بهینه سازی مصرف انرژی در مراکز داده ابر نظامی با استفاده از الگوریتم رقابت استعماری و استراتژی تکامل

رضا عاصمی[1]، علیرضا قنبری[2]

[1] کارشناسی ارشد، مهندسی کامپیوتر، نرم افزار، مرکز مطالعات قرارگاه پدافند هوایی خاتم الانبیا(ص)،
R.asemi86@yahoo.com

[2] کارشناسی ارشد، علوم کامپیوتر، هوش مصنوعی، دانشگاه پدافند هوایی خاتم الانبیا(ص)،
ARGHanbari@gmail.com

چکیده

محاسبات ابری در سال‌های اخیر به عنوان یکی از مهم‌ترین تکنولوژی‌ها برای تحویل تقاضای سرویس‌های پیشرفته از طریق اینترنت عمومی، ضروری شده است و محاسبات ابری نظامی یکی از شاخه‌های آن می باشد که کاربردهای گوناگونی در علوم نظامی دارد. محاسبات ابری نظامی اولین بار با پروژه‌ای با نامMilitary Cloud در January 2012 مطرح شد. امروزه بهینه کردن مصرف انرژی در محاسبات ابری یکی از چالش‌های مهم ارائه دهندگان این سرویس می‌باشد. در روش پیشنهادی سعی شده است با تخصیص ماشین‌های مجازی به میزبان‌های فیزیکی تعداد میزبان‌های فیزیکی روشن کمینه شود و با این کار میزان مصرف انرژی در مراکز داده ابری کاهش یابد. در حالت کلی مسئله تخصیص بهینه ماشین های مجازی به میزبان های فیزیکی را می توان به سه زیر مسئله تقسیم نمود. زیر مسئله اول این است که چه زمانی باید ماشین مجازی مهاجرت کند؟ زیر مسئله دوم به اینکه کدام ماشین مجازی مهاجرت کند؟ می پردازد. زیر مسئله سوم به پرسش ماشین های مجازی به کجا مهاجرت کنند؟ باید پاسخ دهد. تمرکز اصلی این تحقیق بر روی حل زیر مسئله دوم می باشد. در این مقاله با استفاده از الگوریتم های تکاملی روشی مبتنی بر الگوریتم رقابت استعماری برای تخصیص بهینه ماشین های مجازی به میزبان های فیزیکی ارائه شده است. برای ارزیابی عملکرد راه حل ارائه شده، طیف وسیعی از راهکارهای ارائه شده برای زیر مسائل اول، دوم و روش پیشنهادی ترکیب و کارایی آنها مورد تحلیل قرار گرفته است. شبیه سازی روش پیشنهادی با استفاده از شبیه ساز CloudSim انجام شده و نشان می دهد که بکارگیری روش پیشنهادی موجب نقض شدن کمتر قراردادهای SLA نیز شده است. همچنین این روش موجب شده است تا مصرف انرژی توسط میزبان های فیزیکی نسبت به الگوریتم های مشابه 22,54 درصد کاهش داشته باشد.

کلمات کلیدی

محاسبات ابری، الگوریتم رقابت استعماری، ماشین مجازی، انرژی.

1- مقدمه

محاسبات ابری نظامی اخیراً به‌عنوان یک رویکرد برای تحویل دادن خدمات فناوری ارتباطات و اطلاعات در سازمان های نظامی توجه زیادی را به خود جلب کرده است. در سال های اخیر توسعه‌دهندگان خدمات فناوری اطلاعات، مثل آی‌بی‌ام، مایکروسافت، گوگل و سایر سازمان های بزرگ مشابه، مراکز داده را برای خدمات محاسبات ابری توسعه داده‌اند. همچنین مراکز داده دارای هزاران سرویس‌دهنده و سوئیچ می‌باشد. از طرف دیگر به دلیل گرمای تولید شده توسط این مراکز داده، نیاز به تجهیزات خنک‌کننده می باشد. در حقیقت مراکز داده در محاسبات ابری مقدار زیادی از انرژی الکتریکی را مصرف می‌کنند و این به نوبه خود باعث افزایش هزینه‌های عملیاتی و تولید دی اکسید کربن در محیط پیرامون ما می‌شود. در نتیجه برای کاهش انتشار دی

اکسید کربن در محیط اطراف و برای داشتن محاسبات سبز نیاز به تکنیک هایی برای کاهش مصرف انرژی در مراکز داده بیش از پیش احساس می‌شود. سرورها در زمان های بیکاری که روشن هستند 70 درصد از انرژی که در زمان اوج کار مصرف می کنند را مصرف می کنند. بنابراین روشن بودن سرور با حجم بار کم اقتصادی به نظر نمیرسد. با تخصیص بهینه ماشین های مجازی به میزبان های فیزیکی می توان تعداد میزبان های فیزیکی روشن را کمینه کرد و از این طریق مصرف انرژی را در دیتاسنترها کاهش داد. ولی باید این عمل باعث نقض شدن قراردادهای SLA یا پایین آمدن کیفیت سرویس نشود. عمل تخصیص ماشین های مجازی به میزبان های فیزیکی از طریق یکی از اعضای کلود به نام دیتا براکر انجام می گیرد[2].

برای پاسخ به مسئله تخصیص بهینه ماشین های مجازی به میزبان های فیزیکی باید به چندین زیر مسئله پرداخت. تحقیق [3] چالش های اصلی این تخصیص بهینه را به سه زیر مسئله تقسیم کرده است.

1-1 چه زمانی باید ماشین مجازی مهاجرت کند؟

در طول عمر یک مرکز داده ابری به صورت مکرر نیاز است که ماشین های مجازی از یک میزبان فیزیکی به میزبان فیزیکی دیگری منتقل شود. که در زیر مسئله اول در مورد زمان های این مهاجرت ها بحث می شود.

2-1 کدام ماشین مجازی مهاجرت کند؟

پس از آنکه در زیر مسئله اول زمان مهاجرت تشخیص داده شد. باید تصمیم گرفت که کدام یک از ماشین های مجازی برای انتقال داده شدن به ماشین فیزیکی جدید مناسب است که در این زیر مسئله در این خصوص بحث می شود.

3-1 ماشین های مجازی کجا مهاجرت کنند؟

تعیین محل جدید و مناسب برای ماشین های مجازی یا به عبارتی مشخص کردن میزبان های فیزیکی مناسب برای هر کدام از ماشین های مجازی پس از اینکه ماشین های مجازی مناسب توسط زیر مسئله دوم برای مهاجرت انتخاب شد چالش زیر مسئله سوم است. مسئله تخصیص ماشین های مجازی به میزبان های فیزیکی یا به عبارتی زیر مسئله سوم شباهت زیادی به مسئله کلاسیک Bin Packing دارد که یک مسئله NP-Hard محسوب می شود[4].

در این مقاله زیرمسئله دوم با استفاده از الگوریتم ICA-ES [1] حل شده است. الگوریتم ICA-ES ترکیبی از الگوریتم رقابت استعماری و استراتژی تکامل می باشد. در این روش در ابتدا به الگوریتم استاندار د عملگر جهش به منظور فرار از بهینه های محلی اضافه شده است و با استفاده از استراتژی تکامل چگونگی تغییرات نرخ جهش به صورت تطبیقی توسط بازخورد از فضای مسئله تغییر می یابد. در واقع روش تکامل یافته ی رقابت استعماری، موازنه ای میان جست و جوی سراسری و محلی در الگوریتم رقابت استعماری برقرار می کند. نتایج بدست آمده کاهش 22,54 درصدی انرژی را نسبت به الگوریتم های PSO و ژنتیک نشان می دهد. هدف این مقاله ارائه یک راه حل مبتنی بر الگوریتم ICA-ES برای زیر مسئله دوم است تا تعداد میزبان های فیزیکی روشن کمینه شود و مصرف انرژی کاهش یابد. همچنین در این مقاله بررسی شده است که از بین روش های پرکاربرد و شناخته شده برای زیر مسئله اول کدام روش در کنار روش پیشنهادی برای زیر مسئله دوم مناسب تر است. کارهای انجام شده در این تحقیق به صورت خلاصه وار شامل مراحل زیر می باشد:

- استفاده از الگوریتم ICA_ES برای حل مسئله جا گذاری ماشین ها مجازی در میزبان های فیزیکی با هدف کاهش مصرف انرژی
- تعیین مناسب ترین روش از بین روش های موجود برای زیرمسئله تشخیص سرریز بار میزبان های فیزیکی
- تعیین مناسب ترین روش از بین روش های موجود برای تشخیص سرریز در میزبان های فیزیکی

ادامه ساختار این مقاله بصورت زیر است. در بخش 2 به مرور کارهای پیشین پرداخته شده و در بخش 3 مسئله به صورت دقیق بیان شده است. در بخش 4 راهکار پیشنهادی شرح داده خواهد شد. معرفی شبیه ساز، پارامترهای ارزیابی و تنظیمات مربوط به شبیه سازی در بخش 5 آورده شده است. بخش 6 به تحلیل کارایی و عملکرد روش پیشنهادی اختصاص یافته و در پایان نیز جمع بندی این تحقیق ارائه شده است.

2- مرورکارهای پیشین

اختصاص دادن بهینه ماشین های مجازی به میزبان های فیزیکی به سه زیر مسئله تقسیم می شود. در این بخش به مرور برخی از راهکارهای ارائه شده برای هر یک از این زیر مسائل پرداخته خواهد شد.

2- 1چه زمانی باید ماشین مجازی مهاجرت کند؟

زیر مسئله اول مربوط به زمان مهاجرت ماشین مجازی به یک میزبان فیزیکی دیگر است. در حالت کلی این جابجایی در 2 وضعیت باید اتفاق بیافتد. وضعیت اول مربوط به زمانی است که میزبان دچار Hostoverload شده است. به عبارت دیگر زمانی که بار یک میزبان فیزیکی بیش از آستانه تعیین شده برای آن میزبان باشد، به منظور جلوگیری از احتمال نقض شدن قراردادهای SLA* بدلیل کم بودن منابع میزبان فیزیکی باید ماشین مجازی به میزبان دیگر انتقال یابد. دومین وضعیتی که در آن ماشین مجازی باید مهاجرت نماید مربوط به حالتی است که در آن میزبان دچار Host underload شده است. این حالت زمانی اتفاق می افتدکه بار پردازشی یک میزبان به میزانی کاهش می‌یابد و می‌توان کل توان پردازشی آن یعنی ماشین‌های مجازی آن را به یک یا چند میزبان فیزیکی روشن انتقال داد و آن را خاموش کرد. از الگوریتم های موجود برای تشخیص سربار میزبان های فیزیکی می توان به الگوریتم های زیر اشاره کرد.

- **الگوریتم رگرسیون‌محلی (LR) [6]**

الگوریتم LR براساس روش leoess ای است که Cleveland در[5] پیشنهاد داده است. ایده اصلی برای روش رگرسیون محلی، مدل های اتصالات ساده برای زیرمجموعه های محلی داده به منظور ساختن یک منحنی تقریبی برای داده های اصلی است. مشاهدات (xi,yi) وزن همسایه ها را با استفاده از تابع وزن tricube تخصیص داده است که در رابطه (1) نشان داده شده است.

$$[6]\quad T(u) = \begin{cases} (1-|u|^3 \; if \; |u| < 1, \\ 0 \qquad Otherwise \end{cases} \qquad (1)$$

* Service-level agreement
† Local Regression algorithm

- **الگوریتم انحراف مطلق میانه (MAD) [7]**

MAD یک تحلیل آماری قوی است که بیشترین انعطاف پذیری را به نقاط دور افتاده در یک مجموعه را نسبت انحراف استاندارد دارا می باشد. در انحراف استاندارد فاصله از میانه به توان می رسد که منجر به افزایش انحراف می شود که برای متوسط وزن سنگین است. این به این معنی است که نقاط دورافتاده به طور قابل توجهی ارزش انحراف استاندارد را تحت تاثیر قرار می دهد. در MAD مقدار فاصله از نقاط دور افتاده بی ربط است. برای یک مجموعه داده مجموعه تک متغیری X_1، X_2 و ... X_n به عنوان متوسط انحراف مطلق از میانگین مجموعه داده ها تعریف می شود. نحوه محاسبه MAD در رابطه 2 نشان داده شده است.

$$[6]\ MAD = Median_i(|X_i - Median_j(X_j)|) \quad (2)$$

MAD متوسط ارزش انحراف مطلق (باقیمانده) از متوسط داده است. در الگوریتم تشخیص اضافه بار MAD، آستانه استفاده از پردازنده با T_u مشخص می شود که در(3) نشان داده شده است

$$[6]\ T_u = 1 - S.MAD \quad (3)$$

- **الگوریتم رگرسیون‌محلی‌مقاوم (LRR) [6]**

نسخه loess که پیشتر شرح داده شد به نقاط دور افتاده آسیب پذیر است که می توانند توسط توزیع heavy-tailed یا leptokurtic ایجاد شوند. برای ایجاد یک loess قوی، Cleveland یک روش تخمین قوی دیگر به روش حداقل مربعات برای مناسب بودن برای یک خانواده پارامتری پیشنهاد داد.

این تغییر loess را به یک روش تکرار شونده انتقال داد. مناسب بودن اولیه با وزن تعریف شده توسط تابع وزن tricube انجام شده است. مناسب بودن ارزیابی در xi برای رسیدن به مقدار مناسب byi و باقیمانده bei=yi است. در مرحله بعد، هر (xi,yi) یک وزن زیاد ri را تخصیص داده است ، که مقدار آن به مقدار bei بستگی دارد و همچنین وزن riwi(x) را که مقدار ri در (4) تعریف شده است.

$$[6]\ r_i = B\left(\frac{\hat{\epsilon}_i}{6s}\right) \quad (4)$$

- **الگوریتم الگوریتم برد میان چارکی (IQR)** [7]:

در آمار توصیفی، دامنه میان چارکی (IQR) همچنین midspread یا متوسط پنجاه نامیده می شود، یک ابزار اندازه گیری برای پراکندگی آماری است. که آن مساوی تفاوت بین سومین و اولین چارک است.که در (5) نشان داده شده است.

$$[7]\ IQR=Q3-Q1 \quad (5)$$

برخلاف (کل) دامنه؛ دامنه میان چارکی یک چارک آماری قوی است با داشتن یک نقطه شکست 25% و از این رو اغلب کل دامنه ترجیح داده می شود. برای یک توزیع متقارن(به عنوان مثال، به طوری که متوسط برابر میانگین چارک های اول و سوم است)، نصف IQR مساوی است با MAD. با استفاده از IQR، به طور مشابه با (4)، آستانه بهره برداری CPU در (6) تعریف شده است.

$$[7]\ T_u = 1 - S.IQR \quad (6)$$

2-2 کدام ماشین مجازی مهاجرت کند؟

پس از آنکه زمان مهاجرت تشخیص داده شد، اگر رویداد کم‌کاری سربار میزبان فیزیکی اتفاق افتاده باشد، باید تمامی ماشین‌های مجازی مهاجرت کنند تا میزبان فیزیکی خاموش شود و اگر سریز سربار میزبان فیزیکی اتفاق افتاده باشد، زیر مسئله دوم روی می دهد که باید مشخص کنیم کدام ماشین-های مجازی از میزبان فیزیکی مورد نظر مهاجرت داده شوند. سه سیاست مطرح که از آنها برای انتخاب ماشین های مجازی از روی میزبان های فرستنده در وضعیت استفاده بیش از حد استفاده می شود، به شرح زیر هستند[7].

- **به حداقل رساندن مهاجرت(MMT) [7]**

انتخاب ماشین‌های مجازی که مهاجرت آنها به حداقل زمان نیاز دارد، برای به حداقل رساندن سربار حاصل از مهاجرت، از آنجایی که ماشین مجازی سریعتر می تواند مهاجرت کند که Memory و CPU کمی داشته باشد در این روش در واقع ماشین های مجازی کوچک جهت مهاجرت انتخاب می شوند و این سیاست باعث می شود اگر برای خارج شدن از حالت سریزبار نیاز آزادسازی زیاد حافظه یا پردازنده نباشد این روش عملکرد خوبی داشت باشد ولی اگر خلاف این باشد، یعنی برای خارج شدن از حالت سربار نیاز به آزادسازی زیاد باشد در این روش باید تعداد زیادی از ماشین های مجازی مهاجرت داده شوند که در این صورت روش MMT کارائی خوبی نخواهد داشت.

- **انتخاب تصادفی(RS)[2]**

مهاجرت تعداد لازم از ماشین های مجازی، که آنها بر اساس یک متغیر تصادفی با توزیع یکنواخت بسته بندی شده اند. این روش برای مراکز داده ای که تعداد ماشین های مجازی زیادی داشته باشد یا به عبارتی مرکز محاسبات ابر عمومی می تواند یک راه کار مناسب باشد.

3-2 ماشین های مجازی کجا مهاجرت کنند؟

زیر مسئله سوم در مورد سوال "ماشین های مجازی کجا مهاجرت کنند؟" تحقیق می‌کند. پس از تشخیص زمان مهاجرت و انتخاب ماشین‌های مجازی که باید مهاجرت کنند، زیر مسئله سوم مطرح می شود که باید مقصد هر یک از ماشین های مجازی را که کاندید مهاجرت هستند مشخص کند. به الگوریتم‌هایی که برای این منظور استفاده می‌شود الگوریتم‌های جایگذاری ماشین‌های مجازی گفته می‌شود. زیاد بودن تعداد میزبان‌های فیزیکی و ماشین‌های مجازی باعث شده است که ایده استفاده از الگوریتم‌های تکاملی برای مسئله جایگذاری ماشین‌های مجازی به میزبان‌های فیزیکی قدرت

‡ Median Absolute Deviation algorithm
§ Robust Local Regression algorithm
** Interquartile Range algorithm

گرفته شود. مقاله [8] الگوریتم جایگذاری ماشین‌های مجازی مبتنی بر هوش توده‌ای را ارائه کرده است و یک راه حل خود تطبیقی مبتنی بر الگوریتم بهسازی ذرات ارائه داده است والگوریتم خود را با عنوان SAPSO معرفی کرده است یکی از ویژگی های این روش که باعث شده نسبت به روش های مشابه خود به نتایج بهتری برسسد ویژگی خود تطبیقی آن می باشد که باعث می شود الگوریتم در هر لحظه نسبت به آن شرایط تصمیم بگیرد که ماشین مجازی مورد نظر بر روی کدام میزبان فیزیکی قرار بگیرد. مقاله [2] یک روش کلاسیک با عنوان PABFD [‡‡] ارائه کرده است. این روش همواره سعی دارد میزبان فیزیکی را انتخاب کند که با تخصیص ماشین مجازی به آن کمترین افزایش در مصرف انرژی آن میزبان فیزیکی به وجود آید. این روش یکی از متداولترین روش هایی می باشد که تا کنون مورد استفاده قرار گرفته است. مقاله [9] یک راهکار مبتنی بر الگوریتم ژنتیک ارائه داده است. در راهکار ارائه شده در این مقاله سعی شده است بوسیله یک تابع هدف مناسب مصرف انرژی برای پاسخ های احتمالی سنجیده شود. همچنین یکی از مزیت های این روش این می باشد که سعی دارد مصرف انرژی را هم در سرورها کاهش دهد و هم در ارتباطات شبکه ای. در [10] با اشاره به اینکه بهبود مصرف انرژی به امری مهم در مراکزداده‌ها در سال های اخیر تبدیل شده است الگوریتمی مبتنی بر سرد کردن فلزات پیشنهاد شده است که هدفش بهبود جایگذاریهای ماشین های مجازی به میزبان های فیزیکی است. در واقع با استفاده از الگوریتم SA زیر مسئله سوم حل شده است SA جستجو را در Stateهای همسایه‌هایش برای رسیدن به جواب های بهتر انجام می‌دهد. در هر مرحله از تکامل، پیکربندی نیاز به تغییر وضعیت همسایه دارد. نتایج تجربی الگوریتم نشان می‌دهد که این الگوریتم SA می‌تواند نتایج بهتری را تولید کند.

از معایب کلی برای الگوریتم های تکاملی چون ژنتیک، PSO ، SA و... همچمون وابسته بودن به مقدار اولیه پارامترها در الگوریتم SA، به کار گیری بیشتر برای مسائل نامحدود در الگوریتم PSO، انتخاب مقدار نامناسب برای مقدار اولیه و افتادن در بهینه های محلی در اکثر الگوریتم های تکاملی اشاره کرد. اگرچه الگوریتم ژنتیک با استفاده از خاصیت جست و جوی Global تا حدی توانسته است از بهینه های محلی فرار کند، اما این الگوریتم نیز به علت تولید هزاران کروموزوم هزینه و زمان اجرای زیادی دارد. استفاده از الگوریتم رقابت استعماری ضمن پوشش دادن معایب شرح داده شده، از مزایای زیر نیز برخوردار می باشد:

- نو بودن ایده.
- مبتنی بر رفتار اجتماعی انسان که هوشمندانه تر از رفتار های بیولوژیکی است.
- سرعت همگرایی بالا.
- توانایی بهینه سازی توابعی با تعداد متغیر های بسیار زیاد.

3- صورت مسئله

در این تحقیق به حل زیر مسئله انتخاب ماشین های مجازی به نحوی که حداقل یک ماشین مجازی و حداکثر تمام ماشین های مجازی یک میزبان فیزیکی می تواند برای انتقال انتخاب شود و تعداد کمینه‌ای از میزبان‌های فیزیکی روشن‌باشند، پرداخته شده است. به عبارت دیگر یک مرکز داده با M ماشین مجازی و N میزبان فیزیکی در نظر گرفته می شود (M>N). V به عنوان مجموعه ماشین های مجازی متشکل M ماشین تعریف می گردد که در آن v_i یک ماشین مجازی نمونه است. همچنین P را بصورت مجموعه ای میزبان های فیزیکی که در آن p_j یک میزبان فیزیک نمونه می باشد، نشان می دهیم.

$$V=\{v_1,v_2,\ldots,v_m\} \qquad (7)$$

$$P=\{p_1,p_2,\ldots,p_n\} \qquad (8)$$

میزان پردازنده مورد نیاز ماشین مجازی i را با V_i^{Cpu} و میزان حافظه مورد نیاز برای آن را با V_i^{mem} نشان می دهیم و P_j^{Cpu} بیانگر توان پردازنده ماشین فیزیکی j ، P_j^{mem} بیانگر میزان حافظه ماشین فیزیکی j می باشند و P_j^{wcpu} و P_j^{wmem} به ترتیب نمایانگر مجموع کل بارکاری پردازنده و مجموع حافظه اصلی استفاده شده میزبان فیزیکی Pj می باشند. همچنین می توان مجموعه ماشین های مجازی تخصیص داده شده به میزبان های فیزیکی را با مجموعه VP نشان داد. اندیس نشان گر شماره ماشین مجازی و عدد کنار P نمایانگر شماره میزبان فیزیکی مورد نظر می باشد.

$$Vp=\{p2, p5,\ldots, pm\} \qquad (9)$$

نرخ بهره وری پردازنده در میزبان فیزیکی Pj براساس رابطه 10 بدست می آید. از این‌رو می توان مصرف انرژی میزبان فیزیکی را با استفاده از رابطه 11 تخمین زد[1].

$$\mu j = P_j^{wcpu} / P_j^{cpu} \qquad (10)$$

در رابطه 10، P_j^{wcpu} مجموع کل بار کاری پردازنده میزبان فیزیکی pj و P_j^{Cpu} بیانگر توان پردازنده ماشین فیزیکی j را نشان می دهد.

$$E(p_j) = k_j . e_j^{max} + (1 - k_j) . e_j^{max} . \mu_j \qquad (11)$$

در دو رابطه(10) و (11) k_j برابر است با میزان مصرف انرژی میزبان فیزیکی pj در زمان بیکاری، e_j^{max} مصرف انرژی میزبان فیزیکی p_j در اوج بار پردازشی، و μj میزان کاروری پردازنده میزبان فیزیکی P_j را نشان می‌دهد. هدف این تحقیق مشخص کردن یک میزبان فیزیکی برای هر یک از ماشین های مجازی با استفاده از روابط فوق است به نحوی که منجر به کاهش میزان مصرف انرژی گردد.

4- روش پیشنهادی

در این بخش روش پیشنهادی شرح داده شده است به این ترتیب که در ابتدا توضیحات مقدماتی در مورد الگوریتم رقابت استعماری آورده شده است، سپس استراتژی تکامل به صورت مختصر معرفی شده و پس از آن الگوریتم ICA-ES [1] که حاصل ترکیب الگوریتم رقابت استعماری و استراتژی تکامل است شرح داده شده است و در انتها روش حل مسئله انتخاب ماشین های مجازی با الگوریتم ICA-ES ارائه شده است.

[††] Particle swarm optimization

[‡‡] Power-Aware Best Fit Decreasing algorithm

[§§] Simulated Annealing

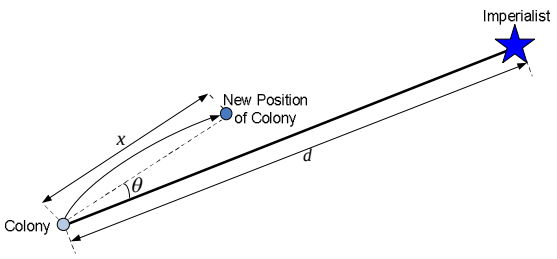
1-4 مقدمه ای بر الگوریتم رقابت استعماری

الگوریتم رقابت استعماری(ICA) یکی از الگوریتم های جدید در زمینه محاسبات تکاملی است که بر مبنای تکامل اجتماعی-سیاسی انسان پایه گذاری شده است[11]. همانند دیگر الگوریتم های تکاملی، این الگوریتم، نیز با تعدادی جمعیت اولیه تصادفی که هر کدام از آنها یک "کشور" نامیده می شوند و به دو نوع مستعمره و امپریالیست تقسیم بندی می شود که با همدیگر تشکیل یک امپراطوری جدید را می دهند. گام های الگوریتم رقابت استعماری عبارتند از:

1) چند کشور در فضای مسئله انتخاب کرده و امپراطوری‌های اولیه را تشکیل بده.
2) مستعمرات را به سمت کشور امپریالیست حرکت بده (سیاست همسان‌سازی).
3) اگر مستعمره‌ای در یک امپراطوری، وجود داشته باشد که هزینه‌ای کمتر از امپریالیست داشته باشد؛ جای مستعمره و امپریالیست را با هم عوض کن.
4) هزینه‌ی کل یک امپراطوری را حساب کن (با در نظر گرفتن هزینه‌ی امپریالیست و مستعمراتشان).
5) یک مستعمره از ضعیف‌ترین امپراطوری انتخاب کرده و آن را به امپراطوری‌ای که بیشترین احتمال تصاحب را دارد، بده. این گام به عنوان گام کشش شناخته می شود که در شکل 1 نمایش داده می شود.
6) امپراطوری‌های ضعیف را حذف کن.
7) اگر تنها یک امپراطوری باقی مانده باشد، توقف کن وگرنه به 2 برو.

رقابت استعماری بین امپراطوری های الگوریتم تکاملی پیشنهادی را شکل می هد. در طول این رقابت امپراطوری های ضعیف فروپاشی می شوند و آنهایی که قدرمتند هستند برای خود مستعمره اختیار می کنند. رقابت استعماری همگرا به یک کشور است که در آن تنها یک امپراطوری وجود دارد و مستعمرات همان مقدار تابع هزینه را مانند امپراطوری دارند.

بعد از تقسیم همه مستعمره ها بین امپراطوری ها و ایجاد امپراطوری اولیه، این مستعمرات شروع به حرکت به سمت امپراطوری مربوطه می کنند براساس قانون جذب می کنند. شکل 1 حرکت یک مستعمره را به سمت یک امپراطوری نشان می دهد. مطابق این شکل کشور امپریالیست کشور مستعمره را در راستای محورهای فرهنگ و زبان به سمت خود جذب می‌کند. همانگونه که در این شکل نشان داده شده است، کشور مستعمره (Colony)، به اندازه x واحد در جهت خط واصل مستعمره به استعمارگر (Imperialist)، حرکت کرده و به موقعیت جدید (New Position of Colony)، کشانده می‌شود. در این شکل، فاصله میان استعمارگر و مستعمره با d نشان داده شده است. x نیز عددی تصادفی با توزیع یکنواخت (و یا هر توزیع مناسب دیگر) می‌باشد. یعنی برای x داریم: $x \sim U(0, \beta \times d)$ که در آن β عددی بزرگتر از یک و نزدیک به 2 می‌باشد. یک انتخاب مناسب می‌تواند $\beta = 2$ باشد. وجود ضریب $\beta > 1$ باعث می‌شود تا کشور مستعمره در حین حرکت به سمت کشور استعمارگر، از جهت‌های مختلف به آن نزدیک شود.



شکل 1 حرکت مستعمره به سمت امپراطوری

در رقابت استعماری، همه امپراطوری ها تلاش می کنند مستعمرات دیگر امپراطوری ها را جذب کرده و آنها را کنترل کنند. این رقابت رفته رفته کاهش قدرت را برای امپراطوری ها ضعیف و افزایش قدرت را برای امپراطوری های قدرتمند به ارمغان می آورد. این کار با برداشتن تعدادی از مستعمرات ضعیف از امپراطوری های ضعیف و ایجاد یک رقابت بین همه ی امپراطوری ها مدل می شود. براساس قدرت کل آنها، در این رقابت، هرکدام از امپراطوری ها احتمالی برای گرفتن موقعیت جذب مستعمرات ذکر شده را دارد. قویترین امپراطوری، بیشترین احتمال را برای جذب مستعمرات خواهد داشت. به عبارت دیگر این مستعمرات دقیقا جذب بیشترین قویترین امپراطوری نمی شوند، اما این امپراطوری ها احتمال بیشتری برای جذب آنها خواهد داشت. هر امپراطوری که قادر نیست در رقابت استعماری موفق شود و قادر نیست قدرتش را افزایش دهد (یا حداقل از کاهش قدرتش جلوگیری نماید) حذف خواهد شد. الگوریتم رقابت استعماری تا کنون برای حل مسائل مختلفی استفاده شده و به نتایج قابل قبولی دست یافته است مانند[۱۳،۱۴،۱۵].

ولی در بعضی مواقع دیده می شود مانند سایر الگوریتم های تکاملی در بهینه های محلی گرفتار می شود هر چند نسبت به الگوریتم های مشابه کمتر است. برای آنکه احتمال گرفتار شدن این الگوریتم را در بهینه های محلی را کاهش داد می توان از روش ارائه شده در [1] استفاده کرد. این تحقیق با استفاده از نظریه استراتژی تکامل الگوریتم رقابت استعماری را بهینه تر کرده است.

2-4 نظریه استراتژی تکامل

استراتژی تکامل با استفاده از مبانی نظری محکمی توسط دو محقق آلمانی به صورت بازنمایی حقیقی معرفی و توسعه یافت و به طور وسیعی در حل مسائل بهینه سازی مورد توجه واقع شد. علاوه بر تفاوت بازنمایی در این الگوریتم و الگوریتم ژنتیک می توان به تنها عملگر تغییر در این الگوریتم به صورت جهش تطبیقی اشاره نمود که یکی از مهمترین ویژگی های این الگوریتم محسوب می شود. تا به حال از استراتژی تکامل نسخه های مختلفی ارائه شده است و از ترکیب آن الگوریتم جدیدی نیز ارائه شده است. در این الگوریتم از قدرت استراتژی تکامل در بهبود الگوریتم رقابت استعماری استفاده شده است[9].

3-4 الگوریتم ICA-ES

چنانچه ذکر شد در الگوریتم رقابت استعماری راه حل های فضای مسئله به صورت کشور در نظر گرفته می شود. در الگوریتم ICA-ES که در [1] شرح داده شد در ابتدای هر جابجایی یک جهش نیز وجود دارد تا از افتادن در بهینه های محلی جلوگیری نموده و به صورت تطبیقی کشورها را به نقاط بهینه

نزدیک نماید. از آنجا که جابجایی کشورهای مستعمره تابعی از کشورهای استعمارگر می باشد. بنابراین جهش صرفا بر روی کشورهای استعمارگر و قبل از همسان سازی کشورهای مستعمره صورت می پذیرد. عملگر جهش در استراتژی تکامل با جمع برداری با توزیع نرمال هر راه حل صورت می پذیرد که در اینجا جمع برداری با توزیع نرمال هر استعمارگر خواهد بود که از رابطه (10) استفاده می گردد.

$$C'_i = C_I + N(0,\sigma) \qquad (12)$$

به طوریکه در رابطه (12)، C'I کشور استعمارگر تکامل یافته، Ci کشور اولیه استعمارگر و σ_i اندازه گام جهش می باشد. بنابراین با توضیحات اراده شده می توان ساختار کلی الگوریتم پیشنهادی را به صورت زیر در نظر گرفت.

1) مقدار دهی اولیه کشورها و نرخ جهش
2) ارزیابی کشورها و تعیین کشورهای استعمارگر و مستعمره
3) تعیین نرخ جدید جهش و انجام جهش تکاملی بر روی کشورهای استعمارگر
4) حرکت کشورهای مستعمره به سمت کشورهای استعمارگر
5) تغییر استعمارگر و مستعمره در صورت لزوم
6) محاسبه ارزش امپراطوری
7) ضعیفترین مستعمره از ضعیفترین امپراتوری گرفته شودو به قوی ترین امپراطوری همسایه داده شود.
8) اگر امپراطوری بدونه مستعمره وجود دارد نابود شود.
9) تا رسیدن به شرایط توقف بازگشت به گام 3

4-4 حل مسئله انتخاب ماشین های مجازی با الگوریتم ICA-ES

روش پیشنهادی با تولید n امپراتوری و m کشور برای هر امپراتوری به‌صورت تصادفی که هر کشور به‌عنوان یک جواب مسئله می‌باشد، الگوریتم شروع می شود. در نتیجه nm جواب مختلف برای مسئله تولید خواهد شد. در مرحله بعد در هر امپراتوری ارزش هر کشور (جواب مسئله) با استفاده از رابطه 13 اندازه‌گیری می شود و براساس برازندگی بدست آمده کشورهای استعمارگر و مستعمره هر امپراطوری تعیین می شود. E(pj) به کار رفته در رابطه 13 قبلا تشریح شده است.

$$fitness = \frac{1}{\sum_1^N E(p_j)} \qquad (13)$$

کشوری با بالاترین مقدار برازندگی در هر امپراتوری به‌عنوان استعمارگر و بقیه $m-1$ کشور به عنوان مستعمره انتخاب می‌شوند. در مرحله بعد، عملگر جذب را بر روی کشورهای هر امپراتوری اعمال می‌کنیم به این صورت که به‌صورت تصادفی β درصد از اندیس های استعمارگر را انتخاب (که بهتر است میزان β پایین در نظر گرفته شود تا از همگرایی به یک باره مستعمرات به سمت استعمارگر و گیرکردن در بهینه محلی جلوگیری شود) و با اندیس های معادل آن در مستعمرات جایگزین می‌کنیم. سپس دوباره میزان برازندگی جواب‌ها را اندازه‌گیری می‌کنیم تا اگر تغییری در امپراتوری‌ها صورت گرفته بود (بهتر شدن مستعمره از استعمارگر و تعویض جای این دو) در الگوریتم اعمال شود. پس از همگرایی نوبت به انقلاب می‌رسد در این مرحله درصد کمی از کشورها (میزان 10 درصد مناسب است) به‌صورت تصادفی انتخاب در یک اندیس آنها تغییر کوچکی (میزبان گیرنده ماشین مجازی مورد نظر تغییر می‌کند) اعمال می‌شود. پس از انقلاب دوباره برای تست تغییرات در هر امپراتوری برازندگی کشورها اندازه‌گیری می‌شود. در مرحله بعد قدرت هر امپراتوری محاسبه‌شده و ضعیف‌ترین امپراتوری برای از دست دادن یکی از مستعمرات خود انتخاب می‌شود.سپس یک امپراتوری برای به‌دست آوردن مستعمره ضعیف‌ترین امپراتوری انتخاب و مستعمره مورد نظر به‌عنوان مستعمره جدید به امپراتوری منتخب منتقل می‌شود. در مرحله بهبود، θ و x شماره های تصادفی هستند با توزیع یکنواخت، همانطور که در رابطه 14 نشان داده شده است و d فاصله بین مستعمره و امپراطوری است.

$$x \sim U(0,\beta \times d),\ \theta \sim U(-\gamma,\gamma) \qquad (14)$$

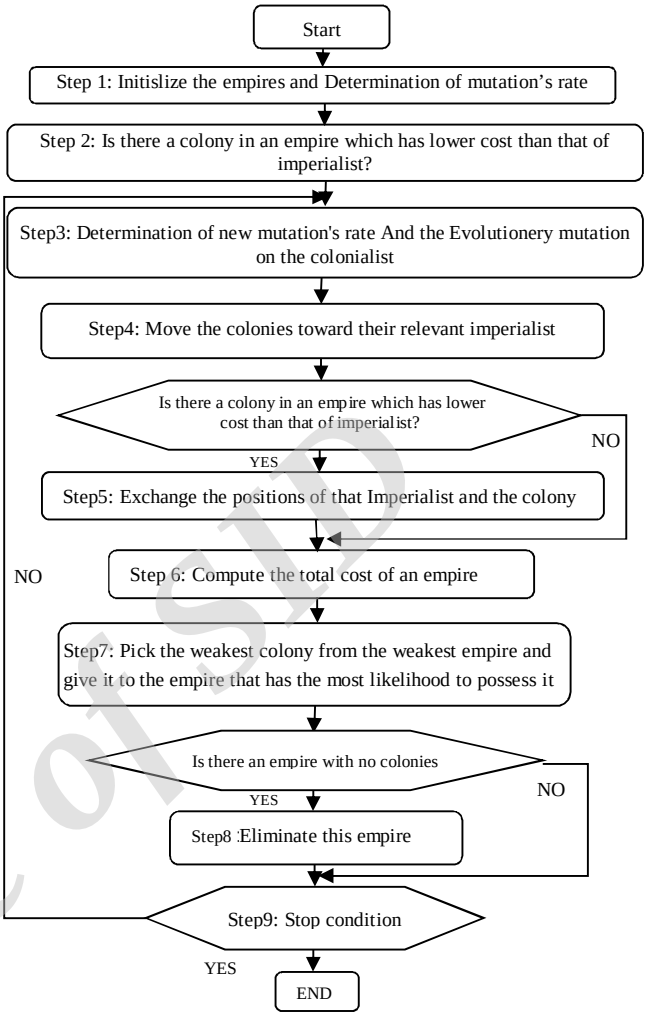
در حالیکه β وγ پارامترهایی برای تغییر ناحیه ای هستند که مستعمرات به طور تصادفی جستجو می کنند در حول امپراطوری. در این تحقیق β و γ عنوان 2 و 0,5 (رادیان) به ترتیب در نظر گرفته شده اند. در ICA، انقلاب باعث می شود یک کشور به صورت اتفاقی خصوصیات سیاستی-اجتماعی خودش را تغییر دهد. این است، مقابل آن چه که توسط یک امپراطوری تخمین زده شده است، که یک مستعمره به صورت تصادفی موقعیت خود را به محور سیاسی-اجتماعی تغییر می دهد. انقلاب باعث افزایش جست و جوی الگوریتم می شود و از همگرایی زودهنگام کشورها به مینیمم محلی جلوگیری می کند. قدرت کلی یک امپراطوری هم بستگی به قدرت کشور امپراطوری و هم بستگی به قدرت مستعمرات آن دارد که در فرمول 15 نشان داده شده است.

$$T.C._n = Cost(imperialist_n) + \xi mean\{Cost(colonies\ of\ empire_n)\} \qquad (15)$$

رابطه 15 قدرت کل یک امپراطوری را به صورت مجموع قدرت کشور استعمارگر به اضافه درصدی از قدرت میانگین مستعمرات آن بیان می کند. در واقع با قوی تر شدن مستعمرات هر امپراطوری قدرت امپراطوری هم افزایش می یابد. این الگوریتم تا برقرار شدن یکی از شروط پایانی که در زیر به آنها اشاره شده تکرار می‌شود:

1. در پایان پس از چند دهه (تکرار الگوریتم) *** از میان امپراتوری های باقی‌مانده بهترین استعمار گر انتخاب و به‌عنوان جواب نهایی مسئله نمایش داده می‌شود.
2. تا زمانی که تنها یک امپراتوری باقی بماند الگوریتم اجرا می‌شود و پس از آنکه شرط مورد نظر برقرار شد، استعمارگر امپراتوری باقی‌مانده به‌عنوان جواب نمایش داده می‌شود.
3. پس از گذشت زمان مشخصی از شروع اجرای الگوریتم، الگوریتم متوقف شده و بهترین جواب (کشور) از میان کشورهای باقی‌مانده به‌عنوان جواب نمایش داده می‌شود. فلوچارت روش پیشنهادی در شکل 2 نشان داده شده است. این مقاله مسئله انتخاب ماشین‌های مجازی با هدف کاهش مصرف انرژی را توسط الگوریتم ICA-ES در بستر محاسبات ابری حل کرده است. برای حل مسئله انتخاب ماشین‌های مجازی مهم‌ترین کار، نحوه مدل کردن کشورهاست، برای این منظور یک آرایه به طول N در نظر گرفته شده است. N تعداد ماشین های مجازی است.

*** *Decades-*



شکل 2 فلوچارت الگوریتم ICA-ES

در واقع هر کشور یک آرایه است. که هر اندیس نماینده یک ماشین مجازی می باشد و عدد صفر داخل هر اندیس نشانگر این می باشد که ماشین مجازی مربوطه برای مهاجرات انتخاب نشده است و عدد یک نشانگر انتخاب ماشین مجازی مربوطه برای مهاجرت می باشد. به صورت مثال اگر اندیس 1 برابر صفر باشد و اندیس 2 برابر عدد 1 باشد. یعنی باید ماشین مجازی شماره 1 در محل خود باقی بماند و لی ماشین مجازی شماره 2 جهت مهاجرت به لیست ماشین های مجازی مهاجر اضافه شود. در ابتدا به صورت تصادفی اندیس هایی که باید مقدار یک بگیرند انتخاب می شوند.

5 شبیه سازی

برای ارزیابی و تحلیل کارایی روش پیشنهادی از شبیه‌ساز CloudSim استفاده شده است. این شبیه‌ساز یک جعبه‌ابزار به زبان جاوا می‌باشد که برای شبیه سازی محیط ابری استفاده می‌شود این جعبه‌ابزار که از مجموعه چندین کلاس تشکیل شده است؛ توسط Anton Beloglazov و همکارانش در سال 2009 طراحی شده است[15]. CloudSim مجموعه‌ای از کلاس‌ها می‌باشد که به زبان جاوا نوشته شده است. در این تحقیق شبیه‌سازی الگوریتم‌ها در دو سناریو A و B انجام شده است که به ترتیب حالت ابرهای خصوصی و عمومی را شبیه سازی می نماید. داده‌های سناریو-های A و B با توجه به مقالات مطالعه شده در این زمینه از جمله مرجع [2] و [3] داده شده است. در این تحقیق روش پیشنهادی از لحاظ مصرف بهینه انرژی، میزان نقض شدن قراردادهای SLA مورد بررسی قرار گرفته است. دو سناریو ارزیابی برای بررسی عملکرد روش پیشنهادئی در نظر گرفته شده است. مصرف انرژی یک میزبان فیزیکی در یک مرکز داده معمولا به پردازنده، حافظه اصلی، دیسک ذخیره سازی، منبع انرژی و سیستم خنک کننده ان بستگی دارد[16]. از جمله آخرین مطالعاتی که در خصوص چگونگی مصرف انرژی در سرورها پرداخته شده است می توان به [18،17] اشاره کرد.

• سناریو A

این سناریو سعی در شبیه سازی یک مرکز داده کوچک دارد. در این سناریو یک مرکز داده با 100 میزبان فیزیکی ناهمگون شبیه سازی شده است. که هر کدام از میزبان های فیزیکی دارای یک پردازنده تک هسته با قدرت 1000، 2000 یا 3000 MIPS و حافظه اصلی 8 گیگابایت و هارد دیسک 1 ترابایتی می باشند. هر کدام از میزبان های فیزیکی وقتی میزان کاروری پردازنده شان 0٪ می باشد 175 وات بر ساعت انرژی و وقتی کاروری پردازنده شان 100% باشد 250 وات بر ساعت انرژی مصرف می کنند. هر کدام از ماشین های مجازی تعریف شده در این سناریو دارای پردازنده تک هسته ای می باشند که قدرت پردازنده آنها 250، 500، 750 یا 1000 MIPS و میزان حافظه اصلی آنها 128 مگابایت و میزان هارد دیسک آنها 1 گیگابایت می باشد. در این شبیه سازی 290 ماشین مجازی ناهمگون در نظر گرفته شده است [19].

• سناریو B

در این سناریو یک مرکز داده با 800 ماشین فیزیکی ناهمگون. که نیمی از آنها سرورهای مدل HP ProLiant ML110 G4 و نیم دیگر سرورهای مدل HP ProLiant ML110 G5 می باشند. داده های مربوط به مصرف انرژی سرورها در این سناریو به این ترتیب می باشند. فرکانس پردازنده هر یک از سرورهای مدل HP ProLiant ML110 G4 1860 MIPS و فرکانس پردازنده هریک از سرورهای مدل HP ProLiant ML110 G5 2660 MIPS می باشد. هر دو مدل دارای 1 گیگابایت بر ثانیه پهنای باند و 4 گیگابایت حافظه اصلی می باشند. هر کدام از میزبان های فیزیکی وقتی میزان کاروری پردازنده شان 0٪ می باشد 175 وات بر ساعت انرژی و وقتی کاروری پردازنده شان 100٪ باشد 250 وات بر ساعت انرژی مصرف می کنند. ویژگی های ماشین های مجازی این سناریو که مطابقت دارند با نمونه های Amazon EC2 به ترتیب زیر می باشند. پردازنده تمام ماشین های مجازی تک هسته ای می باشند ماشین های مجازی براساس میزان حافظه اصلی و قدرت پردازنده به چند دسته تقسیم می شوند.دسته اول با پردازنده

††† Toolkit

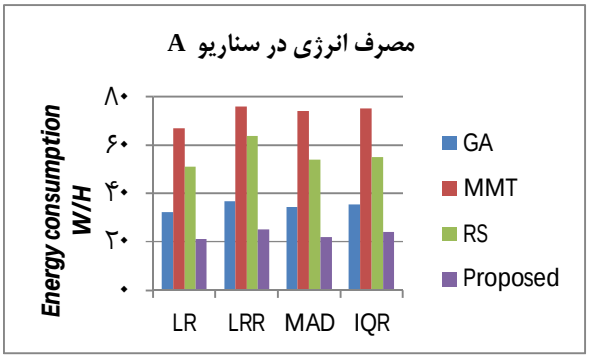
MIPS 2500 و حافظه اصلی 0.85 گیگابایت. دسته دوم با پردازنده 2000 MIPS و حافظه 3,75 گیگابایت، دسته سوم با پردازنده 1000 MIPS و حافظه اصلی 1,7 گیگابایت و دسته چهارم دارای پردازنده 500 MIPS و حافظه اصلی 613 مگابایتی می باشند. در این سناریو 1052 ماشین مجازی در نظر گرفته شده است [3].

6 ارزیابی کارایی

در این تحقیق از الگوریتم ICA-ES برای انتخاب ماشین‌های مجازی استفاده شده است. همانطور که در بخش‌های قبل اشاره شد در یک مرکز داده ابری سه زیر مسئله وجود دارد که بر عملکرد هم تاثیر دارند (چهار روش پرکاربرد MAD، IQR، LR، LRR برای زیر مسئله «چه زمانی مهاجرت انجام گیرد» و روش PABFD برای زیر مسئله « ماشین های مجازی کجا مهاجرت کند » شبیه‌سازی شده است.) در این تحقیق الگوریتم ICA-ES به عنوان راه حلی برای زیرمسئله دوم انتخاب ماشین های مجازی" مطرح شده است. کاهش مصرف انرژی نیاز به قرار گرفتن پاسخ های مناسب برای هر سه زیر مسئله می باشد در واقع روش هایی که برای هر زیر مسئله در نظر گرفته می شود در نتیجه کلی تاثیر خواهد داشت و اینکه روش پیشنهادی در کنار کدام یک از روش های زیر مسئله های دیگر جواب مناسب می دهد مهم می باشد. در این تحیق عملکرد روش پیشنهادی در کنار الگوریتم‌های تشخیص سربار میزبان‌های فیزیکی نیز مورد ارزیابی قرار گرفته است. همچنین برای اینکه بتوانیم یک روش کامل برای بهینه کردن مصرف انرژی داشته باشیم باید بررسی کنیم که برای هر زیر مسئله کدام یک از روش‌های مطرح‌شده بهتر است در واقع ترکیب روش‌ها نیز مهم می باشد. در ادامه مقایسه های الگوریتم پیشنهادی از نظر مصرف انرژی و میزان نقض SLA بیان شده اند.

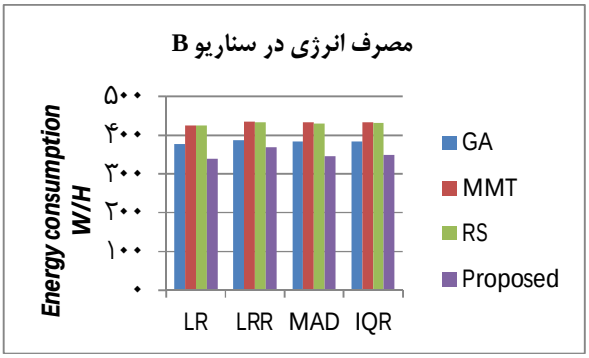
1-6 مقایسه الگوریتم پیشنهادی از نظر مصرف انرژی با الگوریتم های GA ، MMT و RS

شکل 4 مصرف انرژی روش های مختلف را در سناریو A نشان می دهد. محور عمودی این نمودار میزان مصرف انرژی را برحسب وات بر ساعت نشان می دهد و محور افقی روش های تشخیص سرریز بار میزبان های فیزیکی را نشان می دهد. همانطور که در شکل 4 مشخص است کمترین میزان مصرف انرژی را روش پیشنهادی همراه با روش LR برای تشخیص سرریز بار میزبان های فیزیکی با 21 وات بر ساعت بدست آورده است.



شکل 4 مصرف انرژی در سناریو A

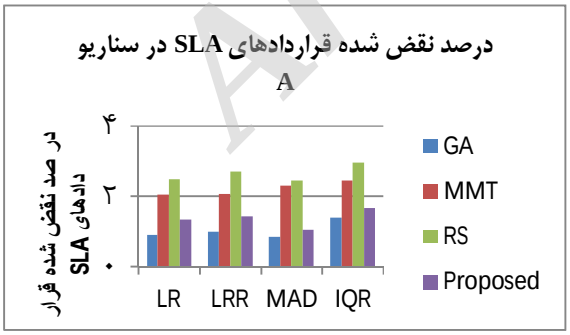
همچنین با دقت در این نمودار مشاهده می شود که روش LR برای تشخیص سرریز بار عملکرد بهتری داشته است و همچنین روش پیشنهادی در همه ترکیب ها بهتر از سایر روش ها عمل کرده است. نمودار شکل 5 مصرف انرژی را در سناریو B نشان میدهد. همانطور که در نمودار شکل 5 مشخص است روش پیشنهادی با 339 وات بر ساعت به همراه الگوریتم LR بدست آورده است. همچنین رفتار مشابهی بین نمودار های شکل های 4 و 5 مشاهده می شودد یعنی همانند نمودار شکل 4 در نمودار شکل 5 نیز روش LR بهترین عملکرد را داشته و روش پیشنهادی نیز در همه ترکیب ها بهتر از روش های مشابه عمل کرده است.



شکل 5 مصرف انرژی در سناریو B

2-6 مقایسه الگوریتم پیشنهادی از نظر میزان نقض شدن قرار دادهای SLA با الگوریتم های GA ، MMT و RS

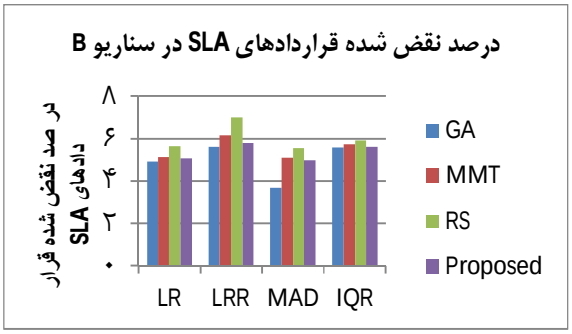
نمودار شکل 6 میانگین نقض شدن قراردادهایSLA را برای سناریو A نمایش می دهد. محور عمودی این نمودار میزان نقض شدن قراردادهای SLA بر حسب درصد را نمایش می دهد و محور افقی روش هایی را که برای زیر مسئله تشخیص سرریز بار میزبان های فیزیکی می توان تعیین کرد. همانطور که در نمودار شکل 6 مشاهده می شود کمترین میزان نقض شدن قرار دادهای SLA توسط در الگوریتم GA اتفاق افتاده است. و روش پیشنهادی در رتبه دوم نقض شدن کمتر قراردادها دارد.



شکل 6 میزان نقض شدن قرار دادهای SLA در سناریو A

نمودار شکل 7 نیز میزان نقض شدن قراردادهای SLA را در سناریو B نمایش می دهد. با توجه به نمودار شکل 7 مشاهده می شود که رفتار نمودار شکل 7 نیز مشابه نمودار شکل 6 می باشد. با دقت در نمودارهای شکل 6 و 7

مشاهده می شود که روش MAD کمترین میزان نقض شدن قراردادهای SLA را داشته است.



شکل 7 میزان نقض شدن قرار دادهای SLA در سناریو B

7 نتیجه گیری

همانطور که از نمودارهای شکل ۴،۵، 6 و 7 مشاهده می شود روش پیشنهادی در کنار روش LR برای تشخیص سر ریز بار میزبان های فیزیکی کمترین میزان مصرف انرژی را داشته است. روش پیشنهادی در سناریو A بهبود 34,82 در صدی و در سناریو B بهبود 10,26 درصدی را داشته که به صورت میانگین می توان گفت روش پیشنهادی بهبود 22,54 درصدی داشته است. همچنین از نمودار های می توان نتیجه گرفت که روش LR برای تشخیص سر ریز بار میزبان های فیزیکی کمک بیشتری برای کاهش مصرف انرژی می کند، الگوریتم پیشنهادی از لحاظ درصد نقض شدن قراردادهای SLA کارایی کمتری نسبت به الگوریتم ژنتیک داشته، البته این اختلاف در حد 208 صدم در صد می باشد که کاملا جزیی می باشد. با دقت در نمودارهای شکل های 4، 5، 6 و 7 مشاهده می شود که رابطه عکس بین کاهش مصرف انرژی و میزان نقض شدن قرار دادهای SLA برقرار می باشد. در واقع با کاهش مصرف انرژی میزان نقض شدن قراردادهای SLA افزایش می یابد. بهبود 22,54 درصدی مصرف انرژی باعث می شود 208 صدم درصد بیشتر نقض شدن قرار دادهای SLA را بتوان نادیده گرفت.

مراجع

[۱] رضوانیان، علیرضا، حسنیه صفی اریان، صبا رضوانیان، دانیال یزدانی، "**بهبود الگوریتم رقابت استعماری با استفاده از استراتژی تکامل**" *سومین همایش ملی مهندسی کامپیوتر و فناوری اطلاعات*، همدان، 1389.

[2] Buyya, R., Beloglazov, A., & Abawajy, J., *Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges*, arXiv preprint arXiv:1006.0308, 2010.

[3] Beloglazov, A., Energy-efficient management of virtual machines in data centers for cloud computing, Submitted in total fulfilment of the requirements of the degree of Doctor of Philosophy, Department of Computing and Information Systems The University Of Melbourne, 2013.

[4] Falkenauer, E., & Delchambre, A., *A genetic algorithm for bin packing and line balancing. In Robotics and Automation,*. Proceedings 1992 IEEE International Conference on (pp. 1186-1192). IEEE, 1992, May.

[5] W Cleveland, W. S. *Robust locally weighted regression and smoothing scatterplots.* Journal of the American statistical association, 74(368), 829-836, 1992, May.

[6] Guenter, B., Jain, N., & Williams, C., *Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning*. In INFOCOM, 2011 Proceedings IEEE (pp. 1332-1340). IEEE, 2011, April.

[7] Beloglazov, A., & Buyya, R., *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 and Experience, 24(13), 1397-1420, 2012.

[8] R. Jeyarani, N. Nagaveni, R. Vasanth Ram., *Self Adaptive Particle Swarm Optimization for Efficient Virtual Machine Provisioning in Cloud*, International Journal of Intelligent Information Technologies, Volume ۷ Issue ۲,pp ۴۴-۲۵, 2011.

[9] Wu, G., Tang, M., Tian, Y. C., & Li, W., *Energy-efficient virtual machine placement in data centers by genetic algorithm.* In Neural Information Processing (pp. 315-323). Springer Berlin Heidelberg, 2012, January.

[10] Wu, Y., Tang, M., & Fraser, W., *A simulated annealing algorithm for energy efficient virtual machine placement*. In Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on (pp. 1245-1250). IEEE, 2012, October.

[11] Atashpaz-Gargari, E., & Lucas, C., *Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition*. In Evolutionary computation, 2007. CEC 2007. IEEE Congress on (pp. 4661-4667). IEEE, 2007, September.

[12] Nazari-Shirkouhi, S., Eivazy, H., Ghodsi, R., Rezaie, K., & Atashpaz-Gargari, E., *Solving the integrated product mix-outsourcing problem using the imperialist competitive algorithm.* Expert Systems with Applications, 37(12), 7615-7626, 2010.

[13] Lucas, C., Nasiri-Gheidari, Z., & Tootoonchian, F. *Application of an imperialist competitive algorithm to the design of a linear induction motor.* Energy Conversion and Management, 51(7), 1407-1411, 2007, September.

[14] Forouharfard, S., & Zandieh, M., *An imperialist competitive algorithm to schedule of receiving and shipping trucks in cross-docking systems*. The International Journal of Advanced Manufacturing Technology, 51(9-12), 1179-1193, 2007, September.

[15] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R., *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience*, 41(1), 23-50, 2007, *September*.

[16] *Minas, L., & Ellison, B., Energy efficiency for information technology: How to reduce power consumption in servers and data centers*. Intel Press. 2007, September.

[17] Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., & Jiang, G., *Power and performance management of*

virtualized computing environments via lookahead control. *Cluster computing*, 12(1), 1-15, 2009.

[18] Fan, X., Weber, W. D., & Barroso, L. A., *Power provisioning for a warehouse-sized computer*. In ACM SIGARCH Computer Architecture News (Vol. 35, No. 2, pp. 13-23). ACM, 2007, June.

Beloglazov, A., Abawajy, J., & Buyya, R., *Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing*. Future generation computer systems, 28(5), 755-768, 2012.