توازن بار در محیط رایانش ابری با در نظر گرفتن وابستگی میان وظایف و استفاده از الگوریتم ژنتیک تطبیقی

یلدا درخشانیان[۱]، سید جواد میرعابدینی[۲*]، علی هارون‌آبادی[۳]

۱: گروه مهندسی کامپیوتر، نرم‌افزار، واحد بوشهر، دانشگاه آزاد اسلامی، بوشهر، ایران y.derakhshanian@gmail.com
۲*: گروه مهندسی کامپیوتر، واحد تهران مرکز، دانشگاه آزاد اسلامی، تهران، ایران j_mirabedini@iauctb.ac.ir
۳: گروه مهندسی کامپیوتر، واحد تهران مرکز، دانشگاه آزاد اسلامی، تهران، ایران a.harounabadi@iauctb.ac.ir

تاریخ دریافت: ۱۳۹۶/۱/۲۶ تاریخ پذیرش: ۱۳۹۶/۶/۹

چکیده

گسترش روز افزون نیازهای محاسباتی، اهمیت استفاده از رایانش ابری را روز به روز بیشتر می‌کند. رایانش ابری، یک مدل رایانشی بر مبنای شبکه‌های رایانه‌ای است که الگویی تازه برای عرضه‌ی منابع را ارائه می‌دهد، بگونه‌ای که کاربران بر اساس نیاز خود، منابع را درخواست نموده یا آن‌ها را آزاد می‌سازند. هنگامیکه تقاضاها برای استفاده از منابع رایانشی افزایش می‌یابند، توزیع مناسب آنها از اهمیت بالایی برخوردار می‌شود، چراکه اگر یک واحد پردازشی دارای حجم زیادی از وظایف، و واحدی دیگر تقریباً بیکار باشد، از منابع بخوبی استفاده نمی‌شود و همچنین زمان اتمام کل وظایف می‌تواند بسیار افزایش بیابد. لذا برای غلبه بر این مشکل، از تکنیک توازن بار استفاده می‌شود. بطورکلی از دید محاسباتی، به فرایند توزیع متعادل بار بر روی واحدهای پردازشی، توازن‌بار گفته می‌شود. در اکثر پژوهش‌های انجام‌شده در رابطه با توازن‌بار، تعاملات میان وظایف در حال اجرا، در نظر گرفته نشده، لذا در صورتیکه وظایف در تعامل با یکدیگر، در واحدهای پردازشی مجزا، در یک شبکه توزیع‌شده قرار گرفته باشند، تعاملات میان آن‌ها می‌تواند در زمان اتمام کل وظایف، تاثیرگذار باشد. هدف از این پژوهش، ارائه روشی است که بتواند با در نظر گرفتن تعاملات میان وظایف، به یک توازن‌بار مطلوب در شبکه دست یابد، بطوریکه زمان اتمام کل و زمان بیکاری ماشین‌ها به حداقل برسند. برای این منظور، از الگوریتم ژنتیک استفاده می‌شود. نتایج آزمایشی بدست‌آمده نشان می‌دهند که محلی‌کردن تعاملات، تاثیر قابل توجهی در کاهش زمان اتمام کل خواهد داشت.

واژه های کلیدی: توازن‌بار، رایانش ابری، وابستگی وظایف، ژنتیک تطبیقی

۱- مقدمه

رشد روز افزون محاسبات موجب شده که برای پردازش آن‌ها از شبکه‌های توزیع‌شده استفاده شود. همچنین، استفاده موثر از منابع، می‌تواند باعث افزایش بهره‌وری سیستم شود. رایانش ابری، مدلی است که توانسته بنحوی منابع را بر اساس تقاضا، برای وظایف موجود در شبکه فراهم نموده [۴، ۱۹، ۲۰، ۲۵]، و موجب دستیابی به مزایایی از قبیل صرفه‌جویی در هزینه‌ها، بهبود بهره‌وری، افزایش مقیاس‌پذیری و... گردد [۹]. از طرفی، برای بهبود کارایی، بارهای کاری باید بطور مناسب بر روی سرورها و مراکز داده ارسال شوند [۱۷]. به همین دلیل، الگوریتم‌های توازن بار متعددی پیشنهاد شده‌اند [۷، ۸، ۱۱، ۱۲، ۱۵، ۱۶، ۱۸، ۲۱، ۲۳، ۲۴]. توازن بار، برای بالا بردن کارایی و سرعت سیستم استفاده می‌شود بطوریکه با متعادل نمودن حجم کاری بر روی واحدهای پردازشی، می‌توان از ظرفیت سیستم استفاده بیشتری کرد. همچنین متوازن بودن بارهای قرار گرفته بر روی واحدهای پردازشی، باعث می‌شود که سیستم به زمان اتمام کل کمتری دست‌یابد. بطور کلی الگوریتم‌های توازن‌بار به دو دسته‌ی ایستا و پویا تقسیم می‌شوند [۴، ۱۳]. الگوریتم‌های ایستا، حالت جاری سیستم را در نظر نمی‌گیرند و تصمیمات توازن‌بار، صرفا براساس دانش و آگاهی قبلی از سیستم می‌باشد.

این نوع الگوریتم‌ها نمی‌توانند با تغییرات بار در زمان اجرا سازگار شوند. الگوریتم‌های پویا، حالت جاری سیستم را در نظر می‌گیرند [۱۷] و توزیع بار را در زمان اجرا انجام می‌دهند، بنابراین می‌توانند با تغییرات بار در زمان اجرا سازگار شوند. بعضی از الگوریتم‌های پویا، تطبیقی هستند؛ به این معنی که، این الگوریتم‌ها می‌توانند با تغییرات حالت سیستم، تغییر یابند [۸].

اگر وظایفِ در حال اجرا در یک محیط پردازشی توزیع‌شده، با یکدیگر در ارتباط باشند بطوریکه یک وظیفه، برای ادامه‌ی روند کار خود، به تغییری یا پاسخی از جانب وظیفه‌ای دیگر، نیاز داشته باشد؛ می‌توان گفت که در چنین سیستمی، وظایف به یکدیگر وابسته هستند. اغلب الگوریتم‌های ارائه شده برای مساله توازن‌بار در سیستم‌های توزیع‌شده، مبحث تعاملات میان وظایف را در نظر نگرفته‌اند، لذا در صورت وجود وابستگی در بین گروهی از وظایف، حتی اگر بارها بصورت متوازن در سیستم، توزیع شده باشند، باز هم زمان زیادی از اجرای آن‌ها، صرف تاخیرهای ناشی از گلوگاه‌های ارتباطی توزیع‌شده در شبکه می‌شود [۵]. این مقاله، با استفاده از یک الگوریتم ژنتیک، سعی می‌کند بدون درگیر شدن با مسائل پیچیده، ارتباطات و وابستگی‌های میان وظایف را در نظر بگیرد و از سرعت این الگوریتم، در رسیدن به یک توازن‌بار سراسری بهره بگیرد.

ادامه این مقاله بدین شرح سازماندهی شده است: در بخش ۲، ادبیات تحقیق در رابطه با توازن‌بار گزارش شده است. در بخش ۳، یک الگوریتم ژنتیک بمنظور برقراری توازن بار در محیط رایانش ابری پیشنهاد می‌شود که تعاملات میان وظایف را در نظر می‌گیرد. بخش ۴ نتایج آزمایشی را ارائه می‌دهد و سرانجام، نتیجه‌گیری این مقاله در بخش ۵ صورت می‌گیرد.

۲- ادبیات تحقیق

تا کنون تحقیقات بسیار زیادی در رابطه با توازن‌بار در انواع سیستم‌های توزیع‌شده مورد مطالعه قرار گرفته است که گزیده‌ای از آن‌ها در این بخش ارائه می‌گردد. قاضی‌پور و همکاران [۱۴] یک الگوریتم زمانبندی کار را پیشنهاد داده‌اند که بوسیله‌ی آن، کارهای موجود در محیط گرید به منابع دردسترس، تخصیص داده می‌شوند. این الگوریتم بر اساس الگوریتم بهینه‌سازی کلونی مورچگان است که با الگوریتم زمانبندیِ حق انتخاب، ترکیب شده بطوریکه از نتیجه‌ی حق انتخاب در الگوریتم بهینه‌سازی کلونی مورچگان پیشنهادی، استفاده میکند. هدف اصلی از این مقاله، به حداقل رساندن زمان اتمام کلمجموعه‌ای از کارهای داده شده است.

سعادتی و میرعابدینی [۲]، یک روش فازی جهت برقراری توازن بار در رایانش ابری را پیشنهاد داده‌اند که در آن با استفاده از یک کنترل کننده‌ی فازی، گره‌ها وزن دهی می‌شوند.

پژوهش حاتمیان و همکاران [۱]، با استفاده از ایده‌ی ترکیب نقاط قوت الگوریتم‌های ژنتیک و کلونی مورچه‌ها، روشی برای حل مساله توازن بار در رایانش ابری پیشنهاد نموده است.

در پژوهش میر و فاضلی [۳]، یک الگوریتم زمانبندی وظایف برای محیط‌های ابری بر اساس الگوریتم ژنتیک بهینه‌سازی شده، ارائه گردیده که میتواند توازن بار را در سرور برقرار کرده و هزینه را بهینه نماید.

در [۲۲]، یک الگوریتم زمانبندی بهینه‌ی ترکیبی به نام GPSO ارائه شده است. GPSO از الگوریتم‌های بهینه‌سازی ازدحام ذرات و جستجوی محلی گرانشی تشکیل شده است. از آنجا که الگوریتم‌های بهینه‌سازی ازدحام ذرات در جستجوی محلی ضعیف است، الگوریتم جستجوی محلی گرانشی برای بهبود آن استفاده شده و از به دام افتادن در یک بهینه محلی جلوگیری می‌کند. این الگوریتم، زمان اتمام کل را کاهش داده و وظایف از دست رفته را به حداقل می‌رساند.

در [۱۱]، یک استراتژی توازن بار، با استفاده از الگوریتم ژنتیک برای رایانش ابری ارائه شده بطوریکه پردازنده‌ی بهینه‌ی سراسری برای کار موجود در یک ابر را پیدا می‌کند. این الگوریتم، بار موجود بر روی زیرساخت ابر را متوازن ساخته و در عین حال سعی می‌کند زمان اتمام کل را برای یک مجموعه از وظایف داده شده به حداقل برساند.

۳- معرفی الگوریتم پیشنهادی

در یک محیط توزیع‌شده (مانند رایانش ابری)، وابستگی میان وظایف، می‌تواند بطور قابل توجهی، زمان اتمام کل را افزایش دهد. بعبارتی، زمان اتمام کل می‌تواند وابسته به عوامل ارتباطی مانند پهنای باند شبکه و تاخیرهای ارتباطی گردد. برای برطرف کردن این مشکل، در [۵، ۶، ۱۰] پیشنهاد شده که با محلی کردن ارتباطات خارجی، نرخ ارتباطات محلی را افزایش دهیم. یعنی اینکه با قرار دادن وظایف وابسته به هم، بر روی ماشین‌های یکسان، این امکان فراهم می‌شود که ارتباطات خارجی میان وظایف به ارتباطات محلی

تبدیل گردد. در نتیجه، وظایف برای ارتباط با یکدیگر لازم نیست بابت محدودیت‌های شبکه‌ای معطل گردند و می‌توانند با استفاده از ارتباطات حافظه‌ی مشترک بر روی یک ماشین یکسان، با یکدیگر در تعامل باشند. هر چه که تعاملات محلی میان وظایف وابسته، بیشتر باشد، زمان اتمام کل، کمتر می‌گردد. ما یک ارتباط حافظه اشتراکی که بر روی یک ماشین فیزیکی قرار گرفته را ارتباط محلی، و یک ارتباط از راه دور که شامل یک ارتباط شبکه‌ای فیزیکی، در بین ماشین‌های فیزیکی مختلف می‌شود را ارتباط خارجی تعریف می‌نماییم [۵، ۶].

در روش پیشنهادی، هنگام شروع الگوریتم، تعداد ارتباطات داخلی (محلی) و ارتباطات خارجی وظایف، تا زمان جاری، به همراه مشخصه‌ی میلیون دستورالعمل در ثانیه مربوط به هر ماشین مجازی و مشخصه‌ی میلیون دستورالعملِ دستورالعمل مربوط به هر وظیفه، جمع‌آوری می‌شوند. سپس با استفاده از یک تابع برازش، زمان اتمام کل، زمان بیکاری ماشین‌های مجازی و نرخ ارتباطات محلی [۵، ۶، ۱۰] ارزیابی می‌گردند. هر چه که مقدار تابع برازش بدست آمده بیشتر باشد، نرخ ارتباطات، محلی‌تر، زمان بیکاری ماشین‌ها، کمتر و زمان اتمام کل وظایف نیز کمتر خواهد بود.

۳-۱- تشریح مراحل الگوریتم

در ابتدا مفروضاتی که در روش پیشنهادی در نظر گرفته شده‌اند، شرح داده می‌شوند. در این الگوریتم، برای ساده‌تر کردن صورت مساله جهت ارایه‌ی رویکرد پیشنهادی، فرض می‌کنیم که تعداد وظایف موجود بر روی هر ماشین فیزیکی، یکسان است. بدیهی است که در صورت ناهمگن بودن قدرت پردازشی ماشین‌های فیزیکی یا تعداد دستورالعمل‌های وظایف، با قرار دادن تعداد وظایف متفاوت بر روی ماشین‌های فیزیکی، می‌توان معیارهای متفاوتی از جمله بهره‌وری و زمان اتمام کل را بهبود داد. در پژوهش‌های آینده با بهره‌گیری از تکنیک وزن‌دهی به هر ماشین‌فیزیکی می‌توان از توانایی هر ماشین به نحو بهتری استفاده نمود. همچنین برای ارایه بهتر عملکرد رویکرد پیشنهادی فرض می‌شود که وظایف، بطور تصادفی در سراسر ماشین‌ها پخش شده‌اند. با در نظر گرفتن این فرض می‌توان اطمینان داشت که الگوریتم پیشنهاد شده می‌تواند در شرایط متفاوت به یک راه‌حل قابل قبول دست پیدا کند.

الگوریتم با یک جمعیت اولیه شروع به کار می‌کند. هر یک از اعضای جمعیت (کروموزوم)، حاوی تعدادی مساوی از وظایف هستند که بصورت تصادفی در سراسر ماشین‌های فیزیکیِ شبیه‌سازی‌شده، پخش شده‌اند، بنابراین از مساوی بودن تعداد وظایف بر روی هر ماشین فیزیکی، اطمینان داریم. سپس با استفاده از یک تابع برازش، سعی در بهینه‌سازی چینش وظایف در کروموزوم می‌نماییم.

قبل از اینکه الگوریتم شروع شود، تعدادی پارامتر ورودی برای آن مشخص می‌گردد. اولین پارامترهای مساله، تعداد ماشین‌های فیزیکی، تعداد ماشین‌های مجازی بر روی هر ماشین فیزیکی و تعداد وظایف هستند. در این الگوریتم، فرض کردیم که تعداد ماشین‌های مجازی موجود در هر ماشین فیزیکی، برابر است. تعداد کل ماشین‌های مجازی از فرمول ۱ بدست می‌آید.

$$VM_{no} = M_{no} \times VM_{no(m)} \qquad (۱)$$

در فرمول ۱، M_{no} بیانگر تعداد ماشین‌های فیزیکی، $VM_{no(m)}$ ، بیانگر تعداد ماشین‌های مجازی بر روی هر ماشین فیزیکی و VM_{no} تعداد کل وظایف را نشان می‌دهد. دو پارامتر مهم دیگر، درصد تقاطع و درصد جهشی هستند که قرار است توسط، الگوریتم ژنتیک، بکار گرفته شود. تعدادی از پارامترها، برای مدیریت ارتباطات میان وظایف، تعریف شده‌اند که عبارتند از: الف) درصد ارتباطات، که مشخص می‌کند بطورکلی چند درصد گروه تعاملی در میان وظایف وجود داشته باشد. این پارامتر، بسیار به پارامتر درجه ارتباط، وابسته است، ب) درجه ارتباط، مشخص می‌کند که در هر گروه تعاملی، چند وظیفه، می‌تواند حضور داشته باشد، برای مثال اگر درجه ارتباط ۳ باشد، یعنی در هر گروه تعاملی ۴ وظیفه، حضور خواهند داشت، ج) وضعیت درجه ارتباط، ما دو وضعیت ایستا و پویا را برای درجه ارتباط در نظر گرفته‌ایم، درجه ارتباط ایستا، به این معنا است که تعداد وظایف موجود در هر گروه تعاملی، با هم برابر خواهند بود، و درجه ارتباط پویا، بیان می‌کند که حداکثر تعداد وظایف موجود در یک گروه تعاملی، برابر با درجه ارتباط خواهد بود. در حالتی که درجه ارتباط، پویا باشد، حداقل تعداد وظایف موجود در یک گروه تعاملی، ۲ است، و د) جریمه، بیانگر مقدار جریمه‌ای است که به ازای هر ارتباط خارجی، در نظر گرفته می‌شود.

پس از آنکه پارامترهای ورودی الگوریتم مشخص شدند، ماشین‌ها بهمراه ماشین‌های مجازیشان، وظایف، گروه‌های تعاملی و تعاملات آن‌ها مدلسازی می‌شود و در نهایت جمعیت اولیه، تولید می‌گردد. ما اندازه‌ی جمعیت را ۱۰ قرار دادیم. این جمعیت به الگوریتم ژنتیک داده می‌شود و هر یک از اعضای جمعیت، با استفاده از تابع برازش ارائه شده در فرمول ۲، ارزیابی می‌گردند.

$$Fitness = w1 \times \left(\frac{1}{Makespan}\right) + w2 \times \left(\frac{1}{IT_{mean}}\right) + w3 \times (LCR_{mean}) \quad (۲)$$

وزن‌های w1=0.5 ، w1=0.5 و w1=0.4، برای کاهش یا افزایش تاثیر هر یک از پارامترهای تاثیرگذار در تابع برازش استفاده می‌شوند. با توجه به اجراهای متعددی که در شرایط مختلف با تعداد دورها و وزن‌های متفاوت انجام دادیم، به این نتیجه رسیدیم که این مقادیر تعیین شده برای وزن‌ها می‌توانند در اغلب شرایط، پاسخ بهتری را ارائه دهند. بنابراین می‌توان گفت که مقادیر وزن‌ها با استفاده از روش سعی و خطا بدست آمده‌اند. هر یک از پارامترهای Makespan، IT_{mean} (متوسط زمان بیکاری) و LCR_{mean} (متوسط نرخ ارتباطات محلی)، قبل از قرار گرفتن در تابع برازش، بر اساس موقعیت فعلی وظایف در کروموزوم، محاسبه می‌شوند و حاصل آن‌ها در تابع برازش، تعیین کننده‌ی میزان برازندگی کروموزوم خواهد بود. نحوه‌ی محاسبه‌ی هر یک از این پارامترها در ادامه شرح داده می‌شود.

$$Makespan = Max\left(Makespan_{EM}(m)\right) \quad (۳)$$

$$Makespan_{EM}(m) = Makespan(m) + EC_{no}(m) \times p \quad (۴)$$

$$Makespan(m) = Max\left(FT_{VM}(m.v.t)\right) \quad (۵)$$

$$EC_{no}(m) = \sum_{t=1}^{Task_{no}(m)} EC_{no}(t) \quad (۶)$$

آوردن زمان اتمام کل مورد استفاده قرار می‌گیرند. در فرمول ۳ که برای محاسبه زمان اتمام کلِ تمام سیستم استفاده می‌شود، $Makespan_{EM}(m)$ بیانگر زمان اتمام کلِ ماشین m است که پیغام‌های خارجی ماشین را در نظر می‌گیرد. در فرمول ۴، $EC_{no}(m)$ بیانگر تعداد پیغام‌های خارجی ماشین m می‌باشد و p، مقدار جریمه‌ای است که برای هر ارتباط خارجی در نظر گرفته می‌شود. فرمول ۵، زمان اتمام کل هر ماشین فیزیکی در شبکه را بدست می‌آورد. در این فرمول $FT_{VM}(m.v.t)$، زمان جریان هر ماشین مجازی موجود در ماشین فیزیکی m را بدست می‌آورد که شماره ماشین مجازی، با v و شماره وظیفه موجود بر روی آن با t نشان داده می‌شود. $EC_{no}(m)$ از طریق فرمول ۶، با بدست آورد مجموع تعداد پیغام‌های خارجی وظایف موجود بر روی ماشین m محاسبه می‌گردد. در این فرمول $EC_{no}(t)$، تعداد پیغام‌های خارجی وظیفه t را نشان می‌دهد. جهت بدست آوردن زمانِ‌جریان هر ماشین مجازی (به میلی ثانیه)، از فرمول ۷ استفاده می‌شود. این فرمول، زمان اتمام اجرای هر وظیفه موجود بر روی ماشین مجازی را نشان می‌دهد.

$$FT_{VM}(m, v, t) = \left(\frac{MI(t)}{MIPS(m,v)}\right) \quad (۷)$$

در فرمول ۷، m بیانگر اندیس ماشین فیزیکی، v بیانگر اندیس ماشین مجازی موجود در ماشین فیزیکی، t، اندیس وظیفه‌ی موجود در ماشین مجازی، MI(t) تعداد دستورالعمل‌های وظیفه t و MIPS(m,v) تعداد دستورالعمل‌هایی که توسط ماشین مجازی v (موجود در ماشین فیزیکیm) در هر ثانیه قابل پردازش است را به واحد میلیون نشان می‌دهد. برای بدست آوردن زمانِ‌جریان مربوط به هر ماشین فیزیکی، کافیست که مجموع زمانِ‌جریان ماشین‌های مجازی موجود در آن ماشین فیزیکی را بدست بیاوریم. زمانِ‌جریان مربوط به هر ماشین فیزیکی از طریق فرمول ۸ بدست می‌آید.

$$FT(m) = \sum_{v=1}^{VM_{no}(m)} FT_{VM}(m, v, t) \quad (۸)$$

در فرمول ۹، با در نظر گرفتن اینکه وظایف می‌توانند با یکدیگر در تعامل باشند، میزان تعاملات وظایف را در محاسبه زمان جریان، تاثیر می‌دهیم.

$$FT_{EM}(m) = FT(m) + EC_{no}(m) \times p \quad (۹)$$

در فرمول ۹، برای در نظر گرفتن هزینه‌ی ارتباطات خارجی، به ازای هر تعامل خارجی، یک مقدار جریمه p را تعیین نموده‌ایم. ما مقدار این جریمه را ۱۰ در نظر گرفته‌ایم. برای بدست آوردن پارامتر IT_{mean} از تابع برازش (فرمول ۲)، از فرمول۱۰ استفاده می‌شود.

$$IT_{mean} = \frac{IT_{total}}{M_{no}}$$
$$If\ IT_{mean} = 0, IT_{mean} = 0.1 \quad (۱۰)$$

در فرمول ۱۰، M_{no}، تعداد ماشین‌های فیزیکی را نشان می‌دهد، و IT_{total} که زمان بیکاری کل ماشین‌ها را نشان می‌دهد، از فرمول ۱۱ بدست می‌آید. در این فرمول، VM_{no} ، بیانگر تعداد کل ماشین‌های‌مجازی است.

$$IT_{total} = (Makespan \times VM_{no}) - \sum_{m=1}^{M_{no}} FT_{EM}(m) \quad (۱۱)$$

پارامتر LCR_{mean} در تابع‌برازش (فرمول ۲)، متوسط نرخ ارتباطات محلی در شبکه را نشان می‌دهد. هر چه مقدار این پارامتر بیشتر باشد، به این معنا است که تعداد ارتباطات محلی موجود در شبکه بیشتر است. هر چه که مقدار این پارامتر، بیشتر باشد، به این معنا است که زمان اتمام کل، کاهش میابد. این پارامتر از طریق فرمول‌های ۱۲ و ۱۳ بدست می‌آید [۵، ۶].

$$LCR_{mean} = \frac{\sum_{m=1}^{M_{no}} LCR(m)}{M_{no}} \quad (۱۲)$$

$$LCR(m) = \frac{LC_{no}(m)}{LC_{no}(m)+EC_{no}(m)} \quad (۱۳)$$

در فرمول ۱۳، LCR(m) ، نرخ ارتباطات محلی، $LC_{no}(m)$ تعداد ارتباطات محلی و $EC_{no}(m)$ تعداد ارتباطات خارجی برای ماشین m را نشان می‌دهند. پس از اینکه پارامترهای ورودی و مولفه‌های سیستم، تعیین شدند، الگوریتم ژنتیک بر روی جمعیت اولیه اجرا می‌شود و در نهایت خروجی، برابر با کروموزومی خواهد بود که دارای بیشترین مقدار تابع برازش باشد. خروجی الگوریتم پیشنهاد شده، یک چینش بهینه از وظایف (راه‌حل) برای واحدهای پردازشی توزیع شده در سیستم را ارائه می‌دهد. با توجه به اینکه در راه‌حل ارائه شده توسط الگوریتم، بار قرار گرفته بر روی ماشین‌های فیزیکی، متعادل هستند، بنابراین هر کاندیدی که برای مهاجرت وظایف، توسط الگوریتم، پیشنهاد می‌گردد، قابلیت مهاجرت را خواهد داشت. بنابراین نیاز نیست که قبل از مهاجرتِ هر وظیفه، شرایط برقراری توازن بار در سیستم، ارزیابی گردد. همچنین، از آنجا که ممکن است الگوی ارتباطی میان وظایف، با گذر زمان، تغییر کند و در آن هنگام، چینشی که توسط الگوریتم ژنتیک، انتخاب شده، یک راه‌حل مناسب نباشد، بنابراین، میتوان بصورت دوره‌ای، با نگه داشتن کروموزوم و مقدار تابع برازش بدست آمده از آخرین باری که الگوریتم اجرا شده، این الگوریتم را مجددا اجرا کرد. در صورتی که پاسخِ حاصل از اجرای مجدد الگوریتم، از وضعیت جاری سیستم بهتر باشد، میتوان راه‌حل جدید را جایگزین راه‌حل قبلی نمود. اگر حاصل تابع برازش، همواره از یک حد بالای غیر بهینه، کمتر نشود، بیانگر این است که منابع پردازشی‌ای که در حال اجرای وظایف هستند، نیازمند ارتقا می‌باشند، زیرا توانایی اجرای وظایف در وضعیت بهتری را ندارند. این منابع پردازشی، می‌توانند از جانب تامین‌کننده‌های منابع ابری تامین شوند [۱۰].

۴- تعریف مدل و نتایج آزمایشی

۴-۱- تعریف مدل سیستم

مدل سیستم، شامل تعدادی واحد اجرایی فیزیکی است که با عنوان ماشین، تعریف می‌شوند. هر ماشین، دارای تعدادی ماشین مجازی می‌باشد. فرض می‌شود که تعداد ماشین‌های مجازی موجود در هر ماشین با هم برابر هستند. بر روی هر ماشین مجازی، یک وظیفه می‌تواند در حال اجرا باشد. هر ماشین مجازی، دارای یک مشخصه‌ی میلیون دستورالعمل در ثانیه می‌باشد که بیانگر تعداد دستورالعمل‌هایی است که در ثانیه، قادر به اجرای آن‌ها خواهد بود. هر وظیفه، دارای یک مشخصه‌ی میلیون دستورالعمل است که تعداد دستورالعمل‌های آن را مشخص می‌کند. درصدی از وظایف می‌توانند با یکدیگر در تعامل باشند. ارتباط مدل شده‌ی میان وظایف، به این صورت است که درصدی از آنها، در گروه‌هایی ارتباطی، گروه‌بندی می‌شوند. وظایفی که در هر گروه ارتباطی قرار می‌گیرند، می‌توانند در زمان اجرا با یکدیگر در تعامل باشند. حداکثر تعداد پیغام‌هایی که توسط هر وظیفه می‌تواند به یک وظیفه‌ی دیگر ارسال شود، یک حد بالای دلخواه خواهد بود. در این روش، علاوه بر پیاده‌سازی آسان و سرعت آن در رسیدن به پاسخ بهینه سراسری، تمام کاندیدهای مهاجرت، می‌توانند در مقصدهای تعیین شده، مهاجرت داده شوند؛ زیرا الگوریتم ژنتیک در ساختار کروموزوم، همواره برابری تعداد وظایف موجود در هر ماشین را در نظر می‌گیرد.

۴-۲- نتایج آزمایشی

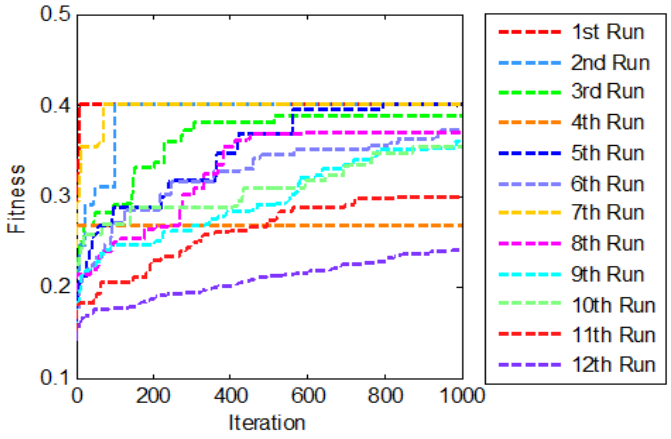
مجموعه‌ی نتایج بدست‌آمده از آزمایش‌ها و تحلیل‌های آن‌ها، در این بخش ارائه داده می‌شود. آزمایش‌ها بوسیله‌ی نرم‌افزار متلب، بر روی مجموعه‌هایی از ماشین‌ها و وظایف، با در نظر گرفتن حالت‌های متفاوت تعامل میان وظایف، انجام شده است. بطور کلی، دو سناریو برای الگوریتم پیشنهادی تعریف شده که با استفاده از آن‌ها، به ارزیابی عملکرد الگوریتم می‌پردازیم.

در سناریو اول، فرض می‌شود که پردازنده‌ها در محیط رایانش ابری، از لحاظ قدرت پردازشی در شرایط یکسان قرار دارند. سناریو برای تعداد وظایف متفاوتی اجرا می‌گردد و در کلیه اجراهای آن، تعداد ماشین‌های فیزیکی برابر با ۳، مشخصه‌ی میلیون دستورالعمل در ثانیه مربوط به هر ماشین مجازی برابر با ۱۰ و مشخصه‌ی میلیون دستورالعمل هر وظیفه، یک عدد تصادفی بین ۱ تا ۱۰ می‌باشد. درصد تقاطع، ۹۰٪ و درصد جهش، ۱۰٪ در نظر گرفته‌شده است. همچنین تعداد تکرارها در هر اجرای برنامه، ۱۰۰۰ دور است. این سناریو شامل دو زیر مجموعه است. الف) وظایف استقلال دارند و فاقد هر گونه وابستگی می‌باشند. بعبارتی پارامترهایی مانند درصد تعاملات و درجه ارتباط در اینجا معنی ندارد. ب) ارتباطات خارجی در نظر گرفته شوند. در این حالت، پارامترهای درصد تعاملات، درجه ارتباط و وضعیت درجه ارتباط مشخص می‌شوند. جدول‌های ۱ و ۲، بترتیب، نتایج بدست‌آمده از اجراهای مختلف در حالت‌های الف و ب از سناریو اول را نشان می‌دهند.

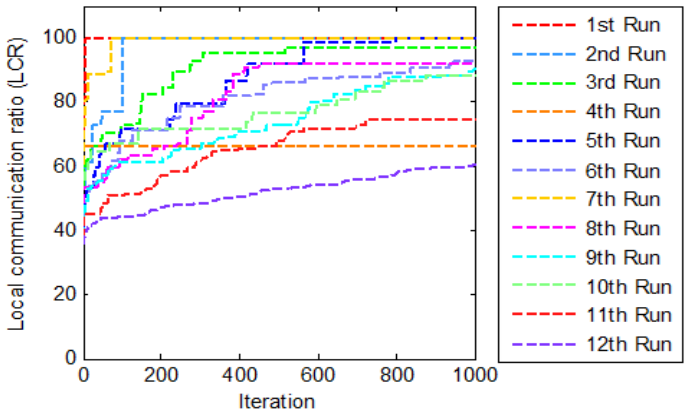
جدول۱: نتایج اجراهای مختلف در حالتی که وظایف مستقل هستند و پردازنده‌ها قدرت پردازشی یکسانی دارند

تعداد ماشین‌های مجازی در هر ماشین فیزیکی	تعداد کل وظایف در حال اجرا	زمان اتمام کل (میلی ثانیه)	متوسط نرخ ارتباطات محلی	متوسط زمان بیکاری هر ماشین (میلی ثانیه)
۲۵	۷۵	۹۹۲	معنی ندارد	۱۰۷۰۱,۴۲
۵۰	۱۵۰	۹۸۶,۸۷	معنی ندارد	۲۱۱۰۶,۶۷
۱۰۰	۳۰۰	۹۹۵,۱۳	معنی ندارد	۴۳۹۲۴,۳۳
۳۰۰	۹۰۰	۹۹۹,۴۶	معنی ندارد	۱۳۵۴۶۰,۸۳

در حالت الف از سناریو اول، با توجه به اینکه وظایف مستقل هستند و در هر ماشین مجازی، در زمان جاری، تنها یک وظیفه در حال اجرا است و قدرت پردازشی ماشین‌های مجازی نیز، با یکدیگر برابر می‌باشد، لذا در هر صورت، زمان اتمام کل، برابر با زمان جریان مربوط به ماشین مجازی‌ای است که حاوی وظیفه‌ای با بیشترین مقدار مشخصه‌ی میلیون دستورالعمل در شبکه می‌باشد. بنابراین در این حالت، مقادیر بدست آمده توسط تابع برازش در تمامی اجراها بسیار مشابه به هم، و برابر با یک عدد بسیار کوچک نزدیک به صفر می‌باشند. همچنین، مشاهده می‌شود که متوسط زمان بیکاری بالایی در شبکه وجود دارد. در مورد این معیار میتوان بیان کرد که هرچه اختلاف میان مشخصه‌ی میلیون دستورالعمل‌های وظایف بیشتر باشد، در نتیجه مقدار زمان بیکاری کل شبکه بیشتر می‌شود. برای جلوگیری از افزایش زمان بیکاری ماشین‌ها، صف‌های جدیدی از وظایف می‌توانند برای تخصیص به آنها در نظر گرفته شوند که بعنوان تحقیقات آینده می‌تواند مورد بررسی قرار بگیرد.



شکل ۱: تابع برازش بدست‌آمده توسط اجراهای موجود در جدول ۲



شکل ۲: نرخ ارتباطات محلی بدست‌آمده بوسیله‌ی اجراهای موجود در جدول ۲

جدول ۲: نتایج اجراهای مختلف در حالتی که وظایف با یکدیگر در تعامل هستند و پردازنده‌ها دارای قدرت پردازشی یکسانی می‌باشند

شماره اجرا	تعداد ماشین‌های مجازی در هر ماشین فیزیکی	تعداد کل وظایف در حال اجرا	درصد ارتباطات	درجه ارتباط	نوع درجه ارتباط	زمان اتمام کل (میلی ثانیه)	متوسط نرخ ارتباطات محلی	متوسط زمان بیکاری هر ماشین (میلی ثانیه)
۱	۲۵	۷۵	٪۴	۱	ایستا	۹۸۸.۷۱	۱۰۰٪	۱۰۹۶۹.۲۸
۲	۲۵	۷۵	٪۸	۳	پویا	۹۹۹.۵۴	۱۰۰٪	۱۰۱۶۱.۸۷
۳	۲۵	۷۵	٪۲۰	۳	پویا	۲۱۰۶.۸۶	۹۷.۲۳٪	۳۷۷۸۵.۶۴
۴	۵۰	۱۵۰	۴٪	۱	ایستا	۹۹۹.۳۲	۶۶.۶۶٪	۲۲۱۶۶.۲۳
۵	۵۰	۱۵۰	۸٪	۳	پویا	۹۸۲.۸۱	۱۰۰٪	۲۱۳۴۳.۱۲
۶	۵۰	۱۵۰	٪۲۰	۳	پویا	۴۲۲۶.۰۷	۹۳.۱۲٪	۱۸۱۴۸۰.۵
۷	۱۰۰	۳۰۰	٪۴	۱	ایستا	۹۹۹.۱۷	۱۰۰٪	۴۵۷۵۰.۴۳
۸	۱۰۰	۳۰۰	٪۸	۳	پویا	۴۰۷۸.۵۹	۹۲.۵۳٪	۳۵۳۱۰۷.۷۹
۹	۱۰۰	۳۰۰	٪۲۰	۳	پویا	۸۳۹۴.۸۲	۹۰.۳۵٪	۷۸۰۰۲۷.۵۷
۱۰	۳۰۰	۹۰۰	٪۴	۱	ایستا	۴۳۵۷.۱۲	۸۸.۶۵٪	۱۱۴۱۴۴۶.۸۱
۱۱	۳۰۰	۹۰۰	٪۸	۲	ایستا	۱۹۵۵۴.۸۲	۷۴.۹۶٪	۵۶۸۳۰۶۶.۲۶
۱۲	۳۰۰	۹۰۰	٪۲۰	۳	ایستا	۱۰۲۹۵۹.۳۴	۶۰.۶۶٪	۳۰۶۳۴۱۲۲.۴۲

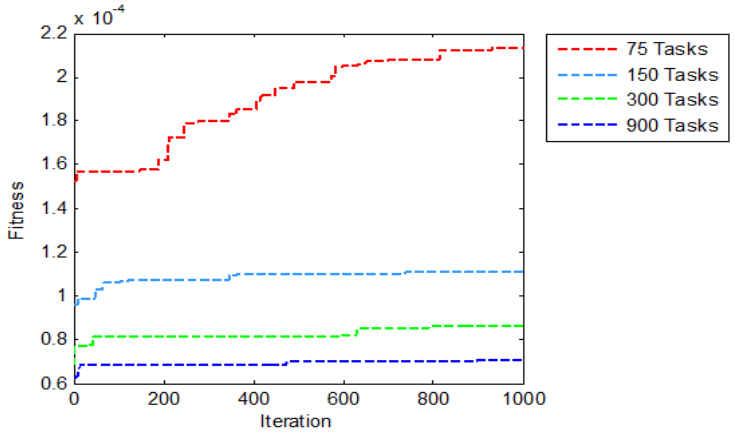
جدول ۲، تاثیر میزان محلی بودن تعاملات در حاصل زمان اتمام کل را نشان می‌دهد. زمانیکه نرخ ارتباطات محلی ۱۰۰٪ باشد، تابع برازش می‌تواند بهترین راه‌حل را ارائه دهد و حاصل آن، برابر با شرایطی باشد که وظایف بصورت مستقل از یکدیگر در حال اجرا باشند، بنابراین می‌توان نتیجه گرفت که محلی کردن تعاملات خارجی بر روی زمان اتمام کل تاثیر قابل توجهی خواهد داشت. شکل‌های ۱ و ۲، پیشرفت تابع برازش و نرخ ارتباطات محلی در جدول ۲ را نشان می‌دهند.

شکل‌های ۱و۲ نشان می‌دهند که با افزایش نرخ ارتباطات محلی، تابع برازش نیز افزایش میابد. همچنین از این شکل‌ها می‌توان متوجه شد که با افزایش تعداد وظایف، نرخ همگرایی به پاسخ مطلوب، کندتر می‌شود ولی در نهایت به راه‌حل مورد نظر، دست خواهیم یافت. در سناریو دوم فرض می‌شود که پردازنده‌های با قدرت پردازشی متفاوتی در رایانش ابری وجود دارند. این سناریو شامل دو زیر مجموعه می‌باشد که دقیقا مانند سناریو قبلی هستند. الف) وظایف استقلال دارند، ب) وظایف به یکدیگر وابستگی دارند. در تمام اجراهای مربوط به سناریو دوم، تعداد ماشین‌های فیزیکی برابر با ۳، مشخصه‌ی میلیون دستورالعمل در ثانیه مربوط به هر ماشین مجازی برابر با یک عدد تصادفی بین ۱ تا ۱۰ (میلیون دستورالعمل در ثانیه) و مشخصه‌ی میلیون دستورالعمل هر وظیفه، یک عدد تصادفی بین ۱ تا ۱۰ می‌باشد. درصد تقاطع، ۹۰٪ و درصد جهش، ۱۰٪ در نظر گرفته‌شده است و تعداد تکرارها در هر اجرای برنامه، ۱۰۰۰ دور می‌باشد. جدول ۳، خروجی‌های بدست‌آمده برای حالت الف از سناریو دوم را نشان می‌دهد.

جدول ۳: نتایج اجراهای مختلف در حالتیکه وظایف دارای استقلال می‌باشند و پردازنده‌ها قدرت پردازشی متفاوتی دارند

تعداد ماشین‌های مجازی در هر ماشین فیزیکی	تعداد کل وظایف در حال اجرا	زمان اتمام کل (میلی ثانیه)	متوسط نرخ ارتباطات محلی	متوسط زمان بیکاری هر ماشین (میلی ثانیه)
۲۵	۷۵	۲۳۷۹,۱۷	معنی ندارد	۳۰۱۰۸,۹۰
۵۰	۱۵۰	۴۵۳۳,۲۹	معنی ندارد	۱۴۶۴۵۶,۳۳
۱۰۰	۳۰۰	۵۷۹۹,۲۱	معنی ندارد	۴۲۲۹۶۱,۷۳
۳۰۰	۹۰۰	۷۰۸۵,۳۵	معنی ندارد	۱۶۸۷۸۴۵,۷۴

در جدول ۳، به این دلیل که وظایف، مستقل از یکدیگر هستند، ارتباط خارجی‌ای برای آن‌ها وجود ندارد که بر روی زمان اتمام کل تاثیر گذار باشد اما در این سناریو، متفاوت بودن قدرت پردازشی در ماشین‌های مجازی موجب می‌شود که تخصیص اولیه وظایف به ماشین‌های مجازی، یک تخصیص بهینه نباشد، لذا الگوریتم با جابجا کردن وظایف، بر روی ماشین‌ها، سعی در متوازن کردن بار پردازشی شبکه می‌نماید. شکل ۳، مقدار برازش بدست آمده از اجراهای موجود در جدول ۳ را نشان می‌دهد.

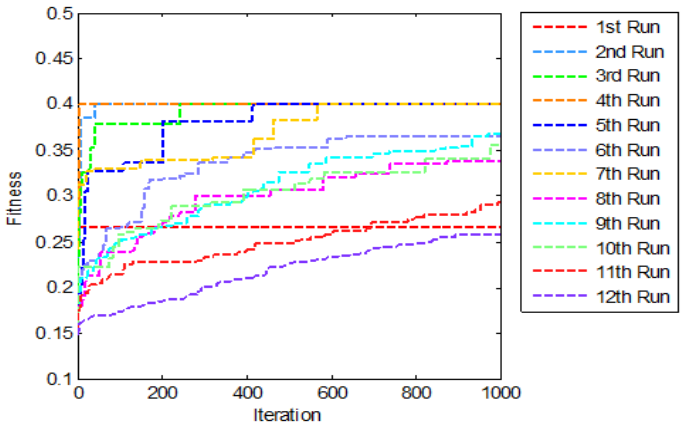


شکل ۳: تابع برازش حاصل از وظایف در حال اجرا در حالتی که وظایف مستقل هستند و ماشین‌های مجازی می‌توانند دارای توان پردازشی متفاوتی باشند

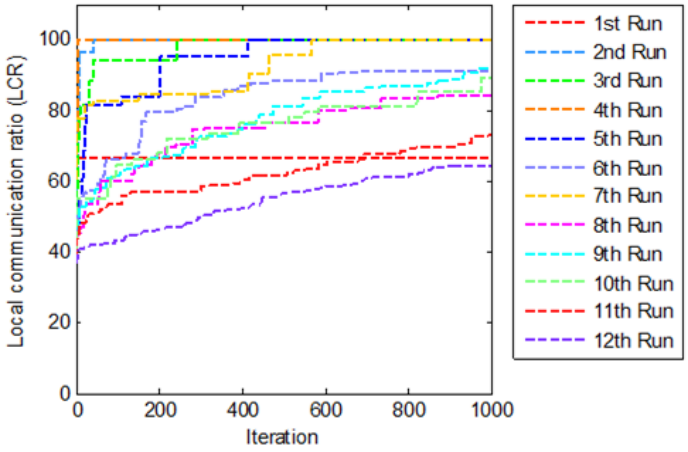
نتایج حاصل از آزمایش‌های مربوط به حالت ب از سناریو دوم در جدول ۴ ارائه شده است. جدول ۴ نشان می‌دهد که با وجود ناهمگن بودن واحدهای پردازشی، عملیات توازن بارِ وظایف به همراه محلی کردن ارتباطات خارجی آن‌ها انجام می‌شود؛ گرچه در این حالت، زمان اتمام کل، به قدرت واحدهای پردازشی و میزان ناهمگنی آن‌ها (به ویژه نسبت به میزان بارهایی که در شبکه توزیع شده) وابسته است. شکل‌های ۴ و ۵، پیشرفت تابع برازش و نرخ ارتباطات محلی موجود در جدول ۴ را ارائه می‌دهند.

جدول ۴: نتایج اجراهای مختلف در حالتی که وظایف با یکدیگر در تعامل هستند و پردازنده‌ها می‌توانند دارای قدرت پردازشی نابرابری باشند

شماره اجرا	تعداد ماشین‌های مجازی در هر ماشین فیزیکی	تعداد کل وظایف در حال اجرا	درصد ارتباطات	درجه ارتباط	نوع درجه ارتباط	زمان اتمام کل (میلی ثانیه)	متوسط نرخ ارتباطات محلی	متوسط زمان بیکاری هر ماشین (میلی ثانیه)
۱	۲۵	۷۵	٪۴	۱	ایستا	۲۴۸۸.۵۵	۶۶.۶۶٪	۳۰۹۰۰.۳۹
۲	۲۵	۷۵	٪۸	۳	پویا	۳۱۹۹.۰۶	۱۰۰٪	۴۹۰۸۲.۷۳
۳	۲۵	۷۵	٪۲۰	۳	پویا	۳۴۳۱.۰۷	۱۰۰٪	۴۹۶۲۷.۵۶
۴	۵۰	۱۵۰	۴٪	۱	ایستا	۳۹۶۳.۴۶	۱۰۰٪	۱۱۸۹۹۹.۹۳
۵	۵۰	۱۵۰	۸٪	۳	پویا	۵۶۰۲.۴۱	۱۰۰٪	۱۹۹۳۱۴.۹۹
۶	۵۰	۱۵۰	٪۲۰	۳	پویا	۸۵۴۷.۶۶	۹۱.۲۴٪	۳۴۶۸۴۶.۹۲
۷	۱۰۰	۳۰۰	٪۴	۱	ایستا	۶۰۱۱.۸	۱۰۰٪	۴۴۵۷۶۳.۷۲
۸	۱۰۰	۳۰۰	٪۸	۳	پویا	۱۳۱۲۶.۸۲	۸۴.۵۶٪	۱۱۶۲۴۴۱.۳۲
۹	۱۰۰	۳۰۰	٪۲۰	۳	پویا	۱۳۹۱۶.۵۹	۹۲.۰۳٪	۱۲۳۷۷۵۷.۳
۱۰	۳۰۰	۹۰۰	٪۴	۱	ایستا	۹۵۲۳.۴۱	۸۹.۰۶٪	۲۴۱۲۲۹۰.۱۶
۱۱	۳۰۰	۹۰۰	٪۸	۲	ایستا	۲۸۷۸۵.۰۴	۷۳.۲۹٪	۸۱۵۶۴۲۳.۵۴
۱۲	۳۰۰	۹۰۰	٪۲۰	۳	ایستا	۹۱۲۸۳.۷۳	۶۴.۳۶٪	۲۶۸۶۱۶۱۷.۲۷



شکل ۴: تابع برازش حاصل‌شده از اجراهای موجود در جدول۴



شکل ۵: نرخ ارتباطات محلی حاصل‌شده از اجراهای موجود در جدول ۴

۵- نتیجه‌گیری

در این پژوهش، الگوریتمی بر مبنای الگوریتم ژنتیک برای برقرای توازن‌بار در میان واحدهای پردازشی توزیع‌شده ارائه شد. این روش، کاهش مقادیر زمان اتمام کل، ارتباطات خارجی و زمان بیکاری شبکه را ملاک عملکرد خود قرار داد و ثابت کرد که با کاهش ارتباطات خارجی می‌توان موجب کاهش زمان اتمام کل در شبکه شد. آنچه که از این پژوهش می‌توان بعنوان کارهای آینده ارائه نمود، تغییر تعداد ماشین‌های مجازی موجود بر روی هر ماشین فیزیکی بگونه‌ای است که تعداد وظایف موجود بر روی هر ماشین‌فیزیکی وابسته به توان آن ماشین باشد. همچنین، هنگامیکه که در طول اجرای سناریو، از زمان آغاز الگوریتم، تنها یک وظیفه بر روی هر ماشین‌مجازی در حال اجرا باشد، زمان بیکاری تحمیل شده به سیستم برابر با مقدار قابل توجهی است لذا در کارهای آتی با بررسی امکان ورود وظایف جدید در هنگام اتمام اجرای پردازش سایر وظایف، سعی در بهبود مقدار این پارامتر خواهیم بخشید.

مراجع

[۱] حاتمیان، آناهیتا؛ بزرگی راد، سیدیاسر و نعمتی، سمیرا، "ارائه یک روش جدید مبتنی بر ترکیب الگوریتم‌های ژنتیک و کلونی مورچه‌ها جهت توازن بار در رایانش ابری"، *دومین کنفرانس ملی توسعه علوم مهندسی*، دوره ۲، ۱۳۹۴.

[۲] سعادتی، فاطمه و میرعابدینی، سیدجواد، "بهبود روند توازن بار در پردازش ابری به کمک متغیرهای وزنی با رویکرد فازی"، *کنفرانس بین المللی سیستمهای غیر خطی و بهینه سازی مهندسی برق و کامپیوتر*، دوره ۱، ۱۳۹۴.

[۳] میر، سجاد و فاضلی، مهدی، "ارائه یک الگوریتم زمانبندی وظایف کارا در محیط‌های رایانش ابری بر مبنای الگوریتم ژنتیک بهینه‌سازی شده"، *اولین کنفرانس بین المللی وب پژوهی*، دوره ۱، ۱۳۹۴.

[4] A.Agarwal, G.Manisha, R. N.Milind, & S. S.Shylaja,“ A Survey of Cloud Based Load Balancing Techniques”, *In proc. Proceedings of Int. Conf. on Electrical, Electronics, Computer Science & Mechanical Engg*, 2014.

[5] L.Bononi, M.Bracuto, G. D'Angelo, & L.Donatiello, “A new adaptive middleware for parallel and distributed simulation of dynamically interacting systems”, *In proc. Distributed Simulation and Real-Time Applications, Eighth IEEE International Symposium on*, pp. 178-187, 2004.

[6] L.Bononi, G.D'Angelo, & L.Donatiello, “ HLA-based adaptive distributed simulation of wireless mobile systems”, *In proc. Proceedings of the seventeenth workshop on Parallel and distributed simulation*, 2003, pp. 40-49.

[7] J.Cao, K.Li, & I.Stojmenovic, “Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers”, *IEEE Transactions on Computers*, Vol. 63, No.1, pp. 45-58, 2014.

[8] N. K.Chien, N. H.Son, & H. D.Loc,"Load balancing algorithm based on estimating finish time of services in cloud computing", *In proc. 2016 18th International Conference on Advanced Communication Technology (ICACT)*, January 2016, pp. 228-233.
[9] D.C.Chou," Cloud computing: A value creation model", *Computer Standards & Interfaces*, Vol.38, pp. 72-77, 2015.
[10] G. D'Angelo, & M.Marzolla, "*New trends in parallel and distributed simulation: From many-cores to Cloud Computing", Simulation Modelling Practice and Theory*, Vol.49, pp. 320-335, 2014.
[11] K.Dasgupta, B.Mandal, P.Dutta, J. K.Mandal, & S. Dam, "A genetic algorithm (ga) based load balancing strategy for cloud computing", *Procedia Technology*, Vol. 10, pp. 340-347, 2013.
[12] I.D. Falco, E.Laskowski, R.Olejnik, U.Scafuri, E. Tarantino, & M.Tudruj, "Extremal Optimization applied to load balancing in execution of distributed programs", *Applied Soft Computing*, Vol. 30, pp. 501-513, 2015.
[13] T.Desai, & J.Prajapati, "A survey of various load balancing techniques and challenges in cloud computing", *International Journal of Scientific & Technology Research*, Vol. 2, No. 11, pp. 158-161, 2013.
[14] F.Ghazipour, S. J. Mirabedini, & A. Harounabadi, " Proposing a new Job Scheduling Algorithm in Grid Environment Using a Combination of Ant Colony Optimization Algorithm (ACO) and Suffrage", *International Journal of Computer Applications Technology and Research*, Vol. 5, No. 1, pp. 20-25, 2016.
[15] J.O. Gutierrez-Garcia, & A. Ramirez-Nafarrate, "Agent-based load balancing in Cloud data centers", *Cluster Computing*, Vol. 18, No. 3, pp. 1041-1062, 2015.
[16] M.Katyal, & A.Mishra, "A comparative study of load balancing algorithms in cloud computing environment", *arXiv preprint arXiv*, pp.1403.6918, 2014.
[17] A.Khiyaita, H.El Bakkali, M.Zbakh, & D.El Kettani, " Load balancing cloud computing: state of art", *In proc. Network Security and Systems (JNS2)*, April 2012, pp. 106-109.
[18] Y.Lu, Q.Xie, G.Kliot, A.Geller, J. R. Larus, & A.Greenberg, "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services", *Performance Evaluation*, Vol. 68, No. 11, pp. 1056-1071, 2011.
[19] T.Mastelic, A.Oleksiak, H.Claussen, I.Brandic, J.M.Pierson, & A. V.Vasilakos, "Cloud computing: Survey on energy efficiency", *ACM Computing Surveys (CSUR)*, Vol. 47, No. 2, pp. 1-36, 2015.
[20] P.Mell, & T. Grance, " The NIST definition of cloud computing", 2011.
[21] R. K.Naha, & M.Othman," Optimized load balancing for efficient resource provisioning in the cloud", *In proc. Telecommunication Technologies (ISTT), 2014 IEEE 2nd International Symposium on*, November 2014, pp. 442-445.
[22] Z.Pooranian, A.Harounabadi, M.Shojafar, & J.Mirabedini," Hybrid pso for independent task scheduling in grid computing to decrease makespan", *In Proc. of International Conference on Future Information Technology, IPCSIT'11*, Vol. 13, 2011, pp. 435-439.
[23] Y.W.Qiu, & J. I. G.Hwang, " A Two-Level Load Balancing Method with Dynamic Strategy for Cloud Computing", *In proc. Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C*, August 2016 , pp. 565-571.
[24] B.Radojević, & M. Žagar, " Analysis of issues with load balancing algorithms in hosted (cloud) environments", *In proc. MIPRO, 2011 Proceedings of the 34th International Convention*, May 2011, pp. 416-420.
[25] B. L.Sahu, & R.Tiwari, " A comprehensive study on Cloud computing", *International journal of Advanced Research in Computer science and Software engineering*, Vol. 2, No.9, pp. 33-37, 2012.

Load Balancing in Cloud Computing Environment by Considering the Dependency among Tasks and Using Adaptive Genetic Algorithm

Yalda Derakhshanian[1], Seyed Javad Mirabedini[2]*, Ali Harounabadi[2]

[1]Department of Computer Engineering-Software, Bushehr Branch, Islamic Azad University, Bushehr, Iran
[2*]Department of Computer, Central Tehran Branch, Islamic Azad University, Tehran, Iran
[3]Department of Computer, Central Tehran Branch, Islamic Azad University, Tehran, Iran
1: y.derakhshanian@gmail.com
2*: j_mirabedini@iauctb.ac.ir
3: a.harounabadi@iauctb.ac.ir

ABSTRACT:

The increasing computing needs leads to an increase in the importance of using cloud computing day by day. Cloud computing is a computing model based on computer networks that presents a new pattern to supply resources, so that users request or release resources based on their needs. When the requests for computing resources increase, proper distribution of resources becomes very important, because if a computational unit has a large number of tasks and the other one is almost idle, resources are not used well and also makespan would be very high. Therefore, in order to overcome this problem, load balancing technique is used. In general, from the computing point of view, the process of distributing the loads on the processing units in a balanced way is called load balancing. In most researches, interactions between running tasks are not considered, so if the tasks in interaction with each other are located in separate processing units in a distributed network, the interactions between them would be effective on makespan. The aim of this research is to present an approach which can achieve a desirable load balancing in the network, such that the makespan and idle time of machines minimized, taking into account the interactions between the tasks. For this purpose, the genetic algorithm is used. The obtained experimental results show that localizing the interactions will have a significant impact in reducing the makespan.

KEYWORDS: Load balancing, Cloud computing, Tasks dependency, Adaptive Genetic