بهبود کشسانی سرویس‌های میزبانی شده روی زیرساخت ابری از طریق موتور تطبیق‌پذیری

علی کاظمی[۱]، فریدون شمس علیئی[۲]

[۱] دانشجوی دکتری، دانشکده علوم و مهندسی کامپیوتر، دانشگاه شهید بهشتی، تهران،
Ali.kazemi@sigma.ir

[۲] دانشیار، دانشکده علوم و مهندسی کامپیوتر، دانشگاه شهید بهشتی، تهران
F_shams@sbu.ac.ir

چکیده

رایانش ابری یکی از فناوری‌های اصلی لایه زیرساخت معماری برای ایجاد تحول دیجیتال (Digital Transformation) در سازمان‌ها و شرکت‌های سنتی و مجازی می‌باشد و همچنین در حال حاضر بستری برای بسیاری دیگر از تکنولوژی‌های نوظهور مانند اینترنت اشیاء، یادگیری ماشینی، داده‌های عظیم و غیره می‌باشد. باتوجه به نقش کلیدی فناوری رایانش ابری در تحول دیجیتالی سازمان‌ها و شرکت‌ها، در کنار نقش کلیدی آن در تامین زیرساخت سایر فناوری‌ها، تطبیق‌پذیری آن از منظر کارایی و سرعت برای پوشش انواع نیازهای مختلف پردازشی و ذخیره‌سازی سرویس‌های تطبیق‌پذیر امری بسیار مهم است. رایانش ابری عملاً بدون وجود قابلیت کشسانی به منظور تطبیق‌پذیری با نیازمندی‌های پویای کارایی نمی‌تواند نیاز سازمان‌ها را برآورده سازد. در این مقاله به طراحی و پیاده سازی یک موتور تطبیق‌پذیری خودمختار مطابق با چرخه MAPE برای اکوسیستم‌های ابری پرداخته می‌شود. موتور تطبیق‌پذیری ارائه شده، از طریق پایش سرویس و زیرساخت ابری، منابع سرویس‌ها را به صورت کشسان و مطابق با بار کنونی آنها تخصیص می‌دهد.

کلمات کلیدی

کشسانی، کارایی، تطبیق‌پذیری، چرخه MAPE، رایانش ابری

۱- مقدمه

مهیا کردن منابع (resource provisioning) یکی از مسائل مهم زیرساخت‌های رایانش ابری است. سرویس‌های ابری با مفاهیم جدیدی مانند کشسانی (elasticity)، پرداخت به میزان مصرف (pay-as-you-go) و مقیاس‌پذیری (scalability) همراه هستند. یکی از مهمترین وجه‌های تمایز بین سرویس‌های سنتی و سرویس‌های ابری ویژگی کشسانی است. کشسانی را می‌توان پیکربندی مجدد (reconfiguration) منابع تخصیص داده شده به یک سرویس ابری برای تطبیق‌پذیری بین بارکاری کنونی و منابع تخصیص داده شده کنونی دانست.

سند رسمی NIST [۱]، پنج ویژگی اساسی را برای سرویس‌های رایانش ابری بر می‌شمارد که عبارتند از: on-demand self-service، broad network access ، resource pooling، rapid elasticity و measured service [۱]. در این تعریف، ویژگی rapid elasticity به عنوان قابلیت‌هایی تعریف شده‌اند که می‌توانند به صورت کشسان مهیا و آزاد (release) شوند.

سرویس‌های کشسان را می توان نقطه مقابل سرویس‌هایی دانست که طراحی منابع آنها به صورت over-provisioned یا under-provisioned

انجام شده است. در طراحی منابع سرویس به صورت over-provisioned، منابع تخصیص داده شده به یک سرویس به میزان حداکثر منابع مورد نیاز تخمین زده شده برای آن سرویس در نظر گرفته می‌شوند بنابراین مهمترین ضعف این مدل هدر رفتن منابع است. برای مثال سرویسی را در نظر بگیرید که فقط در روزهای آخر هفته با ازدحام مواجه می‌شود و در این بازه خاص نیاز به پنجاه درصد منابع اضافی نسبت به سایر روزها دارد. بنابراین در طراحی over-provisioned به دلیل ازدحام آخر هفته، پنجاه درصد منابع اضافی به این سرویس تخصیص داده می‌شوند که در سایر روزهای هفته از این منابع اضافی هیچ استفاده‌ای نمی‌شود. به دلیل اینکه هزینه‌های سرمایه گذاری و نگهداری منابع هزینه‌های بالایی هستند این مدل از ارائه منابع نمی تواند مدل مناسب و مقرون به صرفه‌ای باشد. در طراحی منابع سرویس به صورت under-provisioning، منابعی کمتر از میزان بیشینه مورد نیاز یک سرویس به آن سرویس تخصیص داده می‌شوند. در این نوع طراحی، از هدررفت منابع جلوگیری می‌شود ولی آن سرویس ممکن خوشنامی خود را از دست بدهد زیرا به دلیل تخصیص منابع کمتر از میزان بیشینه موردنیاز، بسیاری از درخواست‌های کاربران سرویس در شرایط ازدحام با شکست مواجه خواهند شد و این موضوع می‌تواند باعث مهاجرت مشتریان این سرویس به سرویس‌های دیگر شود.

کشسانی منابع این امکان را بوجود می‌آورد که میزان منابع تخصیص داده شده به یک سرویس با توجه به نیازمندی‌های آن سرویس در طول زمان متغیر باشند. به عبارت دیگر هیچ نیازی نیست که منابع مورد نیاز یک سرویس در ابتدای کار تخمین زده شوند و سپس تخصیص داده شوند. بلکه سرویس با تخمینی از مصرف منابع اولیه شروع به کار خواهد کرد، اگر میزان بار روی سرویس افزایش پیدا کرد، منابع تخصیص داده شده نیز افزایش پیدا خواهند کرد و بالعکس.

برای ارائه سرویس‌های کشسان نیاز به مولفه‌ای در اکوسیستم ابری وجود دارد که تمامی اطلاعات مورد نیاز را در اختیار داشته باشد. به عبارت دیگر برای اتخاذ تصمیمات کشسانی نیاز به مجموعه‌ای از اطلاعات درباره وضعیت کنونی سرویس‌ها و وضعیت کنونی زیرساخت ابری است. ذکر این نکته ضروری است که فقط اتخاذ تصمیمات کشسانی به تنهایی کافی نمی‌باشد، بلکه باید بازخورد حاصل از تصمیمات اتخاذ شده برای اتخاذ تصمیمات بعدی در نظر گرفته شوند. تامین کشسانی در اکوسیستم ابری می‌تواند بوسیله یک موتور تطبیق‌پذیری (adaptation engine) که اطلاعات لازم را در اختیار دارد، انجام شود. نمونه‌ای از این موتورهای تطبیق‌پذیری در مراجع مختلف [۲، ۳، ۴، ۵، ۶] پیشنهاد داده شده‌اند.

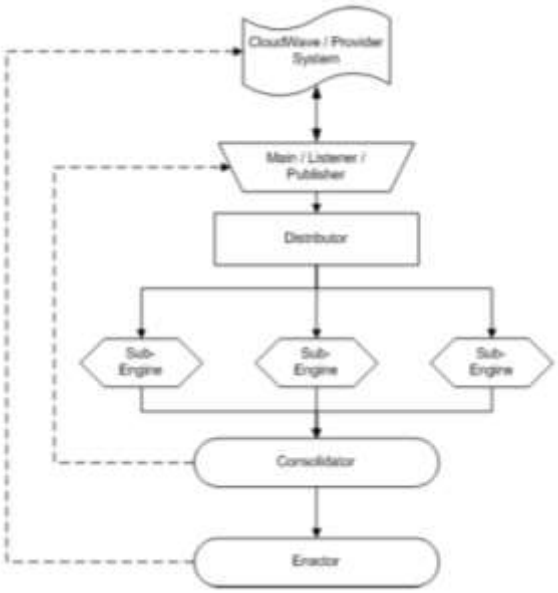
اغلب سیستم‌ها و معماری‌هایی که در حال حاضر برای ارائه سرویس‌های کشسان در مراجع مختلف وجود دارند وابسته به اطلاعات قبلی کاربر از رفتار سرویس (تخمینی از رفتار و میزان نیازمندی‌های منابع) هستند و معمولا فقط از دو نوع از اقدامات برای ارائه سرویس‌های کشسان شامل مقیاس‌پذیری افقی و عمودی پشتیبانی می‌کنند. همچنین برخی از این اقدامات ممکن است باعث ریبوت (reboot) ماشین مجازی و تحمیل downtime به سرویس شوند. با در نظر گرفتن این موارد در این مقاله به طراحی یک موتور خودمختار تطبیق‌پذیری پرداخته شده است که علاوه بر ویژگی خودمختاری از اقدامات مختلف برای مهیا کردن کشسانی پشتیبانی می‌کند و همچنین تضمین می‌کند که ماشین مجازی نیازی به ریبوت نخواهد داشت و سرویس با downtime مواجه نخواهد شد.

در ادامه این مقاله در بخش دوم به بررسی پژوهش‌های مرتبط پرداخته می‌شود. در بخش سوم به تعریف کشسانی پرداخته می‌شود. در بخش چهارم به معماری کلی موتور تطبیق‌پذیری پیشنهادی و مولفه تصمیم‌گیری آن پرداخته می‌شود. در بخش پنجم به دسته‌بندی سیستم‌های کشسانی و در بخش ششم به نتایج اولیه حاصل از تست موتور تطبیق‌پذیری پیشنهادی پرداخته می‌شود. بخش هفتم، بخش جمع‌بندی مقاله است.

۲- بررسی پژوهش‌های مرتبط با سیستم‌های کشسانی

کشسانی یکی از ویژگی‌های مهم رایانش ابری است که هم در صنعت و هم در پژوهش‌ها به آن پرداخته شده است. آمازون، کشسانی را از طریق امکان تعریف نرم‌افزاری زیرساخت (cloudformation) و گروه‌های مقیاس‌پذیری خودکار(auto-scaling groups) مهیا می‌کنند.

CloudWave [۴، ۵] یکی از پروژه‌های مطرح اتحادیه اروپا در حوزه رایانش ابری است. هدف اصلی این پروژه ایجاد تطبیق‌پذیری بین برنامه‌های کاربردی و سرویس‌های ابری با زیرساخت ابری با استفاده از یک موتور تطبیق‌پذیری است. سازوکار CloudWave مبتنی بر rule-set (مجموعه قواعد) است، به عبارت دیگر قواعد مختلف مطابق با شرایط مختلف تعریف شده‌اند و زمانی که هر یک از شرایط تعیین شده به وجود آیند اقدامات مناسب مطابق با آن شرایط انجام می‌شوند. پروسه تصمیم‌گیری CloudWave به صورت توزیع شده می‌باشد، به این صورت که مجموعه‌ای از sub-engine شرایط پیش آمده را بررسی می‌کنند و مطابق با قواعد موجود تصمیم‌گیری خود برای اقدامات مناسب را اعلام می‌کنند. پس از اینکه هر sub-engine تصمیم خود را اتخاذ کرد، از یک شیوه رای‌گیری برای اتخاذ تصمیم نهایی استفاده می‌شود. شکل ۱ پروسه تصمیم‌گیری موتور تطبیق‌پذیری CloudWave را نشان می‌دهد.



شکل (۱): ساختار موتور تطبیق‌پذیری CloudWave **[۵]**

پژوهش [۲، ۳] به معرفی معماری Cloudiator برای مدیریت پایش و تطبیق‌پذیری می‌پردازد. این معماری از مولفه‌های مختلفی تشکیل شده است. وظیفه auto-scaling در این معماری بر عهده مولفه scaling engine است. این مولفه طبق مقادیر آستانه از پیش تعریف شده روی میزان مصرف پردازنده و حافظه تصمیم‌گیری می‌کند. زمانی که میزان مصرف از مقادیر آستانه تعریف شده بیشتر شود، اقدامات لازم اجرا می‌شوند. Cloudiator فقط از مقیاس‌پذیری افقی پشتیبانی می‌کند.

پژوهش [۶] برای مدیریت کشسانی در محیط ابری یک مدل توصیف منابع ابری (cloud resource description model) را ارائه می‌دهد. هدف از ارائه این مدل این است که کاربران بتوانند منابع و ویژگی‌های کشسانی سرویس‌ها را به شیوه‌ای ساده و بدون رویارویی با جزییات فنی یا اسکریپت‌های سطح پایین مشخص کنند. این پژوهش نیز از سازوکار مقایسه با مقادیر آستانه از پیش تعین شده استفاده می‌کند. این پژوهش از مقیاس‌پذیری عمودی، مقیاس‌پذیری افقی، مهاجرت و پیکربندی مجدد برنامه کاربردی (application reconfiguration) پشتیبانی می‌کند.

پژوهش‌های فوق مبتنی بر rule-set هستند. به عبارت دیگر سیستم کشسانی روی مقادیر آستانه از پیش تعریف شده‌ای روی متریک‌های میزان مصرف منابع مانند میزان مصرف پردازنده و میزان مصرف حافظه عمل می‌کند. پژوهش‌های اشاره شده بیشتر از اقدامات مقیاس‌پذیری افقی و مهاجرت پشتیبانی می‌کنند. در هیچ یک از پژوهش‌ها تضمین نشده است که اقدامات کشسانی انجام شده منجر به ریبوت سرور مجازی و در نتیجه downtime سرویس نخواهد شد.

در این پژوهش ما یک موتور تطبیق‌پذیری برای تطبیق‌پذیری میزان منابع تخصیص داده شده به یک سرویس بر اساس بارکاری کنونی روی آن سرویس ارائه می‌دهیم. سیستم پیشنهادی در این پژوهش از جهت‌های اشاره شده در ادامه متمایز از سایر پژوهش‌هاست:

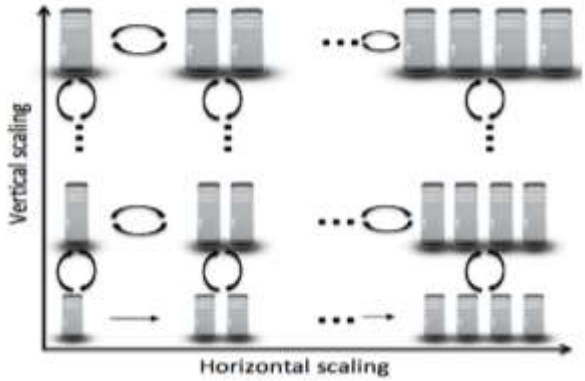
۱. سیستم پیشنهادی علاوه بر مقیاس‌پذیری افقی از اقدامات مقیاس‌پذیری عمودی، مهاجرت زنده و تغییر تنظیمات ابرناظر (hypervisor) پشتیبانی می‌کند.
۲. سیستم پیشنهادی تمامی اقدامات را به صورت زنده (live) اجرا می‌کند. به عبارت دیگر سرور مجازی در نتیجه انجام اقدامات کشسانی با ریبوت یا downtime مواجه نخواهد شد.
۳. سیستم پیشنهادی مبتنی بر rule-set است. ولی به جای مقایسه مقادیر کنونی متریک‌ها با مقادیر آستانه ثابت، با استفاده از تاریخچه مقادیر متریک‌های مختلف مانند زمان پاسخ یا میزان مصرف منابع مختلف، یک خط پایه برای هر متریک تولید می‌کند و مقادیر متریک‌ها را نسبت به مقادیر آن می‌سنجد و براساس میزان اختلاف بین مقادیر خط پایه و مقادیر کنونی متریک‌ها تصمیمات خود را اتخاذ می‌کند.
۴. سیستم ارائه شده در این پژوهش کاملا مبتنی بر چرخه MAPE [۷] و خودمختار است. چرخه MAPE در شکل ۲ نشان داده شده است. این چرخه دارای چهار فاز پایش (monitoring)، تحلیل (analysis)، برنامه‌ریزی (planning) و اجرا (execution) است.

شکل (۲): چرخه MAPE [۷]

موتور ارائه شده در این پژوهش علاوه بر در نظر گرفتن متریک‌های مربوط به میزان استفاده از منابعی مانند حافظه و پردازنده، متریک‌هایی مانند زمان پاسخ (زمان پاسخ سرویس اجرا شونده روی سرورهای مجازی ابری)، میزان کاربران همزمان و طول صف درخواست‌های سرورهای مجازی ابری را در نظر می‌گیرد. این ویژگی مهم باعث می‌شود که موتور تطبیق‌پذیری عکس‌العمل مناسبی در برابر انواع بارهای کاری از خود نشان دهد و کاربران نهایی تجربه خوبی از استفاده از این سرویس داشته باشند. در بخش چهار معماری موتور تطبیق‌پذیری پیشنهادی توضیح داده خواهد شد.

۳- تعریف کشسانی

در مراجع مختلف تعاریف متعددی از کشسانی آمده است. مرجع [۸] کشسانی را به عنوان توانایی یک سیستم در اضافه و حذف کردن منابع (مانند هسته های پردازنده، حافظه، ماشین مجازی و container) به صورت "on the fly" برای تطبیق‌پذیری تغییرات بار کاری به صورت بلادرنگ تعریف می‌کند. در اغلب مراجع مانند [۸] به دو نوع از اقدامات برای مهیا کردن کشسانی اشاره شده است که عبارتند از: vertical scaling و horizontal scaling. در vertical scaling منابعی مانند هسته‌های پردازنده، میزان سهم از پردازنده، حافظه و پهنای‌باند شبکه طبق نیازمندی‌های متغیر بارکاری افزایش یا کاهش داده می‌شوند. درhorizontal scaling ماشین‌های مجازی یا containers تخصیص داده شده به یک سرویس، با توجه به نیازمندی‌های متغیر بارکاری اضافه یا حذف می‌شوند. شکل ۳ از [۸] نمایی از vertical scaling و horizontal scaling را نشان می دهد.

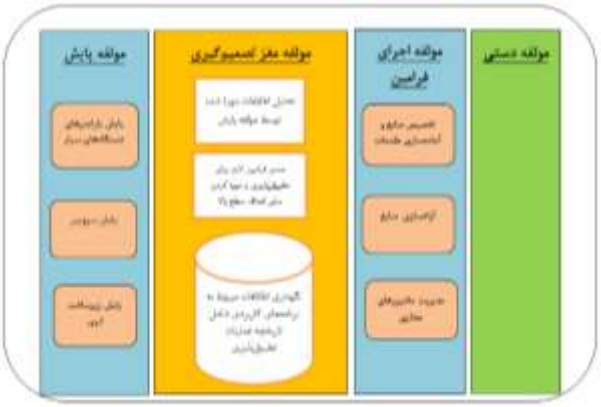


شکل ۳: مقیاس‌پذیری عمودی و افقی [۷]

در مرجع [۹] که کار قبلی ما درباره روش‌های تطبیق‌پذیری در محیط‌های ابری است، لیست جامعی از مکانیزم‌های تطبیق‌پذیری ارائه شده است.

٤- ساختار موتور تطبیق‌پذیری پیشنهادی

در این بخش به صورت خلاصه به بررسی ساختار موتور تطبیق‌پذیری پیشنهادی پرداخته می‌شود. شکل ۴ معماری موتور تطبیق‌پذیری را نشان می دهد.



شکل (٤): معماری موتور تطبیق‌پذیری پیشنهادی

همانطور که قبلا اشاره شد، موتور تطبیق‌پذیری مبتنی بر چرخه MAPE [۷] است که از چهار مولفه پایش (مطابق با فاز پایش چرخه MAPE)، مولفه تصمیم‌گیری (مطابق با فاز تحلیل و برنامه‌ریزی چرخه MAPE)، مولفه اجرای فرامین (مطابق با فاز اجرای چرخه MAPE) تشکیل شده است. مولفه دستی برای مدیر سیستم پیش‌بینی و تعبیه شده است.

شکل ۵ ورودی‌ها و خروجی‌های مولفه مغز تصمیم‌گیری را نشان می‌دهد. ورودی‌ها از مولفه پایش یا از سیاست‌های از پیش تعیین شده بدست می‌آیند و خروجی‌ها برای اجرا به مولفه اجرای فرامین ارسال می‌شوند. در ادامه هر یک از ورودی‌ها و خروجی‌ها توضیح داده خواهند شد که در این فرآیند جزییات هر سه مولفه پایش، مغز تصمیم‌گیری و اجرای فرامین پوشش داده می‌شوند.

شکل (٥): مولفه مغز تصمیم‌گیری

٤-١- ورودی‌های مولفه مغز تصمیم‌گیری

ورودی‌های موردنیاز شامل اطلاعات حاصل از پایش‌هایی هستند که در ادامه به آنها اشاره شده است:

۱. پایش سرویس: پایش سرویس شامل پایش متریک‌های مهم سرویس مانند زمان پاسخ، تعداد درخواست‌های ناموفق، طول صف درخواست‌ها و تعداد کانکشن‌های برقرار شده است.
۲. پایش ماشین مجازی: پایش ماشین مجازی شامل پایش میزان استفاده از منابع پردازنده، حافظه، دیسک (متریک‌هایی مانند تاخیر و نرخ خواندن و نوشتن)، پهنای‌باند شبکه، میزان ازدحام شبکه هستند.
۳. پایش سیستم عامل: پایش سیستم عامل شامل پایش سیستم عامل ماشین مجازی برای کسب اطلاعات لازم از آن است. در برخی موارد پایش سیستم عامل کاربران نیاز به کسب اجازه از آنها دارد.
۴. پایش زیرساخت: منظور از زیرساخت، زیرساخت ابری است که ماشینهای مجازی روی آن قرار دارند و از منابع آن زیرساخت استفاده می‌کنند. متریک‌های مختلفی مربوط به میزان استفاده شده از منابع و میزان منابع در دسترس روی زیرساخت ابری باید بررسی شوند. برای مثال زمانیکه در نتیجه یک horizontal scaling تصمیم به ایجاد یک ماشین مجازی جدید گرفته می شود، موتور تطبیق‌پذیری باید از منابع موجود روی زیرساخت ابری باخبر باشد تا بتواند نسبت به مکان ساخت ماشین مجازی جدید تصمیم مناسب اتخاذ کند.
۵. سیاست‌ها: یکی دیگر از ورودی‌های مهم مغز تصمیم‌گیری سیاست‌های موجود در اکوسیستم ابری هستند که باید موقع اتخاذ تصمیمات کشسانی در نظر گرفته شوند. برای مثال زمانی که نیاز به انجام مهاجرت زنده ماشین مجازی به دلیل افزایش ازدحام مسیر شبکه یا افزایش رقابت بین ماشین‌های مجازی روی یک میزبان فیزیکی (host) برای استفاده از منابع وجود دارد، باید طبق سیاست‌های موجود مقصد ماشین مجازی تعیین شود. طبق این سیاست‌ها ممکن است بتوان ماشین مجازی را به مقصدی با منابع قویتر (مثلا مقصدی که دارای هسته‌های پردازنده قویتر و البته گرانتر) است، مهاجرت داد. یکی دیگر از سیاست‌ها می‌تواند در راستای مصرف بهینه انرژی باشد به این صورت که ماشین‌های مجازی از روی میزبان فیزیکی که بیشتر منابع آن آزاد هستند به سایر میزبان‌های فیزیکی مهاجرت داده شوند و سپس آن میزبان فیزیکی خاموش شود.
۶. سازگاری اقدامات: برای تمامی تصمیماتی که اتخاذ می‌شوند باید سازگاری‌های لازم بررسی شوند. برای مثال برای vertical scaling، سیستم عامل ماشین مجازی باید از قابلیت hot-add منابع پشتیبانی کند.
۷. مانیتور خط پایه: برای تمامی متریک‌های مهم یک خط پایه تولید می‌شود. تولید خط پایه در بازه‌های زمانی (معمولا) یکسان و با استفاده از تاریخچه مقادیر آن متریک است. برای مثال با استفاده

از تاریخچه مقادیر متریک میزان استفاده از پردازنده، می‌توان خط پایه میزان مصرف پردازنده را تولید کرد.

۸. سابقه تصمیمات: تاریخچه‌ای از اقداماتی که انجام شده نگهداری می‌شود تا در حین اتخاذ تصمیمات جدید مورد توجه قرار گیرند.

۹. نوع برنامه کاربردی یا سرویس: با توجه به نوع سرویس می‌توان اقدامات مناسب را اتخاذ نمود. برای مثال اگر یک برنامه کاربردی cpu-intensive یا ram-intensive باشد باید اقدام مناسب که به ترتیب اضافه کردن پردازنده و اضافه کردن حافظه است انجام شود. حالت پیچیده می‌تواند زمانی رخ دهد که نوع یک برنامه کاربردی ترکیبی باشد. نوع برنامه را می توان با توجه تاریخچه الگوی مصرف منابع و مقادیر خط پایه تشخیص داد. با استفاده از روش‌های پیشرفته‌تر مانند تبدیل فوریه می‌توان الگویی از رفتار برنامه کربردی را بدست آورد.

۱۰. امکانات موجود و منابع آزاد: هنگام اتخاذ تصمیمات باید به منابع موجود توجه کرد. مثلا برای مهاجرت زنده ماشین مجازی بهتر است از مسیری که ازدحام کمتری دارد استفاده شود.

۱۱. تبعات و الزامات تصمیمات: قبل از اتخاذ تصمیمات باید بررسی روی تبعات و الزامات آن صورت گیرد. برای مثال باید بررسی کرد که آیا مهاجرت زنده ماشین مجازی ممکن است در شرایط موجود باعث downtime سرویس شود.

۱۲. بازخورد آنی: تصمیمات اتخاذ شده توسط مغز تصمیم‌گیری تاثیراتی روی اکوسیستم ابری دارند. با بررسی این تغییرات می‌توان تصمیمات نزدیک به بهینه اتخاذ کرد. به عنوان راه‌حلی پیشرفته برای بازخورد تصمیمات، می توان از روش‌های یادگیری مانند reinforcement learning استفاده کرد.

۲-٤- خروجی‌های مولفه مغز تصمیم‌گیری

خروجی‌های مولفه مغز تصمیم‌گیری به شرح زیر هستند:

۱. اقدامات پیش دستانه: برای برخی از اقدامات نهایی نیاز به اقدامات پیش دستانه‌ای وجود دارد. برای مثال برای horizontal scaling (افزودن یک ماشین مجازی جدید) باید بررسی کرد که اگر از قبل ماشین مجازی متعلق به این سرویس وجود دارد که suspend شده است، آن را resume کرد و از ساختن ماشین مجازی جدید اجتناب کرد زیرا ایجاد ماشین مجازی زمانبر است (تاخیر ایجاد می کند) و ممکن است شرایط پیش آمده قابلیت تحمل این میزان از تاخیر را نداشته باشند. می‌توان با نگهداری تاریخچه، از روش‌های پیش‌بینی مثل رگرسیون برای پیش‌بینی الگوی بارکاری آینده استفاده کرد.

۲. آماده سازی: برای برخی از اقدامات نهایی نیاز به آماده‌سازی‌هایی وجود دارد، برای مثال برای مهاجرت زنده باید اطلاعات آن از مبدا به مقصد کپی شوند.

۳. اقدام نهایی: اقدام نهایی اقدامی است که مغز تصمیم‌گیری با توجه به تمامی شرایط و ورودی‌ها آن اقدام را برای اجرا اتخاذ می‌کند. برای مثال این اقدام می‌تواند افزودن حافظه به یک ماشین مجازی باشد.

۵- دسته‌بندی سیستم‌های کشسانی

سیستم‌های کشسانی را می‌توان بر اساس سازوکار استفاده شده برای تصمیم‌گیری، دسته‌بندی کرد. مرجع [۱۰] سازوکارهای موجود تصمیم‌گیری را در ۵ دسته، تقسیم‌بندی کرده است که عبارتند از : ۱. مبتنی بر قواعد مانند مانند [۱۱] ۲. استفاده از سری‌های زمانی مانند [۱۲] ۳. استفاده از تئوری کنترل مانند [۱۳] ۴. استفاده از تئوری صف مانند [۱۴] ۵. استفاده از روش‌های یادگیری مانند [۱۵].

مرجع [۱۰] اتخاذ تصمیمات تطبیق‌پذیری سیستم‌های کشسانی را مطابق با چرخه MAPE بیان می‌کند. سیستم کشسانی مبتنی بر چرخه MAPE یک سیستم کشسانی خودمختار است که قادر به گرفتن و تحلیل ورودی‌ها و اتخاذ و اجرای تصمیمات است که می‌تواند با توجه به بازخوردهایی که از محیط می‌گیرد تصمیمات مناسبتری اتخاذ کند.

بیشتر سرویس‌دهندگان ابری مانند آمازون از روش مبتنی بر قواعد استفاده می‌کنند. این روش بسیار ساده است و پیاده‌سازی آن آسان است ولی ممکن است با اشتباهاتی در تصمیم گیری همراه باشد. عملکرد این روش بسیار بستگی به مجموعه قواعد (rule-set) دارد که استفاده می شود.

در یک دسته بندی دیگر از نظر زمان اتخاذ تصمیمات تطبیق‌پذیری، سیستم‌های کشسانی به دو دسته reactive و proactive دسته‌بندی می‌شوند. سیستم‌های کشسانی reactive سیستم‌هایی هستند که در ابتدا شرایطی را مشاهده می‌کنند و سپس نسبت به آن شرایط واکنش نشان می‌دهند به همین دلیل به این سیستم‌ها reactive گفته می‌شود. سیستم‌های کشسانی proactive سیستم‌هایی هستند که قبل از قرار گرفتن در شرایطی، آن شرایط را پیش‌بینی می‌کنند و تصمیمات لازم را اتخاذ می‌کنند. این نوع از سیستم‌های کشسانی حتما نیاز به نگهداری تاریخچه و استفاده از روش‌های پیش‌بینی مانند روش‌های مبتنی بر سری‌های زمانی دارند مانند [۱۲].

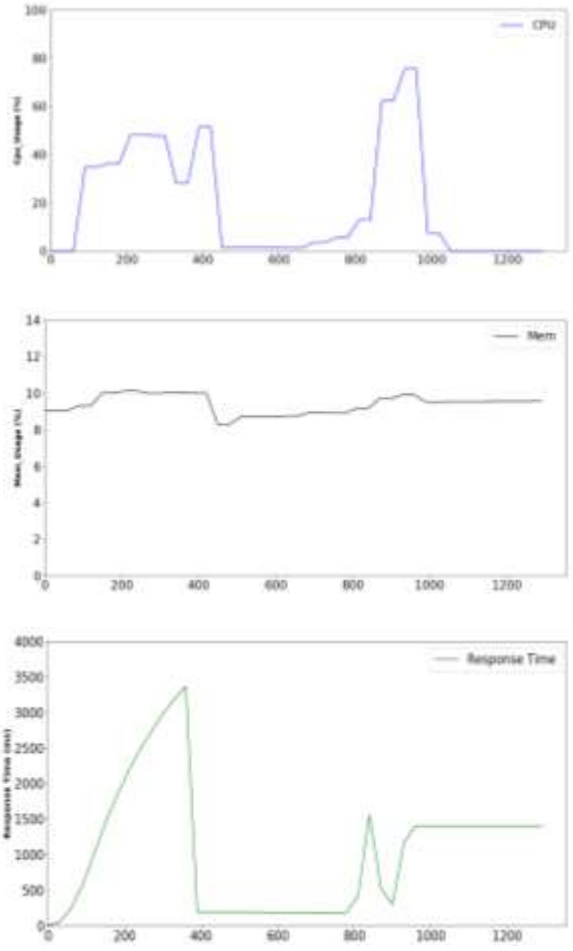
سیستم‌های کشسانی را می‌توان از نظر پیچیدگی روند تصمیم‌گیری نیز دسته‌بندی کرد. برخی از سیستم‌های کشسانی یک دسته ورودی، یک روش تصمیم‌گیری و یک دسته خروجی دارند مانند [۱۱]. برخی از سیستم‌های کشسانی مانند موتور پیشنهاد شده در پروژه cloudwave [۵] دارای ساختار پیچیده‌تری هستند. این موتور در بخش۲ توضیح داده می‌شود.

٦- پیاده‌سازی یک نمونه اولیه از موتور تطبیق‌پذیری پیشنهادی

یک نمونه از موتور تطبیق‌پذیری پیشنهادی روی اکوسیستم ابری OpenStack [۱۶] پیاده‌سازی شده است. برای پیاده‌سازی از ترکیبی از سرویس‌های OpenStack مانند compute (nova)، SDN (neutron) و Authentication (keystone) بسیاری از سایر ویژگی‌های OpenStack استفاده شده است.

در ادامه گراف مربوط به نتایج تست‌های اولیه موتور تطبیق‌پذیری پیشنهادی در شکل ۶ آورده شده است. برای این تست‌ها با استفاده از ابزار JMeter روی ماشین‌های مجازی استرس تست گذاشته شد. در این سه گراف مقادیر زمان پاسخ، مصرف حافظه و پردازنده در طول زمان نشان داده شده است. همانطور که در گراف زمان پاسخ نشان داده شده است، موقعی که زمان پاسخ در حال افزایش بیش از مقادیر مشخص شده است، موتور تطبیق‌پذیری

واکنشی مانند اضافه کردن ماشین مجازی جدید نشان می‌دهد و زمان پاسخ به حد نرمال برمی‌گردد. گراف اول مربوط به میزان استفاده از cpu گراف دوم مربوط به میزان استفاده از memory و گراف سوم مربوط به زمان پاسخ است.



شکل (٦): نتایج تغییرات زمان پاسخ ناشی از عکس‌العمل موتور تطبیق‌پذیری پیشنهادی

۷- نتیجه‌گیری و پژوهش‌های آتی

در این پژوهش یک موتور تطبیق‌پذیری مبتنی بر چرخه MAPE پیشنهاد داده شده است. این موتور تطبیق‌پذیری دارای سه مولفه اصلی پایش، مغز تصمیم‌گیری و اجرای فرامین است. هدف از پیشنهاد این موتور تطبیق‌پذیری بین منابع اختصاص یافته به یک سرویس و بار کنونی روی آن سرویس است. به این ترتیب نیاز به over-provisioned یا under-provisioned منابع نخواهیم داشت. یکی از ویژگی‌های اصلی این موتور پشتیبانی از اقدامات تطبیق‌پذیری مختلفی مانند مقیاس‌پذیری افقی و عمودی، مهاجرت زنده و تنظیمات مربوط به پارامترهای ابرناظر است.

گراف‌های نتایج ارائه شده در این مقاله، گراف‌هایی از نتایج موتور تطبیق‌پذیر هستند. در پژوهش‌های آتی موتور تطبیق‌پذیری با انواع مختلفی از بارهای کاری تست خواهد شد.

مراجع

[1] Liu F., Tong J., Mao J., Bohn R., Messina J., Badger L., and Leaf D., *"NIST Cloud Computing Reference Architecture"*, National Institute of Standards and Technology, Information Technology Laboratory, 2011.

[2] J. Domaschka, D. Seybold, F. Griesinger and D. Baur, *"Axe: A Novel Approach for Generic, Flexible, and Comprehensive Monitoring and Adaptation of Cross-Cloud Applications"*, Communications in Computer and Information Science, pp. 184-196, 2016.

[3] K. Kritikos, J. Domaschka and A. Rossini, "SRL: A Scalability Rule Language for Multi-cloud Environments", *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, 2014.

[4] D. Bruneo, T. Fritz, S. Keidar-Barner, P. Leitner, F. Longo, C. Marquezan, A. Metzger, K. Pohl, A. Puliafito, D. Raz, A. Roth, E. Salant, I. Segall, M. Villari, Y. Wolfsthal and C. Woods, *"CloudWave: Where adaptive cloud management meets DevOps"*, 2014 IEEE Symposium on Computers and Communications (ISCC), 2014.

[5] D. Doyele, G. Katsaros, L. Feehan, E. Garrido, A. SHarifloo, A. Metzgar, Z. mann, K. Barabash, D. Lorenz,*"Deliverable D4.1.3, Coordinated Adptation: Scientific Report"*, 2016

[6] H. Brabra, A. Mtibaa, W. Gaaloul and B. Benatallah, *"Model-Driven Elasticity for Cloud Resources"*, Advanced Information Systems Engineering, pp. 187-202, 2018.

[7] B. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Di Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns and J. Whittle, *"Software Engineering for Self-Adaptive Systems: A Research Roadmap"*, *Software Engineering for Self-Adaptive Systems*, pp. 1-26, 2009.

[8] 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, no. 2, pp. 430-447, 2018.

[۹] کاظمی, علی، ۱۳۹۶، مروری بر تکنیک های رایانش ابری تطبیق پذیر، *اولین همایش ملی پیشرفتهای معماری سازمانی*، تهران، دانشگاه شهید بهشتی، https://www.civilica.com/Paper-NCAEA01-NCAEA01_018.html

[10] T. Lorido-Botran, J. Miguel-Alonso and J. Lozano, *"A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments"*, Journal of Grid Computing, vol. 12, no. 4, pp. 559-592, 2014.

[11] R. Han, L. Guo, M. Ghanem and Y. Guo, *"Lightweight Resource Scaling for Cloud Applications"*, 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), 2012.

[12] S. Gupta and D. Dinesh, *"Online adaptation models for resource usage prediction in cloud network"*, 2017 Twenty-third National Conference on Communications (NCC), 2017.

[13] A. Ali-Eldin, J. Tordsson and E. Elmroth, *"An adaptive hybrid elasticity controller for cloud infrastructures"*, 2012 IEEE Network Operations and Management Symposium, 2012.

[14] L. Wang, J. Xu, M. Zhao, Y. Tu and J. Fortes, *"Fuzzy Modeling Based Resource Management for Virtualized Database Systems*", 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, 2011.

[15] H. Arabnejad, C. Pahl, P. Jamshidi and G. Estrada, *"A Comparison of Reinforcement Learning Techniques for Fuzzy Cloud Auto-Scaling"*, 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2017.

[16] "Open source software for creating private and public clouds.", *OpenStack*, 2018. [Online]. Available: https://www.openstack.org/.