکل ۱۳: مقایسه میزان تأخیر مقیاس‌بندی در بار کاری Google Cluster

تأخیر فرآیند مقیاس‌بندی در روش پیشنهادی نسبت به دو روش CTMC و Auto Elastic-RL تحت بار کاری Google Cluster به طور میانگین به ترتیب ۲۶ درصد و ۴۴ درصد کاهش داشته است. در نتیجه، روش پیشنهادی به‌طور قابل ملاحظه‌ای موجب افزایش سرعت مقیاس‌بندی می‌شود.

۵- نتیجه‌گیری

در این مقاله، مدلی برای بهبود خاصیت کشسانی با استفاده از شبکه پتری رنگی برای تأمین منابع در شبکه‌های ابری ارائه شد. در مدل پیشنهادی مدیریت کشسانی با استفاده شبکه پتری رنگی و در قالب کنترل صف‌های M/M/N صورت گرفت. بدین ترتیب که به ازای ورود هر درخواست یا ارائه سرویس در صف حرکت افقی و به ازای نیاز به افزایش یا کاهش ماشین مجازی حرکت عمودی در صف وجود دارد. ساختار کنترل درخواست پیشنهادی، ارائه سرویس و کشسانی را به سادگی مدیریت می‌کند. در این ساختار درخواست کاربران برای سرویس در صف ابتدایی درخواست قرار می‌گیرد. سپس درخواست‌ها در اختیار واحد کنترل قرار گرفته تا پس از کنترل وضعیت میزبان‌ها به‌تناسب سرویس مورد نیاز کاربر در اختیار میزبان مربوطه قرار گیرد. با توجه به این‌که هر ماشین مجازی توانایی پاسخگویی به چندین درخواست را دارد، ساختار هر ماشین مجازی به‌صورت یک صف M/M/1 در نظر گرفته شده است. روش پیشنهادی بر روی بار کاری واقعی Google Cluster ارزیابی شد. نتایج ارزیابی عملکرد روش پیشنهادی افزایش خاصیت کشسانی را دربرداشت. به‌منظور تکمیل و توسعه روش پیشنهادی در این مقاله، برای بارهای کاری که دارای الگوی خاصی در ارائه درخواست هستند، می‌توان از پتری زمان‌دار استفاده کرد تا زمان‌بندی دقیق‌تری روی بارهای کاری انجام گیرد. همچنین می‌توان به‌جای مدل شبکه پتری رنگی از پتری فازی برای تصمیم‌گیری در مورد تأمین منبع استفاده شود که پیش‌بینی می‌شود، کنترل بالاتری بر روال انجام درخواست‌ها خواهد داشت.

مراجع

[۱] شهرام جمالی، سمیرا حورعلی، «موازنه‌گر نامتمرکز بار در محیط ابر با بهره‌گیری از سیاست تصمیم‌گیری چندشاخصه»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۶، شماره ۳، صفحات ۱۰۶-۹۵، ۱۳۹۵.

[2] 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.

[3] 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.

[4] 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.

[5] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan, "Prepare: Predictive performance anomaly prevention for virtualized cloud systems," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, 2012, pp. 285-294.

[6] L. R. Moore, K. Bean, and T. Ellahi, "Transforming reactive auto-scaling into proactive auto-scaling," in *Proceedings of the 3rd International Workshop on Cloud Data and Platforms*, 2013, pp. 7-12.

[7] G. A. Moreno J. Cámara, D. Garlan, and B. Schmerl, "Efficient decision-making under uncertainty for proactive self-adaptation," in *Autonomic Computing (ICAC), 2016 IEEE International Conference on*, 2016, pp. 147-156.

[8] G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl, "Proactive self-adaptation under uncertainty: a probabilistic model checking approach," in *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, 2015, pp. 1-12.

[9] E. B. Lakew, C. Klein, F. Hernandez-Rodriguez, and E .Elmroth, "Towards faster response time models for vertical elasticity," in *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, 2014, pp. 560-565.

[10] L. Baresi, S. Guinea, A. Leva, and G. Quattrocchi, "A discrete-time feedback controller for containerized cloud applications," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 217-228.

[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 *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, 2012, pp. 581-586.

[13] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *MGC@ Middleware*, 2010, p. 4.

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

[15] M. Beltrán, "BECloud: A new approach to analyse elasticity enablers of cloud services," *Future Generation Computer Systems,* vol. 64, pp. 39-49, 2016.

[16] 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, 2018, pp. 182-211.

[17] S. M.-K. Gueye, N. De Palma, É. Rutten, A. Tchana, and N. Berthier, "Coordinating self-sizing and self-repair managers for multi-tier systems," *Future Generation Computer Systems,* vol. 35, pp. 14-26, 2014.

[18] L. M. Vaquero, D. Morán, F. Galán, and J. M. Alcaraz-Calero, "Towards runtime reconfiguration of application control policies in the cloud," *Journal of Network and Systems Management,* vol. 20, pp. 489-512, 2012.

[19] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications," *Future Generation Computer Systems,* vol. 32, pp. 82-98, 2014.

[20] A. Al-Shishtawy and V. Vlassov, "Elastman: elasticity manager for elastic key-value stores in the cloud," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, 2013, p. 7.

[21] D. Serrano, S. Bouchenak, Y. Kouki, T. Ledoux, J. Lejeune, J. Sopena*, et al.*, "Towards qos-oriented sla guarantees for online cloud services," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, 2013, pp. 50-57.

[22] Y. Al-Dhuraibi, F. Zalila, N. Djarallah, and P .Merle, "Coordinating Vertical Elasticity of both Containers and Virtual Machines," in *CLOSER 2018-8th International Conference on Cloud Computing and Services Science*, 2018, pp. 1-8.

[23] A. da Silva Dias, L. H. Nakamura, J. C. Estrella, R. H. Santana, and M. J. Santana, "Providing IaaS resources automatically through prediction and monitoring approaches," in *Computers and Communication (ISCC), 2014 IEEE Symposium on*, 2014, pp. 1-7.

[24] E. Caron, F. Desprez, and A. Muresan, "Forecasting for Cloud computing on-demand resources based on pattern matching," INRIA, 2010.

[25] 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,* 2018.

[26] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," in *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, 2012, pp .2060-2056.

[27] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," *Future Generation Computer Systems,* vol. 27, pp. 871-879, 2011.

[28] C. Kan, "DoCloud: An elastic cloud platform for Web applications based on Docker," in *Advanced Communication Technology (ICACT), 2016 18th International Conference on*, 2016, pp. 478-483.

[29] K. Salah, K. Elbadawi, and R. Boutaba, "An analytical model for estimating cloud resources of elastic services," *Journal of Network and Systems Management,* vol. 24, pp. 285-308, 2016.

[30] C.-Z. Xu, J. Rao, and X. Bu, "URL: A unified reinforcement learning approach for autonomic cloud management," *Journal of Parallel and Distributed Computing* ،vol. 72, pp. 95-105, 2012.

[31] 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, 2018.

[32] W. M. Van der Aalst, "The application of Petri nets to workflow management," *Journal of circuits, systems, and computers,* vol. 8, pp. 21-66, 1998.

[33] K. Jensen and G. Rozenberg, *High-level Petri nets: theory and application*: Springer Science & Business Media, 2012.

[۳۴] سعید پاشازاده، «تحلیل خودکار بازی رایانه‌ای با استفاده از شبکه پتری رنگی»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۶، شماره ۲، صفحات ۴۸-۳۷، ۱۳۹۵.

[35] W. Ai, K. Li, S. Lan, F. Zhang, J. Mei, K. Li*, et al.*, "On elasticity measurement in cloud computing," *Scientific Programming,* vol .2016.

[36] N. R. Herbst, S. Kounev, and R. H. Reussner, "Elasticity in Cloud Computing: What It Is, and What It Is Not," in *ICAC*, 2013, pp. 23-27.

[37] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper,* pp. 1-14, 2011.