ارائه یک الگوریتم توازن بار نامتمرکز در محیط‌های ناهمگن ابر

سمیرا حورعلی، شهرام جمالی و فاطمه حورعلی

چکیده: یکی از راهکارهای اساسی برای ارتقای کارایی در محیط ابر، موازنه بار می‌باشد. انتخاب VM مناسب برای انجام هر کار، تابع پارامترهای مختلفی مانند میزان منابع مورد نیاز کار نظیر CPU، حافظه، حجم منابع در اختیار VMها، هزینه و سررسید VMها می‌باشد. در این مقاله با در نظر گرفتن تک‌تک این معیارها و اهداف طراحی مانند توازن بار، کاهش نرخ ایجاد VM جدید و مهاجرت VM ها، مسأله را در قالب پارامترهای مؤثر در کارایی مدل کرده و سپس مدل فوق را با استفاده از روش پرومته که یکی از پرکاربردترین روش‌های تصمیم‌گیری چندشاخصه است، حل می‌کنیم. در این روش انتخاب بهترین VM بر اساس ارزش اختصاص‌یافته به هر یک از معیارها صورت می‌گیرد که بر اساس منطق فازی تعیین می‌شود. جهت بررسی کارایی این روش، شبیه‌سازی‌های گسترده‌ای در محیط CloudSim صورت گرفته که نشان می‌دهد روش پیشنهادی نسبت به روش‌های موجود مانند FIFO، DLB و WRR از نقطه نظرات زمان پاسخ، نرخ موفقیت کارها، انحراف بار و نرخ مهاجرت VM ها عملکرد بسیار بهتری دارد.

کلیدواژه: **تکنیک فازی، روش پرومته، ماشین مجازی، محاسبات ابری، موازنه بار.**

۱- مقدمه

امروزه فناوری اطلاعات و اینترنت عنصر جدایی‌ناپذیر زندگی مردم شده است. با تغییر شیوه زندگی افراد جامعه نیازهایی مانند امنیت اطلاعات، پردازش سریع، دسترسی فوری به اطلاعات و از همه مهم‌تر صرفه‌جویی در هزینه‌ها نیز تغییر پیدا کرده است. با گسترش این نیازها، سازمان‌ها و افراد نیازهای کاملاً متفاوت در زمینه خدمات الکترونیکی با گذشته دارند و این امر موتور محرکه‌ای برای ظهور فناوری‌های جدیدی مانند محاسبات ابری می‌باشد. محاسبات ابری یک فناوری توسعه‌یافته است که صنایع IT را قادر می‌سازد هزینه‌های محاسباتی را کاهش دهند. کاربران ابر بر حسب تقاضا منابع را در اختیار می‌گیرند و به اندازه‌ای که از سرویس‌ها استفاده می‌کنند، هزینه‌ای را پرداخت می‌کنند و بنابراین محاسبات ابری یک نوع محاسبات سودمند شناخته می‌شود.

از دید زیرساخت نیز ابر به نوعی از سیستم‌های موازی و توزیع‌شده گفته می‌شود که شامل مجموعه‌ای از کامپیوترهای مجازی به هم متصل است [۱]. در واقع رایانش ابری به معنای استفاده اشتراکی از برنامه‌ها و منابع می‌باشد. ابرها انبار بزرگی از منابع مجازی هستند که به راحتی قابل استفاده و در دسترس هستند (مانند سخت‌افزار، پلتفرم‌های توسعه‌یافته و/یا سرویس‌ها). این منابع می‌توانند به صورت پویا پیکربندی مجدد شوند تا یک بار متغیر (مقیاس) را تنظیم و همچنین بهره‌برداری مطلوبی از منابع را فراهم کنند.

این منابع معمولاً به وسیله یک مدل پرداخت- هزینه که در آن توسط ارائه‌دهنده زیرساخت و براساس توافقات سطح سرویس حداقل هزینه دسترسی تضمین شده است، بهره‌برداری می‌شود. [۲].

از آنجایی که منابع موجود در ابر در هر زمان در حال تغییر هستند، مسئله زمان‌بندی وظایف امر مهمی است که تأثیر زیادی در عملکرد محیط محاسبات ابری دارد. الگوریتم زمان‌بندی روشی است که به وسیله آن وظایف به منابع موجود در مراکز داده تخصیص داده می‌شود. در محیط ابر نیاز است منابع محاسباتی طوری زمان‌بندی شوند که هم ارائه‌دهندگان حداکثر استفاده را از منابعشان ببرند و هم کاربران برنامه‌های کاربردی مورد نیاز خود را با کمترین هزینه در اختیار بگیرند. بنابراین زمان‌بندی یکی از مهم‌ترین مسایل در ابر محسوب می‌شود.

انتخاب یک زمان‌بندی نامناسب می‌تواند باعث ناکارآمدی سخت‌افزار یا کندشدن برنامه ابر شود. در مواردی انتخاب نادرست الگوریتم باعث می‌شود مسئله‌ای که چند ثانیه زمان می‌برد در چندین ساعت حل شود و بنابراین یک زمان‌بند خوب باید در شرایط مختلف رفتار مناسبی داشته باشد. استراتژی‌های زمان‌بند وظیفه بر عدالت یا بهره‌وری منابع تمرکز دارند که هزینه، زمان، فضا، توان عملیاتی و کیفیت سرویس در محاسبات ابری را بهبود می‌دهند.

در این مقاله زمان‌بندی کارها در محیط ناهمگن ابر با استفاده از روش پرومته که یکی از روش‌های پرکابرد در تصمیم‌گیری چندمعیاره است انجام می‌شود. همچنین سعی شده تمامی ویژگی‌های منابع و کارها که میزان اهمیت آنها توسط تکنیک فازی محاسبه می‌شود برای اختصاص منبع مناسب به کارهای ورودی به ابر در نظر گرفته شود.

در بخش ۲ تعدادی از کارهای مطرح که در این زمینه انجام می‌شود، مورد بررسی قرار می‌گیرد. در بخش ۳ ساختار مدل در نظر گرفته شده برای محیط ناهمگن در نظر گرفته شده برای ابر و نحوه رتبه‌بندی منابع بر اساس تکنیک فازی و روش تصمیم‌گیری پرومته توصیف می‌شود. در بخش ۴ روش پیشنهادی مورد بررسی قرار می‌گیرد و آزمایشات و نتایج در بخش ۵ ارائه می‌شود.

۲- کارهای مرتبط

انواع متنوعی از الگوریتم‌های زمان‌بندی در سیستم‌های توزیع‌شده وجود دارد. هدف اصلی الگوریتم‌های زمان‌بندی به دست آوردن عملکرد محاسباتی بالا و بهترین توان عملیاتی سیستم است. الگوریتم‌های ارائه‌شده در این زمینه به دو دسته کلی ایستا و پویا تقسیم می‌شوند و در این قسمت به بررسی چندین الگوریتم زمان‌بندی می‌پردازیم. روش اکتشافی ایستا هنگامی استفاده می‌شود که مجموعه کامل از وظایف قبل از اجرا شناخته شوند. این استراتژی‌ها تحت دو فرض اجرا می‌شوند. اول این که وظایف به طور هم‌زمان می‌رسند و فرض دوم این است که زمان ماشین‌های موجود، بعد از هر زمان‌بندی وظیفه به روز می‌شوند.

این مقاله در تاریخ ۱۴ آبان ماه ۱۳۹۳ دریافت و در تاریخ ۱۴ تیر ماه ۱۳۹۴ بازنگری شد.

سمیرا حورعلی، دانشکده مهندسی کامپیوتر و الکترونیک، دانشگاه غیر انتفاعی- غیر دولتی شاهرود، شاهرود، (email: s.hourali68@gmail.com).

شهرام جمالی، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه محقق اردبیلی، اردبیل، (email: jamali@iust.ac.ir).

فاطمه حورعلی، دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اسفراین، اسفراین، (email: hourali@esfarayen.ac.ir).

در [۳] یک روش ایستا تحت عنوان WRR برای موازنه بار ارائه شده که در این روش، کارهای رسیده به محیط ابر بر اساس نیازهای متفاوتی که دارند ابتدا در کلاس‌های مختلفی طبقه‌بندی می‌شوند. سپس کارهای هر کلاس به صف مربوط به همان کلاس که پهنای باند و وزن مخصوص به خود را دارد وارد می‌شوند. در نهایت کارها از صف‌ها وارد زمان‌بند می‌شوند و این زمان‌بند بر اساس الگوریتم RounRobin عمل می‌کند. این الگوریتم رفع مشکل گرسنگی را تضمین می‌کند چون بر این باور است که سرویس‌های هر کلاس حداقل مقدار پهنای باند لازم برای انجام کارها را دارند.

در [۴] یک موازنه‌گر ایستا ارائه شده که طبق این روش هر وظیفه را به ترتیب دلخواه به ماشینی با بهترین زمان اجرای مورد انتظار آن وظیفه اختصاص می‌دهد بدون این که به دسترس‌بودن آن ماشین توجه کند. هدف MET این است که به هر وظیفه بهترین ماشین مربوط به آن داده شود. این می‌تواند باعث عدم توازن جدی بین ماشین‌ها شود. به طور کلی این الگوریتم برای محیط‌های محاسباتی ناهمگن مناسب نیست.

روش ایستای دیگری برای موازنه بار در [۵] ارائه شده است. روند این روش بر اساس اولویت‌بندی وظایف یا تولید زمان‌بندی بر اساس اولویت است. این اولویت بر اساس زمان اتمام وظیفه مورد انتظار بر روی یک منبع تولید می‌شود. این روش وظایف را در چند گروه وظایف مستقل تنظیم می‌کند. پس از آن این گروه‌ها مکرراً زمان‌بندی می‌شوند. هر تکرار مجموعه‌ای از وظایف مستقل نگاشت‌نشده را می‌گیرد و برای هر وظیفه، حداقل زمان تکمیل مورد انتظار (MECT) را تولید می‌کند. وظیفه‌ای که کوچک‌ترین مقدار MECT بیش از تمام وظایف انتخاب‌شده به منابع مربوط را دارد، در این تکرار اول زمان‌بندی می‌شود. این تا زمانی که تمام وظایف زمان‌بندی شوند، ادامه می‌یابد. هدف این الگوریتم رسیدن به کمترین پاسخ است و برای رسیدن به این هدف، ابتدا وظایفی با زمان تکمیل کم و سپس وظایفی با زمان بیشتر را زمان‌بندی می‌کند.

در [۶] یک الگوریتم ایستای دیگر برای زمان‌بندی جریان کار ارائه شده است. این الگوریتم با مجموعه‌ای از تمام وظایف زمان‌بندی نشده شروع و سپس برای هر وظیفه در مجموعه زمان تکمیل کمینه را محاسبه می‌کند. تفاوت این الگوریتم با روش قبلی در این است که ابتدا وظایف دارای بیشترین زمان تکمیل انتخاب شده و به ماشین نگاشت می‌شوند. بعد وظیفه زمان‌بندی شده از مجموعه حذف می‌شود. این فرایند تا زمانی که همه وظایف زمان‌بندی شوند ادامه می‌یابد.

روش اکتشافی پویا زمانی که مجموعه وظایف و یا مجموعه ماشین‌ها ثابت نیستد، ضروری است. به عنوان مثال، همه وظایف به طور هم‌زمان نرسند و یا برخی از ماشین‌های در فواصل زمانی به حالت آفلاین بروند. روش اکتشافی پویا در دو حالت آنلاین و دسته‌ای استفاده می‌شود. در حالت اول، زمانی که یک وظیفه برسد به یک ماشین زمان‌بندی می‌شود و در حالت دوم، وظایف ابتدا در یک مجموعه جمع‌آوری شده و زمان‌بندی در زمان از پیش برنامه‌ریزی‌شده اجرا می‌شود.

یک الگوریتم بهبود هزینه برای زمان‌بندی جریان کار در ابرهای ترکیبی در [۷] ارائه شده و دارای دو مرحله اصلی انتخاب وظایف و انتخاب منابع از ابر عمومی و تشکیل ابر ترکیبی می‌باشند. در حالی که زمان‌بند تصمیم می‌گیرد که کدام یک از وظایف باعث کاهش زمان اجرا با استفاده از منابع ابر عمومی می‌شوند، تعیین کارایی و هزینه‌های اجرا، نقش عمده‌ای را در زمان‌بندی جدید ایفا می‌کنند.

در [۸] یک استراتژی پویای توازن بار مبنی بر الگوریتم ژنتیک (GA) ارائه شده است. سرعت زمان‌بند افزایش و تغییر بین پردازنده‌ها کاهش می‌یابد و وظایف برای اجرا انتخاب و در حالت پیشرفته خصوصیات وظایف شناخته می‌شود. آستانه سازگاری به توازن بار پویای پردازنده‌ها کمک می‌کند.

در [۹] نوعی زمان‌بند اکتشافی پویا ارائه شده که بین زمان‌بندی عدالت و محلیت داده برخورد ایجاد می‌کند. این زمان‌بند به طور موقت برای بهبود محلیت، عدالت را رها می‌کند و برای این که یک گره با داده‌های محلی را زمانبندی کند، از کارها درخواست انتظار می‌نماید. اگر کارها براساس عدالت زمان‌بندی شود، یک کار محلی نمی‌تواند راه‌اندازی شود و باید مدت کوتاهی انتظار بکشد، در این حین به دیگر کارها اجازه داده می‌شود که وظایف خود را راه‌اندازی کنند. با این حال اگر یک کار به مدت طولانی نادیده گرفته شود، آنگاه مجاز است برای جلوگیری از قحطی، کارهای غیر محلی را راه‌اندازی کند. این زمان‌بند زمانی که وظایف در مقایسه با کارها کوتاه باشند و شکاف‌های بسیاری در هر گره وجود داشته باشد، کارآمد است.)

در [۱۰] روشی پویا تحت عنوان DLB ارائه شده که در این روش محیط به صورت یک مدل $G/S/M$ در نظر گرفته شده است و در آن G تعداد کلاسترهایی است که محیط توزیع‌شده را تشکیل داده‌اند، S تعداد مکان‌های محیط توزیع‌شده است و M تعداد عناصر محاسباتی یا گره‌های پردازشی این محیط است. زمان‌بندی کارها در یک درخت ۴سطحی انجام می‌شود. در سطح اول یک نود مجازی در ریشه درخت قرار دارد که دو وظیفه را بر عهده دارد: ۱) مدیریت اطلاعات بارکاری محیط توزیع شده و ۲) با دریافت کارها از کاربران بر اساس نیاز هر کاربر و بار جاری محیط، تصمیم می‌گیرد که کارهای دریافتی باید به کدام قسمت فرستاده شوند. سطح دوم شامل G نود مجازی است و هر یک از نودها به یک کلاستر فیزیکی از محیط توزیع‌شده متصل هستند. هر یک از این نودها پاسخ‌گوی بارکاری یا درخواست‌های کلاستر مربوط به خود هستند. در سطح سوم از درخت، S نود وجود دارد که هر کدام از این نودها به مکان‌های فیزیکی تمام کلاسترهای محیط توزیع‌شده متصل هستند. وظیفه اصلی این نودها، مدیریت بارکاری گره‌های پردازشی است. در آخرین سطح از درخت M عنصر محاسباتی وجود دارد که هر کدام از این عناصر به کلاسترها و مکان‌های مربوط به خود متصل هستند.

در مقایسه با روش‌های ذکرشده در این بخش، روش ارائه‌شده در این مقاله در دسته روش‌های اکتشافی پویا قرار دارد که در محیط‌های ناهمگن ابر به خوبی عمل می‌کند. در این روش سعی بر این بوده که مشکلات روش‌های قبلی تا حد ممکن مرتفع شود.

۳- روش پیشنهادی

در این بخش ابتدا مدل مفروض برای محیط ابر و سپس چگونگی فرموله‌سازی مسأله مورد اشاره قرار می‌گیرد و در ادامه مسأله فوق با استفاده از روش پرومته حل می‌شود.

۳-۱ مدل در نظر گرفته شده برای محیط ابر

فضای تعریف‌شده برای ابرمحاسباتی شامل K خوشه پردازشی (سرویس) است. تعداد سرورهای فیزیکی m است که به صورت $R=\{R_1,R_2,\ldots,R_m\}$ می‌باشد.

تعداد VM ها در هر سرور فیزیکی n می‌باشد که به صورت $VM(R_f)=\{VM_1,VM_2,\ldots,VM_n\}, f\in[1,m]$ نمایش داده می‌شود. هر کاربر نیز در محیط ابر دارای تعداد محدودی کار بوده و تمام کارهای ارائه‌شده به ابر عبارت است از $J=\{J_1,J_2,\ldots,J_z\}$ که در زمان T به

جدول ۱: ماتریس اولیه P.

گزینه \ شاخص	X_1	X_2	...	X_n
A_1	N_{11}	N_{12}	...	N_{1n}
A_2	N_{21}	N_{22}	...	N_{2n}
⋮	⋮	⋮	⋱	⋮
A_m	N_{m1}	N_{m2}	...	N_{mn}

ابر ارسال می‌شوند $(0 \le t_j \le T)$. T بیانگر مدت زمان شبیه‌سازی است.

هر کار J_i در فضای ابر محاسباتی دارای ۵ مشخصه است که عبارت است از $J_i = (cp_i, m_i, IO_i, b_i, d_i, bw_i)$. cp_i که بر حسب MI است نشان‌دهنده میزان بهره‌برداری کار از CPU می‌باشد، m_i بر حسب MB نشان‌دهنده میزان بهره‌برداری از حافظه و IO_i بر حسب MB نشان‌دهنده میزان بهره‌برداری از ورودی/ خروجی است. سایر مشخصات به ترتیب عبارت است از بودجه تخصیص داده شده به کار مورد نظر بر حسب G$، سررسید اجرای کار بر حسب واحد زمان (ثانیه) و پهنای باند مورد نیاز برای انجام کار بر حسب MB/Sec. مشخصات ذکرشده برای کار توسط کاربر هنگام ارسال وظیفه مورد نظر به ابر مشخص می‌شود.

برای هر گره پردازشی r_j (سرور فیزیکی یا VM) نیز در این محیط پنج مشخصه به صورت $r_j = (rcp_j, rm_j, rIO_j, rc_j, rn_j, bw_j)$ تعریف شده است. rcp_j معرف قدرت پردازشی هر گره است که در واقع تعداد دستورات قابل اجرا توسط هر یک از اجزای پردازشی منبع بر حسب میلیون در هر ثانیه است (MIPS). rm_j و rIO_j به ترتیب نشان‌دهنده میزان بهره‌برداری از حافظه و ورودی/ خروجی است که بر مبنای مگابایت بر ثانیه محاسبه می‌شود (MB/s). مقدار rc_j قیمت منبع را بر حسب G$ نشان می‌دهد، مقدار rn_j معرف میزان تأخیر (ترافیک) شبکه در دستیابی به گره پردازشی است که با واحد ثانیه اندازه‌گیری می‌شود و در نهایت bw_j پهنای باند گره پردازشی مورد نظر را نشان می‌دهد و با واحد (MB/S) اندازه‌گیری می‌شود.

انتخاب VM مناسب برای انجام هر کار، تابع پارامترهای مختلفی مانند میزان منابع مورد نیاز کار نظیر CPU، حافظه، حجم منابع در اختیار VM ها، هزینه و سررسید VM ها می‌باشد. باید با در نظر گرفتن تک‌تک این معیارها، مناسب‌ترین VM برای کار مورد نظر انتخاب شود. بنابراین می‌توان این انتخاب را به صورت یک مسئله تصمیم‌گیری چندشاخصه در نظر گرفت. در این مقاله گزینه‌ها، گره‌های پردازشی موجود در ابر هستند و شاخص‌های تصمیم‌گیری شامل هزینه پردازش، زمان لازم برای پردازش و سرعت پردازش که شامل CPU، حافظه و ورودی/ خروجی می‌باشد، است. با ورود هر کار به یک سرور، مدیر زیرساخت مجازی آن سرور ابتدا با استفاده از روش تصمیم‌گیری PROMETHEE شروع به یافتن بهترین VM برای تخصیص به آن کار در خوشه مربوطه می‌کند، در صورت پربودن تمام VM های موجود در خوشه، سرورها با استفاده از روش تصمیم‌گیری فوق اقدام به انتخاب بهترین سرور (از سایر خوشه‌ها) برای انتقال سایر VM های خود به آن سرور می‌کنند تا ظرفیت جدید برای ایجاد VM به وجود آید.

۳-۲ فرموله‌سازی مسئله

برای اتخاذ تصمیم نیاز به فرموله‌کردن مسئله است که این کار توسط ماتریسی به نام P انجام می‌شود و بنابراین لازم است تمام اطلاعات (کمی یا کیفی) مسئله بر روی ماتریس اولیه P پیاده شود (جدول ۱).

جدول ۲: ماتریس تصمیم بی‌مقیاس شده D.

گزینه \ شاخص	X_1	X_2	...	X_n
A_1	r_{11}	r_{12}	...	r_{1n}
A_2	r_{21}	r_{22}	...	r_{2n}
⋮	⋮	⋮	⋱	⋮
A_m	r_{m1}	r_{m2}	...	r_{mn}

در جدول ۱ A_i گزینه iام، X_j شاخص jام و r_{ij} ارزش شاخص jام برای گزینه iام می‌باشد. تمام شاخص‌های در نظر گرفته شده در این مقاله کمّی می‌باشند و گزینه‌ها VM ها یا سرورها و شاخص‌ها $rcp_j, rm_j, rIO_j, rc_j, rn_j, bw_j$ می‌باشند.

۳-۳ محاسبه ارزش (وزن) شاخص‌ها برای هر کار بر اساس تکنیک فازی

بی‌مقیاس کردن ماتریس P: مقیاس اندازه‌گیری شاخص‌های کمی می‌توانند با یکدیگر متفاوت باشند (مانند هزینه، تأخیر و غیره)، به این دلیل انجام عملیات اصلی ریاضی قبل از بی‌مقیاس کردن یا یکسان‌سازی مقیاس‌ها مجاز نیست. بنابراین طبق (۱) هر عنصر r_{ij} از ماتریس تصمیم‌گیری مفروض بر نرم موجود از ستون jام (به ازای شاخص X_j) تقسیم می‌شود، یعنی

$$N_{ij} = \frac{r_{ij}}{\sqrt{\sum_{i=1}^{m} r_{ij}^2}} \qquad (1)$$

بدین طریق کلیه ستون‌های ماتریس مفروض دارای واحد طول مشابه شده و در نتیجه مقایسه کلی آنها آسان می‌شود. این ماتریس در جدول ۲ نمایش داده شده است.

میانگین هندسی هر یک از سطرها و وزن شاخص iام به وسیله (۲) محاسبه می‌گردد [۱۱]

$$z_1 = [\prod_{j=1}^{n} a_{ij}]^{\frac{1}{n}} \qquad (2)$$

w_i' بیانگر وزن و اهمیت گزینه یا معیار iام برای هر VM می‌باشد و از (۳) به دست می‌آید

$$\forall i : w_i' = \frac{z_i}{(z_1 + z_2 + \ldots + z_n)'} \qquad (3)$$

برای تعمیم روش فوق به حالت فازی باید در رابطه بالا به جای استفاده از عملگرهای کلاسیک از عملگرهایی مانند جمع فازی، ضرب فازی، تبدیل اعداد به اعداد ذوزنقه‌ای فازی و غیره استفاده نمود. بنابراین باید به صورت زیر عمل کرد:

الف) در این مرحله ماتریس زوجی توسط شخص تصمیم‌گیرنده مشخص می‌گردد و اجزای این ماتریس‌ها اعداد فازی ذوزنقه‌ای خواهند بود. چنانچه ارجحیت جزء iام با $\tilde{a}_{ij} = (a_{ij}, b_{ij}, c_{ij}, d_{ij})$ نشان داده شود، در این صورت ارجحیت جزء iام به صورت $\tilde{a}_{ij} = (1/a_{ij}, 1/b_{ij}, 1/c_{ij}, 1/d_{ij})$ خواهد بود. در صورتی که $i = j$ باشد می‌توان نوشت $\tilde{a}_{ij} = (1,1,1,1)$.

ب) برای محاسبه وزن شاخص‌ها بر اساس تکنیک فازی ابتدا میانگین هندسی هر سطر از ماتریس‌های مقایسات زوجی با استفاده از (۴) تعیین می‌شود

جدول ۳: بردار J برای کار i ام.

گزینه / شاخص	X_1	X_2	...	X_5
J_i	P_1	P_2	...	P_5

$$\tilde{z}_i = (\tilde{a}_{i1}, \tilde{a}_{i2}, \ldots, \tilde{a}_{in})^{\frac{1}{n}} \quad (۴)$$

سپس وزن $\tilde{w}_i$ فازی از (۵) محاسبه می‌گردد [۱۲]

$$\tilde{w}_i = \tilde{z}_i . (\tilde{z}_1 \oplus, \tilde{z}_2 \oplus, \ldots, \tilde{z}_n \oplus)^{\frac{1}{n}} \quad (۵)$$

ج) برای محاسبه وزن نهایی شاخص‌ها بر اساس روش ترکیبی برای هر کار، بر اساس میزان اهمیت نسبی‌ای که هر یک از شاخص‌ها برای تصمیم‌گیرنده (کاربر) دارند، وزنی به آنها تعلق می‌گیرد به طوری که مجموع آنها برابر یک شود. این اهمیت نسبی بیانگر درجه ارجحیت هر شاخص نسبت به بقیه شاخص‌ها برای تصمیم‌گیری مورد نظر است. این وزن‌ها از طریق (۶) و تلفیق بردار J (جدول ۳) با بردار w' محاسبه می‌شوند

$$W_i = \frac{p_i \tilde{w}_i}{\sum_{i=1}^{5} p_i \tilde{w}_i} \quad (۶)$$

$$\sum_{i=1}^{k} w_i = 1 \quad (۷)$$

۳-٤ رتبه‌بندی نهایی گزینه‌ها بر اساس تکنیک پرومته

فرض کنید A مجموعه‌ای از گزینه است که باید از میان آنها انتخاب صورت گیرد. با فرض وجود K معیار مؤثر در تصمیم‌گیری، برای هر $a \in A$ مقدار $f_j(a)$ نشان‌دهنده ارزش معیار j ام در گزینه a است. رتبه‌بندی در سه مرحله انجام می‌شود:

مرحله اول: تابع ترجیح P_j به هر یک از معیارهای j اختصاص داده می‌شود. مقدار $P_j(a,b)$ برای هر زوج گزینه محاسبه می‌شود و با افزایش $f_j(a) - f_j(b)$ این مقدار بیشتر می‌شود و هنگامی که اختلاف به اندازه کافی زیاد شود، مقدار $P_j(a,b)$ هم به یک می‌رسد. شکل‌های مختلفی را می‌توان برای تابع P_j فرض کرد که به چگونگی مدل‌سازی معیار j ام بستگی دارد. روش RPOMETHEE شش تابع ترجیح (معیار تعمیم‌یافته) که عبارتند از عادی، Uشکل، Vشکل، هم‌سطح، Vشکل با ناحیه خنثی و گوسی را به تصمیم‌گیرندگان پیشنهاد داده است. انتخاب درست این تابع به تصمیم‌گیرندگان و تحلیلگر و درک آنها از رابطه میان گزینه‌ها و شاخص ها بستگی دارد [۱۳].

مرحله دوم: میزان اولویت کلی $\pi(a,b)$ (نماد ترجیح) برای هر گزینه a بر روی گزینه b محاسبه می‌شود. هرچه میزان $\pi(a,b)$ بیشتر باشد، گزینه a ترجیح بیشتری دارد. $\pi(a,b)$ به این ترتیب محاسبه می‌شود

$$\pi(a,b) = \sum_{j=1}^{k} w_j p_j(a,b) \quad (۸)$$

مرحله سوم: $\pi(a,b)$ نشان‌دهنده درجه اولویت گزینه a نسبت به گزینه b است. برای محاسبه قدرت ترجیح کلی گزینه a به دیگر گزینه‌ها، جریان خروجی محاسبه می‌شود

$$\phi^+(a) = \frac{1}{n-1} \sum_{x \in A} \pi(a,x) \quad (۹)$$

این جریان نشان می‌دهد که گزینه a چقدر بر سایر گزینه‌ها اولویت دارد. این جریان در حقیقت، قدرت گزینه a است. بزرگ‌ترین $\phi^+(a)$ نشان‌دهنده بهترین گزینه است. میزان ترجیح سایر گزینه‌ها بر گزینه a که جریان ورودی نامیده می‌شود حاصل محاسبه زیر است

$$\phi^-(a) = \frac{1}{n-1} \sum_{x \in A} \pi(x,a) \quad (۱۰)$$

این جریان نشان می‌دهد که سایر گزینه‌ها تا چه میزان بر گزینه a اولویت دارند. این جریان در حقیقت ضعف گزینه a است. کوچک‌ترین $\phi^-(a)$ نشان‌دهنده بهترین گزینه است و بنابراین با داشتن و بررسی جداگانه دو جریان ϕ^+ و ϕ^- می‌توان یک رتبه‌بندی جزئی را انجام داد (رتبه‌بندی PROMETHEE I). برای رتبه‌بندی کامل گزینه‌ها، جریان خالص برای هر گزینه به صورت زیر تعریف می‌شود

$$\phi(a) = \phi^+(a) - \phi^-(a) \quad (۱۱)$$

جریان خالص میان جریان‌های ورودی و خروجی توازن ایجاد می‌کند. این محاسبات را رتبه‌بندی کامل PROMETHEE II نامیده اند. جریان خالص بیشتر نشان‌دهنده گزینه بهتر است.

٤- الگوریتم پیشنهادی (PLB)

در هر سرور، مدیر زیرساخت، دو جدول را تنظیم می‌کند که یکی جدول بار و دیگری جدول سرور است.

۱) جدول بار جدولی است که میزان بار منابع تمام VMهای موجود بر روی آن سرور و همچنین میزان بار کلی خود سرور در آن قرار دارد. میزان بار هر یک از منابع یک گره پردازشی با استفاده از (۱۲) تا (۱۴) به دست می‌آید

$$load(cpu_j) = \sum_{i=1}^{m} \frac{job\_cpulength(i)(MI)}{computingcapacity(\frac{MI}{\sec})} \quad (۱۲)$$

$$load(Mem_j) = \sum_{i=1}^{m} \frac{job\_Memlength(i)(MB)}{Memorybandwidth(\frac{MB}{\sec})} \quad (۱۳)$$

$$load(IO_j) = \sum_{i=1}^{m} \frac{job\_IOlength(i)(MB)}{IObandwidth(\frac{MB}{\sec})} \quad (۱۴)$$

که در آن j شماره گره پردازشی است. $Job\_cpulength$، $Job\_Memlength$ و $Job\_IOlength$ به ترتیب بیانگر میزان بهره‌برداری هر کار موجود در صف منابع (CPU، حافظه و ورودی/ خروجی) از منابع گره پردازشی است و بر حسب MI، MB و MB می‌باشند. مخرج هر کدام از کسرهای (۱۲) تا (۱۴) نیز بیانگر سرعت منابع موجود است که برای CPU، حافظه و ورودی/ خروجی به ترتیب MI/Sec، MB/Sec و MB/Sec است. مجموع به دست آمده برای هر یک از منابع موجود بر میزان سرعت آن منبع تقسیم می‌شود تا بارکاری موجود در صف آن منبع محاسبه شود. اگر این مقدار از مقدار آستانه (T) بیشتر باشد به معنای زیر بار بودن آن منبع است و در غیر این صورت منبع دارای بار سبک

$$\begin{aligned}&\text{if } \max(CPU, Mem, I/O) \geq T\\&\quad utilization\ Level\ is\ High\\&\text{else}\\&\quad utilization\ Level\ is\ Low\\&\text{end if}\end{aligned}$$

شکل ۱: شبه‌کد تعیین حالت گره پردازشی.

1. Start (With coming task)
2. Collecting load tables and removing VMs with high cost and delay
If (Remains a VM)
Choosing the best VM by PROMETHEE method and assigning task to it.
Else
Collecting server tables and Remove the overloaded servers
If (Remains a server)
1. Choosing the best server by PROMETHEE.
2. Creating VM on selected server and assigning task to it.
3. Finish.
Else
1. Collecting the load tables from other cluster by a server within the cluster, choosing the best server by PROMETHEE method and migrating VM to it.
2. Creating the VM on selected server and assigning task to it.
3. Finish.

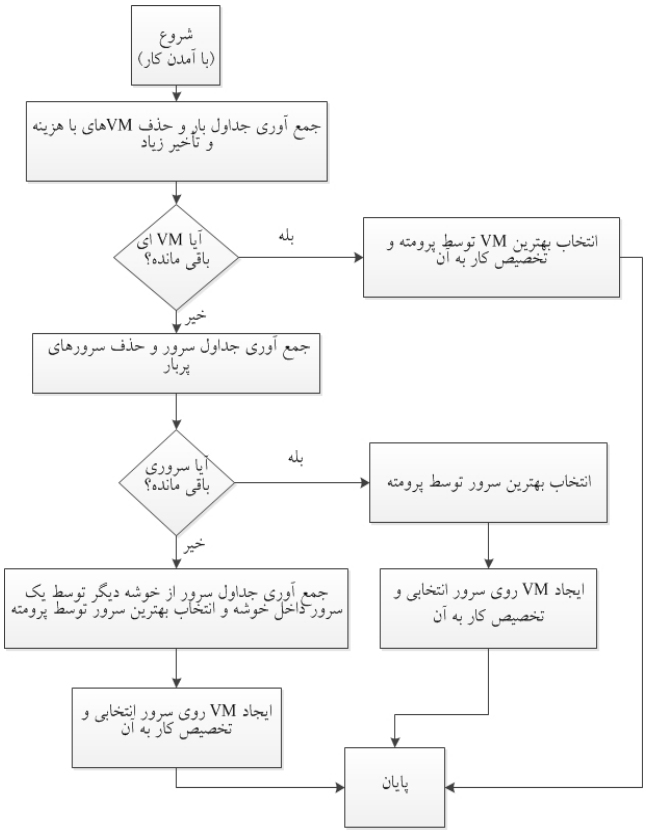
شکل ۲: شبه‌کد روش پیشنهادی.

خواهد بود. اگر یکی از منابع یک گره پردازشی زیر بار باشد به معنای زیر بار بودن کل آن گره است که این مفهوم در شبه‌کد شکل ۱ تعریف شده است.

۲) جدول سرور جدولی است که اطلاعات سرورهای ارائه‌کننده یک سرویس در آن وجود دارد. با اضافه‌شدن هر سرور فیزیکی به ابر، این جدول به روز می‌شود تا اطلاعات سرور اضافه‌شده که شامل تعداد سرویس‌هایی که توسط سرور ارائه می‌شود و قیمت سرور می‌باشد در میان سایر سرورها پخش شود.

مراحل موازنه بار توسط الگوریتم PLB در دو سطح بیان می‌شود:

۱) سطح VM: زمانی که یک کار به سرور (سرور مبدأ) می‌رسد، مدیر زیرساخت مجازی سرور مبدأ با توجه به جدول سرور، فراخوانی را برای جمع‌آوری جداول بار سایر سرورهای درون خوشه‌ای می‌دهد. منظور از سرورهای درون خوشه‌ای، سرورهایی هستند که توسط تعدادی VM سرویس مورد نظر را ارائه می‌دهند. سرورها با دریافت این فراخوان ابتدا اطلاعات آن بخشی از جدول بار که VM های ارائه‌دهنده سرویس در آن وجود دارد را ویرایش کرده و سپس برای سرور مبدأ ارسال می‌کنند. این ویرایش به این صورت است که تمام VM هایی که زیر بار باشند را از جدول حذف می‌کند و فقط آن دسته از VM ها که بار سبک دارند (بار آنها کمتر از T باشد) را به سرور مبدأ ارسال می‌کند. سرور مبدأ با دریافت تمامی جداول بار ویرایش شده و همچنین اندازه‌گیری میانگین زمان تأخیر برای هر سرور، ماتریس تصمیم‌گیری پرومته را تشکیل می‌دهد. سپس با توجه به مشخصه‌های میزان بهره‌برداری از CPU، حافظه و ورودی/خروجی اقدام به تولید بردار وزن (رابطه (۶)) می‌کند و بهترین VM را برای تخصیص به کار ورودی مشخص می‌کند.

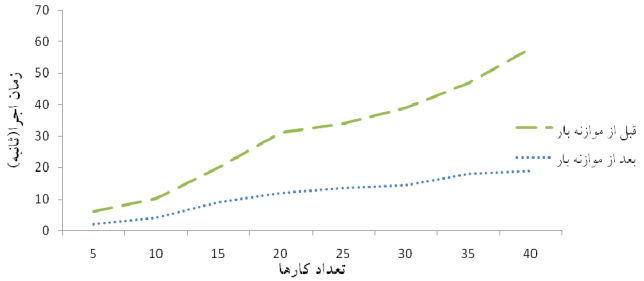


شکل ۳: موازنه بار توسط الگوریتم پیشنهادی در دو سطح.

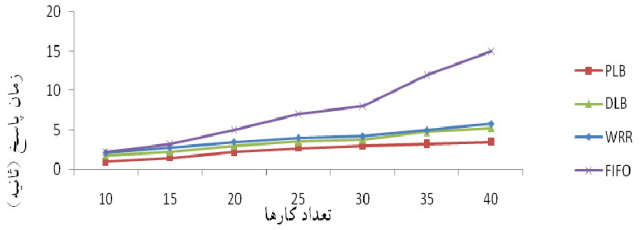
۲) سطح سرور فیزیکی: پس از ورود کار و فراخوانی سرور مبدأ مبنی بر جمع‌آوری جداول بار اگر هیچ VM ای پیدا نشود و به عبارت دیگر اگر جداول بار رسیده به سرور مبدأ خالی باشند به معنای زیر بار بودن تمامی VM های موجود در خوشه است. در این زمان کل خوشه زیر بار است. برای رفع مشکل باید ظرفیت‌های پردازشی (VM) جدید در خوشه ایجاد شود تا قسمتی از بار خوشه به آن انتقال پیدا کند. برای این کار سرور مبدأ اقدام به بررسی میزان بار هر کدام از سرورها در جداول باری دریافتی می‌کند. سرورهایی که دارای ظرفیت خالی باشند موظف به ایجاد VM می‌شوند. در صورتی که سروری برای ایجاد VM پیدا نشد سرورهای موجود در خوشه، اقدام به مهاجرت سایر VM های خود (از سایر خوشه‌ها) به سرورهای آن خوشه‌ها می‌کنند. برای این کار، سرورها نیاز به انتخاب بهترین سرور برای مهاجرت سایر VM های خود به آن را دارند و بنابراین با توجه به الگوریتم پرومته و روش وزن‌دهی ارائه‌شده (رابطه (۶)) اقدام به انتخاب بهترین سرور می‌کند. در اینجا ماتریس تصمیم‌گیری شامل مشخصات سرورهاست. با توجه به میزان بار هر کدام از منابع VM مهاجر، اقدام به وزن‌دهی به هر کدام از مشخصه‌های فوق می‌شود. بدین ترتیب سرورهای داخل خوشه ظرفیت جدیدی برای ایجاد VM داخل خوشه‌ای پیدا می‌کنند و با ایجاد آن می‌توانند از میزان بار موجود در خوشه بکاهند. در شبه‌کد شکل ۲ و شکل ۳ روند موازنه بار در دو سطح ذکرشده نشان داده شده است.

۵- نتایج شبیه‌سازی

در این بخش عملکرد الگوریتم پیشنهادی را با استفاده از نتایجی که از شبیه‌ساز Cloudsim به دست آوردیم، تحلیل می‌کنیم. برای انجام



شکل ۴: زمان اجرا قبل و بعد از انجام موازنه بار با PROMETHEE.



شکل ۵: زمان پاسخ روش‌های PLB، FIFO، DLB و WRR.

شبیه‌سازی در محیط ابر نیاز به اطلاعاتی در مورد ظرفیت سرورهای فیزیکی و VMها است. برای تعیین این مقادیر از دو نوع مرجع استفاده شده که نوع اول منابع موجود در [۱۴] و [۱۵] می‌باشد که در پردازش توزیع‌شده (گرید) مورد استفاده قرار گرفته‌اند.

نوع دوم، منابع مورد استفاده در ابزار CloudSim است. این ابزار توسط آزمایشگاه CLOUDS در دانشگاه ملبورن توسعه داده شده و محیطی برای شبیه‌سازی پردازش ابری است. با در نظر گرفتن مشخصات ذکرشده و با توسعه و تغییر کلاس‌های شبیه‌ساز Cloudsim الگوریتم پیشنهادی خود را شبیه‌سازی نمودیم. در شکل ۷ زمان اجرا، قبل و بعد از پیاده‌سازی روش پیشنهادی آورده شده است. محور X تعداد کارها و محور Y زمان اجرا را نشان می‌دهد.

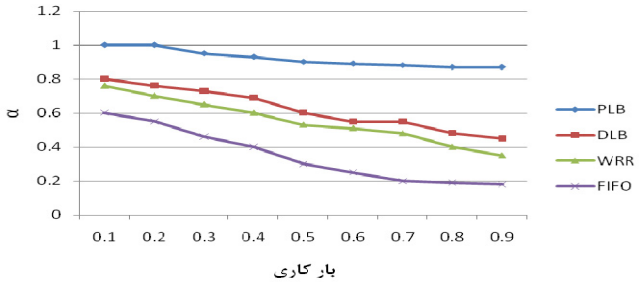
همان طور که در شکل ۴ می‌بینیم زمان اجرا پس از پیاده‌سازی روش پیشنهادی به طور قابل ملاحظه‌ای کاهش یافته است.

در شکل ۵ زمان پاسخ الگوریتم پیشنهادی در مقایسه با روش‌های WRR، FIFO و DLB [۱۰] و [۱۶] تا [۱۸] نشان داده شده است. رشد غیر هماهنگ و سریع بار موجود در صف منابع در روش‌های WRR، FIFO و DLB باعث می‌شود که گره‌های پردازشی زیر فشار بار روند و بنابراین کارها برای اجرا باید مدت زیادی منتظر بمانند. روش پیشنهادی (PLB) دارای کمترین زمان پاسخ در بین الگوریتم‌های فوق است زیرا میزان بهره‌برداری کار از منابع در این روش در نظر گرفته شده است که باعث می‌شود، یک کار به مناسب‌ترین منبع محاسباتی تخصیص یابد. بنابراین بار منابع محاسباتی موجود در گره‌های پردازشی به طور هماهنگ رشد کرده و دیرتر به حالت زیر فشار بار می‌رسند و در نتیجه زمان انتظار و به تبع آن زمان پاسخ کارها تا حد زیادی کاهش می‌یابد.

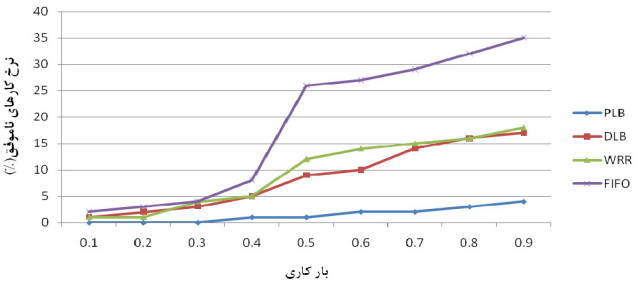
برای مقایسه عملکرد روش PLB در مقایسه با بقیه روش‌ها معیار انحراف بار معرفی شده که آن را با α نمایش می‌دهیم و طبق (۱۵) و (۱۶) محاسبه می‌شود

$$L_{VM_i,t} = \frac{N(T,t)}{S(VM_i,t)} \qquad (۱۵)$$

در رابطه بالا $N(T,t)$ تعداد کارهای موجود در صف VM_i مربوطه در زمان t و $S(VM_i,t)$ نرخ سرویس‌دهی VM_i در زمان t است و بنابراین α به صورت زیر محاسبه می‌شود



شکل ۶: شاخص α برای روش‌های PLB، FIFO، DLB و WRR.

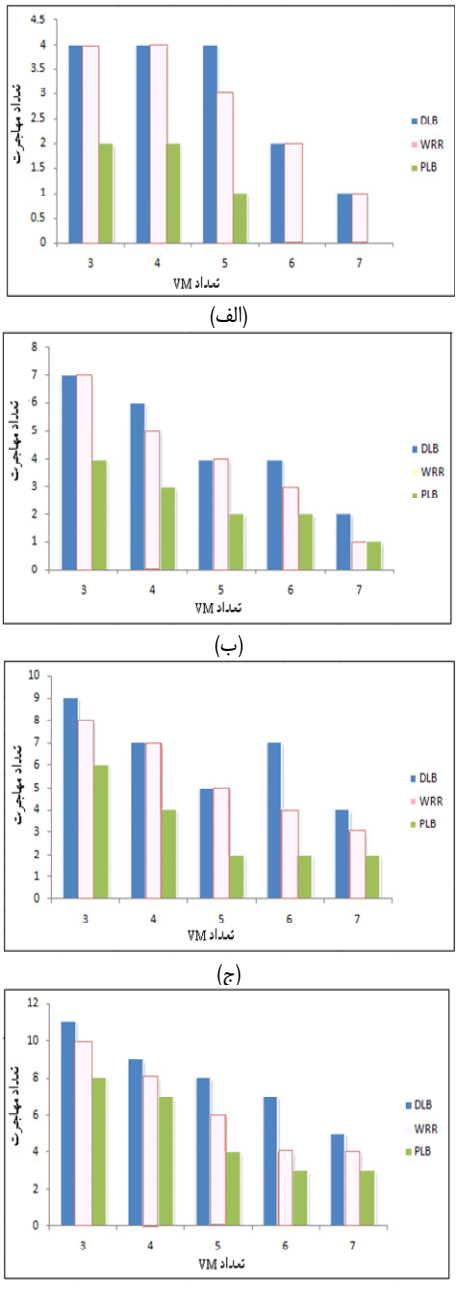


شکل ۷: نرخ عدم موفقیت کارهای برای روش‌های PLB، FIFO، DLB و WRR.

$$\alpha = \frac{L_{VM_i,t_.} - L_{VM_i,t}}{L_{VM_i,t_.}} \qquad (۱۶)$$

که در آن $L_{VM_i,t_.}$ بار اولیه VM_i در لحظه $t_.$ و $L_{VM_i,t}$ بار VM_i در لحظه فعلی (t) است. این پارامتر، معیار خوبی برای تحلیل تأثیر الگوریتم موازنه بار بر روی سیستم و میزان مهاجرت VMها است. مقدار α در بازه [۰,۱] تغییر می‌کند. اگر α برابر یک باشد به این معنی است که گره پردازشی مورد نظر در لحظه t تمام کارهای موجود در صف خود را اجرا نموده است و نرخ سرویس‌دهی بالایی دارد. در شکل ۶ میزان α به ازای تغییر بار کاری نشان داده شده است. مفهوم بار کاری عبارت است از نسبت کل بار ارسال‌شده به ابر به کارهایی که در طول دوره شبیه‌سازی امکان اجرای آنها را داشته است. همان طور که در شکل مشاهده می‌کنیم، روش پیشنهادی دارای بیشترین و روش FIFO دارای کمترین مقدار α است. در نمودار روش پیشنهادی مقدار α همواره بیشینه است زیرا در این روش همواره مناسب‌ترین VM برای کار مورد نظر انتخاب می‌شود و کار مورد نظر در حداقل زمان ممکن اجرا می‌شود. بنابراین همیشه VMها دارای صف‌های کوتاه یا بدون صف هستند و صف مربوطه در مدت زمان کمی خالی می‌شود. اما در روش‌های DLB، FIFO و WRR از آنجا که همه شاخص‌ها برای انتخاب VM مورد نظر لحاظ نمی‌شوند، کارهایی که در صف VMها قرار می‌گیرند تناسب زیادی با توانایی پردازشی VMها ندارند، بنابراین صف مربوط به VMها دیرتر خالی می‌شود و مقدار α تنها زمانی که بار کاری سبک است، تا حدی مطلوب می‌باشد.

در شکل ۷ نرخ عدم موفقیت کارها برای ۴ الگوریتم مورد نظر نشان داده شده و همان طور که می‌بینیم در بارهای کم، این نرخ برای هر چهار الگوریتم کم و تقریباً برابر است ولی در بار ۰/۵ این نرخ برای روش‌های FIFO، WRR و DLB علی‌الخصوص روش FIFO به شدت افزایش یافته است. یک دلیل این موضوع به خاطر این است که VMهای موجود زیر فشار بار رفتند و FIFO مجبور به ایجاد VM های جدید برای انجام‌دادن پردازش‌های خود است که این امر باعث از بین رفتن کارهای بعدی شده است. دلیل دیگر این است که در FIFO فقط زمان ورود کارها به ابر در نظر گرفته می‌شود و بنابراین کارها به سرعت به منابع نامتناسب



(الف)

(ب)

(ج)

(د)

شکل ۸: (الف) تعداد مهاجرت بر اساس تعداد VM برای ۱۰ کار، (ب) تعداد مهاجرت بر اساس تعداد VM برای ۲۰ کار، (ج) تعداد مهاجرت بر اساس تعداد VM برای ۴۰ کار و (د) تعداد مهاجرت بر اساس تعداد VM برای ۸۰ کار.

با نیازهایشان فرستاده می‌شوند و با افزایش بار کاری VMها به سرعت پر می‌شوند، بنابراین کارهای بعدی ارسالی به ابر از بین خواهند رفت. برای الگوریتم WRR نیز وضع تقریباً به صورت مشابه است. در روش پیشنهادی چون تمام معیارها برای فرستادن کار مربوطه به گره پردازشی مورد نظر در نظر گرفته شده است، نرخ کارهای ناموفق کاهش چشمگیری دارد.

شکل ۸ تعداد مهاجرت VMها بر اساس تعداد VMها، زمانی که تعداد کارها از ۱۰ تا ۸۰ تغییر می‌کند را نشان می‌دهد. همان طور که در این شکل مشاهده می‌کنیم در بارهای سبک تعداد مهاجرت برای روش پیشنهادی نزدیک به صفر است و با افزایش تعداد کارها نیز، تعداد مهاجرت به میزان کمی افزایش می‌یابد زیرا در صورت نبودن VM مناسب برای کار مورد نظر، بهترین سرور برای ایجاد VM جدید انتخاب می‌شود و در نهایت اگر سرور مناسبی نیز یافت نشد که احتمال آن در روش پیشنهادی کم است، اقدام به مهاجرت VMها از سایر خوشه‌ها به خوشه مربوطه می‌شود. بنابراین روش پیشنهادی نسبت به دو روش WRR و DLB تعداد مهاجرت کارها را به مقدار زیادی کاهش داده است.

٦- نتیجه

در این مقاله، یک روش موازنه بار نامتمرکز بر پایه روش تصمیم‌گیری پرومته و روش وزن‌دهی فازی برای محیط ناهمگن ابر ارائه شد. در این الگوریتم، انتخاب VM مناسب برای کار مورد نظر بر اساس تمام معیارهای کمی و کیفی و بر پایه تمام نیازهای کاربر انجام شد. روش پیشنهادی با روش‌های دیگر مقایسه شد و نشان داده شد که این روش با کمترین زمان بیکاری VMها محیطهای ناهمگن را به خوبی مدیریت می‌کند و دارای کمترین نرخ مهاجرت VMها است و در کمترین زمان ممکن وظایف را اجرا می‌کند. همچنین چون هر سرور مسئول موازنه بار VMهای خود و سرورهای دیگر است مشکل روش‌های متمرکز و از کار افتادن کل سیستم با یک نقطه از خرابی کاملاً مرتفع شده است.

مراجع

[1] B. Rajkumar, C. Yeoa, and S. Venugopala, "Market oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities," in *Proc. 10th IEEE Int. Conf. on High Performance Computing and Communications,* pp. 5-13, Sep. 2008.

[2] L. M. Vaquero, L. R. Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50-55, Jan. 2009.

[3] K. Sunny and K. Shivani, "Analysis of different scheduling algorithms under cloud computing," *International J. of Computer Science and Information Technologies*, vol. 5, no. 2, pp. 2592-2595, 2012.

[4] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," in *Proc, 7th IEEE Heterogeneous Computing Workshop*, pp. 79-87, Mar. 2012.

[5] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," in *Proc. of the 8th Heterogeneous Computing Workshop, HCW'99*, p. 30, Apr. 2011.

[6] P. Varalakshmi, A. Ramaswamy, A. Balasubramanian, and P. Vijay Kumar, "An optimal workflow based scheduling and resource allocation in cloud," *Advances in Computing and Communications*, vol. 190, pp. 411-420, 2011.

[7] L. F. Bittencourt and E. Madeira, "A cost optimization algorithm for workflow scheduling in hybrid clouds," *J. of Internet Services and Applications*, vol. 2, no. 3, pp. 207-227, Dec. 2011.

[8] A. Zomaya and Y. H. Teh, "Observations on using genetic algorithms for dynamic load-balancing," *IEEE Trans. on Parallel and Distributed Systems*, vol. 12, no. 9, pp. 899-912, Sep. 2001.

[9] M. Zaharia, D. Borthakur, J. Sarma, and K. Elmeleegy, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in *Proc. of 5th European Conf. on Computer Systems, EUROSYS*'10, pp. 265-278, Paris, France, 13-16 Apr. 2010.

[10] B. Yagoubi and Y. Slimani, "Dynamic load balancing strategy for grid computing," *WASET International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 2, no. 7, pp. 260-265, ???. 2008.

[11] P. P. Bonissone and K. S. Decker, "Selecting uncertainly calculi and granularity: an experiment in trading off precision and complexity," Kanal and Lemmer, Eds., *Uncertainty in Artificial Intelligence*, Amsterdam: North Holland: Elsevier Science, 1986, pp. 217-247,

[12] P. P. Bonissone, "A fuzzy set based linguistic approach: theory and applications," in Approximate Reasoning in Decision Analysis. Gupta, MM and Sanchez (Eds), North-Holland: 1982, pp. 329-339.

[۱۳] م. ج. اصغرپور، *تصمیم‌گیری‌های چندمعیاره*، انتشارات دانشگاه تهران، چاپ دهم، ۱۳۹۰.

[14] A. Sulistio and R. Buyya, "A grid simulation infrastructure supporting advance reservation," in *Proc. of the 16th Int. Conf. on Parallel and Distributed Computing Systems*, 7 pp., Nov. 2004.
[15] C. Dumitrescu and I. Foster, "Gangsim: a simulator for grid scheduling studies," in *Proc. of the IEEE Int. Symp. on Cluster Computing and the Grid*, *CCGrid'05*, vol. 2, pp. 1151-1158, May. 2005.
[16] B. Yagoubi and Y. Slimani, "Task load balancing strategy for grid computing," *J. of Computer Science*, vol. 3, no. 3, pp. 186-194, Apr. 2012.
[17] A. Revar, M. Andhariya, D. Sutariya, and M. Bhavsar, "Load balancing in grid environment using machine learning-innovative approach," *International J. of Computer Applications*, vol. 8, no. 10, pp. 975-8887, Oct. 2010.
[18] M. Randles, A. Taleb-Bendiab, and D. Lamb, "Scalable self governance using service communities as ambient," in Proc. of the IEEE Workshop on Software and Services Maintenance and Management, SSMM'09 within the 4th IEEE Congress on Services, IEEE SERVICES-I, pp. 813-820, Los Angeles, CA, USA, 6-10 Jul. 2009.

سمیرا حورعلی در سال ۱۳۹۰ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه صنعتی شاهرود و در سال ۱۳۹۳ مدرک کارشناسی ارشد مهندسی کامپیوتر خود را از دانشگاه محقق اردبیلی دریافت نمود. نام‌برده از سال ۱۳۹۳ در گروه مهندسی کامپیوتر و الکترونیک موسسه آموزش عالی شاهرود مشغول به تدریس می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: پردازش ابری، پردازش تصویر، فناوری اطلاعات و شبکه‌های کامپیوتری.

شهرام جمالی تحصیلات خود را در مقطع کارشناسی سخت افزار کامپیوتر در سال ۱۳۷۸ از دانشگاه صنعتی امیرکبیر و در مقاطع کارشناسی ارشد و دکتری معماری سیستم‌های کامپیوتری به‌ترتیب در سال‌های ۱۳۸۰ و ۱۳۸۷ از دانشگاه علم وصنعت ایران به پایان رسانده است و هم اکنون دانشیار دانشکده مهندسی کامپیوتر دانشگاه محقق اردبیلی می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان پردازش توزیع شده، کارایی سیستم‌های کامپیوتری، شبکه‌های کامپیوتری و مهندسی امنیت اطلاعات می‌باشد.

فاطمه حورعلی در سال ۱۳۸۵ مدرک کارشناسی مهندسی برق خود را از دانشگاه صنعتی شاهرود و در سال ۱۳۸۸ مدرک کارشناسی ارشد مهندسی برق خود را از دانشگاه صنعتی سهند تبریز دریافت نمود. ایشان از سال ۱۳۹۳ تا کنون عضو هیأت علمی و مدیر گروه مهندسی برق مجتمع آموزش عالی اسفراین می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: بینایی کامپیوتر، پردازش تصویر، شبکه‌های عصبی و بازشناسی الگو.