The 4th National Conference on Applied Research in Electrical, Mechanical, Computer and IT Engineering
دانشگاه تخصصی فناوری های نوین آمل برگزار می کند.
چهارمین کنفرانس ملی تحقیقات کاربردی در
مهندسی برق، مکانیک، کامپیوتر و فناوری اطلاعات
2018
ISC
کد اختصاصی همایش: ۹۷۱۸۰-۴۱۰۰۵
۱۲ مهر ماه ۱۳۹۷ دانشگاه شیراز
October 4, 2018
www.emce.ir
ثبت شده در پایگاه استنادی علوم جهان اسلام (ISC)

خاصیت کشسانی ابری بر اساس مدل پیش بینی سربار کار سرور بصورت وفق پذیر

یوسف علیدوست [۱]، عبدالرضا رسولی کناری [۲]، محبوبه شمسی [۳]

[۱] دانشکده برق و کامپیوتر، دانشگاه صنعتی قم، قم، ایران / alidoost.y@qut.ac.ir
[۲] دانشکده برق و کامپیوتر، دانشگاه صنعتی قم، قم، ایران / rasouli@qut.ac.ir
[۳] دانشکده برق و کامپیوتر، دانشگاه صنعتی قم، قم، ایران / shamsi@qut.ac.ir

چکیده

یکی از مهمترین ویژگی های پردازش ابری، خاصیت کشسانی است و نقشی کلیدی در آن ایفا می‌کند. این امر باعث شده محققان زیادی در زمینه بهینه سازی خاصیت کشسانی ابری فعالیت کنند. خاصیت کشسانی، همان مقیاس پذیری همراه با تنظیم خودکار و بهینه سازی است. این کار اگر به بهترین حالت انجام شود، هم توافقنامه سطح خدمات کاملا رعایت می‌شود و هم تامین کننده ابری کمترین اتلاف منابع را خواهد داشت. در این پژوهش برای بهبود تجربه کاربر و همچنین عدم اتلاف منابع، از مدل پیش بینی سربار کار سرور استفاده شده است. با پیش بینی سربار کار کاربران، سیستم می‌تواند تصمیمات مقیاس پذیری را با توجه به تاخیر انجام تغییرات، اتخاذ کند بطوریکه کاربر این تغییرات را حس نکند. نتایج حاصل از پژوهش های گذشته نشان می‌دهد که استفاده از یک الگوریتم ثابت برای تمامی الگوهای سربار کار نمی‌تواند ما را به حداکثر سرعت و دقت در پیش بینی برساند. از این رو در این پژوهش معماری پیشنهاد شده است که در آن سه مدل LR، SVM و ARIMA برای پیش بینی سربار کار استفاده شده است و در این معماری با توجه به دو ویژگی میانگین مصرف cpu و الگوی فصلی سربار کار، مدل متناسب برای هر سربار کار انتخاب می‌شود نتایج نشان می‌دهد که استفاده از این معماری منجر به بهبود ۱۲ درصدی دقت پیش بینی و همچنین کاهش ۲۵ درصدی زمان میانگین پیش بینی سربار کار می‌شود.

واژگان کلیدی: محاسبات ابری، خاصیت کشسانی ابری، سری زمانی، پیش بینی سربار کار

مقدمه

در دهه اخیر پردازش ابری محبوبیت زیادی پیدا کرده و توجه محققین را در دنیای آکادمیک و صنعت، به خود جلب کرده است. فاکتور مهم که دلیل استفاده از پردازش ابری است، قابلیت تامین منابع بر اساس میزان نیاز کاربران است. که تحت عنوان قابلیت کشسانی پردازش ابری شناخته شده است. لذا قابلیت کشسانی یکی از کلیدی ترین ویژگی های پردازش ابری است که میزان منابع اختصاص داده شده به کاربران را، بر اساس سربار کار آنها مطابقت می‌دهد (Badger, Grance, Patt-Corner, & Voas, 2011).

تامین کنندگان سرویس های ابری معمولا رویکردهای مجازی سازی را برای ساختن سازوکار محصولاتشان استفاده می‌کنند. با استفاده از مجازی سازی می‌توان چندین سیستم عامل و برنامه های کاربردی را در یک سرور خاص بصورت همزمان اجرا کرد و با ساختن یک لایه مجرد پیچیدگی های سخت‌افزاری و نرم‌افزاری را از هم پنهان کرد. مجازی سازی اغلب با استفاده از Hypervisor ها پیاده سازی می‌شود. Hypervisor یکی از تکنیک های مجازی سازی است که با آن چند سیستم عامل می‌توانند یک سخت‌افزار را به اشتراک بگذارند. بطوری

که هر سیستم عامل می‌تواند منابع مستقل خود را داشته باشد. VMware, ESX, KVM, Xen, نمونه هایی از Hypervisor های معروف می‌باشند.

مجازی سازی پیمانه ای ، یا مجازی سازی سیستم عاملی، رویکرد دیگری از مجازی سازی است که در آن لایه مجازی سازی به عنوان یک برنامه کاربردی در سیستم عامل اجرا می‌شود. پیمانه ها یک راهکار سبک تر هستند که زمان شروع به کار سریع تر می‌کنند و سربار را کاهش می‌دهند (Pahl, 2015).

قابلیت کشسانی توسط پژوهشگران زیادی در صنایع و دانشگاه ها مورد تحقیق بررسی قرار دارد. تکنولوژی‌های مجازی سازی زیادی برای برآورده کردن خاصیت کشسانی، در حال توسعه و تکامل هستند. این موضوع در دو حوزه ماشین های مجازی و پیمانه ها مورد بحث قرار دارد.

مبانی نظری و تعاریف

تعاریف زیادی برای خاصیت کشسانی در ادبیات تحقیق وجود دارد. توانایی افزودن یا حذف منابع (از قبیل هسته های CPU، حافظه، ماشین های مجازی و پیمانه ها) در حال اجرا توسط سیستم برای مطابقت با تغییرات سربار کار را خاصیت کشسانی می‌گویند (Al-Dhuraibi, Paraiso, Djarallah, & Merle, 2017). واژه های دیگری از قبیل مقیاس پذیری و بهره-وری وجود دارد که با خاصیت کشسانی مرتبط هستند ولی معانی مختلفی دارند.مقیاس پذیری به معنی توانایی تحمل افزایش سربار کار با استفاده از افزودن منابع است (Herbst, Kounev, & Reussner, 2013) و یک معیار مستقل از زمان است. برای واضح تر کردن موضوع معادله زیر ارایه شده است.

خاصیت کشسانی = مقیاس پذیری + تنظیم خودکار + بهینه سازی

به این معنی که خاصیت کشسانی یک لایه بالاتر از مقیاس پذیری است. می‌توان اینگونه فرض کرد که خاصیت کشسانی اتوماتیک شده مفهوم مقیاس پذیری است طوری که منابع را به صورت بهینه به کاربران اختصاص دهد.

واژه دیگر مرتبط با این موضوع بهره وری است. بهره وری تعیین می‌کند که هنگام مقیاس پذیری منابع ابری چگونه باید استفاده شوند. هر چه میزان کمتری از منابع برای انجام کار معینی، اشغال شود؛ بهینگی بیشتر شده و هزینه ها و مصرف انرژی کاهش می‌یابد (Lehrig, Eikerling, & Becker, 2015).

روش مشخص و متداولی برای اندازه گیری معیار های کمی برای خاصیت کشسانی وجود ندارد. می‌توان میزان تاخیر تدارک و تخلیه منابع را به عنوان معیار کمی خاصیت کشسانی در نظر گرفت (Islam, Lee, Fekete, & Liu, 2012).

برای پیاده سازی راهکارهای کشسانی یک یا ترکیبی از دو روش زیر استفاده می‌شود:

مقیاس پذیری افقی، مقیاس پذیری عمودی.

در مقیاس پذیری افقی تعداد نمونه های ماشین های مجازی و پیمانه ها کم یا زیاد می‌شوند در حالی که در مقیاس پذیری عمودی مقادیر منابع در ماشین های مجازی کم یا زیاد می‌شود. مقیاس پذیری افقی معایب و مزایای خاص خودش را دارند. مقیاس پذیری افقی پیاده سازی ساده ای دارد و توسط hypervisor ها پشتیبانی می‌شود در حالی که این روش می‌تواند موجب تخصیص غیر بهینه منابع به شود به این دلیل که مقدار منابع هر نمونه ثابت و ایستا است و در بعضی مواقع امکان دارد که منابع دقیقن به اندازه نیاز اختصاص داده نشوند.

پیشینه تحقیق

بهینه سازی و دقیق تر کردن پیش بینی سربار کار سرورهای ابری بر اساس بررسی تاریخچه سربار سرور توجه اغلب محققین در زمینه پردازش ابری را به خود جلب کرده است.

دو رویکرد کلی در پیش بینی سربار کار وجود دارد: تحلیل سری زمانی، یاد گیری ماشین. سری زمانی یکی از معروف ترین روش ها برای پیش بینی سربار کار سرور های ابری است. Khan (Khan, Yan, Tao, & Anerousis, 2012) روشی ارایه کرد که در آن ابتدا شروع به جستجوی الگوهای قابل تکرار در تاریخچه سربار کار ها کرد سپس بر اساس مدل hidden markov دیدگاهی برای پیش بینی سربار کار سرور ارایه کرد. Hoffmann (Hoffmann, Trivedi, & Malek, 2007) روشی تجربی برای براساس الگوریتم سری زمانی Auto Regression برای پیش بینی سربار سرور روی وب سرور apache ارایه کرد. Morias (Morais et al., 2013) یک چهارچوب برای پیاده سازی سیستم مقیاس پذیری اتوماتیک بر اساس دیدگاه های انفعالی و فعال ارایه کرد و همچنین چند مدل برای پیش بینی سربار کار از قبیل Auto Correlation (AC)، linear regression (LR)، Auto Regression(AR) و Auto Regression Integrated Moving Average (ARIMA) پیشنهاد کرد.

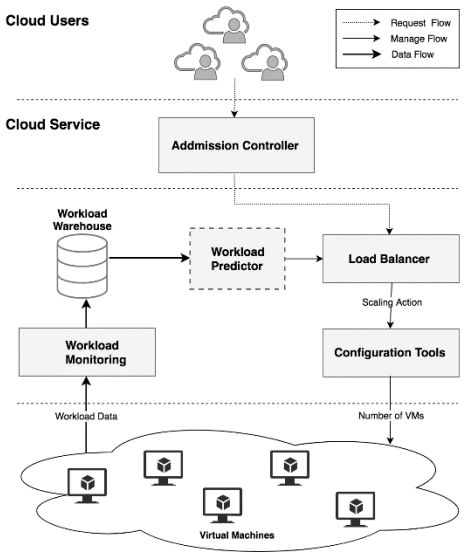
یادگیری ماشین نیز روش دیگر معروف در این زمینه است. Islam (Islam et al., 2012) یک استراتژی برای مدیریت منابع بر اساس شبکه های عصبی ارایه کرد که در آن میتوانست سربار کار سرور را پیش بینی کند. Bankole (Bankole & Ajila, 2013) سیستمی برای پیش بینی سربار کار کاربران توسعه کرد که در آن از سه روش support vector regression ، شبکه عصبی و رگرسیون خطی استفاده شده بود. Yashou (Hu, Deng, Peng, & Wang, 2016) سه روش برای پیش بینی سربار کار سرور ارایه کرد. ابتدا از روش های سری زمانی برای بررسی مانیتورینگ سربار کار استفاده کرد، سپس مدلی بر اساس Kalman Filter برای پیش بینی سربار کار ارایه کرد. و نهایتا یک روش جدید تطابق الگو (pattern matching) برای آنالیز و پیش بینی سربار کار پیشنهاد داد و در نهایت براساس پیش بینی های بدست آمده استراتژی تغییر جدیدی ارایه کرد. Nikravesh (Nikravesh, Ajila, & Lung, 2015) با فرض اینکه دقت پیش بینی سربار کار در سیستم های مقیاس پذیری اتوماتیک را میتوان با انتخاب الگوریتم سری زمانی مناسب بر اساس الگو های اجرایی طی زمان افزایش داد. برای اثبات این فرضیه آزمایشی ترتیب داد که در آن الگوریتم های مختلف سری زمانی با الگو های مختلف اجرایی را با هم مقایسه کرد. در این آزمایش دو تکنیک Support Vector Machine (SVM) و شبکه های عصبی مورد بررسی قرار گرفتند که نتایج حاصل شده نشان داد که روش SVM در الگوهای منتاوب (periodic) و رو به رشد (growing) دقت بالا تری نشان می‌دهد. روش شبکه های عصبی نیز در الگو های غیر قابل پیش بینی بهتر از SVM کار می‌کند. در نهایت یک معماری با روش خودمختار برای تشخیص و انتخاب خودکار الگوریتم مناسب بر اساس الگوهای اجرایی ارایه کرد.

C.Liu (Liu et al., 2017) روشی وفق پذیر بر اساس طبقه بندی سربار کار ارایه کرد که در آن از دو روش LR و SVM جهت پیش بینی سربار کار استفاده کرده. در این پژوهش سربار کار به دو کلاس fast time-scale و slow time-scale طبقه بندی شده اند که سربارها بعد از ورود به سیستم طبقه بندی می‌شوند و سپس با استفاده از روش های حل job assignment یکی از دو مدل LR و SVM برای پیش بینی سربار کار استفاده می‌شود. نتایج این پژوهش نشان میدهد که LR بیشتر برای سربار های slow time-scale بهتر جواب می‌دهد و روش SVM برای سربار های fast time-scale بهتر نتیجه می‌دهد.

مولفه های اصلی معماری مربوط به پیش بینی سربار کار در شکل ۱-۰ آمده است. این معماری شامل بخش های Admission Controller, Load Balancer, Configuration Tools, Workload Predictor, Workload Monitoring می‌باشد زیرساخت ابری شامل منابعی مانند Memory, CPU, IO, Network می‌باشد. Admission controller دسترسی کاربران را به سرویس ابری تعیین می‌کند. Load balancer میزان تخصیص منابع را براساس تغییرات نیازمندیهای کاربران به صورت پویا تغییر میدهد و تعداد VM های مورد نیاز را بر اساس داده‌های دریافتی مولفه Admission controller تعیین می‌کند. این مولفه با مقایسه تعداد VM های موجود و VM های مورد نیاز، دستور افزایش یا کاهش VM ها را به مولفه Configuration tools طول می‌دهد. مولفه Configuration tools نیز عملیات مربوط به مقیاس پذیری افقی را انجام میدهد. میزان استفاده منابع توسط مولفه Monitor مورد دیدبانی قرار می‌گیرد و لاگ های مصرف منابع به پایگاه داده سربار کار منتقل می‌شود. اطلاعات تاریخچه سربار کار برای تعیین نوع سربار کار و همچنین برای پیش بینی منابع مورد نیاز مورد استفاده قرار می‌گیرد.

مولفه Workload predictor مولفه کلیدی این معماری است که در آن ابتدا سربار کار طبقه بندی می‌شود و بر اساس الگوی سربار کار و مولفه های آماری آن، مدل مربوط به پیش بینی سری زمانی برای آنها انتخاب می‌شود.

هنگامی که Job جدید برای تخصیص منابع آماده می‌شود درخواست آن به Load balancer می‌رود. مولفه Load balancer نیز برای پیش‌بینی سربار کار آن Job اطلاعات سربار کار مورد نظر را به Workload classifier می‌دهد. در این زمان اگر داده های مربوط به سربار کار ان Job کم باشد یعنی عمر آن کمتر از یک ساعت باشد مولفه Workload classifier مدل خطی LR را برای پیش بینی سربار کار برمی‌گزیند. به این دلیل که داده های سری زمانی به طول یک ساعت برای train مدل های پیچیده تر پیش بینی سربار کار، خیلی کم است. Job های با عمر طولانی تر بعد از تعیین ویژگی ها و طبقه بندی، مدل مناسب پیش بینی سربار کار برای آن ها اختصاص داده می‌شود.



شکل ۰-۱ معماری سیستم کشسانی ابری

C. Liu (Liu et al., 2017) سربار کار را به دو گروه Fast time-scale و Slow time-scale طبقه بندی کرده است. مبنای این طبقه بندی سرعت تغییرات داده های سربار کار است. داده های سربار کار مربوط به دسته Fast time-scale ویژگی های غیرخطی و تصادفی بیشتری دارند و Burstiness این دسته از داده ها بالاست. در مقابل در Job های مربوط به دسته Slow time-scale فاصله Peak تا میانگین داده های سربار کار، پایین است. به عبارت دیگر در Job های Fast time-scale در مقایسه با Job های Slow time-scale تغییرات با شدت بیشتری انجام می‌شود.

یک مدل پیش بینی سری زمانی ثابت برای پیش بینی سربار کار هر دو دسته نمی‌تواند دقت خوبی داشته باشد. C. Liu (Liu et al., 2017) دو مدل LR و SVM را برای پیش بینی سربار کار برگزیده است چون این دو مدل دقت و بهینگی مناسبی برای پیش بینی سری زمانی دارند. مخصوصاً برای پیش بینی داده های نمونه های کوچک و غیر خطی. مدل پیش بینی سری زمانی متناسب برای هر سربار کار بر اساس راهکاری که در ادامه توضیح داده خواهد شد اختصاص داده می‌شود.

پس از تعیین مدل پیش بینی متناسب توسط مولفه Workload classifier ، میزان سربار کار در چرخه بعدی توسط Workload predictor پیش بینی می‌شود نتایج حاصله به Load balancer داده می‌شود و تعداد VM های مورد نیاز برای چرخه بعدی تعیین می‌شود و بر اساس آن دستور scale-up یا scale-down به مولفه Configuration Tools داده می‌شود. میزان سربار کار در چرخه های ثابت زمانی اندازه گیری می‌شود طول این چرخه ها معمولا ۵ تا ۱۰ دقیقه می‌باشد. در هر چرخه زمانی تمام job هایی که در حال اجرا هستند در Workload Classifier طبقه بندی می‌شوند و میزان سرباره کار هر کدام از آنها برای چرخه زمانی بعدی پیش بینی می‌شود در نتیجه Workload Classifier نقش کلیدی در معماری وفق پذیر خاصیت کشسانی ابری دارد.

C. Liu (Liu et al., 2017) برای طبقه بندی سربار کار یک تابع هدف تعریف کرده است و در تلاش برای یافتن بهینه ترین حالت برای تابع مورد نظر است. مدل داده های سری زمانی به صورت زمان گسسته فرض شده است. مسئله به این صورت فرض شده است که مجموعه ای از سری زمانی وجود دارد که با w_t نشان داده می‌شود که $t \in \{1,2,\dots,k\}$، طوری که w_{t_0} میزان سربار کار در چرخه زمانی t_0 است. با فرض اینکه i نوع اپلیکیشن وجود دارد A_i $(i = 1,\dots,n)$ و j نوع پیش‌بینی مدل سری زمانی M_j $(j = 1,2)$ وجود دارد که باید بر اساس مشخصات و ویژگی‌های A_i مورد انتخاب قرار گیرد. با تخصیص مدل M_j اپلیکیشن A_i زمان پیش بینی را t_{ij} در نظر گرفته می‌شود. و میزان خطای پیش بینی را با تابع MRPE[۱] محاسبه کرده و تحت عنوان ε_{ij} در نظر گرفته می‌شود که در فرمول (۱) آمده.

$$\varepsilon_{ij} = \frac{1}{n}\sum_{s=1}^{n}\left(\left|{}^{(\overline{w}_s - w_s)}/_{w_s}\right|\right) \qquad (۱)$$

[۱] Mean Relative Percentage Error

میزان پیش بینی شده سربار کار $\overline{w}_t$ و میزان واقعی سربار کار w_t در زمان t است. برای اینکه به کمترین میزان از خطای پیش بینی برسیم تخصیص مناسب ترین مدل پیش بینی به اپلیکیشن مورد نظر بسیار مهم است. از آنجا که استخراج ویژگی از سربار کار پیچیده و زمانبر است تابع هدفی تعریف شده است که در آن با انتخاب x_{ij} مناسب سعی در رسیدن به بهترین حالت می‌کنیم.

$$x_{ij} = \begin{cases} 1 & A_i \; is \; not \; assigned \; model \; j \\ 0 & A_i \; is \; assigned \; model \; j \end{cases} \quad (۲)$$

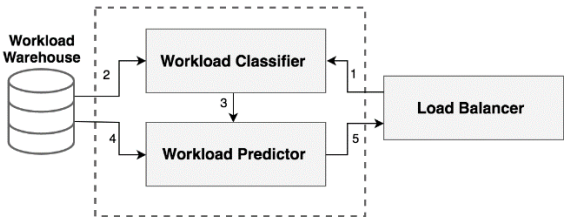
برای رسیدن به بهینگی باید معادله زیر کمینه شود.

$$\sum_{i=1}^{n}\sum_{j=1}^{2} \varepsilon_{ij}.x_{ij} \quad 0 < \varepsilon_{ij} < E_i \quad (۳)$$

ε_{ij} میزان خطای پیش بینی برای حالتی است که مدل j برای اپلیکیشن i انتخاب شده باشد و E_i نیز میزان حداکثر خطای پیش بینی برای اپلیکیشن i است.

معماری پیشنهادی

معماری جدید مولفه Workload Prediction در شکل ۴ ۱ ارایه شده است در این معماری بعد از این که job جدید وارد سیستم می‌شود، Load Balancer شماره Job مورد نظر را به Workload Classifier می‌دهد. مولفه Workload Classifier اطلاعات Job مورد را از Data Warehouse واکشی می‌کند. پس از تعیین فصلی بودن یا نبودن الگوی سری زمانی مربوط به سربار کار آن job، مدل متناسب را برای آن job تعیین می‌کند. در صورتی که سربار کار مورد نظر دارای الگوی فصلی باشد، مدل ARIAM برای آن انتخاب می‌شود در غیر این صورت مدل SVM انتخاب می‌شود. برای job هایی که میانگین مصرف CPU آنها ناچیز (کمتر از 0.002) باشد نیز مدل LR انتخاب می‌شود. مدل تعیین شده به مولفه Workload Predictor ارسال می‌شود و با استفاده از آن مدل مصرف CPU آن Job، در چرخه زمانی بعدی پیش‌بینی می‌شود. نتایج پیش بینی مولفه Load Balancer رفته براساس آن تصمیم متناسب برای مقیاس پذیری اتخاذ می‌شود.



شکل ۰-۱ معماری سیستم کشسانی ابری

در ادامه ساختار مدل های پیش بینی LR ، SVM و ARIMA بررسی شده و پیاده سازی می‌شوند. همچنین تابع هزینه مربوط به انتخاب بهترین مدل، تعریف می‌شود.

مدل LR

LR رابطه میان متغیر ورودی z با متغیر خروجی وابسته y را مدل می‌کند.

$$y_t = \beta_1 + \beta_2 z_t \quad (۴)$$

بطوریکه y به عنوان متغیر هدف که همان پیش بینی سری زمانی در نظر گرفته می‌شود و z نیز چرخه زمانی می‌باشد. ضرایب β_1 و β_2 با حل کردن معادله رگرسیون خطی بر اساس مقادیر کار قبلی y_{t-1} ، y_{t-2} و y_{t-3} تعیین می‌شود. بر اساس قانون cramer مقادیر ضرایب β_1 و β_2 با استفاده از فرمول زیر محاسبه می‌شود.

$$\beta_1 = \frac{\sum x_t^2 \sum y_t - \sum x_t \sum x_t y_t}{n\sum x_t^2 - (\sum x_t)^2} \quad (۵)$$

$$\beta_2 = \frac{n\sum x_t y_t - \sum x_t \sum y_t}{n\sum x_t^2 - (\sum x_t)^2} \quad (۶)$$

با دانستن مقادیر سربار کار قبلی، ضرایب β_1 و β_2 را بر اساس فرمول (۵) و (۶) حساب می‌کنیم و در نهایت با استفاده از فرمول (۴) مقادیر سربار کار چرخه زمانی بعدی را پیش بینی می‌کنیم.

```
1|      for s in range(lr_start - n, lr_end):
2|              x = np.array(df[s:n + s].time)
3|         y = np.array(df[s:n + s].cpu_rate)
4|
5|      b1 = ((sum(x ** 2) * sum(y)) - (sum(x) * sum(x * y))) / ((n * sum(x ** 2)) - (sum(x)
       ** 2))
6|      b2 = ((n * sum(x * y)) - (sum(x) * sum(y))) / ((n * sum(x ** 2)) - (sum(x) ** 2))
7|       df.lr_forecast[n + s + 1] = b1 + (b2 * (n    + s + 1))
```

کد مربوط به مدل LR

در کد مربوط به مدل LR داده ها در متغیر df قرار دارند که در خطوط ۱ تا ۳ مولفه زمان و مصرف CPU آن به آرایه های x و y کپی می‌شود. در خطوط ۵ و۶ طبق فرمول ۳-۲ و فرمول ۳-۳ ضرایب β_1 و β_2 محاسبه شده و در متغیر b1 و b2 ذخیره می‌شود و در نهایت در خط ۷ بر اساس فرمول (۴) مقدار پیش بینی شده محاسبه شده و در مولفه lr_forecast در متغیر df ذخیره می‌شود.

مدل SVM

قبل از پیاده سازی این مدل باید تغییر ساختاری در داده های سری زمانی اعمال شود. در این تغییر ساختار ۳ مقدار قبلی هر مرحله به عنوان ویژگی و مقدار آن مرحله به عنوان مقدار هدف در نظر گرفته می‌شود در نتیجه به ساختاری می‌رسیم` که در آن فضای برداری سه بعدی وجود دارد در هر مرحله به عنوان مقدار هدف برای پیش بینی در نظر گرفته می‌شود. حال با استفاده از کتابخانه LIBSVM به زبان python شروع به اعمال مدل SVM روی داده های سری زمانی می‌کنیم. در محاسباتمان از تابع gaussian RBF به عنوان تابع کرنل SVM استفاده می‌کنیم. این متد داده ها را به دو دسته ۷۰ درصدی train و ۳۰ درصدی test تقسیم می‌کند. پارامترهای مورد استفاده در تابع LIBSVM نیز به صورت زیر مقدار دهی شده است.

m=3, c=100, y= 0.5, s=3, p=0.001, t=2

```
1|  df['lag1'] = df['cpu_rate'].shift(1)
2|  df['lag2'] = df['cpu_rate'].shift(2)
3|  df['lag3'] = df['cpu_rate'].shift(3)

4|  for prediction_start in range(start, end):
5|          data = df.drop(['svm_forecast', 'lr_forecast', 'arima_forecast'], 1)
6|          data = data[:prediction_start]
7|          lags = 3
8|          data.dropna(inplace=True)
9|          X = np.array(data.drop(['cpu_rate', 'time'], 1))
10|         y = np.array(data['cpu_rate'])
11|         x_train = X[:-1]
12|         y_train = y[:-1]
13|          x_test = X[prediction_start - lags - 1:             prediction_start - lags]
14|     y_test = y[prediction_start - lags - 1:   prediction_start - lags]

15|         problem = svm_problem(y_train, x_train)

16|         m = svm_train(problem, '-t 2 -p 0.001 -s 3 -g 0.5 -c 100')
17|         p_lbl, p_acc, p_val= svm_predict(y_test, x_test, m)
18|        df.svm_forecast[prediction_start - 1] = p_val[0][0]
```

کد مربوط به مدل SVM

در کد مربوط به SVM ابتدا در خطوط ۱ تا ۳، سه مقدار قبلی مصرف CPU در مولفه های lag1، lag2 و lag3 ذخیره می‌شود. در خطوط ۵ تا ۸ ساختار متغیر df آماده ساختن داده های «ویژگی» و «هدف» می‌شود. در خطوط ۹ و ۱۰ متغیر های x و y که همان داده های ویژگی و هدف هستند، ساخته می‌شوند. در خطوط ۱۱ تا ۱۴ داده های train و test مربوط به مدل SVM ساخته می‌شود. در خطوط ۱۵ تا ۱۷ مدل svm ساخته شده و train می‌شود. و در نهایت در خط ۱۸ مقدار پیش بینی شده برای سیکل زمانی مربوطه در مولفه svm_forecast در متغیر df قرار می‌گیرد.

مدل ARIMA

مدل ARIMA ترکیبی از AR و MA است. در مدل AR فرض بر این گرفته می‌شود که مقادیر سیکل زمانی به مقادیر قبلی اش وابسته است. AR(p) مدل AR با گام p است.

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \varepsilon_t \qquad (۷)$$

به این صورت که در آن μ یک مقدار ثابت است و γ_p ضریب مقدار t-p است.
در مدل MA نیز فرض بر این است که مقادیر هر سیکل زمانی با مقادیر باقیمانده ها در گام های قبلی وابسته است. MA(q) نیز مدل MA با گام q است.

$$y_t = \mu + \sum_{i=1}^{p} \theta_i \varepsilon_{t-i} + \varepsilon_t \qquad (۸)$$

که در آن θ_i ضریب مقدار باقی مانده در گام t-q است.
حال می‌توان دو مدل AR و MA را با هم ترکیب کرد که به آن ARMA گفته می‌شود. مدل ARMA(p,q) بصورت زیر تعریف می‌شود.

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \sum_{i=1}^{p} \theta_i \varepsilon_{t-i} + \varepsilon_t \qquad (۹)$$

مدل ARIMA(p,d,q) همان مدل ARMA(p,q) است با این تفاوت که به اندازه d بار عمل تفریق را روی داده ها انجام می‌دهد که داده ها به ایستایی برسند. ایستایی داده ها به این معنی می‌باشد که میانگین و واریانس داده ها در گذر زمان تغییر نکند و همچنین داده ها فاقد روند[۲] زمانی باشند. همچنین مدل ARIMA یک مولفه فصلی هم در نظر می‌گیرد. به این صورت که ضریب sp، sd و sq برای بخش پذیری داده ها و همچنین دوره[۳] داده ها را گرفته و عمل AR، تفریق و MA را روی بخش فصلی داده ها اعمال می‌کند. برای تعیین دوره داده ها نیز از تابع ACF استفاده شده است.
در کد مربوط به مدل ARIMA نیز در خطوط ۲ تا ۴ ساختار داده ها در متغیر df ، آماده استفاده در مدل ARIMA می‌شود. در خط ۵ چک میشود که اگر داده های حالت فصلی داشته باشند مقدار ضرایب مولفه فصلی مدل ARIMA نیز به تابع مربوطه داده شود. در خطوط ۶ و ۸ مدل ساخته می‌شود. و در خطوط ۱۰ و ۱۱ مدل fit شده و مقادیر پیش بینی شده محاسبه می‌شود. و در خطوط ۱۲ تا ۱۵ مقادیر پیش بینی شده در مولفه arima_forecast در متغیر df قرار می‌گیرد.

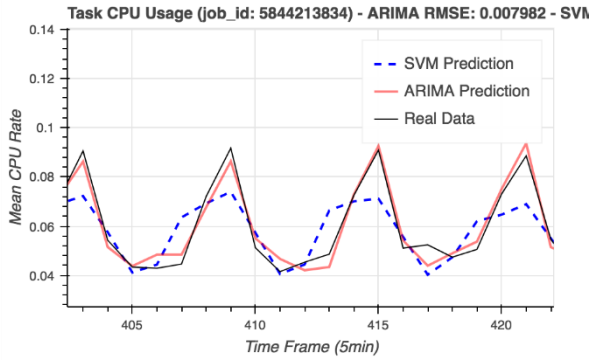
```
1|for prediction_start in range(start, end, 2):
2|      data = df.drop(['svm_forecast', 'lr_forecast', 'arima_forecast'], 1)
3|      data = data[:prediction_start]
4|      data['cpu_rate'].tolist()
5|      if period == 0:
6|             model = sm.tsa.statespace.SARIMAX( data['cpu_rate'].tolist(), order=(1, 1, 0),
                enforce_invertibility=False, enforce_stationarity=False)
7|      else:
8|             model = sm.tsa.statespace.SARIMAX( data['cpu_rate'].tolist(), order=(p, d, q),
                seasonal_order=(sp, sd, sq, period), enforce_invertibility=False,
                enforce_stationarity=False)
9|      model_fit = model.fit()
10|     pred = model_fit.predict(start=prediction_start,  end=prediction_start + steps,
dynamic=True)
11|     forecast_set = pred
12|     j = 1
13| for i in forecast_set:
14|     df['arima_forecast'][prediction_start + j - 1] = i
15|     j = j + 1
```

کد مربوط به مدل ARIMA

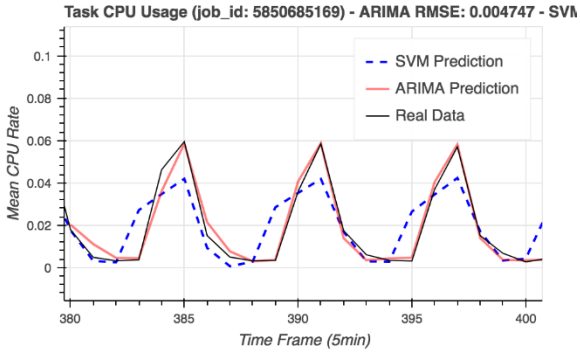
نتیجه گیری

در شکل ۰-۱ و شکل ۰-۲ نتیجه پیش بینی مدل ARIMA و SVM بر روی دو نمونه از job ها که دارای الگوی فصلی هستند آورده شده است. همانطور در این شکل مشهود است مدل ARIMA دقت بیشتری نسبت به مدل پیش بینی سری زمانی SVM، در این الگو ها دارد. به این دلیل که مدل ARIMA یک مولفه فصلی دارد که با بدست آورده دوره فصلی، مدل های AR و MA را روی بخش فصلی داده‌ها اعمال می‌کند. برای تشخیص دوره فصلی از تابع ACF استفاده شده است.

[۲] Trend
[۳] Period



شکل ۰-۱ ارزیابی مدل ARIMA و SVM روی الگوی فصلی



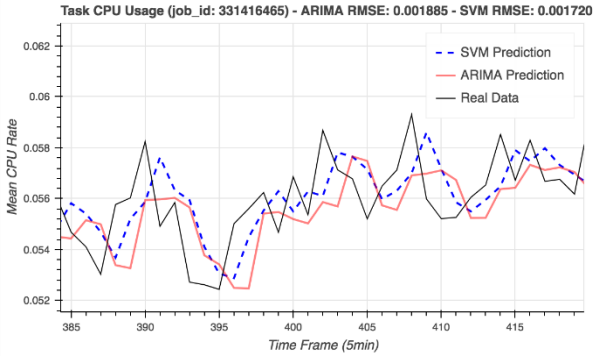
شکل ۰-۲ ارزیابی مدل ARIMA و SVM روی الگوی فصلی

میزان خطای پیش بینی RMSE برای job مربوط به شکل ۰-۱ و شکل ۰-۲ در جدول ۱ زیر آورده شده است. مقدار خطای پیش بینی مدل ARIMA برای این الگو بهتر از مدل SVM است.

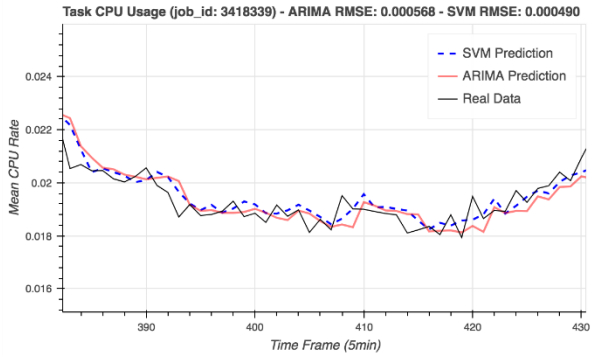
جدول ۱ خطای پیش بینی

Job_id	ARIAM RMSE	SVM RMSE
5850685169	0.004747	0.011361
5844213834	0.007982	0.011367

در شکل ۰-۳ و شکل ۰-۴ دو نمونه از job هایی که الگوی تصادفی دارند مورد بررسی قرار گرفته اند در این الگو ها مدل پیش بینی سری زمانی SVM دقت پیش بینی بهتری نسبت به مدل ARIMA دارد.



شکل ۰-۳ ارزیابی مدل ARIMA و SVM روی الگوی تصادفی



شکل ۰-۴ ارزیابی مدل ARIMA و SVM روی الگوی تصادفی

میزان خطای پیش بینی RMSE برای job مربوط به شکل ۰-۱ و شکل ۰-۲ در جدول ۲ زیر آورده شده است. مقدار خطای پیش بینی مدل ARIMA برای این الگو بهتر از مدل SVM است.

جدول ۲ خطای پیش بینی

Job_id	ARIAM RMSE	SVM RMSE
331416465	0.001885	0.001720
3418339	0.007982	0.011367

ارزیابی زمان اجرا

برای ارزیابی زمان اجرای مدل های پیش بینی سری زمانی، تعدادی از job ها را بصورت تصادفی انتخاب کرده و داده های مربوط به ۱۰۰ سیکل زمانی مشخص را از طریق هر سه مدل پیش بینی کردیم که زمان های اجرای مدل ها در جدول ۳ آمده است.

جدول ۳ زمان اجرای مدل های پیش بینی سری زمانی

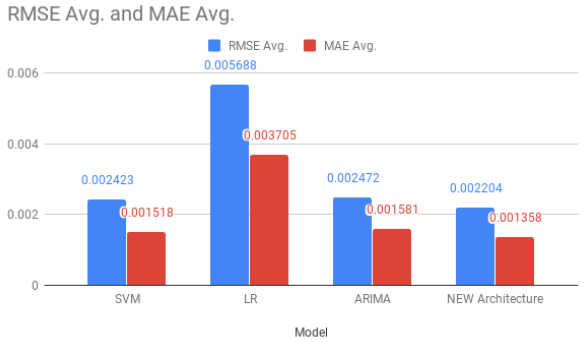
Model	Execution Time Avg. (s)
LR	0.08329286575
SVM	3.846281958
ARIMA	3.583490133

بر اساس نتایج بدست آمده زمان اجرای مدل SVM حدود ۴۶ برابر و مدل ARIMA حدود ۴۳ برابر مدل LR می‌باشد. با استفاده از معماری جدید پیشنهادی زمان متوسط پیش بینی به میزان ۲۵ درصد کاهش میابد.

ارزیابی معماری پیشنهادی

حال معماری پیشنهادی را با حالت هایی که مدل های پیش بینی سری زمانی LR ، SVM و ARIMA را به تنهایی برای تمام job ها استفاده کنیم مقایسه می‌کنیم.

شکل نشان دهنده میزان خطای پیش بینی میانگین RMSE و MAE مدل های LR ، ARIMA و SVM و معماری جدید است. میزان خطای پیش بینی در معماری جدید نسبت به حالت هایی که سه مدل را به تنهایی برای تمامی job ها استفاده کنیم کمتر می‌باشد و شاهد بهبود ۱۲ درصدی دقت پیش بینی سربار کار هستیم.



شکل ۰-۵ نمودار خطای پیش بینی مدل هایLR ، ARIMA و SVM و معماری جدید

جدول ۴ جدول خطای پیش بینی مدل های ARIMA و SVM و معماری جدید

Model	RMSE Avg.	MAE Avg.
SVM	0.002423	0.001518
ARIMA	0.002472	0.001581
LR	0.005688	0.003705
NEW Architecture	0.002204	0.001358

جمع بندی

در این پژوهش برای بهبود تجربه کاربر و همچنین عدم اتلاف منابع از مدل پیش‌بینی سربار کار سرور استفاده شده است. با پیش بینی سربار کار هر job سیستم می‌تواند تصمیمات مقیاس پذیری را با توجه به تاخیر انجام تغییرات اتخاذ کند بطوریکه کاربر این تغییرات را حس نکند. استفاده از یک مدل ثابت برای تمامی الگوهای سربار کار دقت نسبتا پایینی خواهد داشت. از این رو معماری جدیدی پیشنهاد شده است که در آن از سه مدل LR ، SVM و ARIMA استفاده شده است. نتایج نشان داده که مدل ARIMA برای الگو های فصلی سربار کار دقت بالاتری نسبت به SVM دارد در نتیجه با تشخیص الگوی سربار کار هر job، درصورت فصلی بودن مدل ARIMA را برای آن job انتخاب می‌کنیم. استفاده از این معماری منجر به افزایش ۱۲ درصدی دقت پیش بینی سربار کار گردید. و همچنین در صورت تخصیص مدل LR به job هایی که دارای میانگین میزان مصرف CPU ناچیز هستند زمان میانگین پیش بینی را به میزان ۲۵ درصد کاهش می‌دهد.

منابع

Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., & Merle, P. (2017). Elasticity in Cloud Computing: State of the Art and Research Challenges. *IEEE Transactions on Services Computing*.

Badger, L., Grance, T., Patt-Corner, R., & Voas, J. (2011). Draft cloud computing synopsis and recommendations. *NIST special publication, 800*, 146.

Bankole, A. A., & Ajila, S. A. (2013). *Cloud client prediction models for cloud resource provisioning in a multitier web application environment.* Paper presented at the Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on.

Herbst, N. R., Kounev, S., & Reussner, R. H. (2013). *Elasticity in Cloud Computing: What It Is, and What It Is Not.* Paper presented at the ICAC.

Hoffmann, G. A., Trivedi, K. S., & Malek, M. (2007). A best practice guide to resource forecasting for computing systems. *IEEE Transactions on Reliability, 56*(4), 615-628.

Hu, Y., Deng, B., Peng, F., & Wang, D. (2016). *Workload prediction for cloud computing elasticity mechanism.* Paper presented at the Cloud Computing and Big Data Analysis (ICCCBDA), 2016 IEEE International Conference on.

Islam, S., Lee, K., Fekete, A., & Liu, A. (2012). *How a consumer can measure elasticity for cloud platforms.* Paper presented at the Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering.

Khan, A., Yan, X., Tao, S., & Anerousis, N. (2012). *Workload characterization and prediction in the cloud: A multiple time series approach.* Paper presented at the Network Operations and Management Symposium (NOMS), 2012 IEEE.

Lehrig, S., Eikerling, H., & Becker, S. (2015). *Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics.* Paper presented at the Quality of Software Architectures (QoSA), 2015 11th International ACM SIGSOFT Conference on.

Liu, C., Liu, C., Shang, Y., Chen, S., Cheng, B., & Chen, J. (2017). An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications, 80*, 35-44.

Morais, F. J. A., Brasileiro, F. V., Lopes, R. V., Santos, R. A., Satterfield, W., & Rosa, L. (2013). *Autoflex: Service agnostic auto-scaling framework for iaas deployment models.* Paper presented at the Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on.

Nikravesh, A. Y., Ajila, S. A., & Lung, C.-H. (2015). *Towards an autonomic auto-scaling prediction system for cloud resource provisioning.* Paper presented at the Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.

Pahl, C. (2015). Containerization and the paas cloud. *IEEE Cloud Computing, 2*(3), 24-31.