Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 10, Number 2, Summer 2021, pp. 39-52
Received: 09/25/2020, Revised: 01/19/2021, Accepted: 06/01/2021

An ELM-based Load Balancing Algorithm for Cloud Computing Platforms

Sedigheh Bagheri[1], Seyedakbar Mostafavi[1*] and Fazlollah Adibnia[1]
1- Department of Computer Engineering, Yazd University, Yazd, Iran.
[1]s.bagheri@stu.yazd.ac.ir, [2*]a.mostafavi@yazd.ac.ir, and [3]fadib@yazd.ac.ir

Corresponding author's address: Seyedakbar Mostafavi, Department of Computer Engineering, Yazd University, Yazd, Iran.

Abstract- Since the workload of the end users and the provisioned cloud resources are dynamically changed over time, the workload is not evenly distributed over the cloud. Therefore, designing appropriate mechanisms to detect the status of the cloud and properly distribute the load on each host can play an effective role in improving system performance and energy consumption in cloud data centers. Reactive load balancing approaches don't prevent load-imbalance in cloud and make virtual machines (VM) migrate after load imbalance and increase energy consumption and job response time. Also, in proactive load balancing methods, some problems, such as host state detection with insufficient accuracy and fixed threshold of cpu utilization without considering the host current and future states in VM migrations, prevent the optimal number of balanced hosts and energy consumption in datacenters. In this paper, a proactive approach to the early detection of host states is presented which is based on Extreme Learning Machine (ELM). The proposed approach predict the CPU utilization of each host over time and applies an adaptive threshold to determine the future status of each host (i.e., overload, underload, secure and normal state). Then, a subset of VMs are migrated to hosts with minimum overload probability in future to avoid overloaded hosts. Implementation of the proposed method and its evaluation on the real data sets in Cloudsim show that the proposed method improves energy consumption, response time, the number of VM migrations and non-violation of the Service Level Agreement (SLA) in comparison to competitive algorithms including RF-LB [7] and ANN-LB [13].

Keywords- Load Balancing, Load Prediction, Extreme Learning Machine (ELM), Virtual Machine Migration, Adaptive Threshold

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۱۰، شماره ۲، تابستان ۱۴۰۰، صفحه ۳۹-۵۲
دریافت: ۱۳۹۹/۰۷/۰۴، بازنگری: ۱۳۹۹/۱۰/۳۰، پذیرش: ۱۴۰۰/۰۳/۱۱

الگوریتم توازن‌بار مبتنی بر پیش‌بینی ELM در محاسبات ابری

صدیقه باقری[۱]، سید اکبر مصطفوی*[۲]، فضل الله ادیب نیا[۳]
۱- دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران.
*۲- دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران.
۳- دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران.
[1]s.bagheri@stu.yazd.ac.ir, [2*]a.mostafavi@yazd.ac.ir, [3]fadib@yazd.ac.ir

* نشانی نویسنده مسئول: سید اکبر مصطفوی، یزد، بلوار دانشگاه، دانشگاه یزد، دانشکده مهندسی کامپیوتر.

چکیده- از آنجا که تقاضای کاربران و رفتار سیستم از نظر تخصیص منابع، پویا و متغیر با زمان است، بار کاری به شکل متوازن روی منابع ابر توزیع نمی‌شود. طراحی مکانیزم‌های مناسب جهت تشخیص وضعیت و توزین مناسب بار روی هر میزبان می‌تواند نقش موثری در بهبود کارایی سیستم و مصرف انرژی در مراکز داده ابر داشته باشد. روش‌های توازن بار ارائه شده به صورت واکنشی از ورود سیستم به حالت عدم توازن جلوگیری نکرده و متناسب با شرایط ایجاد شده دست به مهاجرت ماشین مجازی(VM) می‌زنند. در این روش‌ها، با ورود سیستم به حالت عدم توازن، انرژی مصرفی و همچنین زمان پاسخ کارها افزایش می‌یابد. همچنین در روش‌های توازن‌بار پیش‌دستانه، عدم دقت کافی برای تشخیص وضعیت میزبان‌ها، استفاده از آستانه‌های ثابت و همچنین مهاجرت ماشین‌های مجازی به میزبان‌ها، بدون در نظر گرفتن وضعیت کنونی و آینده آنها، احتمال پربار شدن میزبان‌ها و افزایش انرژی مصرفی در مراکز داده را بالا می‌برد. از این رو، روش پیشنهادی این مقاله، بکارگیری یک رویکرد پیش‌دستانه با هدف تشخیص زودهنگام وضعیت میزبان‌ها است که مقدار مصرف پردازنده هر میزبان در آینده، توسط روش ماشین یادگیری افراطی (ELM) پیش‌بینی می شود و با استفاده از سه آستانه تطبیقی وضعیت آتی میزبان‌ها مشخص می‌شود، سپس ماشین‌های مجازی از میزبان‌های پربار و درصورت نیاز میزبان‌های کم بار به آن دسته از میزبان‌هایی انتقال پیدا می‌کنند که احتمال پربار شدن آنها بعد از تخصیص کمینه باشد. پیاده‌سازی روش پیشنهادی و ارزیابی آن روی مجموعه داده واقعی با استفاده از شبیه ساز Cloudsim نشان داده است که روش پیشنهادی در مقایسه با روش پیش‌دستانه و واکنشی رقیب، در انرژی مصرفی، زمان پاسخ، تعداد مهاجرت‌های ماشین مجازی و عدم نقض توافقنامه سطح سرویس (SLA) بهبود ایجاد کرده است.

واژه‌های کلیدی: توازن‌بار، پیش‌بینی بار، ماشین یادگیری افراطی، مهاجرت ماشین‌های مجازی ، آستانه تطبیقی.

۱- مقدمه

سیستم‌های محاسباتی به طور گسترده‌ای درحال کامل شدن هستند تا بتوانند پاسخگوی نیازهای بشر در مسائل و کاربردهای مختلف علمی، تجاری و غیره باشند. رایانش ابری[۱] یکی از رویکردهای محاسباتی است که طی چند سال اخیر مورد توجه قرار گرفته است. ابر منابع محاسباتی را با توجه به نیاز کاربر و بر حسب تقاضا، با دریافت هزینه از مشتری ارائه می‌دهد. محاسبات ابری از نظر الگوی بارکاری و رفتار سیستم‌ها در محیط‌های مختلف بسیار پویا هستند. برای مدیریت موثر برنامه‌ها و منابع موجود از روش‌های کارآمد برای تعیین مناسب‌ترین مقدار هر

منبع برای هر بارکاری استفاده می‌شود. یکی از روش‌های مدیریت منابع در ابر، الگوریتم‌های توازن‌بار است که از پربارشدن میزبان‌ها در شرایطی که میزبان‌های دیگر، کم‌بار و یا بیکار باشند، جلوگیری می‌کند. اضافه بار در هر میزبان باعث کاهش کارایی سرویس‌ها می‌شود، در حالی که کمبود بار باعث به هدر رفتن منابع می‌شود. از این رو کنترل وضعیت بار منابع در مراکز داده برای رسیدن به کیفیت سرویس (QoS[۲]) و استفاده کارآمد از منابع، امری ضروری است. همچنین استفاده بهینه از منابع و انرژی، هزینه‌های عملیاتی و پیاده‌سازی را کنترل می‌کند[۱]. از اهداف رویکردهای توازن‌بار می‌توان به کاهش زمان پاسخ، پایدار نگه‌داشتن سیستم، قابلیت تحمل خطا، افزایش کارایی سیستم برای بدست آوردن بهره‌وری بهینه منابع، بهبود دسترس‌پذیری منابع و کاهش انرژی مصرفی و به دنبال آن کاهش انتشار کربن در محیط، اشاره کرد[۲].

ممکن است استفاده از یک مرکزداده در طول زمان به دلیل ایجاد ماشین‌های مجازی (VM[3]) و میزبان‌های جدید، شکست میزبان‌های موجود در آن و یا حذف VMهای موجود تغییرکند. بنابراین نیاز به سازماندهی مجدد VMها و میزبان‌ها برای ایجاد توازن‌بار و ادغام سرورها با توجه به معیارهای توافقنامه سطح سرویس (SLA[۴]) است. در مدیریت منابع در ابر و ایجاد توازن‌بار چهار مسئله مهم وجود دارد که به مهاجرت VMها مرتبط است. مهاجرت VM یک ویژگی بسیار قدرتمند مجازی سازیست که با توجه به نیازهای فعلی به منابع، VMها را در حداقل تعداد میزبان‌های فعال به منظور کاهش انرژی مصرفی ادغام می‌کند. در ادامه این چهار مسئله توضیح داده می‌شود[۳و۴]:

الف) تشخیص میزبان‌های پربار[۵]: تشخیص وضعیت هر میزبان بر اساس میزان بهره‌وری CPU آن انجام می‌شود. اگر این مقدار در یک میزبان خیلی زیاد شود، آن میزبان در حالت پربار قرار می‌گیرد. الگوریتم‌های متفاوتی برای تشخیص وضعیت میزبان‌ها وجود دارد. استفاده از آستانه‌های تطبیقی و ثابت، روش‌های یادگیری ماشین و راهکارهای پیش‌بینی و غیره می‌توانند استفاده شوند. هنگامی که یک میزبان پربار می‌شود، باید تعدادی از VMهای آن به میزبان‌های دیگر منتقل شود تا از وضعیت پربار شده خارج شود.

مقدار بهره‌وری CPU هر میزبان (CPU_u) با استفاده از رابطه (۱) بدست می‌آید که در آن Total.Req.Mips کل Mips[۶]های درخواست شده توسط VMهای درون میزبان، و TotalHostMips نیز کل Mips در میزبان مورد نظر است[۳]:

$$CPU_u(H_j) = \frac{Total.Req.Mips}{TotalHostMips} \quad (۱)$$

اگر مقدار $CPU_u(H_j)$ به ۱ نزدیک شود، کل Mipsهای میزبان در حال استفاده است و درخواست‌های جدید ممکن است با تأخیر روبه‌رو شوند. از طرفی، اگر این مقدار بسیار پایین باشد، منابع در میزبان اتلاف می‌شود و انرژی مصرفی بالا می‌رود.

ب) تشخیص میزبان‌های کم‌بار[۷]: وقتی میزان بهره‌وری CPU یک میزبان کم باشد، آن میزبان کم‌بار شناخته می‌شود. در میزبان‌های کم‌بار اغلب، همه VMهای درون میزبان به میزبان‌های دیگر منتقل می‌شوند و میزبان به حالت خواب، به منظور صرفه‌جویی در مصرف انرژی تغییر حالت می‌دهد، و یا یک گزینه مناسب برای جایگذاری VMهای مهاجرتی میزبان پربار شده، در نظر گرفته می‌شود.

ج) انتخاب VMها برای مهاجرت از میزبان‌های پربار و میزبان‌های کم‌بار: پس از تصمیم‌گیری برای مهاجرت VMها از یک میزبان مشخص، مرحله انتخاب VM آغاز می‌شود و یک یا چند VM را برای انتقال به میزبان دیگر انتخاب می‌کند. انتخاب VMها برای مهاجرت، براساس پارامترهای متفاوتی انجام می‌شود که شامل بهره‌وری CPU، میزان حافظه، مدت زمان مهاجرت و تعداد کارهای در حال اجرا بر روی آنها می‌باشد. فرآیند مهاجرت VM، منابع شبکه و CPU را هم در میزبان مبدأ و هم در میزبان مقصد مصرف می‌کند و همچنین برای مدت معینی نیز VM را از دسترس خارج می‌کند که تا حدودی SLA را نقض می‌کند. کارایی دیگر VMهای موجود بر روی میزبان مبدأ و مقصد نیز تحت تأثیر افزایش درخواست منابع برای مهاجرت قرار می‌گیرند[۵].

د) انتخاب میزبان مناسب برای جایگذاری VMهای انتخاب شده برای مهاجرت[۸]: سرانجام مرحله انتخاب مقصد برای VMهای مهاجرتی آغاز می‌شود. جایگذاری VMها بر اساس عوامل مختلفی مانند منابع در دسترس میزبان (شامل CPU، حافظه، پهنای‌باند وغیره) انرژی مصرفی در مراکز داده و همچنین ترافیک کارها بر روی VM انجام می‌شوند. هدف از الگوریتم‌های جایگذاری VM، ارائه بهترین کیفیت سرویس به برنامه‌های در حال اجرا بر روی VM است. پس از انتخاب میزبان مبدأ، فرآیند مهاجرت آغاز می‌شود.

هدف اصلی این مقاله کاهش انرژی مصرفی و زمان پاسخ کارها طی دو مرحله تشخیص وضعیت میزبان و جایگذاری VMها است. با توجه به اینکه مصرف CPU بیشترین تأثیر در مصرف انرژی مراکز داده دارد، در این روش از الگوریتم ماشین یادگیری افراطی

یا ELM[۹] برای پیش‌بینی مقدار مصرف CPU هر میزبان در آینده استفاده می‌شود. الگوریتم پیشنهادی وضعیت آتی مصرف CPU را در یک تکرار و با دقت بالا محاسبه می کند. سپس مکانیزمی برای تشخیص وضعیت میزبان ها بر اساس سه آستانه تطبیقی محاسبه می شود. بر اساس وضعیت میزبان ها، الگوریتم های جایگذاری ماشین مجازی در میزبان ها با در نظر گرفتن وضعیت فعلی و آتی آنها با هدف حداقل کردن میزبان های پربار در آینده انجام می شود.

در ادامه، بخش ۲، شامل کارهای مرتبط گذشته است. در بخش ۳، روش پیش‌بینی ارائه شده توضیح داده می‌شود. در بخش ۴، الگوریتم توازن‌بار پیشنهادی مطرح شده است. در قسمت ۵، شبیه‌سازی انجام شده و عملکرد روش پیشنهادی در مقایسه با دیگر روش‌ها و در بخش ۶ نیز نتیجه‌گیری و کارهای آینده آورده شده است.

۲-کارهای پیشین

در دو دهه گذشته مدیریت منابع مراکز داده، مهاجرت VMها و تخصیص مجدد آنها اهمیت قابل‌توجهی داشته است. الگوریتم‌های زیادی برای تشخیص وضعیت میزبان‌ها برای بدست آوردن بهره‌وری بهینه منابع محاسباتی و کاهش مصرف انرژی ارائه شده‌اند. در ادامه نمونه‌هایی از این الگوریتم‌ها آورده شده است:

در [۴] چند الگوریتم جایگذاری VM بر اساس آستانه تطبیقی و با استفاده از مهاجرت زنده VMها ارائه شده است. برای تشخیص گره‌های پربار از آستانه بهره‌وری تطبیقی برای بدست آوردن آستانه بالایی در میزبان‌ها استفاده شده که شامل انحراف مطلق از میانگین (MAD)، محدوده چارک‌ها (IQR)، رگرسیون محلی (LR) و رگرسیون محلی مقاوم (RLR) است. این روش‌ها بر اساس تحلیل داده‌های گذشته مرتبط با میزان منابع مورد استفاده هر VM در طول عمر آن محاسبه می‌شوند. انتخاب VM برای مهاجرت نیز طبق روش‌های کمترین زمان مهاجرت (MMT)، سیاست انتخاب تصادفی (RS) و سیاست بیشترین ضریب همبستگی (MC) انتخاب می‌شوند.

انتخاب مقصد برای جایگذاری VMها بر اساس مسئله Bin-Packing انجام می‌شود که در این روش همه VMهای انتخاب شده برای مهاجرت، به ترتیب کاهش میزان بهره‌وری CPU مرتب می‌شوند و هر VM را به میزبانی که کمترین افزایش انرژی ناشی از تخصیص VM مورد نظر دارد، منتقل می‌کند. VMها در میزبان‌هایی که دارای بهره‌وری CPU کمینه هستند نیز به دیگر میزبان‌ها منتقل می‌شوند و به وضعیت خواب می‌روند تا انرژی مصرفی را کاهش دهند[۶،۴].

مقاله [۳] یک الگوریتم جایگذاری VM بر اساس روش پیش‌بینی مارکوف پیشنهاد داده است که از مهاجرت‌های غیرضروری جهت کمینه کردن نقض SLA استفاده می‌کند. در این مقاله از زنجیره مارکوف مرتبه اول و بر اساس بردار آخرین بهره‌وری CPU میزبان‌ها، وضعیت هر میزبان در آینده مشخص می‌شود. در مدل مارکوف پیشنهادی، سه حالت برای گذار میزبان‌ها وجود دارد که شامل O یا پربار، U یا کم‌بار و N یا نرمال می‌شود. آستانه‌های استفاده شده نیز شامل دو آستانه بالا و پایین می‌شود که آستانه پایین ثابت و برابر با ۱/. و آستانه بالایی نیز طبق الگوریتم MAD در [۴] بدست می‌آید. وضعیت هر میزبان در داده‌های آموزشی با مقایسه میزان بهره‌وری کنونی با این آستانه‌ها مشخص می‌شود. داده استفاده شده دراین مقاله شامل Planetlab و داده‌های تصادفی می‌شود. شبیه‌سازی این روش نیز در CloudSim انجام شده است.

مقاله [۷] یک رویکرد توازن‌بار با استفاده از الگوریتم‌های پیش‌بینی و مهاجرت VMها بیان کرده است. در این مقاله ابتدا ۴ روش یادگیری ماشین برای تعیین وضعیت بار هر میزبان استفاده می‌شود. K نزدیکترین همسایه[۱۰]، ماشین بردار پشتیبان[۱۱]، شبکه‌های عصبی[۱۲] و جنگل تصادفی[۱۳] از جمله الگوریتم‌های یادگیری ماشین هستند که در این مقاله استفاده شده است. مقدار آستانه بالا و پایین به صورت ثابت در نظر گرفته شده است که آستانه پایین برابر با ۱/. و آستانه بالا برابر با ۸/. در نظر گرفته شده است. انتخاب VMهای مهاجرتی در میزبان‌های پربار و کم‌بار طبق سیاست ماکزیموم ضریب همبستگی بین بهره‌وری‌های VMها و ضریب همبستگی درونی بین VMها انجام می‌شود و نسبت به مقاله [۴] انتخاب دقیق‌تری برای مهاجرت VMها داشته است. به این صورت که در میزبان‌های پربار VM که دارای ضریب همبستگی ۱ باشد و ضریب همبستگی درونی آن، غیر۱ باشد، به دیگر میزبان‌ها منتقل می‌شود. در میزبان‌های کم‌بار نیز VM که دارای ضریب همبستگی درونی صفر با دیگر VMهای آن میزبان دارد، به میزبان‌هایی با وضعیت پربار منتقل می‌شود. اگر ضریب همبستگی درونی بین VMهای میزبان کم‌بار، صفر نباشد VMها به میزبان‌های عادی دیگر منتقل می‌شود و میزبان به حالت خواب می‌رود. این روش با استفاده از مجموعه داده Planetlab در شبیه‌ساز CloudSim شبیه‌سازی شده است. در این روش مصرف انرژی و زمان پاسخ کارها در نظر گرفته نشده است.

در مقاله [۸] یک روش پیش‌بینی مصرف CPU بر اساس رگرسیون خطی ارائه شده است که از میزان استفاده CPU هر میزبان در گذشته، استفاده می‌کند و میزان مصرف CPU آن را در کوتاه مدت تخمین می‌زند. روش LIRCUP ابتدا میزبان‌های پربار در آینده نزدیک را، پیش‌بینی می‌کند و قبل از وقوع نقض SLA مهاجرت VM را انجام می‌دهد. میزبان‌های کم‌بار نیز پس از مهاجرت VMهای آن به حالت خواب می‌روند. در داده‌های آموزشی مورد استفاده روش پیش‌بینی، میزبانی پربار در نظر گرفته می‌شود که مصرف CPU آن از ۸۵ درصد کل بهره‌وری CPU در یک ساعت گذشته بیشتر باشد و اگر مقدار بهره‌وری CPU یک میزبان کمتر از ۱۰ درصد بهره‌وری‌های CPU در یک ساعت گذشته باشد، آن میزبان کم‌بار در نظر گرفته می‌شود. بعد جمع‌آوری داده‌های آموزشی روش تخمین رگرسیون خطی برای تعیین وضعیت میزبان در آینده اعمال می‌شود. در این مقاله از مجموعه داده ComonProject از داده‌های Planetlab و داده‌های تصادفی در محیط CloudSim استفاده شده است.

در [۹] یک سیاست پیش‌گویانه مهاجرت VMها با استفاده از مدل پیش‌بینی گروهی ارائه شده است که از پیش‌بینی کوتاه مدت بهره‌وری آینده CPU مربوط به VMها و میزبان‌ها برای کشف میزبان‌های پربار استفاده می‌کند. بارکنونی سیستم مورد نظر در بازه‌های زمانی گسسته کنترل می‌شوند و به صورت سری‌زمانی در یک فایل برای آموزش روش‌های پیش‌بینی ذخیره می‌شوند. روش پیش‌بینی بر اساس یادگیری گروهی طراحی شده است که به ترکیب وزنی چندین روش اشاره دارد. هر روش به صورت مستقل پیش‌بینی را انجام می‌دهد و نتایج به وسیله میانگین‌گیری ساده بین آنها ترکیب می‌شوند تا قابلیت اطمینان پیش‌بینی‌ها بیشتر شود. در این روش اگر درصد بهره‌وری پیش‌بینی شده از ۸۴ درصد کل CPU میزبان بیشتر شود، میزبان پربار و مهاجرت VM آغاز می‌شود. سیاست کمترین زمان مهاجرت نیز برای انتخاب VMهای مهاجرتی در میزبان پربار استفاده می‌شود. هدف اصلی این الگوریتم، ایجاد یک مصالحه بین انرژی مصرفی و نقض SLA است.

در مقاله [۱۰] یک الگوریتم پیش‌بینی بار برای تخصیص پویای منابع ارائه شده است که رویکرد اجتناب از پربارشدن میزبان را در پیش گرفته است. در این مقاله از روش پیش‌بینی میانگین متحرک با وزن نمایی بهبودیافته (EEWMA) برای تخمین نیازهای آینده به منابع و سپس مشخص کردن تعداد میزبان‌های فیزیکی مورد نیاز استفاده می‌شود.

در مرجع [۱۱] یک الگوریتم جایگذاری VM انرژی-آگاه با استفاده از مدل‌های پیش‌بینی ارائه شده است. در این روش، پیش‌بینی بهره‌وری آینده CPU و حافظه VMها و ماشین‌های فیزیکی انجام می‌شود. در روش UP-VMC ارائه شده، جایگذاری VM به عنوان یک مسئله Bin-Packing دوبعدی (CPU و MEMORY) در نظر گرفته می‌شود. روش‌های پیش‌بینی استفاده شده شامل رگرسیون خطی و k نزدیکترین همسایه است که برای تخمین کوتاه مدت بار استفاده می‌شود. اگر حداقل مقدار بهره‌وری پیش‌بینی شده بزرگتر از ظرفیت در دسترس منابع باشد، میزبان پربار محسوب می‌شود و بعضی از VMهای آن مهاجرت داده می‌شود. انتخاب VMهای مهاجرتی بر اساس سیاست‌های کمترین زمان مهاجرت، VM با بار کمینه و VM با بار بیشینه انجام می‌شود. همه VMها در میزبان‌های کم‌بار نیز به دیگر میزبان‌ها مهاجرت می‌کنند و میزبان مورد نظر خاموش می‌شود. روش UP-VMC میزبانی را به عنوان مقصد برای مهاجرت VMها انتخاب می‌کند که ظرفیت مورد نیاز حال و آینده برای تخصیص VM را داشته باشد. هدف اصلی این روش بهینه کردن تعداد مهاجرت‌ها، انرژی مصرفی و نقض SLA است.

مقاله [۱۲] روش رگرسیون چند بعدی را برای پیش‌بینی میزبان‌های پربار در آینده استفاده کرده است. پارامترهای بهره‌وری در نظر گرفته شده شامل CPU، حافظه و پهنای باند می‌باشد که این ۳ پارامتر برای محاسبه بهره‌وری میزبان ترکیب شده‌اند. مجموعه داده استفاده شده برای شبیه‌سازی در محیط Cloudsim شامل UniLu-Gaia-2014-1.swf و PlanetLab است. هدف این روش کمینه کردن مصرف انرژی و نقض SLA با توجه به انواع مختلف درخواست‌های کاربران (مبتنی بر CPU، مبتنی بر حافظه و مبتنی بر ورودی و خروجی) است.

در مرجع [۱۳] یک الگوریتم توازن‌بار به صورت واکنشی ارائه شده است که از آستانه پایینی ثابت برابر با ۰/۲ و یک آستانه تطبیقی مبتنی بر شبکه عصبی استفاده کرده است. در روش ANN-LB، قبل از مهاجرت VMها، مقدار بار تمام میزبان‌ها در آینده پیش‌بینی می‌شود و میانگین مقادیر پیش‌بینی شده به عنوان آستانه بالایی برای تعیین وضعیت میزبان‌ها به کار می‌رود و مهاجرت VM به میزبان‌هایی که مقدار بهره‌وری CPU آنها بین آستانه بالایی و پایینی است، انجام می‌شود. همچنین VM انتخابی برای مهاجرت، VM است که مقدار CPU مورد نیاز آن، بزرگتر یا مساوی اختلاف بین آستانه بالایی و میزبان پربار باشد. ورودی‌های شبکه عصبی مقدار بهره‌وری CPU هر میزبان و تعداد VMهای فعال بر روی آن میزبان است و خروجی بهره‌وری CPU در آینده

است.

۳- روش پیش‌بینی ماشین یادگیری افراطی

ELM، یک الگوریتم یادگیری برای شبکه‌های عصبی پیش‌خور با تک لایه پنهان (SLFN[۱۴]) است. ELM می‌تواند در هنگام آموزش تعداد نرون‌های لایه پنهان را به صورت تطبیقی و وزن بین لایه ورودی و لایه پنهان و همچنین آستانه گره‌های لایه پنهان (Bias) را به صورت تصادفی تنظیم کند (تعداد نرون‌های لایه پنهان می‌تواند به عنوان ورودی به الگوریتم داده شود). وزن بین لایه پنهان و خروجی نیز بر اساس راه‌حل حداقل مربعات بدست می‌آید. کل فرآیند یادگیری در یک تکرار انجام می‌شود و سرعت یادگیری آن در مقابل شبکه‌های عصبی پس‌خور مبتنی بر گرادیان کاهشی به طور قابل توجهی بهتر است [۱۴].

برای N نمونه مجزا و دلخواه به شکل (x_i, t_i) که در آن $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in R^n$، i امین نمونه با n ویژگی متفاوت و $t_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in R^m$ باشد، شبکه‌های عصبی پیش‌خور تک لایه پنهان با $\tilde{N}$ گره درلایه پنهان و تابع فعال‌سازی f(x)، به صورت رابطه (۲) مدل می‌شود [۱۵،۱۴]:

$$\sum_{i=1}^{\tilde{N}} B_i . f_i(x_i) = \sum_{i=1}^{\tilde{N}} B_i . f_i(a_i . x_j + b_i) = t_j \,, \quad j = 1, \dots, N \qquad (۲)$$

که $a_i = [a_{i1}, a_{i2}, \dots, a_{in}]^T$ بردار وزن بین i امین گره لایه پنهان و گره‌های ورودی، b_i آستانه i امین گره در لایه پنهان و $B_i = [B_{i1}, B_{i2}, \dots, B_{im}]^T$ بردار وزن بین i امین گره لایه پنهان و گره‌های خروجی است. $a_i . x_j$ نیز ضرب داخلی دو بردار a_i و x_j است. رابطه (۱) را می‌توان به صورت رابطه (۳) نوشت [۱۵،۱۴]:

$$HB = T \qquad (۳)$$

در این رابطه:

$$H = [a_1, a_2, \dots, a_{\tilde{N}}, b_1, b_2, \dots, b_{\tilde{N}}, x_1, x_2, \dots, x_N]^T \qquad (۴)$$

$$\begin{bmatrix} f(a_1 . x_1 + b_1) & \cdots & f(a_{\tilde{N}} . x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ f(a_1 . x_N + b_1) & \cdots & f(a_{\tilde{N}} . x_N + b_{\tilde{N}}) \end{bmatrix},$$

$$T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}, B = \begin{bmatrix} B_1^T \\ \vdots \\ B_{\tilde{N}}^T \end{bmatrix}$$

است که H ماتریس خروجی لایه پنهان در شبکه عصبی است[۱۵،۱۴]. i امین ستون ماتریس H، خروجی i امین گره لایه پنهان طبق ورودی‌های $x_1, \dots, x_n$ است. هنگامی که آموزش شبکه شروع می‌شود، SLFN به صورت تصادفی وزن بین گره‌های لایه ورودی و لایه پنهان و همچنین Bias مربوط به گره‌های لایه پنهان را مقداردهی می‌کند. بعد از مقداردهی تصادفی در مرحله قبل، ماتریس خروجی لایه پنهان محاسبه و آموزش SLFN به حل معادله خطی حداقل مربعات در رابطه (۴) تبدیل می‌شود[۱۵،۱۴]:

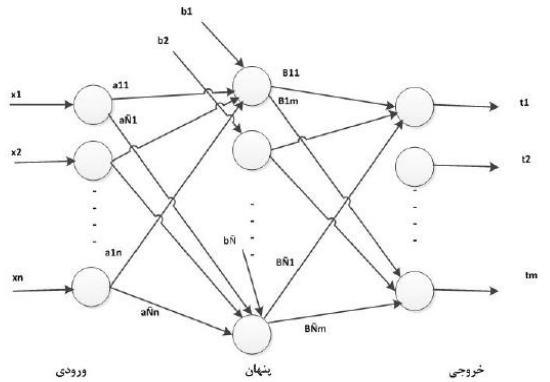
$$\|H(a_1, a_2, \dots, a_{\tilde{N}}, b_1, b_2, \dots, b_{\tilde{N}})B - T\| = \min_B \|H(a_1, a_2, \dots, a_{\tilde{N}}, b_1, b_2, \dots, b_{\tilde{N}})B - T\|, \qquad (۵)$$

$$\hat{B} = H^{\tau}T$$

راه‌حل بهینه $\hat{B}$ دارای حداقل خطای آموزشی است و به دلیل یکتا بودن آن، از راه‌حل بهینه محلی جلوگیری می‌کند.

۴- روش توازن‌بار EBHLPB

با توجه به اهمیت تقاضاهای منابع در آینده برای مدیریت کارآمد منابع، روش EBHLPB از ماشین یادگیری افراطی برای پیش‌بینی سری زمانی بهره‌وری CPU در میزبان‌ها استفاده می‌کند و سپس از نتایج آن برای توازن‌بار پیش‌دستانه بر روی مراکز داده بهره می‌گیرد. در این بخش، در قسمت ۱ معماری سیستم ارائه شده، بیان می‌شود. سپس در قسمت ۲، مدل ELM بکاربرده شده در این روش، نشان داده می‌شود و در ادامه نیز الگوریتم توازن‌بار با استفاده از رویکرد مهاجرت VMها، بیان می‌گردد.

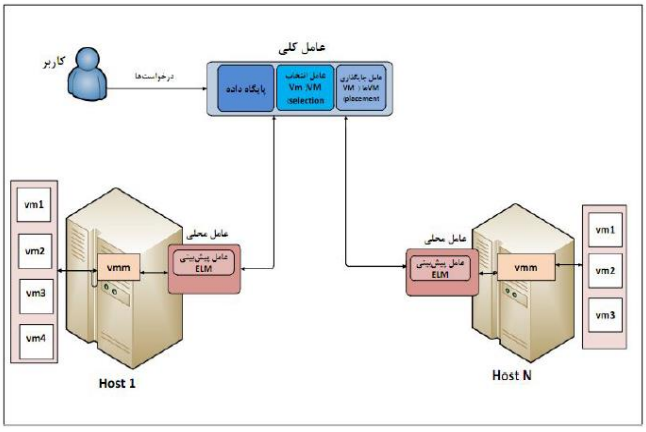


شکل ۱: ساختار کلی **ELM**

۴-۱- معماری سیستم

شکل ۲ معماری سیستم در روش پیشنهادی را نشان می‌دهد. سیستم هدف در محیط زیرساخت به عنوان سرویس پیاده‌سازی

شده است که شامل تعدادی مرکز داده است. هر مرکز داده شامل m میزبان ناهمگن است. میزبان‌ها به وسیله پارامترهایی مانند تعداد VMهای اختصاص داده شده، مقدار Mips، پهنای باند، حافظه و غیره از همدیگر متمایز می‌شوند. بر روی هر میزبان ممکن است چندین VM وجود داشته باشد. هنگامی که درخواستی از طرف کاربر وارد سیستم می‌شود، واسط[۱۵] با توجه به الزامات درخواست، آن را به سمت یک مراکز داده می‌فرستد. درخواست‌ها به VMهای مختلفی که بر روی میزبان‌ها نشسته‌اند، داده می‌شوند. معماری ارائه شده شامل دو مولفه اصلی است، عامل کلی و عامل محلی. عامل محلی به صورت توزیع‌شده درون هر میزبان قرار دارد تا بتواند میزان بهره‌وری CPU هر میزبان را به صورت پیوسته نظارت کند. عامل محلی شامل عامل پیش‌بینی ELM است. عامل پیش‌بینی ELM مسئول پیش‌بینی بهره‌وری CPUآینده هر میزبان است. ناظر VM نیز مسئول جایگذاری و اختصاص VMها به میزبان‌هاست. همچنین با دریافت نقشه مهاجرت از عامل کلی، مهاجرت‌ها را نیز انجام می‌دهد. عامل کلی، در هر مرکز داده قرار دارد و با عامل محلی در میزبان‌ها در ارتباط است تا بتواند دید کلی نسبت به بهره‌وری‌های کنونی و آینده هر میزبان داشته باشد. عامل‌های انتخاب VM، جایگذاری VM و پایگاه داده در عامل کلی قرار دارند. عامل انتخاب VM، مسئول پیدا کردن VM مناسب برای مهاجرت و عامل جایگذاری VM، مسئول پیدا کردن میزبان‌های مناسب برای VMهای مهاجرتی است. پایگاه داده موجود در عامل کلی نیز، تمام اطلاعات مربوط به بهره‌وری CPU هر میزبان و VM را جمع‌آوری و نگه‌داری می‌کند. عامل کلی با استفاده از رویکرد EBHLPB یک نقشه مهاجرت را که شامل VMهای مهاجرتی و میزبان‌های مبدأ و مقصد است، می‌سازد و به ناظر VM دستور می‌دهد که مهاجرت را انجام دهد.

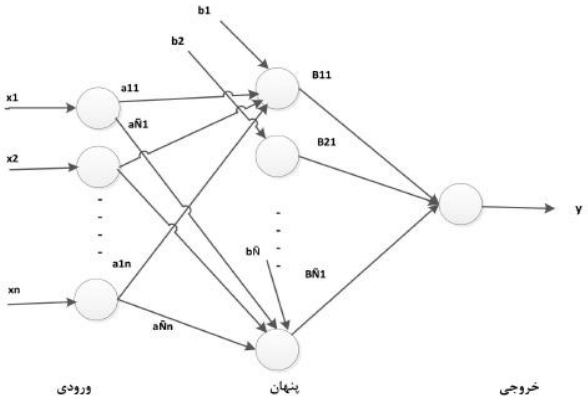


شکل ۲: معماری ارائه شده در روش EBHLPB

۴-۲- تابع پیش‌بینی

در این روش از تحلیل سری‌زمانی برای پیش‌بینی مقدار بهره‌وری هر میزبان در آینده استفاده شده است. از ELM می‌توان برای پیش‌بینی رگرسیون و طبقه‌بندی استفاده کرد. برای نمونه در [۱۶] از تابع رگرسیون مصرف برق به صورت سری زمانی استفاده کرده است. در این روش از پیش‌بینی رگرسیون ELM استفاده شده است. تابع پیش‌بینی ELM در روش EBHLPB مقادیر گذشته بهره‌وری هر میزبان را به عنوان داده‌های آموزشی می‌گیرد و مقدار CPU در آینده نزدیک را پیش‌بینی می‌کند.

در این روش ELM، پیش‌بینی را بر اساس آخرین مقدارهای بهره‌وری CPU هر میزبان است، انجام می‌دهد. داده‌های مربوط به بهره‌وری CPU گذشته هر میزبان در یک فایل ذخیره شده و به عنوان داده‌های آموزشی ELM استفاده می‌شود. ورودی ELM در این مدل ۴ مقدار آخرین بهره‌وری CPU هر میزبان، و خروجی مقدار بهره‌وری CPU در آینده برای آن میزبان است. مدل ELM استفاده شده در این روش در شکل ۳ نشان داده شده است. با توجه به اینکه میزبان‌ها ناهمگن هستند، برای هر نوع میزبان داده‌های آموزشی جداگانه به ELM داده می‌شود.



شکل ۳: ساختار ELM در روش EBHLPB

تابع نگاشت ویژگی ELM در این روش، تابع Sigmoid است که در رابطه (۶) آمده است [۱۵،۱۴]:

$$G(a,b,x) = {}^{1}\!/_{1+\exp(-(a.x+b))} \qquad (۶)$$

۴-۳- الگوریتم توازن‌بار

همانطور که قبلاً ذکر شد، یکی از راه‌حل‌های توازن‌بار برای کمینه کردن مصرف انرژی در مراکز داده، مهاجرت VMها است

که مسئله توازن‌بار را، به چهار زیرمسئله شامل تشخیص میزبان‌های پربار و کم‌بار، انتخاب VM برای مهاجرت و دیگری جایگذاری VMها تقسیم کرده است. الگوریتم EBHLPB با بیان راه‌حل‌های جدید در مرحله تشخیص میزبان‌های پربار و کم‌بار و مرحله جایگذاری VMها، توازن‌بار را انجام می‌دهد. این روش به صورت پیش‌دستانه عمل می‌کند. روش‌های پیش‌دستانه بر خلاف روش‌های واکنشی[۱۶] که بر اساس پارامترهای کنونی، توازن‌بار را انجام می‌دهند، از پیش‌بینی تقاضاها در بازه‌های زمانی آینده سعی در تشخیص وضعیت میزبان‌های درون سیستم در آینده و ایجاد توازن‌بار دارد. به عبارت دیگر، روش EBHLPB تلاش می‌کند قبل از ورود سیستم به حالت نامتوازن و پربار و کم‌بار شدن میزبان‌ها، وضعیت آینده آنها را مشخص کند و با مهاجرت VMها از ورود سیستم به حالت نامتوازن جلوگیری کند تا از این طریق، مصرف انرژی و نقض SLA را کاهش دهد. در ادامه هر یک از مراحل توازن‌بار توضیح داده می‌شود.

تشخیص میزبان‌های پربار و کم‌بار: تشخیص وضعیت میزبان‌ها در آینده بر اساس روش پیش‌بینی و آستانه‌های تطبیقی انجام می‌شود. آستانه منطبق با تغییر بهره وری CPUهای میزبان‌ها در طول زمان می تواند از روش های مختلفی شامل از رگرسیون محلی، میانگین، MAD و IRQ محاسبه شود. با توجه به اینکه در نظر گرفتن آستانه ثابت برای محیط‌های پویا مانند ابر مناسب نیست، در روش EBHLPB از رویکرد تطبیقی برای تنظیم خودکار آستانه‌های بهره‌وری CPU بر اساس داده‌های گذشته جمع‌آوری شده، در طول عمر VMهای درون میزبان‌ها استفاده شده است. برای این منظور، ۳ آستانه پیشنهاد می شود که شامل آستانه بالایی، آستانه امن و آستانه پایین می‌شود. در این جا از چارک ها به عنوان یکی از شاخصهای آماری مقاوم[۱۷] استفاده شده است تا تاثیر کمی از داده های پرت مربوط بهره وری CPU میزبان بپذیرد. آستانه بالایی برابر با چارک سوم، آستانه امن برابر با چارک دوم یا میانه و آستانه پایین برابر با چارک اول مقادیر بهره‌وری CPUهای مربوط به میزبان‌ها می‌شود. شبه کد مربوط به تشخیص وضعیت هر میزبان در شکل ۴ آمده است. در این الگوریتم U نشان‌دهنده وضعیت کم‌بار، O نشان‌دهنده وضعیت پربار، N نشان‌دهنده وضعیت عادی و S نشان دهنده وضعیت امن میزبان‌ها است. با مقایسه بهره‌وری پیش‌بینی شده به وسیله ELM، با آستانه‌های مشخص شده، وضعیت هر میزبان تعیین می‌شود.

اگر میزبانی دارای وضعیت پربار در آینده باشد، برخی VMهای آن میزبان مهاجرت داده می‌شود، تا از پربار شدن میزبان در آینده نزدیک جلوگیری شود. همچنین زمانی که وضعیت آینده یک میزبان کم‌بار پیش‌بینی شود، تمام VMهای آن منتقل و خود به حالت خواب جهت صرفه‌جویی در انرژی می‌رود.

انتخاب VMها: بعد از مشخص شدن وضعیت میزبان‌ها در حال و آینده، مرحله انتخاب VMهای مهاجرتی در میزبان‌های پربار آغاز می‌شود. در این قسمت از سیاست ماکزیمم ضریب همبستگی ارائه شده در [۴] و ضریب همبستگی درونی در [۷] استفاده می‌شود. ابتدا ضریب همبستگی بین پارامترهای بهره‌وری CPU یک VM با دیگر VMها طبق رابطه‌های (۷) و (۸) محاسبه می‌شود. سپس ضریب همبستگی درونی برای VMهایی که دارای ضریب همبستگی ۱ هستند، طبق رابطه (۹) محاسبه می‌شود و VM که دارای ضریب همبستگی درونی ۱ باشد، از لیست مهاجرت حذف می‌شود. در ادامه نحوه محاسبه ضریب همبستگی و ضریب همبستگی درونی آورده شده است.

Algorithm 1: Host State Detection in EBHLPB Algorithm

```
   /* CPU Utilization of
      Host(j)(CPU_util(H_j),
      LowerThreshold, SecureThreshold,
      UpperThreshold) (input)        */
   /* StateofHost (output)           */
   /* determine underloaded Hosts    */
 1 if CPU_util(H_j) ≤ LowerThreshold then
 2 |  StateofHost ⟵ U
 3 end
   /* determine overloaded Hosts     */
 4 if CPU_util(H_j) ≥ UpperThreshold then
 5 |  StateofHost ⟵ O
 6 end
   /* determine secure Hosts         */
 7 if LowerThreshold ≤ CPU_util(H_j) ≤
   SecureThreshold then
 8 |  StateofHost ⟵ S
 9 end
   /* determine Normal Hosts         */
10 if SecureThreshold ≤ CPU_util(H_j) ≤
   UpperThreshold then
11 |  StateofHost ⟵ N
12 end
13 return StateofHost
```

شکل۴: الگوریتم تشخیص میزبان‌های پربار و کم‌بار در روش EBHLPB

ضریب همبستگی: ضریب همبستگی برای تعیین نوع و درجه رابطه یک متغیر با متغیر دیگر است. اگر $x_1, x_2, \ldots, x_n$ n متغیر تصادفی که نشان‌دهنده مقادیر بهره‌وری CPU مربوط به n، VM موجود بر روی میزبان باشد و Y نیز نشان‌دهنده یک VM برای مهاجرت باشد آنگاه، n-1 متغیر مستقل و ۱ متغیر وابسته داریم. m اندازه آرایه مربوط به بهره‌وری‌های هر VM است. ماتریس X یک ماتریس افزونه با ابعاد m*n مربوط به متغیرهای مستقل و y

یک ماتریس m*1 مربوط به متغیر وابسته Y است. بردار مقادیر پیش‌بینی شده متغیر وابسته Y توسط $\hat{y}$ نشان داده می‌شود که، طبق رابطه (۷) محاسبه می‌شود[۴]:

$$X = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,n-1} \\ & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \cdots & x_{m,n-1} \end{bmatrix}$$
$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \quad (۷)$$
$$\hat{y} = Xb, b = (X^T X)^{-1} X^T y$$

با پیدا شدن مقادیر پیش‌بینی حال می‌توان ضریب همبستگی که با R نشان داده می‌شود را، طبق رابطه (۸) محاسبه کرد[۴]:

$$R_{Y,X_1,\ldots,X_{n-1}} = \frac{\sum_{i=1}^{n}(y_i - m_Y)(\hat{y}_i - m_{\hat{Y}})}{\sum_{i=1}^{n}(y_i - m_Y)\sum_{i=1}^{n}(\hat{y}_i - m_{\hat{Y}})} \quad (۸)$$

که در این رابطه m_Y و $m_{\hat{Y}}$ نشان‌دهنده متوسط مقادیر بردارهای Y و $\hat{y}$ هستند.

ضریب همبستگی درونی: ضریب همبستگی درونی بین VMهای ۱و ۲،۳،۴ ($IR^2{}_{1234}$) طبق رابطه (۹) محاسبه می‌شود[۷]:

$$IR^2{}_{1234} = \frac{R^2{}_{12} + R^2{}_{13} + R^2{}_{14} - 2(R_{12})(R_{13})(R_{14})R_{234}}{1 - R^2{}_{234}} \quad (۹)$$

در این رابطه، R_{xyz} ضریب همبستگی بین VMهای شماره x و y و zاست. همچنین $R^2{}_{xyz}$ مربع ضریب همبستگی بین VMهای شماره x و y و z را نشان می‌دهد.

شبه‌کد مربوط به انتخاب VMها در شکل ۵ آمده است.

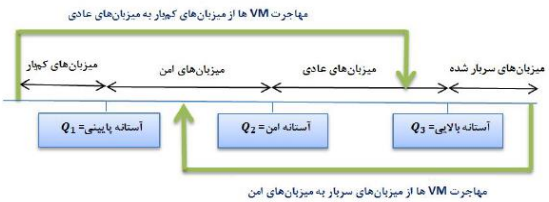
جایگذاری VMها: گام بعد از مشخص شدن میزبان‌های پربار و کم‌بار و تعیین‌VMهای مهاجرتی، مرحله جایگذاری VM بر اساس نتیجه پیش‌بینی ELM است. الگوریتم ارائه شده در جایگذاری VMها هم حالت کنونی و هم حالت آینده میزبان‌ها را در نظر می‌گیرد، تا از این طریق مانع از مهاجرت‌های غیر ضروری شود. برای انتخاب میزبان مناسب برای پذیرفتن VMها، سه شرط زیر بررسی می‌شود:

شرط اول: با توجه به اینکه بهره‌وری‌های یک میزبان تابعی از بهره‌وری‌های VMها است، VMهای انتخاب شده از میزبان‌های پربار بهره‌وری بالاتری دارند و پس از انتقال به میزبان‌های عادی احتمال پربار شدن آنها را زیاد می‌کنند. همچنین VMهای کم‌بار دارای بهره‌وری پایین‌تری هستند و با انتقال به میزبان‌های امن ممکن است بهره‌وری آنها را در حد پایین نگه دارد و این تخصیص کارآمد نباشد. بنابراین VMهای گره‌های پربار به میزبان‌هایی منتقل می‌شوند که وضعیت کنونی آنها امن و نسبت به میزبان‌های عادی میزان بهره‌وری بالایی ندارند. همچنین VMهای گره کم‌بار به میزبان‌ها در ناحیه عادی منتقل می‌شوند که بهره‌وری بالاتری نسبت به میزبان‌های امن دارند، تا از این طریق هم احتمال پربار شدن میزبان‌ها بعد تخصیص کاهش یابد و هم تخصیص کارآمد باشد. نمایی از شرط یک در شکل ۶ آمده است.

Algorithm 2: VM Selection in Over-loaded Hosts in EBHLPB Algorithm

```
/* Host List, VM List (input)                    */
/* Migration List (output)                       */
foreach Host(j) in Host List, j=1 to m do
    if Host is predicted as Overloaded then
        foreach VM(i) in Vm List, i=1 to n
        do
            find the correlation factor between
            CPU utilization of VM(i) and
            CPU utilization of other VM in
            Host(j)
            if correlation factor ==1 then
                find inter-correlation of VM(i)
                with other VMs in Host(j)
                if inter-correlation factor
                ==1 then
                    put VM(i) into ignorelist
                end
                else
                    put VM(i) in to
                    Migration List of VM
                end
            end
        end
    end
end
return Migration List of VM
```

شکل ۵: شبه کد انتخاب VM در میزبان های پربار شده



شکل ۶: نمایی از شرط یک در جایگذاری VMها در روش EBHPLB

شرط دوم: میزبان‌های انتخاب شده در ناحیه‌های مشخص شده باید بعد از تخصیص پربار نشوند که طبق رابطه (۱۰) این شرط بررسی می‌شود.

$$CPU_u(H_j) + CPU_u(VM_i)/(\text{TotalHostMips}) \leq UpperThreshold \quad (۱۰)$$

که در این رابطه $CPU_u(H_j)$ مقدار بهره‌وری کنونی میزبان و $CPU_u(VM_i)$ مقدار CPU درخواستی VM موردنظر است.

شرط سوم: وضعیت آینده میزبان مقصد نباید پربار باشد. شبه‌کد جایگذاری VMها در شکل۷ نشان داده شده است. همچنین در شکل ۸ شبه کد کلی الگوریتم EBHLPB آورده شده است.

۵- ارزیابی نتایج

در این بخش ابتدا پارامترهای روش پیش‌بینی مورد استفاده و سپس پارامترهای روش توازن‌بار EBHLPB ، و ارزیابی آن بیان شده است.

Algorithm 3: VM Placement in EBHLPB Algorithm

```
   /* Host List, Migration List of VM,Sourcehost (input) */
   /* Destinationhost (output) */
 1 if Sourcehost.futurestate == O then
 2     foreach host in Host List do
 3         if HostisActive== true && CPU_u(H_j) + (CPU_Req(VM_i)/TotalHostMIPS) ≤ UpperThreshold(Q3) && host.currentstate==S && Host.futurestate= U or S or N then
 4             TempHostList[] ⟵ add Host
 5         end
 6     end
 7 end
 8 if Sourcehost.futurestate == U then
 9     foreach Host in Host List do
10         if HostisActive== true && CPU_u(H_j) + (CPU_Req(VM_i)/TotalHostMIPS) ≤ UpperThreshold(Q3) && host.currentstate==N && Host.futurestate= U or S or N then
11             TempHostList[] ⟵ add Host
12         end
13     end
14 end
15 Destinationhost = random element of TempHostList[]
16 return Destinationhost
```

شکل ۷: شبه کد الگوریتم جایگذاری VM در روش EBHLPB

۵-۱- پارامتر روش پیش‌بینی

برای ارزیابی الگوریتم‌های توازن‌بار ابتدا باید پارامترهایی که در هر رویکرد برای روش پیش‌بینی تنظیم شده است بیان شود. با توجه به اینکه تنظیمات خاصی در [۷] برای روش RF عنوان نشده است، در اینجا تنظیمات پیش‌فرض استفاده شده است. شبکه عصبی استفاده شده در [۱۳] نیز دارای یک لایه پنهان، تعداد نرون‌های لایه ورودی و لایه پنهان نیز ۲ و تعداد نرون‌های لایه خروجی نیز ۱ است. در روش ELM، تعداد نرون‌های لایه پنهان باید به عنوان ورودی به الگوریتم داده شود. در شکل ۹ تأثیر افزایش تعداد نرون‌های لایه پنهان بر روی دقت مدل پیش‌بینی ELM نشان داده شده است. دقت روش با استفاده از RMSE[۱۸] طبق رابطه (۱۱) محاسبه شده است. در این رابطه، n تعداد نمونه‌ها، $\tilde{y}_i$ مقدار خروجی پیش‌بینی شده و y_i مقدار واقعی خروجی است[۷].

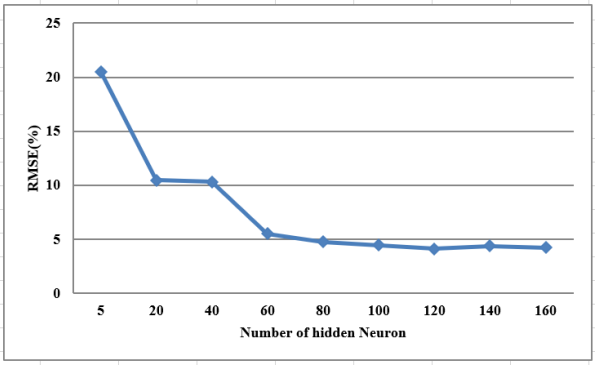
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2} \quad (۱۱)$$

در نمودار شکل ۹ مقدار بهره‌وری CPU پیش‌بینی شده ۲۰ میزبان با تعداد مختلف گره‌های لایه پنهان دیده می‌شود. با توجه به اینکه در این نمودار از تعداد ۱۰۰ به بعد، دقت ELM، تقریباً همگرا شده است، مقدار ۱۰۰ به عنوان تعداد نرون‌های لایه پنهان در ELM انتخاب شده است.

Algorithm 4: EBHLPB Algorithm

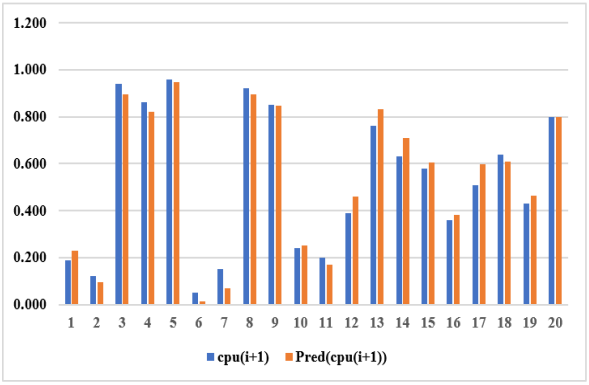
```
   /* Host (input) */
   /* MigrationMap (output) */
   /* calculate Thresholds */
 1 UpperThreshold=Q_3(CPU utilization)
 2 SecureThreshold=Q_2(CPU utilization)
 3 LowerThreshold=Q_1(CPU utilization)
 4 while HostisActive== True do
       /* calculate current CPU utilization of Host */
 5     currentutilization = TotalReq.MIPS / TotalhostMIPS
       /* find current state using Algoritm 1 */
 6     currentstate ⟵ HostStateDetection (currentutilization, LowerThreshold, SecureThreshold, UpperThreshold)
       /* find future utilization whit ELM Prediction */
 7     Futureutilization ⟵ ELM(4 lasts CPU utilizations of host)
 8     Futurestate ⟵ HostDetectionState(futureutilization, LowerThreshold, SecureThreshold, UpperThreshold)
 9     if Futurestate == O then
10         migrateVMs: select some VM from Algorithm 2
11         foreach VM in migrateVMs do
12             VMPlacement (VM, Host)
13             MigrationMap[] ⟵ (migrateVMs,VMPlacement)
14         end
15     end
16     if Futurestate== U then
17         foreach VM in allVMs do
18             VMPlacement (VM, Host)
19             MigrationMap[] ⟵ (allVMs,VMPlacement)
20         end
21     end
22 end
23 return MigrationMap
```

شکل ۸: شبه کد کلی روش EBHLPB

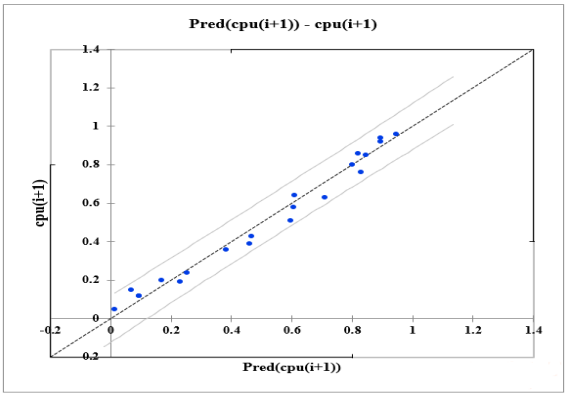


شکل ۹: تأثیر افزایش تعداد نرون‌های لایه پنهان در دقت روش ELM

در شکل ۱۰، نمودار مقادیر پیش‌بینی شده میزان بهره‌وری CPU در ۲۰ میزبان با روش ELM در کنار مقادیر اصلی بعد از ۱۰ تکرار الگوریتم به صورت میانگین نمایش داده شده است. همچنین نمودار رگرسیون مربوط به روش پیش‌بینی در شکل ۱۱ آمده است.



شکل ۱۰: نمودار مقادیر پیش‌بینی شده بهره‌وری CPU با استفاده از روش ELM و مقدار واقعی



شکل ۱۱: نمودار رگرسیون روش پیش‌بینی

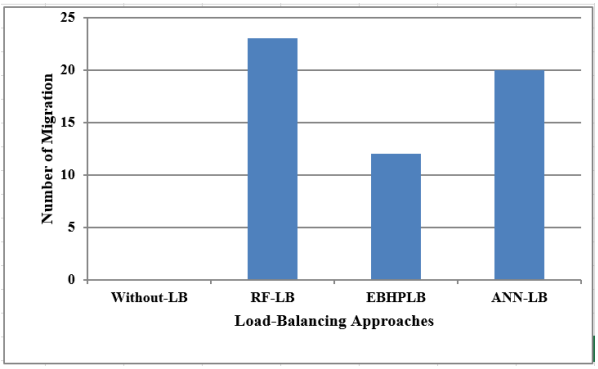
۵-۲- ارزیابی روش EBHLPB

شبیه‌سازی روش EBHLPB درشبیه‌ساز CloudSim3.0 انجام شده است. مجموعه داده استفاده شده در این مقاله UniLu-Gaia-2014-1 می‌باشد. شبیه‌سازی روش EBHLB ارائه شده با روش RF-LB مطرح شده در [۷] و همچنین روش ANN-LB ارائه شده در [۱۳] بر روی ۲۰ میزبان و ۴۰، VM ناهمگن مقایسه شده است. در جدول ۱ ویژگی‌های موجودیت‌های مورد استفاده آورده شده است.

ارزیابی تعداد مهاجرت‌های VM: مهاجرت VMها، علاوه بر مصرف منابع در ماشین مبدأ و مقصد، از منابع شبکه برای انتقال استفاده می‌کند. هرچه تعداد مهاجرت‌ها بیشتر باشد، ترافیک شبکه بیشتر می‌شود و تأثیر منفی بر عملکرد دیگر VMها دارد[۴،۳]. بنابراین مهاجرت VMها، باعث کاهش کارایی سیستم می‌شود. به همین دلیل کمینه کردن مهاجرت VMها یک امر ضروری است. مقایسه انجام شده بین روش ارائه شده، روش مقاله [۷]، روش مقاله [۱۳] و روش بدون توازن‌بار در شکل ۱۲ نشان داده شده است.

جدول ۱- جدول مشخصات موجودیت‌های مورد استفاده در آزمایشات

پارامترها	مقادیر
تعداد میزبان‌ها	۲۰
تعداد ماشین‌های مجازی	۴۰
تعداد مراکز داده	۳
تعداد کارها	۵۰۰
طول کارها	۱۰-۵۰۰۰ Mips
پردازنده مربوط به میزبان‌ها	۱۰۰۰-۴۰۰۰Mips
حافظه مربوط به میزبان‌ها	۲۰۰۰-۴۰۰۰Mbps
پهنای‌باند مربوط به میزبان‌ها	۱۰۰۰۰Gbps
تعداد عناصر محاسباتی در میزبان‌ها	۵-۲ PE
پردازنده مربوط به ماشین‌های مجازی	۳۰۰-۱۰۰۰Mips



شکل ۱۲: مقایسه تعداد مهاجرت VMها

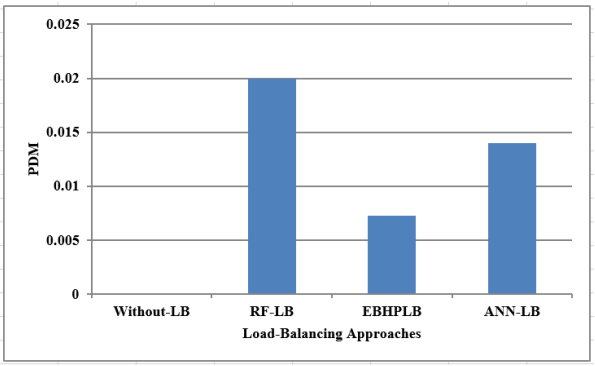
هنگامی که سیستم توازن‌بار را در نظر نمی‌گیرد هیچ‌گونه مهاجرتی در سیستم رخ نمی‌دهد و سیستم همچنان با بهره‌وری‌های کم و زیاد میزبان‌ها به اجرای کارها ادامه می‌دهد. در رویکرد پیشنهادی EBHPLB، روش پیش‌بینی استفاده شده و راه‌حل تطبیقی به کاربرده شده، توانایی بالاتری در تشخیص گره‌های پربار و کم‌بار داشته است. همچنین با توجه به جایگذاری متناسب VMها با در نظر گرفتن حالت آینده و کنونی در رویکردهای ارائه شده، احتمال پربار شدن میزبان‌ها بعد از مهاجرت کم می‌شود و نیاز به مهاجرت کمتر می‌شود. در روش RF-LB دقت روش پیش‌بینی برای تشخیص وضعیت میزبان‌ها و همچنین حساسیت جایگذاری VMها به اندازه روش پیشنهادی نیست. مطابق انتظار، در روش ANN-LB که آستانه بالایی به صورت تطبیقی تنظیم شده است، تعداد مهاجرت‌ها را نسبت به روش RF-LB کاهش داده است زیرا در RF-LB آستانه به صورت ثابت تنظیم شده و تشخیص وضعیت میزبان‌ها با خطا همراه است.

ارزیابی تخریب عملکرد سیستم به واسطه مهاجرت یا PDM[۱۹]: همانطور که گفته شد مهاجرت VM، اثر منفی بر عملکرد سیستم می‌گذارد که این کاهش عملکرد را حدود ۱۰ درصد بهره‌وری VM در هنگام مهاجرت تخمین زده‌اند [۱۷].

بنابراین، مقدار تخریب عملکرد در هر میزبان طبق رابطه (۱۴) محاسبه می‌شود[۶،۴، ۳] :

$$PDM = \frac{1}{V}\sum_{v=1}^{V}\frac{Cd_v}{Cr_v} \qquad (۱۴)$$

که در آن V تعداد VMها، Cd_v برابر با ۱۰ درصد مصرف CPU در VM_v در همه مهاجرت‌ها تخمین زده شده است [۴،۳]. Cr_v نیز کل CPU درخواست شده توسط VM_v است. در شکل ۱۳ نتایج مربوط به پارامتر PDM آورده شده است. سناریو استفاده شده بدون رویکرد توازن‌بار، هیچ گونه تخریب عملکردی ناشی از مهاجرت ندارد، زیرا هیچ مهاجرتی اتفاق نیافتاده است. از طرفی به دلیل کمتر شدن تعداد مهاجرت‌ها در روش EBHPLB، تخریب عملکرد ناشی از آن‌ها نسبت به روش RFLB و ANN-LB کاهش یافته است.



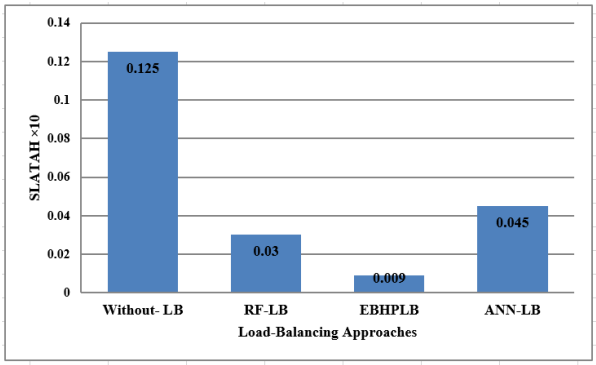
شکل ۱۳: مقایسه میانگین PDM

ارزیابی میزان نقض SLA در هر میزبان یا SLATAH[۲۰]: این پارامتر مقدار نقض SLA در هر میزبان را نشان می‌دهد که برابر است با مدت زمانی که میزبان مورد نظر بهره‌وری CPU، معادل با ۱۰۰ درصد را تجربه می‌کند. SLATAH طبق رابطه (۹) به دست می‌آید[۴،۳].

$$SLATAH = \frac{1}{J}\sum_{j=1}^{J}\frac{T_{sj}}{T_{aj}} \qquad (۱۵)$$

که J تعداد کل میزبان‌ها، T_{sj} کل زمانی که میزبان j دارای بهره‌وری CPU، ۱۰۰ درصد است و T_{aj} نیز کل زمان فعال بودن میزبان j را نشان می‌دهد.

همانطور که در شکل ۱۴ نشان داده شده است، در رویکرد بدون توازن‌بار، اگر بهره‌وری میزبانی به صد درصد برسد یا میزبانی پربار شود، هیچ گونه اقدامی نخواهد شد و تا زمانی که حجم کارها در سیستم بالا باشد، و یا VMها تخریب نشده باشند، همچنان در حالت پربار می‌ماند. در روش RF-LB درصد بیشتری از میزبان‌ها نسبت به روش پیشنهادی وضعیت صد درصد بهره‌وری را تجربه می‌کنند، که حاصل از تشخیص نادرست وضعیت پربار شده میزبان و عدم بررسی وضعیت میزبان بعد از تخصیص در مسئله مهاجرت VM از گره‌های پربار است. روش ANN-LB یک روش واکنشی است و هنگامی که سیستم به وضعیت نامتعادل رسیده باشد، اقدام به توازن بار می‌کند، به همین دلیل تعداد میزبان‌هایی که به بهره‌وری صد در صد می‌رسند افزایش پیدا کرده است.

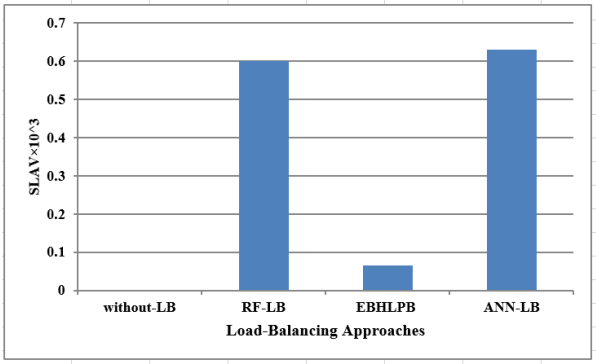


شکل ۱۴: مقایسه میانگین SLATAH

میزان نقض SLA یا SLAV: SLA یا توافقنامه سطح سرویس، بین مشتری و تأمین‌کنندگان سرویس برای اطمینان از سطح خدمات ارائه شده، انجام می‌شود. SLA شامل پارامترهای مختلفی از جمله حداقل ظرفیت CPU و حافظه و پهنای باند است. هنگامی که مقدار درخواستی هر کدام از این پارامترها از ظرفیت موجود بیشتر شود، نقض توافقنامه سطح سرویس اتفاق می‌افتد. SLAV طبق رابطه (۱۶) محاسبه می‌شود [۶،۴،۳]:

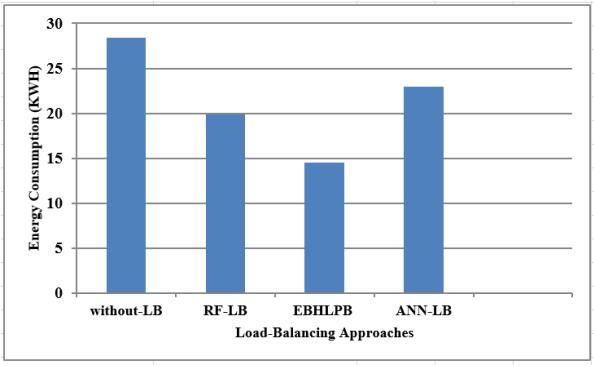
$$SLAV = SLATAH \times PDM \qquad (۱۶)$$

میزان نقض SLA در کل سیستم از ترکیب دو پارامتر بدست آمده در قسمت‌های قبل محاسبه می‌شود. با توجه به رابطه (۱۶) در اینجا میزان نقض SLA در رویکرد بدون توازن‌بار به دلیل صفر بودن PDM برابر صفر است. می‌توان گفت که این معیار برای سنجش میزان نقض SLA حاصل از مهاجرت VM است و نمی‌تواند در روش بدون توازن‌بار همراه با مهاجرت کارایی مناسبی داشته باشد. همانطور که درشکل ۱۵ دیده می‌شود، روش EBHLPB کاهش قابل توجهی نسبت به روش‌های RF-LB و ANN-LB داشته است.



شکل ۱۵: مقایسه میانگین SLAV

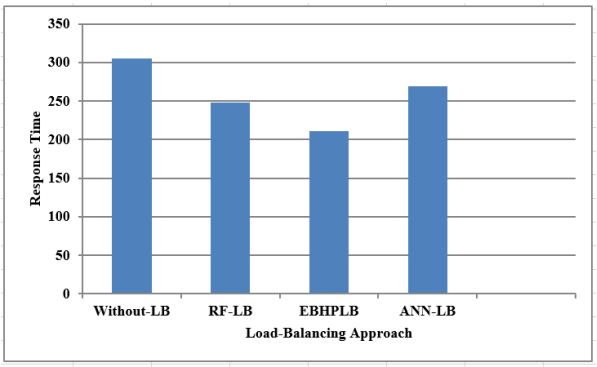
انرژی مصرفی: کمینه کردن میزان انرژی مصرفی، یکی از اهداف مهم الگوریتم‌های توازن‌بار است. انرژی مصرفی کل سیستم، معادل با انرژی مصرف شده توسط همه ماشین‌های فیزیکی موجود در سیستم است[۴]. در شکل ۱۶ نتایج حاصل از مقایسه مصرف انرژی نشان داده شده است. در روش بدون توازن‌بار، هیچ اقدامی برای ادغام VMها جهت خاموش کردن میزبان‌های کم‌بار و کم‌کردن بار میزبان‌های پربار انجام نمی‌شود، به همین دلیل، مصرف انرژی بسیار بالاست. روش RF-LB مصرف انرژی بیشتری نسبت به روش EBHLPB داشته است.



شکل ۱۶: مقایسه میانگین انرژی مصرفی

هرچه آستانه بالایی بیشتر باشد، مصرف انرژی در میزبان‌ها افزایش می‌یابد و چون به آستانه مورد نظر نرسیده‌اند، اقدامی برای متوازن کردن بار انجام نمی‌گیرد و بالا بودن بار آنها سبب کاهش کارایی سیستم می‌شود. از طرفی هر چه آستانه پایینی کمتر باشد، میزبان‌های بیشتری در محدوده عادی بشمار می‌روند، درحالی که می‌توانند به میزبان‌های دیگر منتقل شوند و میزبان خاموش شود. همچنین می‌توان این مصرف انرژی را ناشی از تشخیص اشتباه وضعیت میزبان‌ها و رسیدن میزبان‌ها به بهره‌وری صد درصد دانست. در روش EBHPLB، بار میزبان‌ها نسبت به روش RF-LB متوازن‌تر است، زیرا آستانه به صورت تطبیقی تنظیم شده است، همچنین در این روش، تعداد میزبان‌های خاموش بیشتری نسبت به روش پایه وجود دارد. به همین دلیل مصرف انرژی حدود ۳۰ درصد نسبت به روش RF-LB کاهش داشته است. روش ANN-LB یک روش واکنشی است و تعداد زیادی از میزبان‌ها حالت پربار یا کم‌بار بودن یا حتی میزان بهره‌وری صد در صد را تجربه می‌کنند، در حالی که روش‌های پیش‌دستانه قبل از ایجاد شرایط کم‌بار و پربار بر روی میزبان‌ها تصمیمات را اتخاذ می‌کنند. مقادیر مختلف انرژی در رویکردهای مختلف طی ۱۰ بار اجرای الگوریتم، در جدول ۲ نشان داده شده است.

زمان پاسخ کارها: یکی از پارامترهای SLA مدت زمان پاسخ به درخواست‌های کاربران است. پارامترهای متفاوتی بر زمان پاسخ کل کارهای موجود در یک سیستم، اثر گذارند. مانند مهاجرت VMها و مدت زمان پربار شدن و بهره‌وری صد در صد میزبان‌ها. با کاهش زمان پاسخ تعداد مشتریان موجود در سیستم را نیز می‌توان افزایش داد. از این رو الگوریتمی که زمان پاسخ کارها را کمینه کند، عملکرد بهتری دارد.



شکل ۱۷: مقایسه میانگین زمان پاسخ

نتایج حاصل از مقایسه زمان پاسخ روش‌ها در شکل۱۷ آمده است. مقادیر مختلف زمان پاسخ در رویکردهای مختلف طی ۱۰ بار اجرای الگوریتم، در جدول ۳ نشان داده شده است.

جدول ۲: مقادیر انرژی مصرفی در روش‌های مختلف

تکرار	Without-LB	RF-LB	EBHPLB	ANN-LB
1	28	19.9	14.52	23.2
2	28	19.8	14.51	23.2
3	28	19.7	14.45	23.3
4	28	19.8	14.53	23.35
5	28	19.85	14.55	23.21
6	28	19.7	14.53	23.35
7	28	19.74	14.62	23.19
8	28	19.8	14.53	23.26
9	28	19.9	14.51	23.34
10	28	19.8	14.49	23.28
Mean	28	19.799	14.52	23.268
ST.D	0	0.06789	0.04128	0.062097

۶- نتیجه گیری

پیش‌بینی تقاضای منابع در آینده، یک مسئله مهم در مدیریت کارآمد منابع در مراکز داده ابر است. از مسائل مهم مدیریت منابع ابر و ایجاد توازن‌بار در مراکز داده را، می‌توان تشخیص میزبان‌های پربار و کم‌بار، تعیین VMهای مناسب برای مهاجرت و انتخاب میزبان مناسب برای انتقال VMها به آنها، دانست. با توجه به اینکه مصرف CPU دارای بیشترین تأثیر در مصرف انرژی است، روش ارائه شده در این مقاله، سعی در پیش‌بینی دقیق مصرف CPU هر میزبان در آینده به وسیله روش ELM دارد، تا بتواند میزبان‌های پربار در آینده را به درستی تشخیص دهد و بار موجود بر روی منابع را مدیریت و توازن‌بار را به صورت پیش‌دستانه انجام دهد. این روش همچنین آستانه تطبیقی برای تعیین وضعیت میزبان‌ها و سیاست جایگذاری موثری برای مهاجرت VMها ارائه داده است. شبیه‌سازی در شبیه‌ساز Cloudsim و همچنین روی مجموعه داده UniLu-Gaia-2014 انجام شده است. ارزیابی انجام شده با روش‌های مورد مقایسه، نشان داده است که روش ارائه شده میزان زمان پاسخ کارها، میزان انرژی مصرفی و همچنین نقض SLA را کاهش داده است.

جدول ۳: مقادیر زمان پاسخ در روش‌های مختلف

تکرار	Without-LB	RF-LB	EBHPLB	ANN-LB
1	305.23	247.8	210.62	269.66
2	305.23	235.6	223.1	254.23
3	305.23	260.6	218.9	260.2
4	305.23	257.1	212.33	270.56
5	305.23	233.8	225.2	261.45
6	305.23	258.7	210.1	257.4
7	305.23	245.8	208.3	266.7
8	305.23	240.8	212.5	274.2
9	305.23	253.7	214.25	251.5
10	305.23	248.12	216.2	261.6
Mean	305.23	248.202	215.15	262.75
ST.D	0	8.945817	5.367502	7.019541

۷- مراجع

[1] A.Thakur and M.S. Goraya, "A taxonomic survey on load balancing in cloud", Journal of Network and Computer Applications, vol. 98, pp. 43-57, 2017.

[2] E. Jafarnejad Ghomi, A.M. Rahmani and N.N. Qader, "Load-balancing algorithms in cloud computing: A survey", Journal of Network and Computer Applications, vol. 88, pp. 50-71, 2017.

[3] S.B. Melhem, A. Agarwal, N. Goel and N. Zaman," Markov Prediction Model for Host Load Detection and VM Placement in Live Migration", IEEE Access, vol. 6, pp. 7190-7205,2018.

[4] A. Beloglazov and R. Buyya," Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers", Concurrency and Computation: Practice & Experience, vol. 24, pp. 1397-1420,2012.

[5] S.B. Melhem , A. Agarwal, N.Goel and M. Zaman, "Selection Process Approaches in Live Migration: A Comparative Study", 2017 8th International Conference on Information and Communication Systems (ICICS), pp. 23-28, 2017.

[6] A. Beloglazov, J. Abawajy J, R. Buyya. "Energy-aware resource allocation heuristics for efficient management of datacenters for cloud computing". Future Generation Computer Systems, vol. 28, pp. 755-768, 2011.

[7] A. Bala1, I. Chana, "Prediction-based proactive load balancing approach through VM migration", Engineering with Computers, vol. 32, pp. 581-592, 2016.

[8] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtualmachines in data centers," 39th IEEE Euromicro Conference Series on Software Engineering and Advanced Application, vol. , pp. 357-364, 2013.

[9] M. Sommer, M. Klink, S. Tomforde and J. Hähner, "Predictive load balancing in cloud computing environments based on ensemble forecasting", 2016 IEEE International Conference on Autonomic Computing (ICAC), pp. 300-307, 2016.

[10] M. Lavanya and V. Vaithiyanathan, "load prediction algorithm for dynamic resource allocation", Indian Journal of Science and Technology, vol.8, 2015.

[11] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu and H. Tenhunen, "Energy-aware VM consolidation in cloud data centers using utilization prediction model", IEEE Transactions on Cloud Computing, vol. 7, pp. 524-526, 2019.

[12] A.A. El-Moursy1, A. Abdelsamea , R.Kamran and M. Saad, " Multi-dimensional regression host utilization algorithm (MDRHU) for host overload detection in cloud computing", Journal of Cloud Computing: Advances, Systems and Applications,vol.8, 2019.

[13] D. Patel, R. Gupta, R.K. Pateriya, "Energy-Aware Prediction-Based Load Balancing Approach with VM Migration for the Cloud Environment". Data, Engineering And Applications, pp. 59-74, 2019.

[14] S. Ding, H. Zhao, Y. Zhang, X. Xu and R. Nie," Extreme learning machine: Theory and applications" , Artificial Intelligence Review, vol. 44, pp. 103-115, 2013.

[15] G.B. Huang, Q.Y. Zhu and C.K. Siew, "Extreme learning machine: Theory and applications", Neurocomputing, vol. 70, pp. 489-501, 2006.

[16] O.Ertugrul," Forecasting electricity load by a novel recurrent extreme learning machines approach "International Journal of Electrical Power \& Energy Systems, vol. 78, pp. 429-435, 2016.

[17] W.Voorsluys, J. Broberg, S. Venugopal, R. Buyya . "Cost of virtual machine live migration in clouds: a performance Evaluation", In Proceedings of the 1st International Conference on Cloud Computing (CloudCom), Vol. 2009. 2009.

پاورقی‌ها:

[1] Cloud computing
[2] Quality of Service
[3] Virtual Machine
[4] Service Level Agreement
[5] Overloaded Host Detection
[6] Million Instructions Per Second
[7] Underload Host Detection
[8] Migration
[9] Extreme Learning Machine
[10] k-nearest neighbors
[11] Support vector machine
[12] Neural networks
[13] Random forest
[14] Single Layer Feed Forward Neural Network
[15] Broker
[16] Reactive
[17] Robust statistics
[18] Root Mean Square Error
[19] Performance Degradation Due to Migration
[20] SLAV Time per Active Host