

بهره‌گیری از الگوریتم پرش ترکیبی قورباغه جهت کاهش مصرف انرژی مراکز داده ابری از طریق بهینه‌سازی مدیریت زمان‌بندی کارها و ترکیب مؤثر ماشین‌های مجازی

وحید ستاری نائینی^۱، استادیار؛ یاسمین سالم^۲، کارشناسی ارشد؛ عصمت راشدی^۳، استادیار

۱- بخش مهندسی کامپیوتر- دانشگاه شهید باهنر کرمان- کرمان- ایران - vsnaeini@uk.ac.ir

۲- دانشکده مهندسی برق و کامپیوتر- دانشگاه تحصیلات تکمیلی صنعتی و فناوری پیشرفته - کرمان - ایران - sa.yas1370@gmail.com

۳- دانشکده مهندسی برق و کامپیوتر- دانشگاه تحصیلات تکمیلی صنعتی و فناوری پیشرفته - کرمان - ایران - e.rashedi@kgut.ac.ir

چکیده: امروزه رایانش ابری سبز به دلیل کاهش اثرات زیست‌محیطی مورد توجه قرار گرفته است. یکی از معیارهایی که در رایانش ابری سبز بر آن تاکید شده است، مصرف انرژی مراکز داده است. یکی از راهکارهای کاهش مصرف انرژی، که در این مقاله مورد بررسی قرار گرفته است، مدیریت زمان‌بندی کارها و ترکیب مؤثر ماشین‌های مجازی است. در این مقاله الگوریتمی جهت مدیریت زمان‌بندی کارها و توازن بار ارائه می‌شود. این الگوریتم به نام پرش ترکیبی قورباغه با بهره‌مندی از حافظه، همکاری و اشتراک‌گذاری اطلاعات بین قورباغه‌ها، سرعت همگرایی مناسب و انعطاف‌پذیری بهتر در برابر مشکل بهینه محلی، بهبود قابل‌توجهی نسبت به روش‌های سیستم تجمع مورچگان (ACO) و الگوریتم ژنتیک (GA) جهت مصرف انرژی و مهاجرت ماشین مجازی فراهم می‌آورد. در این مقاله، مدیریت پویای منابع بر اساس ترکیب مؤثر ماشین‌های مجازی انجام می‌شود و توسط الگوریتم پیشنهادی با توجه به قرارداد سطح سرویس پیاده‌سازی می‌شود. تفاوت این روش با روش‌های دیگر در این است که بهبود پارامترهای زمان، سرعت و دقت همگرایی رانشان می‌دهد. نتایج تجربی نشان می‌دهد که، روش ارائه شده موجود، از نظر مصرف انرژی، تعداد مهاجرت‌های ماشین مجازی و نقض قرارداد سطح سرویس عملکرد بهتری دارد.

واژه‌های کلیدی: رایانش ابری سبز، مصرف انرژی، مدیریت زمان‌بندی، ترکیب مؤثر ماشین‌های مجازی، نقض قرارداد سطح سرویس، تعداد مهاجرت‌های ماشین مجازی.

Application of Shuffled Frog-Leaping Algorithm to Reduce Energy Consumption in Cloud Data Centers by Optimizing Scheduling Management and Virtual Machines Consolidation

V. Sattari-Naeini¹, Assistant Professor; Y. Salem², MSc Student; E. Rashedi³, Assistant Professor

1- Faculty of Computer Engineering, Shahid Bahonar University of Kerman, Kerman, Iran, Email: vsnaeini@uk.ac.ir

2- Faculty of Electrical and Computer Engineering, Graduate University of Advanced Technology, Kerman, Iran,

3- Faculty of Electrical and Computer Engineering, Graduate University of Advanced Technology, Kerman, Iran,

Abstract: Today, green cloud computing has been concerned due to the reduction of environmental impacts. One of the criteria that has been emphasized in green cloud computing is energy consumption of data centers. One way to reduce energy consumption, which we is emphasized in this paper, is tasks scheduling management and consolidation of virtual machines. In this paper, an algorithm is presented to manage both tasks scheduling and load balancing. This algorithm, called the Shuffled Frog-Leaping provides a significant improvement against other existing models in terms of energy consumption and migration of virtual machines using memory, collaboration and sharing information among frogs, high convergence speed and better flexibility against local optimum problem. In this paper, the dynamic resource management is based on the consolidation of virtual machines and is implemented according to service level agreement by the proposed method. The difference between this method and other existing methods is that it shows improvement of time, speed and accuracy of convergence parameters. Experimental results show that the proposed method outperforms existing ones in terms of energy consumption, number of virtual machine migrations and service level agreement violation.

Keywords: Green cloud computing, energy consumption, scheduling Management, consolidation of virtual machines, service level agreement violation, number of virtual machine migrations.

تاریخ ارسال مقاله: ۱۳۹۵/۰۹/۲۲

تاریخ اصلاح مقاله: ۱۳۹۵/۱۲/۲۵

تاریخ پذیرش مقاله: ۱۳۹۶/۰۵/۱۰

نام نویسنده وحید ستاری نائینی

نشانی: ایران - کرمان - بزرگراه امام خمینی - میدان افصلی پور - دانشگاه شهید باهنر کرمان - بخش مهندسی کامپیوتر

۱- مقدمه

پردازش ابری مدل ارائه خدمات مقرون به صرفه‌ای است که، مدیریت و نگهداری منابع را بسیار آسان می‌کند و در صورت تغییرات کسب و کار، منابع به سرعت می‌توانند خود را با این تغییرات سازگار سازند. هسته محیط پردازش ابری، مرکز داده آن است که، اغلب از تعداد زیادی امکانات سخت‌افزاری به هم مرتبط تشکیل شده است. میزان قابلیت‌های پردازشی مراکز داده، شاخص اصلی ارائه‌دهندگان خدمات ابری در نظر گرفته می‌شود.

در مراکز داده بزرگ با افزایش داده به‌طور فزاینده، مصرف انرژی افزایش می‌یابد. این افزایش مصرف انرژی در محیط زیست اثر می‌گذارد. هزینه‌های زیست‌محیطی تحمیل شده سبب شده است که، رفته رفته مبحثی به نام راینش ابری سبز مطرح شود. راینش ابری سبز نه تنها باعث حداقل شدن هزینه‌های عملیاتی شده است، بلکه به کاهش اثرات زیست‌محیطی نیز کمک می‌کند. پیش‌بینی می‌شود، مصرف انرژی مراکز داده در سال ۲۰۲۰ توسط راینش ابری ۳۸ درصد کاهش یابد [۱].

پردازش سبز تنها یک روش به‌منظور افزایش بهره‌وری عملیاتی از مراکز داده ابری و کاهش آسیب به محیط زیست است. میزبان‌های مراکز داده وقتی که در وضعیت بدون استفاده هستند، باید انرژی کمتری مصرف کنند. سرورها و امکانات سخت‌افزاری موجود در بازار در دهه اخیر از تغییرات پویای فرکانس ولتاژ پشتیبانی می‌کنند [۲]. در DVFS، سرور می‌تواند فرکانس عامل را بر اساس جریان بار موجود در سیستم برای دستیابی به حداقل مصرف انرژی تنظیم کند. با این وجود اتخاذ یک مکانیزم نظارتی پویا برای عدم اطمینان از منابع ضروری است. در یک محیط ابری در مرکز داده وجود منابع و برنامه‌های بدون استفاده و نامشخص سبب افزایش مقدار اتلاف انرژی می‌شود [۳].

دلیل مصرف زیاد انرژی فقط به میزان منابع استفاده شده و بازده کم سخت‌افزار مربوط نیست، بلکه بیشتر به استفاده ناکارآمد این منابع نیز وابسته است. اطلاعاتی که از بیش از ۵۲۲۲ سرور در یک دوره شش‌ماهه جمع‌آوری شد [۴]، نشان داد که سرورها فقط ۱۲ الی ۱۵ درصد کل ظرفیت پردازشی خود را غالباً استفاده می‌کنند که، موجب هزینه در تهیه سخت‌افزار می‌شود.

استفاده حداکثری از منابع مراکز داده ابری و کاهش مصرف انرژی، نیاز به انتخاب، مهاجرت و ترکیب پویای ماشین‌های مجازی میزبان‌های مراکز داده ابری، درحالت‌های مختلف بارکاری دارد. این فرآیند شامل چهار بخش می‌شود [۵]: ۱- تشخیص میزبان‌های اضافه بار ۲- انتخاب میزبان‌هایی با بارکاری کم ۳- انتخاب ماشین‌های مجازی جهت مهاجرت از میزبان‌های اضافه بار ۴- قرارگیری ماشین‌های مجازی.

پیش‌نیاز اصلی برای مهاجرت این است که، ماشین مجازی که باید مهاجرت داده شود را تعیین کنیم؛ بنابراین وقتی که یک میزبان به‌عنوان میزبان پرکار تشخیص داده می‌شود، فرآیند انتخاب ماشین مجازی

برای آن میزبان اجرا می‌شود. انتخاب این که کدام ماشین مجازی باید مهاجرت کند، بر کارایی میزبان تاثیرگذار است.

اگر ماشین مجازی با بهره‌وری پایین منتقل شود، بار میزبان به‌اندازه کافی کم نمی‌شود و این یعنی احتمال اینکه میزبان در بازه زمانی بعدی نیز پرکار شود، زیاد است. در مقابل، مهاجرت یک ماشین مجازی با بهره‌وری بالا بار میزبان را به‌خوبی کم می‌کند، اما هزینه‌هایی مثل زمان مهاجرت و مصرف انرژی را در پی خواهد داشت. سانگ و همکاران در مرجع [۶]، انتخاب ماشین مجازی را به‌عنوان یک مسئله بهینه‌سازی چند هدفه مدل کرده‌اند. آن‌ها کارایی هر ماشین مجازی را بر اساس وزنی از سه مدل دمای پردازنده، منابع و توان مصرفی ماشین مجازی تعریف کرده‌اند. آن‌ها همچنین ثابت کرده‌اند، که هزینه ناشی از مهاجرت ماشین مجازی، رابطه خطی با داده‌های انتقالی دارد و عمده‌تاً به‌وسیله اندازه حافظه ماشین مجازی تعیین می‌شود؛ بنابراین در تعریف تابع انتخاب ماشین مجازی، هزینه مهاجرت نیز لحاظ می‌گردد. در این مقاله تعداد مهاجرت‌های ماشین مجازی و مصرف انرژی در نظر گرفته نمی‌شود؛ همچنین روی زمان‌بندی ماشین‌های مجازی تمرکز ندارد.

رودرو و همکاران [۷]، پیشنهاد استراتژی مدیریت مصرف انرژی بر اساس کیفیت خدمات برای سیستم‌های سرور خوشه‌ای را مطرح کردند. این استراتژی به دو بخش مدیریت پایان و مدیریت محلی تقسیم می‌شود. مدیریت محلی از DVFS پشتیبانی می‌کند. هنگامی که مدیریت توان سخت‌افزار تشخیص دهد که، سیستم نیاز به خاموش یا روشن شدن یک سرور دارد، مدیریت محلی با ماژول DVFS میزان توان را کنترل می‌کند و امکان تغییر حالت سرور را فراهم می‌کند. در این مقاله تعداد مهاجرت ماشین‌های مجازی در نظر گرفته نشده است.

معصوم زاده و همکاران [۸]، از تکنیک یادگیری فازی برای ایجاد یک سیاست انتخاب ماشین مجازی وفق‌پذیر و پیشگیرانه از روی میزبان پرکار شده استفاده کرده‌اند که، تصمیمات بهینه‌ای را برای انتخاب ماشین مجازی جهت مهاجرت می‌گیرد. آن‌ها ثابت کرده‌اند که، نتیجه سیاست‌های انتخاب ماشین مجازی به رفتار بارکاری و وضعیت میزبان بستگی دارد. از این‌رو، از چندین الگوریتم برای این منظور استفاده کرده‌اند. آن‌ها برای هر الگوریتم یک پاداش در نظر گرفته‌اند که، نشان‌دهنده کارایی الگوریتم در مصرف انرژی و میزان نقض قرارداد سطح سرویس است. همچنین رفتار هر میزبان را با دیدن وضعیتش دنبال می‌کنند. در روش پیشنهادی آن‌ها، منابع و تعداد ماشین‌های مجازی قرار گرفته روی میزبان، به‌عنوان وضعیت آن در نظر گرفته شده است. در این مقاله از زمان‌بندی کارها استفاده نشده است.

در مرجع [۹]، یک الگوریتم ژنتیک به‌عنوان روش بهینه‌سازی برای زمان‌بندی پیشنهاد شده است، که باعث می‌شود تصمیم‌گیری زمان‌بندی توسط ارزیابی کل وظایف در صف کارها انجام شود. نتایج شبیه‌سازی اولیه نشان می‌دهد، که زمان‌بندی پیشنهادی کارایی بهتری از منظر makespan، تأخیر سیاست‌های زمان‌بندی و توازن بار داشته است. در این مقاله نقض قرار دارد سطح سرویس و مصرف انرژی در نظر گرفته نشده است.

توان، باید اطمینان حاصل کنیم که پیکربندی سیستم از مدل قرارداد سطح سرویس پیروی می‌کند. یک روش هوشمند برای این کار استفاده از الگوریتم‌های تکاملی برای رسیدن به حداکثر بهره‌وری برای تخصیص منابع در یک مرکز داده ابری است. ویژگی‌های الگوریتم پرش ترکیبی قورباغه (SFLA) ما را ترغیب به استفاده از این الگوریتم به‌عنوان یک روش بهینه‌سازی برای زمان‌بندی کارها در محیط ابری می‌کند. با توجه به اهمیت تعداد مهاجرت ماشین‌های مجازی، زمان‌بندی کارها، نقض قرارداد سطح سرویس و مصرف انرژی، در این مقاله روشی مبتنی بر SFLA ارائه می‌شود که از طرفی سرعت همگرایی بالایی دارد؛ همچنین با در نظر گرفتن تعداد مهاجرت ماشین‌های مجازی و ترکیب مؤثر آن‌ها همراه با زمان‌بندی کارها، میزان نقض قرارداد سطح سرویس و مصرف انرژی به‌طور مؤثری کاهش می‌یابد.

این مقاله به ۶ بخش تقسیم شده است. در بخش دوم در مورد الگوریتم پرش ترکیبی قورباغه به‌اجمال بحث می‌شود. در بخش سوم و چهارم معماری سیستم پیشنهادی و روش پیشنهادی که زمان‌بندی کارها و ترکیب مؤثر ماشین مجازی را در بر می‌گیرد، بیان می‌شوند. در بخش پنجم شبیه‌سازی و بررسی نتایج بیان می‌شوند و در نهایت در بخش ششم نتیجه‌گیری و کارهای آتی ارائه می‌شوند.

۲- الگوریتم پرش ترکیبی قورباغه SFLA

الگوریتم پرش ترکیبی قورباغه (SFLA) یک الگوریتم مبتنی بر جمعیت فرا ابتکاری است. این الگوریتم توسط Eusuff و Lansey برای حل مسائل بهینه‌سازی ارائه شده است [۱۴]. الگوریتم SFLA از نحوه جست‌وجوی غذای گروه قورباغه‌ها سرچشمه می‌گیرد و برای جست‌وجوی محلی میان زیرگروه‌های قورباغه از روش Memetic استفاده می‌کند. SFLA از استراتژی ترکیب استفاده می‌نماید و امکان مبادله پیام در جست‌وجوی محلی را فراهم می‌سازد. در این الگوریتم نه تنها در جست‌وجوی محلی، بلکه در جست‌وجوی سراسری نیز پیام‌ها مبادله می‌شوند؛ بنابراین جست‌وجوی محلی و سراسری به‌خوبی در این الگوریتم ترکیب می‌شوند [۱۷-۱۵].

در جدول ۱ پارامترهای استفاده شده در مقاله و مرتبط با این الگوریتم تعریف شده‌اند.

۲-۱- شرح الگوریتم SFLA

تابع SFLA به‌صورت تصادفی یک جمعیت اولیه از قورباغه‌ها به تعداد F عدد $P = \{X_1, X_2, \dots, X_F\}$ تولید می‌کند. برای حل مسئله به‌صورت t بعدی، قورباغه i ام $X_i = \{X_{i1}, X_{i2}, \dots, X_{it}\}$ ($1 < i < F$) قرار می‌گیرد. بعد از تولید جمعیت قورباغه، مقدار شایستگی ($f(X_i)$) هر قورباغه (X_i) محاسبه می‌شود، سپس قورباغه‌ها براساس شایستگی‌شان ($f(X_i)$) به‌صورت نزولی طبقه‌بندی می‌شوند. قورباغه‌ها به m گروه تقسیم می‌شوند، که هر گروه n قورباغه دارد؛ بنابراین $F = n \times m$ است. یک جست‌وجوی محلی بر روی هر گروه اعمال می‌شود.

فرحناکیان و همکاران در مرجع [۱۰]، معماری سیستم‌های توزیع‌شده برای اجرای ترکیب مؤثر ماشین مجازی پویا برای کاهش مصرف انرژی مراکز داده ابری، درحالی‌که کیفیت سرویس مورد نظر را حفظ می‌کند، ارائه داده‌اند. در اینجا از الگوریتم فرا ابتکاری بهینه‌سازی آنالین به نام سیستم کلونی مورچگان استفاده شده است. روش ترکیب مؤثر ماشین‌های مجازی بر اساس ACS^۴ (ACS-VMC) یک راه‌حل نزدیک به بهینه را بر اساس یک تابع هدف مشخص پیدا می‌کند. نتایج آزمایش بر روی حجم کار واقعی نشان می‌دهد، که ACS-VMC مصرف انرژی را درحالی‌که سطح عملکرد مورد نیاز در یک مرکز داده ابری حفظ می‌کند، کاهش می‌دهد. در این روش، ترکیب مؤثر ماشین‌های مجازی موجود از نظر مصرف انرژی، تعداد مهاجرت ماشین مجازی و الزامات مربوط به کیفیت سرویس مورد نیاز عملکرد بهتر دارند. در این مقاله سرعت همگرایی مورد بحث قرار نگرفته است.

در مقاله [۱۱]، از بهینه‌سازی چندهدفه قیددار برای مسأله تخصیص وظایف در ابر سیار استفاده شده و روشی دو مرحله‌ای پیشنهاد شده است. در مرحله اول الگوریتم مبتنی بر شاخه و حد چندهدفه (MRABB) طراحی شده است و در مرحله دوم با استفاده از روش رتبه‌بندی ترجیحات از طریق شباهت به جواب ایده‌آل (TOPSIS) و گرفتن ترجیحات کاربر، بهترین جواب مصالحه‌گر تعیین می‌گردد. نتایج آزمایش‌ها نشان می‌دهد که روش پیشنهادی در مقایسه با الگوریتم‌های تخصیص وظایف پیشین، به‌خوبی مصالحه میان زمان و انرژی را مدیریت می‌کند. در این روش، تعداد مهاجرت ماشین‌های مجازی، سرعت همگرایی و میزان نقض قرارداد سطح سرویس در نظر گرفته نشده است. سیستم‌ها در مراکز داده ابری امکان مهاجرت ماشین‌های مجازی بین گره‌های فیزیکی برای رسیدن به بهبود عملکرد و صرفه‌جویی در انرژی را می‌دهند. زمانی که ماشین‌های مجازی کمتر از منابع استفاده می‌کنند و کمتر منابع به آن‌ها تخصیص داده شده است، ماشین‌های مجازی می‌توانند با سایر گروه‌های سرور ترکیب شوند [۱۲]. در مرجع [۱۳]، یک روش زمان‌بندی بر اساس الگوریتم ژنتیک ارائه شده است که، جمعیت اولیه را با روش پیشرفته ماکزیمم مینیمم ایجاد کرده، که می‌تواند از نظر مدت زمانی که طول کشیده است تا کار انجام شود، نتایج بهینه‌تری تولید کند. عملکرد الگوریتم ژنتیکی اصلاح‌شده پیشنهادی و الگوریتم‌های موجود در برابر داده‌های نمونه بررسی شده است. نتایج تجربی نشان می‌دهد که، الگوریتم پیشنهادی عملکردی قابل قبول و بهتر از الگوریتم‌های موجود ارائه می‌دهد. در این مقاله تعداد مهاجرت‌های ماشین مجازی در نظر گرفته نمی‌شود.

دیدگاه مدیریت منابع در یک مرکز داده ابری برای زیرساخت به عنوان یک سرویس ارائه‌دهنده در محیط ابری در این مقاله مورد بررسی قرار می‌گیرد. طرح پیشنهادی نه تنها اطمینان از کیفیت خدمات توسط قرارداد سطح سرویس را به کاربران ارائه می‌دهد، بلکه باعث صرفه‌جویی مؤثر در مصرف انرژی می‌شود، که هدف رایانش ابری سبز است. برای رسیدن به استفاده حداکثری از منابع در حال مهاجرت ماشین مجازی و تغییر وضعیت میزبان از حالت بدون استفاده به حالت ذخیره‌سازی

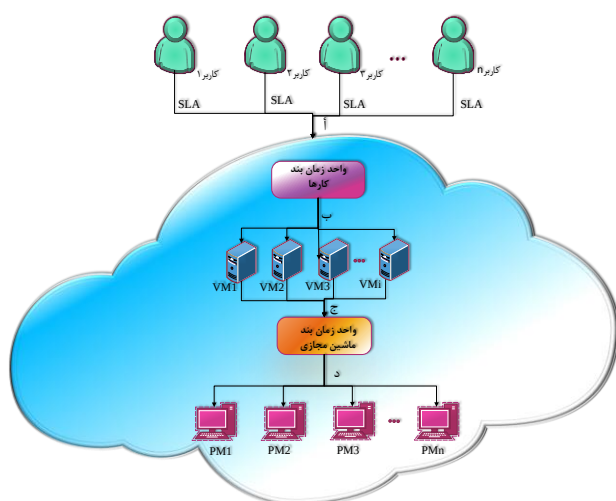
۲-۲- مزایا و ویژگی‌های الگوریتم پرش ترکیبی قورباغه

- ویژگی‌های مهم این الگوریتم که می‌توان به آن در بهینه‌سازی زمان‌بندی و مدیریت منابع ابر تکیه کرد، به شرح زیر است [۱۷-۱۵]:
- بهره‌مندی از حافظه: قورباغه‌ها دارای حافظه هستند، به گونه‌ای که دانش راه‌حل‌های خوب توسط همه عامل‌ها حفظ می‌شود.
 - همکاری و اشتراک‌گذاری اطلاعات بین قورباغه‌ها: در الگوریتم بهینه‌سازی پرش ترکیبی قورباغه، هر عضو گروه، موقعیت خود را با توجه به تجربیات شخصی و تجربیات کل جامعه تغییر می‌دهد. اشتراک اجتماعی اطلاعات بین اعضای یک گروه، یکسری مزیت‌های تکاملی را در پی خواهد داشت و این فرایند، پایه و اساس الگوریتم بهینه‌سازی و توسعه آن محسوب می‌شود، در نتیجه همکاری سودمند بین قورباغه‌ها وجود دارد و آن‌ها در هر گروه، اطلاعاتشان را با همدیگر به اشتراک می‌گذارند.
 - سرعت همگرایی بالا: در این الگوریتم، اعضای جمعیت با هم ارتباط دارند و از طریق تبادل اطلاعات، به حل مسئله می‌رسند که از سرعت همگرایی بالایی برخوردار است.
 - انعطاف‌پذیری بهتر در برابر مشکل بهینه محلی: الگوریتم بهینه‌سازی پرش ترکیبی قورباغه نسبت به استراتژی‌های بهینه‌سازی دیگر، با بهره‌گیری از تعداد فراوان عامل‌ها در برابر مشکل بهینه محلی، انعطاف بیشتری نشان می‌دهد.

۳- معماری سیستم

در این مقاله از معماری شکل ۱ برای توسعه سیستم و اجرای الگوریتم پیشنهادی استفاده می‌شود، که عملکرد آن به‌صورت زیر توصیف می‌شود:

آ- کارهای مربوط به کاربران وارد محیط ابر می‌شود و قراردادی بین کاربر و محیط ابر تنظیم می‌شود، که پارامترهای آن تحت عنوان قرارداد سطح سرویس قابل ارزیابی است.



شکل ۱: معماری سیستم پیشنهادی

جدول ۱: پارامترهای استفاده شده در مقاله

مقدار	مفهوم پارامتر
$X_{i,w}(k)$	مکان قورباغه‌ای، با بدترین شایستگی در تکرار k ام از یک گروه i ام
$X_{i,s}$	بهترین مکان قورباغه در گروه i ام
r	عدد تصادفی بین صفر و یک
$D_{i,w}(k)$	فاصله بدترین قورباغه در گروه i ام در تکرار k ام
X_b	بهترین (b) مکان قورباغه در جمعیت فعلی
MeanRespTim	میانگین زمان پاسخ
noofHosts	تعداد میزبان‌ها
reqTask	تعداد کارها
c	فاکتور جهش
W	وزن پرش
W_{max}	مقدار حداکثر
W_{min}	مقدار حداقل
k_{max}	حداکثر تعداد تکرار
k	شماره تکرار
Upper_Thresh	حداکثر آستانه استفاده از پردازنده
R_{SLA}	میزان قرارداد سطح سرویس برای میزبان
T_{SLA}	مقدار قرارداد سطح سرویس موجود بر روی میزبان
m	مجموعه کلی ماشین‌های مجازی (m) روی میزبان (h)
$\sum_{h \in h_{Um}} Utlz(h)$	مجموعه کلی ماشین‌های مجازی (m) روی میزبان (h)
$Utlz(v)$	میانگین بهره‌برداری پردازنده (ماشین مجازی (v)) فعلی از دوره زمانی
$E_w^{(T)}(h)$	انرژی که میزبان برای انجام کارها مصرف می‌کند
$E_m^{(T)}(h)$	انرژی سربار مصرفی برای مهاجرت ماشین‌های مجازی
T	دوره زمان

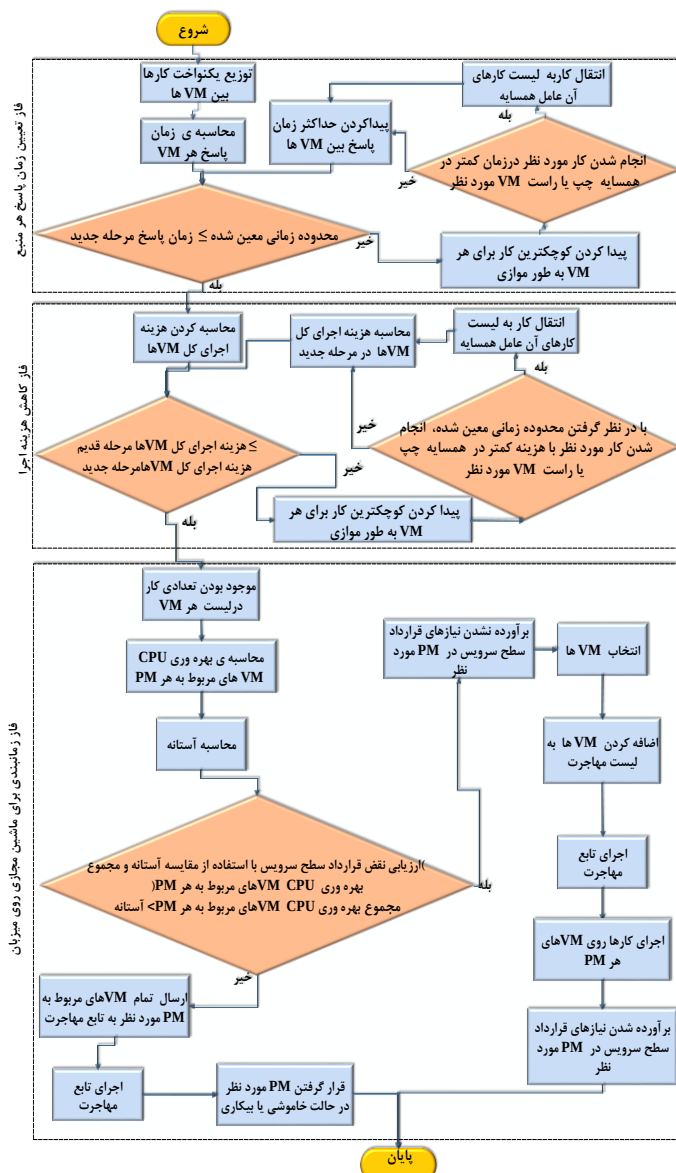
آخرین قورباغه با کمترین شایستگی در هر گروه با رابطه ۱ به‌روزرسانی می‌شود [۱۷-۱۵]:

$$D_{i,w}(k) = r \times (X_{i,s} - X_{i,w}(k)) \quad (1)$$

اگر موقعیت بدترین قورباغه جدید بهتر از موقعیت بدترین قورباغه فعلی باشد، بعد از به‌روزرسانی این مقدار جای مقدار قدیمی را خواهد گرفت. در غیر این صورت فاصله با رابطه ۲ به‌روزرسانی خواهد شد [۱۷-۱۵]:

$$D_{i,w}(k) = r \times (X_b - X_{i,w}(k)) \quad (2)$$

اگر بهبودی پس از به‌روزرسانی مشاهده نشود، مقدار جواب $X_{i,w}$ به‌صورت تصادفی مقداردهی می‌شود. این عمل در هر گروه تکرار می‌شود تا به یک تکرار خاص برسد. بعد از جست‌وجو در عمق هر گروه، جمعیت قورباغه وارد شده، دوباره ترکیب می‌شوند و سپس مرتب خواهند شد. بهترین مکان قورباغه، بر اساس بهترین شایستگی ذخیره می‌شود. جمعیت، دوباره به گروه‌های مختلف تقسیم‌بندی می‌شوند و جست‌وجوی عمقی محلی دوباره بر روی آن‌ها اجرا می‌شود و جست‌وجو زمانی به پایان می‌رسد، که تمام شرایط ارضا می‌شوند [۱۷-۱۵].



شکل ۲: فلوچارت راهکار پیشنهادی

	task1	task2	task3	task4	taskreqTasks			
frog1	VM9	VM1	VM8	VM2	0	0	0	VM9
frog2	VM2	VM9	VMreqVms	VM3	0	0	0	VM15
frog3	VM3	VM1	VM4	VM1	0	0	0	VM20
frog4	VM5	VMreqVms	VM11	VM8	0	0	0	VM15
	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	
frogrf	VMreqVms	VM14	VM14	VM10	0	0	0	VMreqVms

شکل ۳: ماتریس مربوط به روش پیشنهادی

۴-۱- زمان‌بندی کارها بر اساس الگوریتم پرش ترکیبی قورباغه

ماشین‌های مجازی که در یک میزبان قرار می‌گیرند، به‌عنوان هم‌سایه یکدیگر در نظر گرفته می‌شوند، به‌طوری‌که هر ماشین مجازی در مکان j با ماشین مجازی در مکان $j+1$ هم‌سایه است.

ب- بعد از ورود به محیط ابر، کارهای مورد نظر وارد واحد زمان‌بندی می‌شوند. فرآیند زمان‌بندی کارها در این واحد، با هدف کاهش هزینه محاسبات در محدوده مهلت تعیین شده انجام می‌شود و از تابع هزینه برای امتیازدهی به هر عامل استفاده می‌شود. این پژوهش با توجه به اینکه بر اساس روش بهینه‌سازی عمل کرده است، هر عامل نماینده یک ماشین مجازی است. الگوریتم، طی دو فاز انجام می‌شود؛ به‌طور کلی در فاز اول تعیین می‌شود که زمان پاسخ هر منبع در محدوده زمانی معین شده قرار گیرد. در فاز دوم با در نظر گرفتن محدوده زمانی، هزینه اجرا کاهش می‌یابد.

ج- بعد از اینکه واحد زمان‌بند کارها، کارها را به ماشین‌های مجازی اختصاص داد، واحد زمان‌بند ماشین مجازی، زمان‌بندی ماشین‌های مجازی به ماشین‌های فیزیکی را انجام می‌دهد. در این واحد باید وضعیت ماشین‌های فیزیکی تعیین شوند. پس به همین دلیل در ابتدا یک آستانه محاسبه می‌شود که در بخش ۴ به‌تفصیل توضیح داده می‌شود؛ سپس بر اساس این آستانه، وضعیت ماشین‌های فیزیکی تعیین می‌شوند. برای به‌دست آوردن این آستانه در ابتدا، جمع کل بهره‌وری واحد پردازش مرکزی^۷ ماشین‌های مجازی روی ماشین فیزیکی^۸ محاسبه می‌شود. در نهایت برای هر کدام از ماشین‌های فیزیکی، جمع کل بهره‌وری واحد پردازش مرکزی ماشین‌های مجازی روی آن ماشین فیزیکی با آستانه محاسبه شده مقایسه می‌شود. اگر مجموع کل بهره‌وری واحد پردازش مرکزی ماشین‌های مجازی روی آن ماشین فیزیکی بیشتر از حد آستانه باشد، ماشین فیزیکی در حالت اضافه بار قرار دارد و ماشین‌های مجازی مربوط به این ماشین فیزیکی باید وارد تابع مهاجرت شوند. در غیر این صورت مرحله بعدی اجرا می‌شود.

د- در پایان بعد از اجرای مرحله سوم، با توجه به فلوچارت شکل ۲ کارهای مربوط به کاربران روی منابع مورد نظر اجرا می‌شوند.

۴- زمان‌بندی کارها به کمک الگوریتم پرش ترکیبی قورباغه و

ترکیب مؤثر ماشین‌های مجازی

شکل ۲ فلوچارت زمان‌بندی کارها و ترکیب مؤثر ماشین‌های مجازی مبتنی بر الگوریتم پرش قورباغه را نشان می‌دهد. مراحل مختلف این الگوریتم، به‌تفصیل در ادامه بخش توضیح داده شده است:

هر قورباغه بیانگر زوج مرتب $\langle VM_i, T_j \rangle$ حاصل ماشین مجازی

نام و کار نام است. فرآیند انطباق SFLA و منابع ابر در روش پیشنهادی

به کمک ماتریس شکل ۳ توصیف می‌شود. طبق شکل ۳ در ابتدا کار نام

($1 \leq j \leq reqTasks$) به‌صورت تصادفی توسط قورباغه ($1 \leq k \leq f$)

روی ماشین مجازی نام ($1 \leq i \leq reqVms$) قرار می‌گیرد.

سیس این ماتریس توسط زمان‌بندی کارها بر اساس الگوریتم SFLA

تغییر می‌یابد. عمل زمان‌بندی کارها با هدف به حداقل رساندن هزینه

محاسبات در محدوده مهلت تعیین شده انجام می‌شود.

مساوی است. در این فاز باید هزینه اجرا با در نظر گرفتن مهلت زمانی کاهش داده شود. الگوریتم این فاز نیز به صورت مرحله‌ای اجرا می‌شود، به‌طوری‌که در هر مرحله هر ماشین مجازی کوچک‌ترین کار لیست خود را پیدا کرده و اگر هر یک از ماشین‌های مجازی همسایه کار مورد نظر را با هزینه کمتر در محدوده زمانی انجام دهد، آن کار را به لیست آن همسایه منتقل می‌کند.

بهبود SFLA با تنظیم پویای گام جهش

در حقیقت تغذیه قورباغه در SFLA، اشاره به دیگر قورباغه‌ها دارد. چشم انداز جهش قورباغه بین مکان فعلی قورباغه و مکان بهترین غذا (بیشترین شایستگی قورباغه) است. یک مکان با غذای بیشتر، حول مکان با حداکثر غذا در جست‌وجوی فعلی قرار دارد. SFLA معمولاً، قادر به تعیین این منطقه نیست؛ در نتیجه با محدود کردن این بازه در این منطقه، توانایی بهینه‌سازی و اثرگذاری بر سرعت همگرایی این الگوریتم را پیدا می‌کند. برای تکرار k ام با توجه به محدود کردن بازه جست‌وجو رابطه ۴ پیشنهاد می‌شود:

$$D_{i,w}(k) = r \times c \times (X_b - X_{i,w}(k)) \quad (4)$$

در رابطه بالا، c ، برابر ۱ است. این رابطه باعث افزایش جهش قورباغه در هر مرحله می‌شود و مقدار جهش را گسترش می‌دهد که، باعث بهبود بهینه‌سازی این الگوریتم می‌شود. این الگوریتم می‌تواند با گسترش مقدار c ، محدوده جست‌وجوی بیشتری را تحت پوشش قرار دهد. این مقدار نمی‌تواند به صورت نامحدود گسترش داده شود و اگر به صورت نامحدود گسترش پیدا کند، عملکرد موقعیت بهینه فعلی تضعیف خواهد شد. برای کنترل بهتر رابطه ۴ یک وزن W به آن اضافه می‌شود، که این مقدار، رابطه بین نقطه مطلوب محلی و نقطه مطلوب سراسری را در پروسه جهش قورباغه به صورت رابطه ۵ بیان می‌کند:

$$D_{i,w}(k) = W \times r \times c \times (X_b - X_{i,w}(k)) \quad (5)$$

در رابطه بالا، پارامتر W تأثیر قابل توجهی در رفتار همگرایی الگوریتم دارد. با مقدار بزرگ پارامتر W توانایی الگوریتم برای پیدا کردن نقطه سراسری در جست‌وجو زیاد می‌شود و توانایی پیدا کردن نقطه محلی ضعیف خواهد شد. تأثیر سرعت قبلی روی سرعت فعلی الگوریتم را می‌توان با تنظیم W به شکل رابطه ۶ کنترل کرد.

$$W = W_{min} + (k_{max} - k) \times (W_{max} - W_{min}) / k_{max} \quad (6)$$

بنابراین مقدار W رفته رفته از مقدار W_{max} در یک پروسه تکراری خطی به مقدار حداقل W_{min} خواهد رسید. توانایی جست‌وجو در یک پروسه تکرار الگوریتم، قوی است و این الگوریتم قادر به جست‌وجو در یک فضای بزرگ جواب خواهد بود و مناطق جدید به صورت مداوم برای پیدا کردن جواب، بررسی خواهند شد. از دیدگاه تکرار متناوب، الگوریتم به تدریج محدوده جست‌وجوی خود را به یک منطقه کاهش می‌دهد و بدین ترتیب سرعت همگرایی افزایش پیدا خواهد کرد. پس به این ترتیب رابطه ۵ که به کمک W محاسبه شده است در رابطه ۶ به‌روزرسانی می‌شود.

در ابتدا، کارها به صورت تصادفی با توزیع یکنواخت، بین ماشین‌های مجازی توزیع می‌شوند. جمعیت به چندین memplex تقسیم‌بندی می‌شود، که هر کدام شامل چندین قورباغه است. تفاوت memplexها در گروه‌هایشان است، که رفتار هر کدام، یکدیگر را تحت تأثیر قرار می‌دهند. تمام memplexها در یک فرمی به عنوان جمعیت جدید ترکیب می‌شوند و بعد از آن، جست‌وجوی محلی برای هر کدام از آن‌ها پیاده‌سازی می‌شود. پروسه جست‌وجو در این الگوریتم، از استراتژی نخبگان^۹، برای به‌روزرسانی اطلاعات آن مکان در الگوریتم مدل ابری استفاده می‌کند. متناوب بودن جست‌وجوی محلی و سراسری تا زمانی که اطلاعات به همگرایی مورد نظر برسند، ادامه پیدا می‌کند، که به عنوان یک همگرایی خاص با حداکثر میزان دقت تلقی می‌شود. برخی از الگوریتم‌های فرا ابتکاری به آسانی در نقطه بهینه محلی گیر خواهند افتاد، زمانی که این مسئله توسط تابع بهینه حل شود، در دقت و همگرایی سرعت مسئله اثر می‌گذارد؛ بنابراین این روش، فضای جست‌وجو را تقسیم‌بندی می‌کند، که این باعث بهبود وضعیت نقطه محلی خواهد شد؛ در نتیجه، سرعت و دقت همگرایی بهبود می‌یابد.

این الگوریتم، زمان‌بندی کارها را طی دو فاز انجام می‌دهد؛ به‌طور کلی در فاز اول تعیین می‌شود که زمان پاسخ هر منبع در محدوده زمانی معین شده قرار گیرد. در فاز دوم با در نظر گرفتن محدوده زمانی، هزینه اجرا کاهش می‌یابد.

الف - فاز اول تعیین زمان پاسخ هر منبع

ابتدا کارها به‌طور تصادفی با توزیع یکنواخت بین ماشین‌های مجازی توزیع شده و هر کار در داخل لیست مربوط به ماشین مجازی قرار می‌گیرد و زمان پاسخ هر ماشین مجازی با توجه به لیست کارهای انتساب شده، طبق رابطه ۳ محاسبه می‌شود.

$$R = \text{MeanRespTime} / (\text{noofHosts} * \text{reqTask}) \quad (3)$$

حالا الگوریتم به صورت مرحله‌ای اجرا می‌شود، تا اینکه حداکثر زمان پاسخ ماشین‌های مجازی از مهلت زمانی معین شده کمتر یا مساوی شود. در هر مرحله هر ماشین مجازی کوچک‌ترین کار لیست خود را پیدا خواهد کرد و در صورتی که آن کار توسط ماشین مجازی همسایه در زمان کمتر اجرا شود، آن را به لیست ماشین مجازی همسایه انتقال خواهد داد؛ سپس زمان پاسخ هر ماشین مجازی را با توجه به لیست کارهای جدید انتقال یافته محاسبه می‌کند. در صورتی که زمان پاسخ حداکثر در بین ماشین‌های مجازی از محدوده زمانی معین شده کمتر باشد، فاز اول پایان یافته است. در غیر این صورت مرحله بعدی همین فاز اجرا می‌شود. قطعاً باید در هر مرحله زمان پاسخ هر ماشین مجازی کمتر یا مساوی مرحله قبل گردد، به دلیل اینکه هر ماشین مجازی کوچک‌ترین کار خود را انتخاب کرده و در صورتی که همسایه آن ماشین مجازی، کار را سریع‌تر انجام دهد، آن کار را به ذره همسایه خود منتقل می‌کند.

ب- فاز دوم کاهش هزینه اجرا

حالا هر ماشین مجازی تعدادی کار در داخل لیست خودش دارد که، زمان پاسخ هر ماشین مجازی از مهلت زمانی معین شده کمتر و یا

برای کاهش مصرف انرژی مؤثر است. پس از آن که تمام ماشین‌های مجازی منتقل شدند، میزبان برای صرفه‌جویی در انرژی به حالت خواب یا آماده‌باش می‌رود.

۴-۲-۱- مدیریت میزان مصرف انرژی به ازای هر ماشین میزبان

میزبان مصرف انرژی میزبان در مراکز داده ابری عمدتاً ناشی از مصرف پردازنده، حافظه، دیسک، سیستم خنک‌کننده و دیگر ماژول‌های سخت‌افزاری است. با افزایش محبوبیت پردازنده‌های چند هسته‌ای، مصرف انرژی این پردازنده‌ها، آن‌ها را تبدیل به یک جزء مصرف‌کننده انرژی کرده است. بر اساس مطالعات kusic و همکاران [۱۸]، مصرف انرژی میزبان، نسبت تقریباً خطی با مصرف انرژی پردازنده دارد. در این روش یک نسخه پشتیبان^{۱۰} از فایل‌ها و اطلاعات گرفته می‌شود و در شبکه متصل به ماشین مجازی ذخیره می‌شود؛ بنابراین حافظه ماشین مجازی در طول مهاجرت کپی می‌شود و نیازی به کپی اطلاعات پس از ترکیب مؤثر ماشین مجازی نخواهد بود. میزان انرژی که میزبان برای انجام کارها مصرف می‌کند از رابطه ۹ به دست می‌آید.

$$E^{(T)}(h) = E^{(T)}_w(h) + E^{(T)}_m(h) \quad (9)$$

انرژی مصرفی میزبان در تمام دوره‌های زمانی به کمک رابطه ۱۰ بیان می‌شود.

$$E(h) = \sum_{\forall T} E^{(T)}(h) \quad (10)$$

۴-۲-۲- تعیین میزان قرارداد سطح سرویس و نقض آن

کیفیت خدمات که اغلب توسط قراردادهای سطح سرویس ارائه شده است، قابل دستیابی است و این یک نیاز مهم برای سیستم‌های رایانش ابری است. قراردادهای سطح سرویس مشخص، اغلب در توانایی محاسبات، زمان واکنش و حداکثر ظرفیت بازدید باهم تفاوت دارند. میزان نقض قرارداد سطح سرویس در رابطه ۱۱ نشان داده شده است [۱۰]:

$$SLAD = \frac{T_s}{T_a} \times \frac{C_d}{C_r} \quad (11)$$

اولین آیتم $\frac{T_s}{T_a}$ در رابطه بالا، نقض قرارداد سطح سرویس و مقدار سربار بر روی پردازنده میزبان را نشان می‌دهد. در این معادله T_s طول زمان استفاده صد درصدی از میزبان را نشان می‌دهد و T_a اندازه دوره زمانی را نشان می‌دهد. میزان استفاده صد درصد از میزبان به این معنی است که، میزبان دارای بار اضافی است. یک نقض قرارداد سطح سرویس زمانی اتفاق می‌افتد که، زمان اجرا برای دسترس برنامه جدید دشوار باشد؛ یعنی زمانی که میزبان دارای اضافه بار باشد. دومین آیتم $\frac{C_d}{C_r}$ در رابطه ۱۱، میزان تخریب کارایی (کاهش عملکرد میزبان) در هنگام مهاجرت ماشین مجازی است. در این آیتم C_d مقدار MIPS (تعداد میلیون دستور اجرا شده در هر ثانیه) استفاده شده در هنگام مهاجرت VM است. C_r مقدار MIPS کل اعمال شده توسط VM‌ها بر روی میزبان است.

۴-۲- استراتژی زمان‌بندی برای ماشین مجازی روی میزبان

کاربران باید بتوانند از سرویس ماشین‌های مجازی که بر روی مرکز داده نصب شده است، استفاده نمایند. پس از رسیدن به قراردادهای سطح سرویس، مرکز داده ابر بلافاصله ماشین مجازی را به میزبان مناسب اختصاص می‌دهد.

با زمان‌بندی ارائه‌شده، مدیریت منابع، ماشین مجازی به میزبان داده می‌شود و آن ماشین مجازی می‌تواند نیازهای قرارداد سطح سرویس را برآورده کند و به همین ترتیب در انرژی صرفه‌جویی خواهد شد.

در طول هر دوره زمانی، ماشین‌های مجازی با توجه به استراتژی کاهش تعداد مهاجرت انتخاب می‌شوند و در میزبان خود ترکیب مؤثر می‌شوند. برای به دست آوردن نقض قرارداد سطح سرویس، آستانه استفاده از پردازنده بر اساس رابطه ۷ به دست می‌آید:

$$NewUpper_Thrsh = \frac{R_{SLA}}{r_{SLA}} \times Upper_Thrsh \quad (7)$$

آستانه جدید استفاده از پردازنده، به عنوان ضرب آستانه فعلی بر نسبت قراردادهای سطح سرویس برآورد می‌شود. این مقدار آستانه جدیدی که تولید می‌شود، جایگزین آستانه قبلی خواهد شد. مقدار $Upper_Thrsh$ نمی‌تواند تا به نهایت کاهش یا افزایش یابد. اگر مقدار $Upper_Thrsh$ بسیار زیاد شود و کیفیت خدمات برای کاربر تأمین نشود، در این حالت ترکیب مؤثر ماشین مجازی اتفاق نمی‌افتد، چون مقدار استفاده از واحد پردازش مرکزی نزدیک به ۱۰۰ درصد است. این وضعیت بدان معنی است که، میزبان بار اضافی دارد. از سوی دیگر مقدار $Upper_Thrsh$ نباید بسیار کم باشد، چون مصرف انرژی بالا می‌رود.

مرحله بعدی، ارزیابی نقض قرارداد سطح سرویس در میزبان فعلی است و تعیین اینکه آیا نیازهای قرارداد سطح سرویس در این میزبان تأمین می‌شود یا خیر؛ چنانچه این خواسته برآورده نشود، ماشین مجازی انتخاب شده و با توجه به رابطه ۸ این ماشین مجازی به لیست مهاجرت اضافه می‌شود. بلافاصله این ماشین مجازی آماده است که، از میزبان مذکور مهاجرت کند.

$$vm = \{v | v \in h \cup m, \sum_{h \in h \cup m} Utilz(h) - \sum Utilz(v) < Upper_Thrsh, |v| \rightarrow \min\} \quad (8)$$

ماشین‌های مجازی که نیاز به مهاجرت دارند، توسط استراتژی حداقل تعداد ماشین‌های مجازی مهاجر که همان معیار تابع هزینه است، انتخاب می‌شوند. هدف این استراتژی کاهش تخریب ناشی از مهاجرت ماشین‌های مجازی است که، منجر به کاهش عملکرد سیستم می‌شود. هنگامی که مجموع میزان استفاده از پردازنده تمام ماشین‌های مجازی که باقی مانده‌اند، کمتر از میزان حد آستانه بعد از مهاجرت باشد، نیازهای خاص قرارداد سطح سرویس برآورده خواهد شد. روش پیشنهادی تمام ماشین‌های مجازی که بهره‌وری از واحد پردازنده مرکزی آن‌ها کمتر از مقدار آستانه است، را منتقل می‌کند، که این کار

۳-۴- بررسی پیچیدگی الگوریتم پیشنهادی

فاز ۱ و ۲ دارای مرتبه زمانی $O(s \times nIter \times m \times N_t \times K \times N_s)$ است، که s تعداد کل قورباغه‌ها، $nIter$ تعداد کل تکرارها، m تعداد memplex ها، N_t تعداد تکرار در جست‌وجوی محلی، K تعداد قورباغه‌های انتخاب شده به‌صورت تصادفی و N_s تعداد تکرار در جست‌وجوی سراسری است. فاز ۳ دارای مرتبه زمانی $O(M \times N)$ است که در آن M به تعداد میزبان‌ها و N به تعداد ماشین‌های مجازی هر میزبان اشاره دارد.

۵- شبیه‌سازی و بررسی نتایج

CloudSim یک ابزار و یک بسته شبیه‌سازی پردازی رایانه‌ای برای شبیه‌سازی پردازی ابری است که، توسط آزمایشگاه دانشگاه ملیبورن راه‌اندازی شد و پروژه GridBus در سال ۲۰۰۹ با استفاده از این ابزار مورد مطالعه قرار گرفت [۱۹]. این ابزار بر اساس مدل برنامه‌نویسی GridSim از تحقیقات رایانش ابری و توسعه آن‌ها پشتیبانی می‌کند و ایجادکننده ویژگی‌های پردازی‌های ابری و مدیریت زمان‌بندی آن‌ها و زمان‌بندی شبیه‌سازی این پردازی‌ها است. الگوریتم زمان‌بندی پرش ترکیبی قورباغه توسط کاربران می‌تواند، در مراکز داده الگوریتم زمان‌بندی را شبیه‌سازی کند.

۵-۱- پارامترهای شبیه‌سازی

مقادیر مورد استفاده در آزمایش‌ها و شبیه‌سازی و الگوریتم‌ها در جداول ۳ و ۲ آورده شده‌اند. جدول ۲، مربوط به پارامترهای استفاده شده در الگوریتم زمان‌بندی پیشنهادی است. جدول ۳، پارامترهای استفاده شده در شبیه‌ساز Cloudsim را نشان می‌دهد.

برای ارزیابی نتایج از مجموعه داده benchmark تهیه شده در مرجع [۲۰] استفاده شده است

جدول ۲- پارامترهای استفاده شده در الگوریتم

زمان‌بندی پیشنهادی		
پارامتر	مفهوم پارامتر	مقدار
Tmax	تعداد نسل‌ها	۱۰۰
M	تعداد memplex	۱۰
N	تعداد قورباغه‌ها در هر memplex	۱۰
F	تعداد کل قورباغه‌ها	۱۰۰
Wmin	حداقل وزن پرش	۰/۹
Wmax	حداکثر وزن پرش	۳/۲
Iteration	تعداد کل تکرارها	۱۰۰

جدول ۳- پارامترهای استفاده شده در شبیه‌سازی cloudsim

موجودیت‌ها	پارامترها	مقادیر
ماشین‌های مجازی (virtual machine)	حافظه (ram)	۵۱۲
	پهنای باند	۵۰۰-۱۰۰۰
	زمان‌بندی کارها	Space-share
	تعداد عناصر پردازشی (PE)	۱-۴
	تعداد مراکز داده	۱۰
	حداکثر تعداد میزبان‌ها	۵۰۰
مرکز داده	زمان‌بندی ماشین مجازی	Time-share

۵-۲- سناریوهای آزمایش

برای بررسی کارایی الگوریتم پیشنهادی، روی سه الگوریتم SFLA، [۱۰] ACO و [۹] GA تست و آزمایش انجام شده است. آزمایش‌هایی که روی این سه الگوریتم اجرا شده است، چهار سناریوی است که، در جدول ۴ نشان داده شده است. در این سناریوها، زمان اجرا به‌صورت تصادفی تولید شده‌اند. سناریو ۱ کمترین تعداد کار و منابع و سناریو ۴ بیشترین تعداد کار و منابع را دارد.

جدول ۴- سناریوهای آزمایش

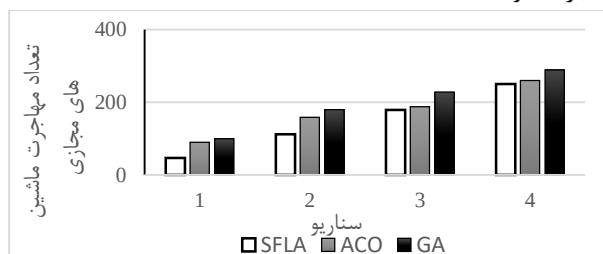
نام سناریو	تعداد کارها (وظایف)	تعداد منابع (ماشین مجازی)
سناریو ۱	۵۰۰	۱۵۰
سناریو ۲	۲۰۰۰	۳۵۰
سناریو ۳	۵۰۰۰	۳۵۰
سناریو ۴	۱۰۰۰۰	۸۰۰

این آزمایش‌ها با استفاده از یک پردازنده چهار هسته‌ای از خانواده پردازنده‌های اینتل و حافظه چهارگیگابایتی و سیستم عامل ویندوز ۷ انجام گردیده است.

۵-۳- مقایسه مصرف انرژی، تعداد مهاجرت‌ها، نقض قرارداد

سطح سرویس و میانگین تعداد ماشین‌های فیزیکی خاموش

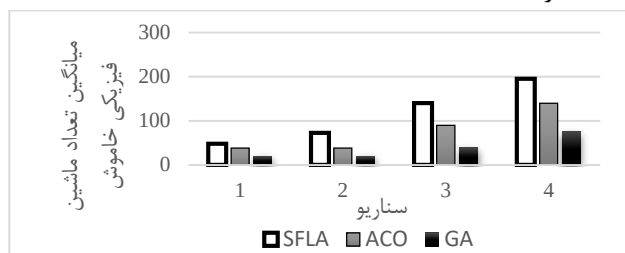
شکل ۱، تعداد مهاجرت ماشین‌های مجازی برای سناریوهای ۱ الی ۴ روی الگوریتم‌های SFLA، ACO و GA نشان می‌دهد. از آن جایی که تعداد تعویض میزبان بیکار یا کم استفاده در الگوریتم SFLA، نسبت به دو الگوریتم ACO و GA کمتر است، از این‌رو همان‌طور که در شکل ۱ مشاهده می‌شود، ماشین‌های مجازی مهاجرت کمتری در الگوریتم SFLA نسبت به دو الگوریتم دیگر دارند. با مشاهده شکل ۱ می‌بینیم، که الگوریتم پیشنهادی نسبت به الگوریتم ACO در سناریو ۱ و ۲ نیز کاهش تعداد مهاجرت نسبتاً کمی داشته است. در سناریوهای ۳ و ۴ نیز تعداد مهاجرت‌های هر دو الگوریتم، حدوداً شبیه به یکدیگر می‌باشند، بنابراین ACO، در کارهای با حجم زیاد به‌درستی کار می‌کند و در کارهای با حجم کم (سناریو ۱ و ۲) به‌درستی کار نمی‌کند، اما SFLA در هر دو حالت خوب کار می‌کند و تعداد مهاجرت‌ها را نسبت به دو الگوریتم دیگر کاهش می‌دهد که موجب کاهش سربار مهاجرت خواهد شد.



شکل ۱: تعداد مهاجرت ماشین‌های مجازی در الگوریتم‌های SFLA، ACO و GA برای سناریوهای مختلف

در شکل ۲، میزان مصرف انرژی برحسب وات برای سناریوهای ۱ الی ۴ روی الگوریتم‌های SFLA، ACO و GA نشان داده شده است.

به هم می‌باشند، اما در سناریو ۲ و ۳ الگوریتم پیشنهادی میانگین تعداد ماشین‌های فیزیکی خاموش بیشتری نسبت به الگوریتم ACO دارد.



شکل ۴: میانگین تعداد ماشین فیزیکی خاموش در الگوریتم‌های

SFLA، ACO و GA برای سناریوهای مختلف

با مقایسه شکل ۲ و ۴ مشاهده می‌شود که، در سناریوی ۲ و ۳ الگوریتم پیشنهادی از نظر مصرف انرژی نسبت به الگوریتم ACO نمودارهای آن‌ها نزدیک به هم می‌باشند، اما این الگوریتم از نظر میانگین تعداد ماشین‌های فیزیکی خاموش نسبت به الگوریتم ACO افزایش نسبتاً خوبی داشته است؛ همچنین در سناریوی ۴، الگوریتم پیشنهادی از نظر انرژی مصرفی و میانگین تعداد ماشین‌های فیزیکی خاموش به یک میزان نسبت به الگوریتم ACO کاهش و افزایش داشته است.

با مقایسه شکل‌های ۳ و ۴ مشاهده می‌شود که، در سناریوی ۲ و ۳ الگوریتم پیشنهادی از نظر میزان نقض قرارداد سطح سرویس نسبت به الگوریتم ACO تقریباً در یک سطح هستند، اما این الگوریتم از نظر میانگین تعداد ماشین‌های فیزیکی خاموش نسبت به الگوریتم ACO افزایش نسبتاً خوبی داشته است، همچنین در سناریوی ۴ الگوریتم پیشنهادی از نظر میزان نقض قرارداد سطح سرویس و میانگین تعداد ماشین‌های فیزیکی خاموش به یک میزان نسبت به الگوریتم ACO کاهش و افزایش داشته است.

۵-۴- بررسی همگرایی الگوریتم‌ها

برای بررسی همگرایی الگوریتم‌ها نیاز است تا تابع هزینه (رابطه ۱۲) محاسبه شود.

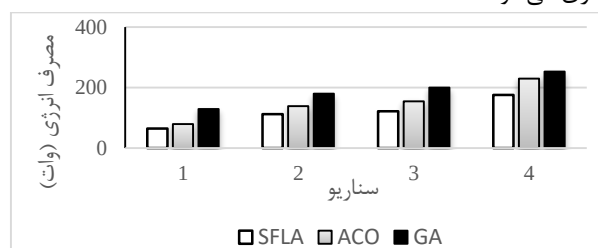
$$costfunction_{i,j} = \frac{TL_Task_i}{\frac{PE_NUM_i \times PE_MIPS_VM_j}{inputfilesize} + \frac{BW_VM_j}{BW_VM_j}} \quad (12)$$

تابع هزینه، هم برای کار و هم برای ماشین مجازی یکسان در نظر گرفته شده است و در هر سه الگوریتم یکسان محاسبه می‌شود. $PE_MIPS_VM_j$ ، تعداد دستور اجرا شده در هر ثانیه است. TL_Task_i طول کار i ام، PE_NUM_i تعداد عناصر پردازشی در ماشین مجازی j ام و BW_VM_j پهنای باند ماشین مجازی j ام است. $inputfilesize$ سایز فایل ورودی، قبل از اجراست. عملیات بهینه‌سازی بر اساس این تابع هزینه انجام شده است.

شکل ۵، فرآیند همگرایی الگوریتم‌ها را بر اساس محاسبه تابع هزینه برحسب تعداد تکرارها نشان می‌دهد. جهت داشتن دید کلی، در

الگوریتم SFLA به دلیل تعداد مهاجرت‌ها منجر به کاهش سربار مهاجرت و کاهش زمان اجرا و نهایتاً کاهش مصرف انرژی نسبت به دو الگوریتم دیگر می‌شود. الگوریتم پیشنهادی نسبت به الگوریتم GA در سناریوهای ۱ الی ۴ کاهش انرژی چشم‌گیری داشته است.

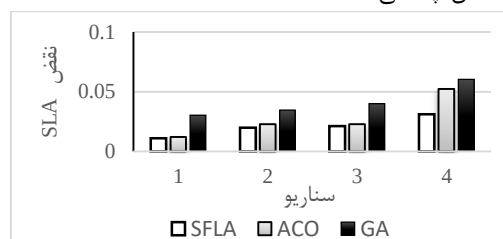
نتایج نشان می‌دهند، در صورتی که مقدار حدود آستانه بسیار کوچک باشد، می‌تواند از حالت خاموش شدن بیش از حد میزبان جلوگیری کند. مقدار حدود آستانه نباید بیش از حد بزرگ باشد؛ زیرا باعث تعویض مکرر میزبان بین حالت‌های بیداری و خاموشی خواهد شد، که این حالت باعث افزایش مصرف انرژی می‌شود؛ همچنین یک مقدار حد آستانه بزرگ باعث ایجاد مشکل در تخصیص ماشین‌های مجازی می‌شود.



شکل ۲: انرژی مصرفی در الگوریتم‌های SFLA، ACO و GA برای

سناریوهای مختلف

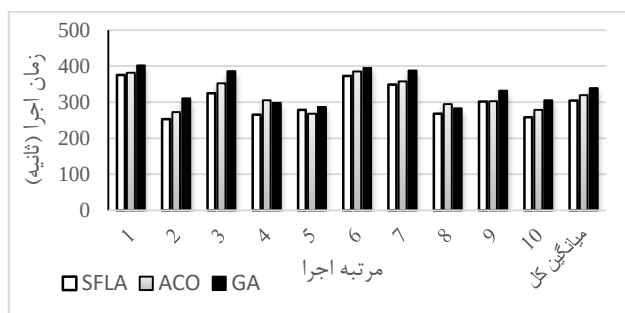
شکل ۳، سطح نقض قرارداد سطح سرویس برای سناریوهای ۱ الی ۴ روی الگوریتم‌های SFLA، ACO و GA را نشان می‌دهد. از آنجایی که SFLA با استفاده از پیش‌بینی ماشین‌های فیزیکی دارای بار اضافی، تضمین می‌کند، که ماشین فیزیکی مقصد زمانی که یک ماشین مجازی به این ماشین فیزیکی مهاجرت می‌کند، در حالت اضافه بار نخواهد بود. همان‌طور که در شکل ۳ مشاهده می‌شود، SFLA نسبت به دو الگوریتم دیگر کاهش سطح نقض قرارداد سطح سرویس خواهد داشت. این الگوریتم نسبت به الگوریتم GA کاهش سطح نقض قرارداد سطح سرویس چشم‌گیری داشته است، اما نسبت به الگوریتم ACO کاهش چندانی نداشته است.



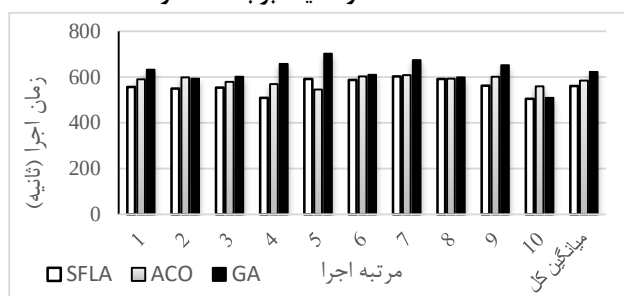
شکل ۳: میزان نقض قرارداد سطح سرویس در الگوریتم‌های

SFLA، ACO و GA برای سناریوهای مختلف

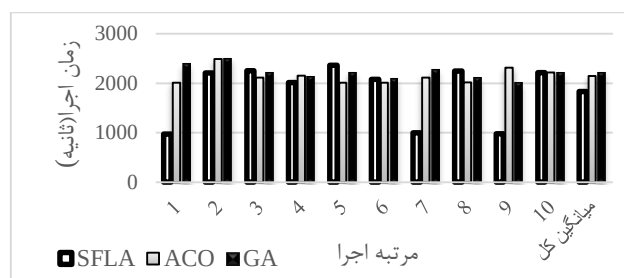
در شکل ۴ میانگین تعداد ماشین‌های فیزیکی خاموش برای سناریوهای ۱ الی ۴ روی الگوریتم‌های SFLA، ACO و GA نشان داده شده است. همان‌طور که در شکل ۴ مشاهده می‌شود، الگوریتم پیشنهادی نسبت به دو الگوریتم دیگر افزایش میانگین تعداد ماشین‌های فیزیکی خاموش را خواهد داشت. در سناریو ۱ میانگین تعداد ماشین فیزیکی خاموش در ACO و الگوریتم پیشنهادی شبیه



شکل ۷: زمان اجرای ۱۰ تکرار اول برای الگوریتم‌های SFLA، ACO و GA در محیط ابر با ۲۰۰۰ کار



شکل ۸: زمان اجرای ۱۰ تکرار اول برای الگوریتم‌های SFLA، ACO و GA در محیط ابر با ۵۰۰۰ کار



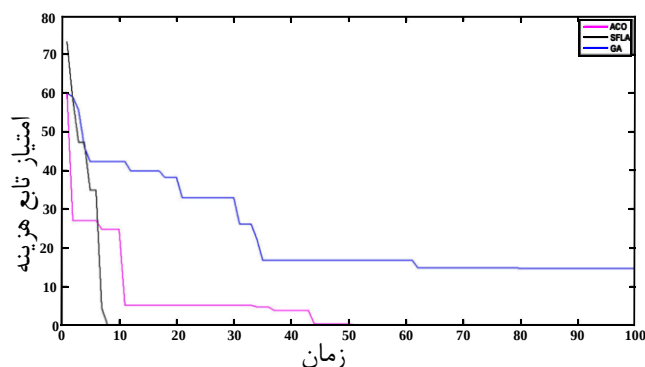
شکل ۹: زمان اجرای ۱۰ تکرار اول برای الگوریتم‌های SFLA، ACO و GA در محیط ابر با ۱۰۰۰۰ کار

با افزایش تعداد وظایف در فضای ابر، عملکرد روش SFLA به روش ACO نزدیک می‌گردد و روش SFLA فاصله بیشتری در تعداد وظایف کم با روش ACO دارد.

همان‌طور که در شکل ۵ مشاهده شد، الگوریتم پیشنهادی نسبت به دو الگوریتم دیگر بهترین زمان همگرایی را از خود نشان می‌دهد. برای این منظور میانگین makespan هر سه الگوریتم مورد مطالعه، برای تعداد کارهای متفاوت آزمایش شده است. شکل ۱۰، میانگین makespan بر تعداد کارها برای الگوریتم‌های SFLA، ACO و GA نشان می‌دهد. همان‌طور که در شکل ۱۰ مشاهده می‌شود، الگوریتم SFLA نسبت به الگوریتم‌های ACO و GA میانگین makespan کمتری دارد، یعنی مدت زمان تکمیل کارها در این الگوریتم پیشنهادی بهتر از دو الگوریتم دیگر است. همان‌طور که در شکل ۱۰ دیده می‌شود، در الگوریتم پیشنهادی، makespan در مقایسه با الگوریتم‌های ACO و GA کمتر شده است، اما کاهش چشم‌گیری ندارد؛ دلیل این مسئله

خصوص نحوه همگرایی الگوریتم‌ها، تغییرات بهترین جواب‌های الگوریتم‌های SFLA، ACO و GA در ۱۰۰ تکرار اول، برای سناریوی اول آورده شده است. در شکل ۵، همگرایی الگوریتم‌های SFLA، ACO و GA آورده شده است، تا بهترین جواب‌ها با هم مقایسه گردد. همان‌طور که در رابطه ۶ بخش ۴ بیان شد، الگوریتم SFLA، از دیدگاه تکرار متناوب، به تدریج محدوده جست‌وجوی خود را به یک منطقه کاهش می‌دهد و بدین ترتیب سرعت همگرایی افزایش پیدا می‌کند. طبق شکل ۵، الگوریتم SFLA در ۹ تکرار همگرا می‌شود، اما ACO در ۴۴ تکرار همگرا می‌شود؛ بنابراین الگوریتم SFLA در تعداد تکرارهای کمتری همگرا شده است و این بدان معنا است که، سرعت همگرایی روش پیشنهادی نسبت به سرعت همگرایی الگوریتم‌های ACO و GA بسیار قابل قبول‌تر است.

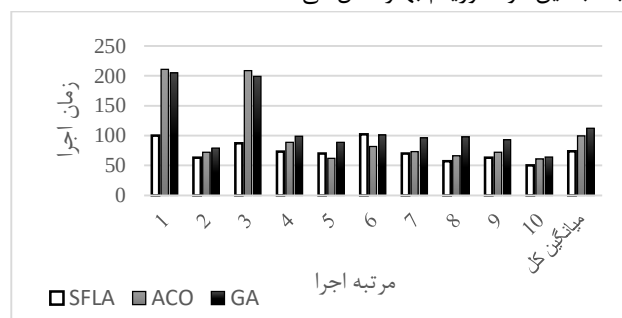
در الگوریتم پیشنهادی، زمانی که جواب بهینه فعلی هیچ بهبودی در مقدار تابع هزینه در تکرارهای مداوم نشان ندهد، الگوریتم فرض بر این می‌کند که، همگرایی لازم حاصل شده و اجرای برنامه پایان می‌یابد.



شکل ۵: نمایش همگرایی الگوریتم‌های SFLA، ACO و GA در سناریو اول

۵-۵- مقایسه میانگین زمان‌های اجرا

شکل‌های ۶، ۷، ۸، ۹ زمان اجرای ۱۰ تکرار اول برای الگوریتم‌های SFLA، ACO و GA در محیط ابر با تعداد کارهای متفاوت را نشان می‌دهند. تمامی نتایج برحسب ثانیه می‌باشند. نتایج به‌دست‌آمده از شکل‌های ۶، ۷، ۸، ۹ نشان می‌دهند که، الگوریتم SFLA زمان کل اجرای کارها را نسبت به دو الگوریتم ACO و GA بهبود داده است و نسبت به این دو الگوریتم بهتر عمل می‌کند.



شکل ۶: زمان اجرای ۱۰ تکرار اول برای الگوریتم‌های SFLA، ACO و GA در محیط ابر با ۵۰۰ کار

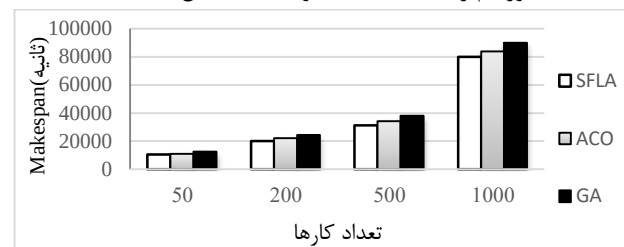
به آسانی قابلیت به‌کارگیری در حل دیگر مسائل بهینه‌سازی ترکیبی را نیز دارد. الگوریتم پرش ترکیبی قورباغه به دنبال کاهش هزینه‌های مربوط به سربراشی از عدم توازن بار در سیستم‌های محاسباتی است، به‌طوری‌که کاهش مصرف انرژی و رضایت کاربران با افزایش توافق‌نامه سطح سرویس را در کنار هم داشته باشد. تمرکز ما بر طراحی و پیاده‌سازی یک چارچوب نرم‌افزاری شامل سرویس‌های پیش‌بینی به‌صورت پویا، با استفاده از سرویس‌های تأمین تطبیقی و مدل‌های زمان‌بندی چند مرحله‌ای است، به‌طوری‌که منابع به‌طور مؤثر برای بهبود مصرف انرژی در مراکز داده اختصاص داده شوند و در جهت توسعه سریع الگوریتم‌های زمان‌بندی ماشین مجازی برای بهینه‌سازی منابع متعدد در سیستم‌های مقیاس بزرگ است.

در آینده ما قصد داریم به بررسی وظایف و کارهایی که نیاز به محاسبات بیشتر در مدل بارگذاری ماشین‌های مجازی دارند، بپردازیم. مدلی که در آن رفتار پویای مرکز داده برای وظایف غیرمستقل و وابسته بر روی میزبان‌های نامتجانس می‌تواند به‌طور چشم‌گیری تغییر کند و طرح مدیریت منابع باید به‌سرعت، قادر به پاسخ‌گویی به این تغییرات باشد. یکی از ضعف‌های این روش پیشنهادی استفاده از تخصیص تصادفی کارها به ماشین‌های مجازی است. به همین دلیل استفاده از روشی که جایگزین روش تخصیص تصادفی کارها شود، یکی دیگر از کارهایی است که در آینده قصد داریم به آن بپردازیم. یکی دیگر از کارهایی که در آینده قصد داریم به آن بپردازیم، مدیریت پویای منابع بر اساس ترکیب مؤثر ماشین‌های مجازی است.

منابع

- [1] J. G. Koomey, "Estimating total power consumption by servers in the US and the world," Stanford University, pp. 1-20, 2007.
- [2] A. Fayyaz, M. U. Khan and S. U. Khan, "Energy efficient resource scheduling through VM consolidation in cloud computing," In 2015 13th International Conference on Frontiers of Information Technology (FIT), IEEE, pp. 65-70, 2015.
- [3] T. C. Ferreto, M. A. Netto, R. N. Calheiros and C. A. De Rose, "Server consolidation with migration control for virtualized data centers" Future Generation Computer Systems, vol. 27, no. 8, pp. 1027-1034, 2011.
- [4] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," The Journal of Supercomputing, vol. 60, no. 2, pp 268-280, 2012.
- [5] 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 and Experience, vol. 24, no. 13, pp. 1397-1420, 2012.
- [6] A. Song, W. Fan, W. Wang, J. Luo and Y. Mo, "Multi-objective virtual machine selection for migrating in virtualized data centers," In Joint International Conference on Pervasive Computing and the Networked World, Springer Berlin Heidelberg, vol. 7719, pp. 426-438, 2012.
- [7] I. Roder, H. Viswanathan, E. K. Lee, M. Gamell, D. Pompili and M. Parashar, "Energy-efficient thermal-aware autonomic management of virtualized HPC cloud

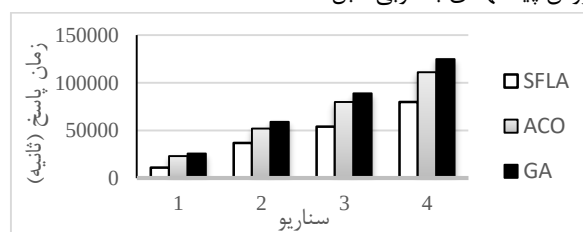
به سربراشی جابجایی ماشین‌های مجازی برمی‌گردد که برای خاموش کردن ماشین‌های فیزیکی و در نتیجه کاهش مصرف انرژی انجام می‌شود. همان‌طور که در شکل ۱۰ دیده می‌شود، الگوریتم پیشنهادی makespan را نسبت به الگوریتم کلونی مورچگان (ACO) حدود ۸٪ و نسبت به الگوریتم ژنتیک (GA) حدود ۱۶٪ کاهش داده است.



شکل ۱۰: میانگین makespan بر تعداد کارها برای هر یک از الگوریتم‌ها

۵-۶- مقایسه زمان پاسخ

شکل ۱۱، هزینه پاسخ‌های به‌دست‌آمده برای سه الگوریتم SFLA، ACO و GA را نشان می‌دهد. همان‌طور که در شکل ۱۱ مشاهده می‌شود، الگوریتم پیشنهادی SFLA زمان پاسخ میانگین را نسبت به الگوریتم ACO تا حدود ۱۵٪ و نسبت به الگوریتم GA تا حدود ۳۰٪ کاهش داده است. بر اساس شکل ۱۱، الگوریتم SFLA نسبت به دو الگوریتم ACO و GA بیشترین دقت را دارا می‌باشند. این دقت در اثر همگرایی خوب می‌تواند رخ داده باشد. هر چقدر اندازه مسئله بزرگتر باشد، تعداد نقاط بهینه‌های محلی فضای جستجو بیشتر می‌شود؛ لذا تأثیر روش پیشنهادی به‌خوبی قابل مشاهده است.



شکل ۱۱: زمان پاسخ عامل‌ها

۶- نتیجه‌گیری

مدیریت متمرکز منابع مرکز داده نه تنها می‌تواند کیفیت سرویس برای کاربران تضمین کند، بلکه باعث حداکثر شدن صرفه‌جویی در انرژی برای رایانش سبز خواهد شد. الگوریتم‌های GA، ACO و SFLA که در این مقاله مورد بررسی قرار گرفته‌اند، سعی در ارضای هر دو کیفیت سرویس و صرفه‌جویی در مصرف انرژی از طریق مدیریت تعداد مهاجرت‌ها و زمان‌بندی وظایف داشته‌اند. سهم اصلی در دیدگاه مدیریت منابع از ترکیب مؤثر منابع میزبان توسط تکنولوژی مهاجرت ماشین‌های مجازی شکل می‌گیرد و دلیل این مهاجرت ماشین‌های مجازی تعویض میزبان بیکار و یا کم استفاده برای صرفه‌جویی در مصرف انرژی در حال تضمین استاندارد توافق‌نامه سطح سرویس است. الگوریتم بهینه‌سازی پرش ترکیبی قورباغه نه تنها باعث بهبود هدف شده است، بلکه توانایی جستجوی محلی را افزایش داده است. این روش پرش ترکیبی قورباغه

- algorithm, Journal of Water Resources Planning and Management, vol. 129, no. 3, pp. 210–225, 2003.
- [15] H. Liu, F. Yi, H. Yang, “Adaptive grouping cloud model shuffled frog leaping algorithm for solving continuous optimization problems, Computational intelligence and neuroscience, vol. 25, pp. 1-8, 2016.
- [16] J. P. Luo, X. Li and M. R. Chen, “Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers,” Expert Systems with Applications, Elsevier, vol. 41, no. 13, pp. 5804-5816, 2014.
- [17] M. Eusuff, K. Lansey and F. Pasha, “Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization,” Engineering Optimization, vol. 38. No. 2, pp. 129-154, 2006.
- [18] D. Kusic, J. Q. Kephart, J. E. Hanson, N. Kandasamy and G. Jiang, “Power and performance management of virtualized computing environments via lookahead control,” Cluster computing, Springer, vol. 12, no. 1, pp. 1-15, 2009.
- [19] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience, vol. 41, no. 1, pp. 23-50, 2011.
- [20] T.D.Braun, H.J.Siegel, N.Beck, L.L.Bölöni, M.Maheswaran, A.I.Reuther, J.P.Robertson, M.D.Theys, B.Yao, D.Hensgen and R.F.Freund, A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, Journal of Parallel and Distributed computing, vol. 61, no. 6, pp.810-837.
- infrastructure,” Journal of Grid Computing, vol. 10, no. 3, pp. 447-473, 2012.
- [8] S. S. Masoumzadeh and H. Hlavacs, “Integrating VM selection criteria in distributed dynamic VM consolidation using Fuzzy Q-Learning,” In Proceedings of the 9th International Conference on Network and Service Management (CNSM), IEEE, pp. 332-338, 2013.
- [9] Y. Ge and G. Wei, “GA-based task scheduler for the cloud computing systems,” In Web Information Systems and Mining (WISM), 2010 International Conference, IEEE, vol. 2, pp. 181-186, 2010.
- [10] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres and H. Tenhunen, “Using ant colony system to consolidate vms for green cloud computing,” IEEE Transactions on Services Computing, vol. 8, no. 2, pp. 187-98, 2015.
- [۱۱] سیمین قاسمی فلاورجانی، محمدعلی نعمت بخش، بهروز شاهقلی قهفرخی، «تخصیص وظایف چند هدف در واگذاری به ابر سیار»، مجله مهندسی برق دانشگاه تبریز، دوره ۴۶، شماره ۴، صفحات ۲۱۷-۲۳۲، ۱۳۹۵.
- [12] B. Speitkamp and M. Bichler, “A mathematical programming approach for server consolidation problems in virtualized data centers,” IEEE Transactions On Services Computing, vol. 3, no. 4, pp. 266-278, 2010.
- [13] F. Tao, Y. Feng, L. Zhang and T. W. Liao, “CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling,” Applied Soft Computing, vol. 19, pp. 264-279, 2014.
- [14] M. M. Eusuff and K. E. Lansey, “Optimization of water distribution network design using the shuffled frog leaping

زیرنویس‌ها

⁷CPU utilization⁸Physical machine⁹Elite¹⁰Backup^{۱۱}Ant Colony System^{۱۲}Genetic Algorithm¹ Dynamic voltage and frequency scaling² Overload³ Underload⁴ Ant Colony System⁵ Ant Colony System -based VM Consolidation^{*}Shuffled Frog-Leaping Algorithm