زمستان ۱۴۰۰، دوره ۶، شماره ۴، پیاپی ۲۳
DOR :20.1001.1.2538161.1400.6.4.3.7

دریافت مقاله: ۱۴۰۰/۰۳/۲۵
پذیرش مقاله: ۱۴۰۰/۰۸/۰۵

نوع مقاله: پژوهشی

رویکرد بهینه‌سازی چند هدفه برای مسئله جایابی سرویس‌های نرم‌افزاری در سیستم‌های مبتنی بر رایانش ابری

مصطفی قبائی‌آرانی*
استادیار،گروه مهندسی کامپیوتر، واحد قم، دانشگاه آزاد اسلامی، قم، ایران
پست الکترونیکی: m.ghobaei@qom-iau.ac.ir

مریم رضائی
کارشناسی ارشد، مهندسی کامپیوتر، واحد محلات، دانشگاه آزاد اسلامی، مرکزی، ایران
پست الکترونیکی: rezaei.maryam0@gmail.com

چکیده

در دهه اخیر رایانش ابری مورد توجه بسیاری از ارائه‌دهندگان و استفاده‌کنندگان فناوری اطلاعات قرار گرفته است. یکی از مدل‌های پرکاربرد ارائه خدمات در حوزه رایانش ابری، مدل نرم‌افزار به‌عنوان خدمت[1] بوده که معمولا به‌صورت ترکیبی از مؤلفه‌های داده و برنامه ارائه می‌شوند. یکی از چالش‌های مهم در این حوزه، یافتن مکان بهینه برای مولفه‌های نرم‌افزاری بر روی زیرساخت‌های ابری است که در آن نرم‌افزار به‌عنوان خدمت بتواند بهترین عملکرد ممکن را داشته باشد. مسئله جایابی نرم‌افزار به‌عنوان خدمت به چالش تعیین این‌که کدام سرویس‌دهنده‌ها در مرکز داده ابر، بدون نقض محدودیت‌های نرم‌افزار به‌عنوان خدمت، می‌توانند میزبان کدام مولفه‌ها باشند اشاره دارد. در این مقاله، راهکار بهینه‌سازی چند هدفه با هدف کاهش هزینه و زمان اجرا جهت جایابی مؤلفه‌های در محیط‌های ابری را ارائه می‌دهیم. راهکار پیشنهادی خود را با استفاده از کتابخانه Cloudsim شبیه‌سازی کرده و در نهایت با دو الگوریتم ازدحام ذرات چندهدفه[2] و فاخته[3] مورد ارزیابی و مقایسه قرار دادیم. نتایج شبیه‌سازی نشان می‌دهد که راهکار پیشنهادی عملکرد بهتری نسبت به دو الگوریتم پایه داشته و موجب کاهش ۹/٤درصدی زمان اجرای جایابی مؤلفه‌های نرم‌افزار به‌عنوان خدمت،کاهش ۷/۱درصدی هزینه و افزایش ۱۵درصدی بهره‌وری می‌گردد.

واژه‌های کلیدی: رایانش ابری، نرم‌افزار به‌عنوان خدمت، جایابی مولفه‌های نرم‌افزاری

۱.مقدمه

رایانش ابری، فناوری جدیدی است که روزبه‌روز بر محبوبیت آن افزوده می‌شود، در آن منابع ابری و انواع مختلف خدمات ابری بر اساس تقاضا و برای دسترسی گسترده به شبکه کاربران آن، موجود می‌باشند [۱]. ویژگی‌های دیگر ابر شامل انعطاف‌پذیری خدمات ارائه داده شده، که طبق آن می‌توانند متناسب با تقاضای کاربران باشند، و مدل پرداخت به ازای استفاده می‌باشد،

* نویسندهٔ مسئول

1- Software- as -a- Service (SaaS)

2- MOPSO

3- CSA

کــه طبق آن به‌جــای پرداخت یک مقدار هزینه مشــخص، کاربــران تنها هزینه چیزی را که از ابر اســتفاده نموده‌اند را پرداخت می‌نمایند. خدمات رایانش ابر به سه دسته کلی طبقه‌بندی شــده است۱-زیرســاخت به‌عنوان خدمت[۴] که به‌عنوان راه‌حلی برای مشکل ذخیره‌سازی کاربران رایانش ارائه شــده است[۲]. ۲- بُن‌سازه به‌عنوان خدمت[۵] که هدف قرار دادن برنامه‌های کاربردی که برای طراحی، توســعه، گســترش و آزمایش فعالیت‌ها در نرم‌افزار سیستم عامل ابراســت[۳]. نرم‌افزار به‌عنوان خدمت کــه جایگزین برای نصب نرم‌افزار است[٤]. ما در این تحقیق بر روی مشکلات حوزه نرم‌افزار به‌عنوان خدمت تمرکز می‌کنیم، بســیاری از محققان به این امر اشــاره دارند کــه نرم‌افزار به‌عنوان خدمت قبل از رایانش ابری وجود داشــته است [۵،٦،۷]، و تقاضای روزافزون برای اســتفاده از آن منجر شده است تا فروشــندگان نرم‌افزار به‌عنوان خدمت در محیط ابر آن را گســترش دهند، زیرا ابر مقیاس‌پذیری مورد نیاز برای نرم‌افــزار به‌عنوان خدمــت را ارائه می‌دهــد[۸]. با وجود احتمال ارائه بدون اســتفاده از رایانش ابــری، این نوع از خدمات ابر، در مقایسه با خدمات دیگر ابر همچون بُن‌سازه به‌عنوان خدمت و زیرساخت به‌عنوان خدمت، در چندسال اخیر محبوبیت گســترده‌ای را به‌دست آورده است [۹،۱۰]. بررســی آمار نشان می‌دهد که درآمد اشــتراک نرم‌افزار به‌عنوان خدمت از تقریبا ٤۲ میلیارد دلار در سال ۲۰۱۲ به ۱۰۷ میلیارد دلار در سال ۲۰۱٦ افزایش داشته است [۱۰].

جهت بهبود بیشــتر انعطاف‌پذیــری نرم‌افزاربه‌عنوان خدمت، ایــن نوع از خدمــات را می‌توان به شــکل مرکب ارائــه داد. نرم‌افزار به‌عنــوان خدمت مرکــب، ترکیبی از زیرسیســتم‌های مرتبط جفت می‌باشد که هرکدام از آن‌ها همراه با تابع تجاری مشــخص، جهت تشــکیل سیســتم عملیاتی ســطح بــالا، می‌باشــند [۱۱]. آن مولفه‌ها عموما مولفه‌های برنامه می‌باشــند که ممکن است به یک یا تعداد زیادی از بســته‌های داده نیاز به دسترسی داشته باشند، یا مولفه‌های داده هســتند که همچنین به‌عنوان بخشــی از نرم‌افزار به‌عنوان خدمت مرکب درنظر گرفته می‌شوند. این شــکل از نرم‌افزار به‌عنوان خدمت ارائه داده شده در مدل مرکب باعث انعطاف‌پذیری بیشتر نرم‌افزار به‌عنوان خدمت می‌گردد تا مناســب با تقاضای کاربران باشد، که این کار را از طریق ترکیب، جداسازی، و ترکیب دوباره مولفه‌ها در هنگام نیاز انجام می‌دهد. همچنین از طریق توانایی استفاده از مولفــه نرم‌افزار به‌عنوان خدمــت در نرم‌افزار به‌عنوان خدمــت چندگانــه دیگر، هنگامــی که به عملکــرد تجاری بخصوص آن نیاز است، و بتوان آن را با مجموعه‌ای دیگر از مولفه‌ها «گردآوری» نمود، قابلیت استفاده مجدد را ممکن می‌سازد. هردو این مزیت‌ها می‌توانند منجر به صرفه‌جویی در هزینه و زمان برای ارائه‌دهندگان ابر شــوند. [۱۱،۷] با این حال، مــدل مرکب جهت ارائه نرم‌افزار به‌عنوان خدمت ممکن اســت باعث بــه وجود آمدن چندیــن چالش از نظر مدیریت منابع به ارائه‌دهنده ابر گردد. به‌عنوان مثال، یافتن راه‌حــل جایابی برای نرم‌افزار به‌عنوان خدمت مرکب که با کمینه‌ســازی زمان اجرا و همچنین کاهش هزینه می‌تواند عملکرد آن را بهینه سازد.

مسئله عنوان شده به‌عنوان مســئله جایابی[٦] نرم‌افزار به‌عنوان خدمت شــناخته شــده و به‌عنوان مسئله مدیریت منابع رایانش ابری درنظر گرفته می‌شــود. مدیریت منابع همیشــه برای رایانش ابری چالش بوده اســت که این امر بخصوص در سال ۲۰۱٦ خود را بیشتر نشان داد، هنگامی کــه کمبود منابــع و متخصصان به‌جای امنیــت تبدیل به نگرانی اصلی برای ارائه‌دهندگان ابر شد [۹]، زیرا بارکاری بیشتری به محیط ابری انتقال داده شد. بسیاری از کارهای پژوهشی به چالش‌های مختلف مدیریت منابع در محاسبه ابر همراه با هوش محاســباتی[۷] و روش‌های بهینه‌ســازی مختلف پرداخته‌اند [۱۲]. ما در این تحقیق با توجه افزایش محبوبیت و اســتفاده از نرم‌افزار به‌عنوان خدمت توســط کاربران، مســئله مدیریت منابع ابر را مسئله اصلی تحقیق

4- Infrastructure-as-a-Service
5- Platform -as- a-service
6- SaaS placement problem
7- Computational intelligence

خود قرار می‌دهیم و با اســتفاده از الگوریتم های متفاوت به دنبال راه‌حل مناسبی برای حل مسئله جایابی مؤلفه‌های نرم‌افزار به‌عنوان خدمت می‌باشیم. با انجام شبیه‌سازی و مقایسه راهکار پیشنهادی با الگوریتم‌های دیگر که پیش‌تر استفاده شــده بودند، این نتیجه حاصل می‌شود که HYPE می‌تواند عــلاوه بر بــرآورده نمودن تقاضــای کاربران، استفاده از منابع ابر را بهینه‌سازی کند. این الگوریتم برای جایابی این‌که کدام سرویس‌دهنده برای میزبانی کدام مؤلفه انتخاب شــود، به‌گونه‌ای که تقاضای کاربران را برآورده کند و استفاده از منابع ابر را از طریق کاهش زمان اجرا و کاهش هزینه بهینه‌سازی کند و به یک جایابی مؤثر و کارا دســت یابد، راهکار مناسبی می‌باشد. دستاوردهای اصلی این پژوهش به شرح زیر می‌باشد:

- ارائــه الگوریتمی برای جایابی بهینه مؤلفه‌های نرم‌افزار به‌عنوان خدمت مرکب، در محیط ابری
- ارزیابــی کارایی الگوریتم پیشــنهادی بــا دو الگوریتم ازدحام ذرات چند هدفه و فاخته و در دو ســناریو متفاوت از نظر بهینه‌بودن در زمان اجرا و هزینه اجرایی

ادامه این مقاله به‌صورت زیر ســازماندهی شده است. در بخش دوم به معرفی کارهای مربوطه که با الگوریتم‌های متفــاوت برای جایابــی بهینه انجام شــده می‌پردازیم در بخش ســوم راهکار پیشنهادی به وســیلهٔ الگوریتم HYPE برای جایابی با اهداف بهینه‌ســازی در زمان و هزینه ارائه می‌گردد. در بخش چهارم نتایج عملکرد جایابی با الگوریتم پیشــنهادی در قالــب نتایج شبیه‌ســازی مــورد ارزیابی قرارگرفته و با دو روش دیگر مقایسـه می‌شود. درنهایت، در بخش آخر نتیجه‌گیری و پیشنهادها مطرح می‌گردد.

۲.کارهای مرتبط

اخیرا مسئله جایابی سرویس در چندین کار پژوهشی مورد بررسی قرار گرفته شده است که از بین آن‌ها چندین پژوهشگر رویکردهای هوشمند محاسباتی مختلف را جهت بهینه‌سازی عملکرد نرم‌افزار به‌عنوان خدمت ارائه داده‌اند. محققان بهینه‌سازی را در دو شاخه جدا مورد بررسی قرار دادند. برخی محققان هدف بهینه‌سازی را کاهش هزینه‌های قرار دادند که خواســتن بــا کاهش هرینه‌های گســترش نرم‌افزار به‌عنوان خدمت در سرویس‌دهنده و افزایش سود ارائه‌دهندگان ابــر به آن بپردازنــد، و برخی دیگر کاهش زمان اجرای کل را هدف بهینه‌ســازی قرار دادند. کارها و آثار زیادی در زمینه جایابــی نرم‌افزار به‌عنوان خدمت با اســتفاده از الگوریتم‌های مختلف ارائه شده است که ما در اینجــا به معرفی هریک از آن‌ها پرداخته و آن‌ها را شــرح می‌دهیم.

۲-۱. ارائه رویکــرد جایابی نرم‌افزار به‌عنوان خدمت با هدف بهینه‌ســازی در هزینه با حداقل رســاندن هزینه‌ها و افزایش سود

چین بی و ساســی [۸][۱۳] رویکردی مبتنی بر الگوریتم چند هدفه بهینه‌سازی ازدحام ذرات [۹] برای جایابی پیشنهاد دادند. نویسندگان براســاس کارهای پیشین به این نتیجه رســیدند کــه بهینه‌ســازی ازدحــام ذرات در محیط‌های ایســتا خوب عمــل می‌کند و برای ایجاد نتایــج قابل قبول در محیط‌های پویا مانند ابر باید ســازگاری پیدا کند و این ناســازگاری به‌خاطر حافظه‌های منسوخ و فقدان تنوع در الگوریتم بهینه‌ســازی ازدحام ذرات[۱۰] بود چون محیط ابر پویا است و درخواســت‌ها و حجم منابع مورد نیاز متغیر می‌بود آن‌ها دریافتند که رویکــرد چندگانه در محیط‌های پویا بســیار ســودمندتر خواهد بود و در فضای جستجو در چنــد منطقه نتایج امیدوارکننده‌تــری ارائه می‌دهد و بر همیــن مبنا از الگوریتــم الگوریتم چند هدفه بهینه‌ســازی ازدحام ذرات اســتفاده کردند که البته ایــن رویکرد را از لحــاظ زمان اجرا با الگوریتم ژنتیــک[۱۱] واز لحاظ هزینه با الگوریتم بهینه‌سازی ازدحام ذرات مقایسه کردند که به این نتیجه رسیدند که از هر دو کارامدتر هست ولی مشکل این روش این بود که با افزایش تعداد ســرویس‌دهنده‌های ابر

8- Chainbi, W. and Sassi,
9- Multi Particle swarm optimization
10- Particle swarm optimization
11- Genetic algorithm

هم هزینه و هم زمان اجرا افزایش پیدا می‌کرد.

آلتمن وکاشف[12][۱۴] یک الگوریتم بهینه‌سازی هزینه برای تصمیم‌گیری در مورد جایابی خدمات در ابرها پیشنهاد دادند. آن‌ها مدل هزینه‌ای جامع که فاکتورهای هزینه و انواع ابرها را پوشش می‌داد، برای تصمیم‌گیری هزینه‌های خدمات با تخمین محاسبات هزینه بر اساس (فرمول هزینه ثابت،فرمول هزینه متغیر، فرمول هزینه کل) ارائه دادند که هزینه کلی همهٔ گزینه‌های جایابی خدمات احتمالی را مقایسه و گزینه جایابی خدمات با حداقل هزینه را شناسایی می‌کند که شامل هزینه انتقال اطلاعات بین ابرهای عمومی و هزینه استقرار خدمات از طریق مهاجرت[13]ماشین مجازی می‌شود. هزینه استقرار می‌تواند هزینه انتقال داده‌ها را جبران کند پس می‌توانیم بیان کنیم که افزایش تعداد مهاجرت خدمات می‌تواند تاثیر منفی در سود کل داشته باشد در این مورد به حداقل رساندن هزینه‌ها اجرا نخواهد شد.

شن و هانگ[14][۱۵]، مسئله جایابی را با هدف دوگانه (۱) کاهش هزینه‌های ارتباط بین کارها و (۲) افزایش همبستگی درنظر گرفتند. با ادغام دو نمودار وابستگی [15]و همزمانی خدمات[16] و تبدیل به مسئله k بخشی و حل آن با استفاده از الگوریتم حریصانه. مشکل این روش این بود که برای کاهش زمان اجرا به کارساز قوی نیاز بود که افزایش هزینه را به‌دنبال داشت و تفاوت بین جایابی مؤلفه داده[17]و مؤلفه برنامه [18]را نیز در نظر نگرفت.

بهارواچ[19] [۱۶]رویکرد مبتنی بر بهینه‌سازی ذرات را پیشنهاد نمود. الگوریتم به گونه‌ای شبیه به الگورتم ژنتیک با یک جمعیت تصادفی آغاز شده و جهت تولید جمعیت بهتر تکامل می‌یابد اما در الگوریتم بهینه‌سازی ذرات هیچ اپراتور ژنتیکی اعمال نمی‌شود، زیرا این «ذرات»اند که جهت یافتن راه‌حل‌های بهینه در فضای جستجو پخش شده و با سرعت مشخص حرکت می‌کنند و پس از بروزرسانی سرعت و موقعیت‌شان، ذراتی که در جای جدید قرار گرفته‌اند، جمعیت جدیدی از راه‌حل‌های جایابی را ارائه می‌دهند و براساس مقادیر سازگاری جدیدشان، در صورت تغییر بهترین راه‌حل‌های محلی و کلی نیز به‌روزرسانی می‌شوند. الگوریتم بهینه‌سازی ذرات در مقایسه با الگوریتم ژنتیک کم هزینه‌تر و مقیاس‌پذیری بهترداشت اما راجع به زمان پردازش الگوریتم هیچ بررسی تجربی وجود ندارد و تنها از کارسازها و مولفه‌های همگن استفاده نموده است.

کوما[20] [۱۷] از الگوریتم ژنتیک مبتنی بر اصلاح جهت اصلاح راه‌حل‌ها و دستیابی به جمعیت دیگر و مناسب‌تراستفاده نمود. هر راه‌حل تولیدی که محدودیت‌های منابع را نقض کند از طریق تولید دوباره تصادفی بخشی از راه‌حل در فضای جستجو صحیح «اصلاح» می‌شد. در مقایسه این روش با (FFD)[21] سود بیشتری به همراه داشت اما زمان محاسبه طولانی‌تر داشت و تفاوت بین جایابی مؤلفه داده و برنامه را درنظر نگرفته بود

۲-۲. ارائه رویکرد جایابی نرم‌افزار به‌عنوان خدمت با هدف بهینه‌سازی در زمان اجرا

یوسح وتانگ [22] اولین کسانی بودند که مجموعه‌ای از پژوهش‌های جایابی را با مولفه‌های ناهمگن انجام دادند[۱۸،۱۹،۲۰]. در اولین پژوهش الگوریتم ژنتیک مبتنی برخطا با شروع تصادفی و تخمین را ارائه دادند که اگر راه‌حل تولیدی کارآیی نداشته وهمچنین محدودیت‌های منابع را نیز نقض کند، الگوریتم آن‌ها به‌جای رهاسازی کامل، آن را «جریمه» کند. عیب این روش زمان اجرای[23] طولانی بود. اولین پژوهش خود را با استفاده از رویکرد جدید مبتنی بر الگوریتم همیاری‌تکاملی[24]بهبود بخشیدند. بررسی‌های مجدد نشان می‌دهند که الگوریتم هم تکاملی تعاونی موازی از نظر کیفیت راه‌حل‌های تولیدی و زمان محاسبه نسبت به الگوریتم ژنتیک و الگوریتم هم تکاملی

12- Altmann, J. and Kashef,
13- Virtual machine
14- Huang and Shen
15- Service Dependencies Graph
16- Service Concurrence Graph
17-Data component
18- Application component
19- Bhardwaj
20- Kumar
21- First-Fit Decreasing
22- Yusoh and Tang
23-Total Execution Time
24- Cooperative Evolutionary Algorithm

تعاونی تکراری برتری دارد. اما مانند دو پژوهش پیشین الگوریتم همیاری تکاملی موازی تنها محدودیت های منابع را درنظر گرفته است.

نی و همکاران[۲۵] [۲۱] الگوریتم کلونی مورچه‌ها[۲۶] را برای جایابی سرویس پیشنهاد دادند. این الگوریتم از حرکت مورچه‌های مصنوعی و فرمون به جا مانده در محیط جایابی انجام می‌شد. این روش از الگوریتم ژنتیک برتر بود اما با افزایش تعداد مولفه‌های نرم‌افزار به‌عنوان خدمت به خوبی عمل نکرد و تفاوت‌های بین جایابی مؤلفه‌های برنامه‌ها یا داده هارا درنظر نگرفتند.

باون و شائوچان[۲۷] [۲۲] از الگوریتم پیوندی انطباقی، که الگوریتم ژنتیک را با گرم کردن شبیه‌سازی شده ترکیب می‌نماید استفاده کردند. نرخ جهش و همگذری به گونه‌ای تنظیم شده‌اند تا بسته به نزدیکی مقادیر سازگاری راه‌حل‌های پیشنهادی تغییر کنند. طبق بررسی معرفی الگوریتم دوم باعث تسریع در بهینه‌سازی شده است.

لیو و همکارانش[۲۸] [۲۳] دومین کسانی بودند که از رویکرد کلونی مورچه‌ها استفاده نمودند، که در حالی‌که زمان اجرای کلی آن‌ها را به حداقل می‌رساند، مطلوبیت جایابی مولفه نرم‌افزار به‌عنوان خدمت در یک ماشین بخصوص تحت تاثیر معادله درجه تطابق عملکرد که خلاصه شدهٔ درجه نزدیکی بین نیازمندی منبع مولفه و عملکرد ماشین‌مجازی می‌باشد. این معادله با به‌کارگیری مولفه فعلی در ماشین کمک می‌نماید که با قوی‌ترین عملکرد همراه است. عملکرد کلونی مورچه‌ها نسبت به ژنتیک کلاسیک بهتر است، محققان تنها نیازمندی‌های منابع و مولفه‌های نرم‌افزار به‌عنوان خدمت همگن ترمیم شده را درنظر گرفته‌اند.

مزنی و همکاران[۲۹] [۲٤] با تغییر در الگوریتم بهینه‌سازی ازدحام ذرات، الگوریتم بهینه‌سازی ازدحام ذرات با ذرات مرکب[۳۰] را ارائه دادند، که در آن هر ذره مرکب در ازدحام نشان‌دهنده طرحی برای جایابی بود، ذرات مرکب جهت بررسی و ارزیابی هر فضا رفتار جمعی را اتخاذ می‌کنند (مانند مراکز داده) و از طریق همکاری با ذرات مرکب دیگر یا ذرات مستقل (مانند سرویس‌دهنده‌ها) ساختار خود را تنظیم می‌کنند. که نتایج آزمایش بر روی پیشنهاد آن‌ها نشان داد که نه‌تنها قابل اجرا در جایابی سرویس می‌باشد بلکه توانست راه‌حل‌های قابل قبولی را درفضای جستجو بزرگ را ارائه دهد و نسبت به الگوریتم ژنتیک برتری داشت.

موساو همکاران[۳۱] [۲۵] یک روش توزیع شده برای نرم‌افزار به‌عنوان خدمت ترکیبی با استفاده از الگوریتم رئیس/ مرئوس ارائه دادند، دراین روش مرئوس‌ها به‌صورت محلی نظارت و سازگاری مؤلفه‌های توزیع شده نرم‌افزار به‌عنوان خدمت را به عهده داشتند و رئیس، مسئول نظارت و بازسازی کار به منظور تحویل سرویس کیفیت خدمات[۳۲] درخواستی می‌باشد. آن‌ها از حلقه MAPE را برای مدیریت پویا در زمان اجرا استفاده کردند و فرایند سازگاری نرم‌افزار به‌عنوان خدمت را به‌عنوان یک مسئله بهینه‌سازی چند هدفی در نظر گرفتند تا گره‌های بهینه را انتخاب کنیم که عملکرد را به حداکثر و هزینه را به حداقل برسانند وسربار را کاهش می‌دهد. و از الگوریتم فیلتر کالمن برای پیش بینی کیفیت خدمات استفاده شد. در این روش هزینه و زمان اجرا کاهش و میزان دسترسی افزایش پیدا کرد ولی با افزایش مؤلفه‌ها زمان افزایش یافت.

۳. راهکار پیشنهادی

در این مقاله راهکاری برای جایابی اجزای نرم‌افزار به‌عنوان خدمت بر روی ابر با استفاده از الگوریتم HYPE ارائه می‌شود. الگوریتم پیشنهادی شامل شش مرحله می‌باشد. در راهکار پیشنهادی ما توابع هدف را زمان و هزینه بیان کردیم و هدف ما بهینه‌سازی در زمان و هزینه می‌باشد. تمامی

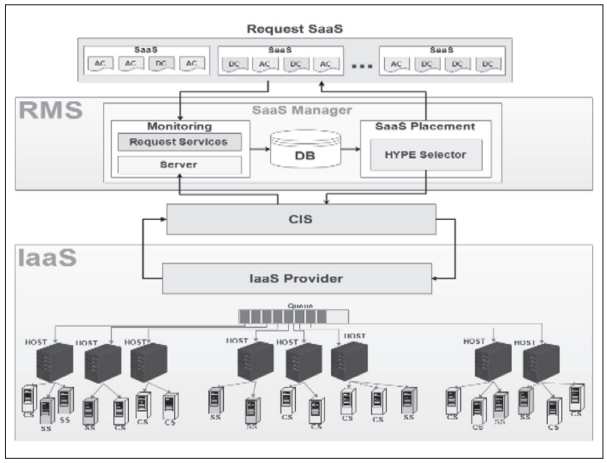
25- Ni and et all
26- Ant Colony Optimization
27- Bowen and Shaochun
28- Liu and et all
29- Mezni and et all
30- 4Particle Swarm Optimization with Composite Partic
31- Mousa and et all
32- Quality Of Service

بخش‌ها عملکرد خاصی دارند. در ادامه به‌صورت مشروح در مورد هر کدام از بخش‌ها توضیحاتی ارائه می‌گردد.

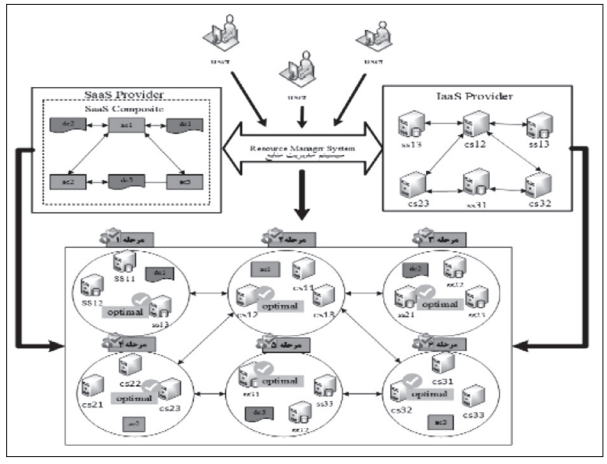
۳-۱.چارچوب پیشنهادی

ساختار معماری راهکار پیشنهادی براساس الگوریتم HYPE دارای چند مرحله مدیریت بر روی منابع ابری می‌باشد. این مدیریت باید به‌گونه‌ای باشد که با رعایت توافقنامه سطح خدمات[۳۳] بتوانیم به حداقل زمان اجرا در جایابی مؤلفه‌های نرم‌افزار به‌عنوان خدمت بر روی محیط ابر دست یابیم. محیط ابری که برای اجرای نرم‌افزار به‌عنوان خدمت مورد نیاز است شامل چند میزبان مختلف می‌باشد که این میزبان‌ها شامل سرویس‌دهنده‌های ذخیره[۳۴]، سرویس‌دهنده‌های محاسباتی[۳۵] هستند که هر کدام قابلیت منابع خاص خود را دارند و از طریق سوئیچ‌ها و مسیریاب‌ها به‌هم متصل می‌شوند. در چارچوب پیشنهادی سرویس‌های درخواست‌شده دارای دو مولفهٔ برنامه کاربردی[۳۶] و مولفهٔ داده[۳۷] هستند. اطلاعات مربوط به درخواست‌ها در اختیار واحد مهم سیستم مدیریت منابع[۳۸] و بخش نظارت[۳۹] قرار می‌گیرد. در ساختار پیشنهادی ارائه‌دهندگان زیرساخت[۴۰] وظیفه مدیریت منابع زیرساخت را به عهده دارد. اطلاعات مربوط به منابع ابر[۴۱] دریافت شده و در اختیار لایه سیستم مدیریت منابع و واحد نظارت قرار می‌گیرد. واحد سیستم مدیریت منابع شامل سه بخش مهم است. واحد نظارت که اطلاعات نرم‌افزار به‌عنوان خدمت درخواستی و منابع سیستم را دریافت می‌کند. واحد دوم پایگاه دانش است که اطلاعات مربوط به درخواست و منابع را در خود نگهداری می‌کند. واحد سوم و مهم‌ترین واحد، واحد جایابی نرم‌افزار به‌عنوان خدمت است. این واحد به کمک اطلاعات دریافت شده از واحد نظارت و به واسطه الگوریتم چندهدفه HYPE تصمیم‌گیری می‌کند به چه صورت جایابی نرم‌افزار به‌عنوان خدمت انجام شود. به‌طور خلاصه برای مدیریت منابع موجود بر روی سرویس‌دهنده‌ها احتیاج به یک مدیریت است تا بتواند اهداف نرم‌افزار به‌عنوان خدمت را ارضا و همچنین توافقنامه سطح خدمات را رعایت کند. در ضمن بتواند با حداقل زمان اجرا با مدیریت خود به یک جایابی مؤثر و کارای مؤلفه‌های نرم‌افزار به‌عنوان خدمت را بر روی سرویس‌دهنده‌ها دست یابد. این مراحل سیستم مدیریت منابع را برای یافتن بهترین جایابی ممکن را با ذکر مثالی شرح می‌دهیم، زمانی که کاربری درخواستی را به ابر می‌فرستد تمام مراحل بالا انجام می‌شود با فرض این‌که اجزای نرم‌افزار به‌عنوان خدمت[۴۲] شامل شش مؤلفه داده و برنامه که آن‌ها شامل مؤلفه‌های برنامه (ac1,ac2,ac3) و مؤلفه‌های داده (dc1,dc2,dc3) باشد برای انجام جایابی بهینه مراحلی انجام می‌شود مثلا برای مؤلفه dc1 با بررسی ظرفیت منابع و ظرفیت منابع مورد نیازش سه سرویس‌دهنده ذخیره (ss11,ss12,ss13) به‌عنوان نامزد در نظر گرفته می‌شود با محاسبه تابع برازش الگوریتم پیشنهادی خود به این به این نتیجه می‌رسیم که سرویس‌دهنده ذخیره ss13 از لحاظ زمان اجرا بهینه است پس این سرویس‌دهنده ذخیره انتخاب می‌شود و مؤلفه داده dc1 بر روی ماشین مجازی روی این سرویس‌دهنده ذخیره مستقر می‌شود. (۱) در جایابی بعدی مؤلفه‌های برنامه برای جایابی مؤلفه ac1 با بررسی ظرفیت منابع و ظرفیت منابع مورد نیاز سه سرویس‌دهنده محاسباتی (cs11,cs12,cs13) به‌عنوان نامزد در نظر گرفته شده و از بین این سه سرویس سرویس‌دهنده محاسباتی با محاسبه تابع برازش از لحاظ زمان اجرا سرویس‌دهنده cs12 بهینه است پس ac1 بر روی ماشین مجازی روی این سرویس‌دهنده محاسباتی مستقر می‌شود(۲) در جایابی بعدی برای مؤلفه dc2 با بررسی ظرفیت منابع و ظرفیت منابع مورد نیازش سه سرویس‌دهنده ذخیره(ss21,ss22,ss23) به‌عنوان نامزد در نظر گرفته می‌شود که بعد از محاسبه تابع برازش، سرویس‌دهنده ss21 ازبین آن‌ها بهینه است

33- service-level degradation (SLA)
34- Storage Server
35- Compute Server
36- Application Component
37- Data Component
38- Resource Management System
39- Monitoring
40- IaaS Provider
41- Cloud Information Service (CIS)
42- SaaS composite



شکل ۱ :تفکیک مؤلفه‌های RMS



شکل ۲ :چارچوب پیشنهادی

پس dc2 بر روی ماشین مجازی روی این سرویس‌دهنده ذخیره مستقر می‌شود (۳)، برای جایابی مؤلفه برنامه ac2با بررسی ظرفیت منابع و ظرفیت منابع مورد نیازش سه سرویس‌دهنده محاسباتی (cs21,cs22,cs32) به‌عنوان نامزد در نظر گرفته شده و با محاسبه تابع برازش سرویس‌دهنده cs23 بهینه است پس ac2 بر روی ماشین مجازی روی این سرویس‌دهنده محاسباتی مستقر می‌شود (٤) در جایابی مؤلفه dc3 با بررسی ظرفیت منابع و ظرفیت منابع مورد نیازش در بین سه سرویس‌دهنده ذخیره (ss31,ss32,ss33) سرویس‌دهنده ss31 بهینه است پس dc3 بر روی ماشین مجازی روی این سرویس‌دهنده ذخیره انتخاب و مستقر می‌شود. (٥) در آخر برای مؤلفه ac3 با بررسی ظرفیت منابع و ظرفیت منابع مورد نیاز سه سرویس‌دهنده محاسباتی (cs31,cs32,cs33) در نظر گرفته

جدول ۱. متغییرها وتعاریف آن ها

نام	تعریف
DC	مرکز داده ابر
SS	سرویس‌دهنده ذخیره
CS	سرویس‌دهنده محاسباتی
E	لبهٔ ارتباطی بین سرویس‌دهنده‌ها
N	تعداد سرویس‌دهنده محاسباتی
VM_S	ماشین مجازی مستقر روی سرویس‌دهنده
pc_i	ظرفیت پردازش سرویس‌دهنده محاسباتی
mem_i	ظرفیت حافظه سرویس‌دهنده محاسباتی
$disk_i$	ظرفیت ذخیره سرویس‌دهنده محاسباتی
VMC_i	مجموعه ماشین‌های مجازی مستقر روی سرویس‌دهنده محاسباتی
pc_{ij}	ظرفیت پردازش ماشین مجازی *j*ام بر روی سرویس‌دهنده *i*ام
mem_{ij}	ظرفیت حافظه ماشین مجازی *j*ام بر روی سرویس‌دهنده *i*ام
$disk_{ij}$	ظرفیت ذخیره ماشین مجازی *j*ام بر روی سرویس‌دهنده *i*ام
M	تعداد سرویس‌دهنده ذخیره
$B_{vi,vj}$	پهنای باند بین رئوس
$L_{vi,vj}$	تاخیر بین رئوس
S	مجموعه اجزای نرم‌افزار به‌عنوان خدمت
AC	مجموعه مؤلفه‌های برنامه نرم‌افزار به‌عنوان خدمت
DC	مجموعه مؤلفه‌های داده نرم‌افزار به‌عنوان خدمت
$PCReq_i$	نیازمندی ظرفیت پردازشی مؤلفه برنامه
$memReq_i$	نیازمندی ظرفیت حافظه مؤلفه برنامه
$size_i$	نیازمندی ظرفیت ذخیره مؤلفه برنامه
AmountRW	میزان خواندن نوشتن اجزا بایکدیگر
DAG_s	گراف گردش کار
DAC	ارتباط مؤلفه برنامه با برنامه
DDC	ارتباط مؤلفه برنامه با داده
TET	زمان اجرای کل
PT	زمان پردازش
DTT	زمان انتقال داده از سرویس‌دهنده ذخیره به سرویس‌دهنده محاسباتی

شده و با محاسبه تابع برازش بین آن‌ها سرویس‌دهنده cs32 بهینه است پس ac2 بر روی ماشین مجازی روی این سرویس‌دهنده محاسباتی مستقر می‌شود.(٦) تمام این مراحل جایابی را در شکل ۲ توضیح دادیم.

۳-۲. قاعده بندی مسئله

در این بخش متغیرها و فرمول‌های مورد استفاده در راهکار پیشنهادی مورد بررسی قرار می‌گیرد. جدول ۱ متغیرها و نمادهای مورد استفاده مورد استفاده در راهکار پیشنهادی معرفی می‌گردد.

۳-۳.ساختار نرم‌افزار به‌عنوان خدمت درخواستی

بر اساس نوع و ویژگی هر مولفه نرم‌افزار به‌عنوان

جدول ۲: ساختار جدول نرم‌افزار به‌عنوان خدمت درخواست شده

Service_ID	Component ID	Component Type	Data	IN_Time	Request_Resource

جدول ۳: قیود منابع درخواستی

Minimum MIPS	Minimum RAM(GB)	Minimum Storage(GB)	Minimum Bandwidth(Mbps)

جدول۴: ساختار جدول کارآیی سرویس‌دهنده‌ها

Server_ID	Bandwidth(Mbps)	Storage(GB)	RAM(GB)	MIPS

جدول۵: ساختار جدول تاخیر ارتباطی سرویس‌دهنده‌ها

Server_ID_1	Server_ID_2	Latency

خدمت، ماشین های مجازی در اختیار آن درخواست قرار می‌گیرد و دسترسی به منابع وجود خواهد داشت. ساختار هر نرم‌افزار به‌عنوان خدمت ورودی به‌صورت جدول۲ است. هر نرم‌افزار به‌عنوان خدمت شامل چندین مولفه است. پس ممکن است شناسه نرم‌افزار به‌عنوان خدمت چندین بار تکرار شود.عنوان سوم در جدول معرف نوع و عنوان چهارم داده‌های مولفه است و در نهایت نیازهای سخت افزاری مولفه که در جدول ۳ آمده است.

عنوان Request_Resource مانند عنوان شرایط پذیرش نرم‌افزار به‌عنوان خدمت، معرف ویژگی های منابع مورد نیاز مولفه نرم‌افزار به‌عنوان خدمت است که در قالب یک ماتریس به‌صورت جدول ۳ استفاده می شود.

۳-۳-۱ واحد نظارت

این بخش در بازه‌های زمانی یکسان اطلاعات مربوط به درخواست‌ها و وضعیت منابع ابر را دریافت می‌کند. بدین ترتیب که اطلاعات مورد نیاز از کارآیی سرویس‌دهنده‌ها را درخواست کرده و پس از دریافت آن‌ها را در پایگاه داده خود ثبت می‌کند. ساختار هر رکورد مجموعه داده مربوط به سرویس‌ها به‌صورت جدول ٤ است

به همین ترتیب جدولی برای نگهداری تاخیر ارتباطی سرویس‌ها وجود دارد که ساختار ٥ را خواهد داشت.

شبه کد الگوریتم ۱ ساختار بخش نظارت را نمایش می‌دهد.

۳-۳-۲ جایابی نرم‌افزار به‌عنوان خدمت با استفاده از الگوریتم Hype

در ابتدا اولویت اجرای نرم‌افزار به‌عنوان خدمت درخواستی بر اساس مهلت زمانی آن‌ها استفاده شده است انجام می‌شود به شکلی که که دارای کمترین(کوچک‌ترین) مهلت زمانی و نزدیک‌ترین زمان ورودی است، دارای بالاترین اولویت (اولین عنصر لیست مرتب شده) جهت اجرای نرم‌افزار به‌عنوان خدمت می‌باشد. جهت مشخص کردن اولویت وظایف از رابطه ۱ استفاده می‌شود.

(۱)

$$Pri_i = \alpha \times \frac{1}{(Current - IN(Service_i)} + \beta \times \frac{1}{D(Service_i)}$$

و هم مقدار وزن در نظر گرفته شده برای هر یک از پارامترها است. محدوده مقادیر و بین صفر و یک در نظر گرفته می‌شود، به‌گونه‌ای که جمع این دو برابر یک شود.

بعد از اولویت‌بندی لیستی از سرویس‌های آزاد تهیه می‌شود. تا در مورد نرم‌افزار به‌عنوان خدمت ساده‌تر تصمیم‌گیری بشود که روی کدام گره قرار بگیرند. جدول (٦) ساختار این ماتریس را نشان می‌دهد.

بدین ترتیب هر سرویس‌دهنده به واسطه شاخص شناسه[43] در دسترس خواهد بود. مهم‌ترین بخش در راهکار پیشنهادی این جایابی نرم‌افزار به‌عنوان خدمت است. در ادامه با استفاده از الگوریتم HYPE سرویس‌دهنده

43- ID

Algorithm 1: Pseudo code for Monitoring Phase (S_i)

1:Input: (cloud service S_i)
2:Output : (R)
3: for (every cloud service S_i in SaaS Provider at the interval Δt) do
Monitor (Request of SaaS S_i in interval Δt);← 4: Req
Storage,Bandwidth, Server_ID of cloud service S_i in Monitor (MIPS,RAM, 5: R ←
interval Δt);
6: end for

جدول ۶: متغیرها و تعاریف آن‌ها

ID	Host_ID	Server_ID	Server_Type
1	1	1	1
2	.	2	2
.	.	.	.
.	1	K	1
.	.	.	.
.	.	.	.
.	M	1	.
.	.	2	.
.	.	.	.
N	M	K	.

مناسب برای هر مولفه از نرم‌افزار به‌عنوان خدمت جهت جایابی انتخاب می‌شود.

نحوه عملکرد الگوریتم Hype را در شبه کد الگوریتم ۲ نشان می‌دهیم، در این الگوریتم ابتدا جمعیت اولیه به تعداد مشخص (خط ۱) و به‌صورت تصادفی تولید (خط۲) و ارزش‌گذاری می‌شوند (خط ۳). سپس بر اساس مفهوم غلبگی عناصر مرتب می‌شوند (خط ۵). در ادامه با استفاده از شاخص ابرحجم عناصرهم‌رده جمعیت سنجش شده و مجدد مرتب‌سازی بر اساس این شاخص صورت می‌گیرد (خط٦). عمل ترکیب (خط ۸) و جهش (خط ۹) روی عناصر جمعیت انجام می‌شود. سپس با اجتماع عناصر قبلی، عناصر ترکیب شده و جهش یافته جمعیت جدید شکل می‌گیرد (خط ۱۰). مجدد جمعیت جدید ارزش‌گذاری می‌شود (خطوط ۱۱ تا ۱۳). سپس بر اساس مفهوم غلبگی عناصر مرتب می‌شوند (خط ١٤). در ادامه با استفاده از شاخص ابرحجم عناصرهم‌رده جمعیت سنجش شده و مجدد مرتب‌سازی بر اساس این شاخص صورت می‌گیرد

Algorithm2: Pseudo code for Hype_Select(R_Cl , Ser);

1:Input:(R)
2:Output: (Pop)
3: for i =1 to Pop_count do
4: Pop[i].pos=Random(1,size(R_Cl));
5: Pop[i].rank=Fitness(R_Cl[Pop[i].pos],Ser);
6: end for
7: Pop=none_dominated_sort(pop);
8 Pop=HyperVolume(pop);
9: for i=1 to Generation_Count do
10: Cross_pop=Cross_Over(pop);
11: Mut_pop=Mutation(Cross_pop);
12: Pop=Pop ∪ Cross_pop ∪ Mut_pop;
13: for j=1 to size(pop) do
14: Pop[j].rank=Fitness(R_Cl[Pop[j].pos],Ser)
15: end for
Pop=none_dominated_sort(pop) 17:
18: Pop=HyperVolume(pop);
19: Pop=Pop[1]…[Pop_count]
20: end for;
21: return Pop[1];

(خط۱۵). جمعیت آرشیو یاجمعیت بهینه به‌عنوان عناصر نخبه و نسل بعد انتخاب می‌شود (خط ١٦). اعمال خطوط ۸ تا ١٦ به تعداد نسل تکرار می‌شود و در نهایت عنصر برتر (ابتدایی) جمعیت به‌عنوان برترین راه‌حل استخراج می‌گردد (خط ۱۸). در ذیل این مراحل را به تفصیل شرح می‌دهیم.

۳-۳-۲-۱ ایجاد جمعیت اولیه

ابعاد هر عنصر از جمعیت به تعداد مولفه‌های موجود درنرم‌افزار به‌عنوان خدمت، یعنی عدد K وابسته است. در هر عضو جمعیت یا بردار، به‌صورت تصادفی شناسه K سرویس از کل سرویس‌دهنده‌های مجاز برای ارائه نرم‌افزار به‌عنوان خدمت بدون تکرار قرار می‌گیرد.

بدین ترتیب Si =[Xi,1 , Xi,2 , … , Xi,k] معرف i امین عضو از جمعیت اولیه است. به‌طوری‌که هر جزء Xi,d با توجه به این که 1<= d<= K است، به معنی سرویس‌دهنده شماره d در عنصر i ام است.

۳-۳-۲-۲ ارزیابی عناصر

بعد از ایجاد جمعیت اولیه باید ابتدا با توجه به شاخص‌های سخت افزاری درخواست تناسب ماشین مجازی و درخواست به‌صورت زیر محاسبه می‌گردد.

ابتدا مقدار سازگاری برای سه شاخص پردازنده، حافظه اصلی و دیسک با رابطه ۲ محاسبه می‌گردد، این سازگاری باید به نحوی باشد که میزان پردازنده،حافظه اصلی، دیسک و پهنای باند ماشین مجازی از میزان ظرفیت مورد نیاز درخواست به پردازنده، حافظه، پهنای باند درخواست فرستاده شده بیشتر یا برابر باشد که عدد یک به معنای پذیرش وسازگاری درخواست با منابع موجود و در غیر این صورت عدد صفر و به معنای عدم سازگاری و رد درخواست می‌باشد.

$$Comp_{K_{MIPS}} = \begin{cases} 1 \; VM_K.MIPS \geq Req.MIPS \\ 0 \; Otherwise \end{cases} \quad (۲)$$

$$Comp_{K_{Mem}} = \begin{cases} 1 \; VM_K.Mem \geq Req.Mem \\ 0 \; Otherwise \end{cases}$$

$$Comp_{K_{Disk}} = \begin{cases} 1 \; VM_K.Disk \geq Req.Disk \\ 0 \; Otherwise \end{cases}$$

$$Comp_{K_{BW}} = \begin{cases} 1 \; VM_K.BW \geq Req.BW \\ 0 \; Otherwise \end{cases}$$

در نهایت، سازگاری نهایی را با استفاده از رابطه ۳ محاسبه می‌کنیم. سازگاری نهایی با ضرب کردن تک تک سازگاری‌هایی که در رابطه۲ محاسبه نمودیم به دست می‌آید یعنی حتی اگر یک مورد ناسازگاری باشد و حجم مورد نیاز درخواست (پردازش، ذخیره، دیسک) بیشتر از حجم ماشین (پردازش، ذخیره، دیسک) باشد و صفر یا عدم پذیرش حاصل شود، عدم سازگاری نهایی و صفر جواب نهایی می‌شود، پس عمل نمی‌تواند انجام شود.

$$Comp_k = Comp_{K_{MIPS}} \times Comp_{K_{Mem}} \times Comp_{K_{Disk}} \times Comp_{K_{BW}} \quad (۳)$$

میزان مناسب بودن (یا مقدار سود) عنصرها با استفاده از تابع ارزشگذاری به دست می‌آید. در راهکار پیشنهادی برای هر عنصر از جمعیت اولیه ۲ هدف یا ارزش حاصل می‌شود.

۳-٤ هدف اول: زمان اجرای کل

برای محاسبه زمان اجرای کل ابتدا باید مقادیر زمان انتقال داده(DTT) و زمان پردازش (PT) محاسبه گردد. در صورتی که تعداد درخواست‌هاK باشد محاسبات در مورد TET به‌صورت زیر خواهد بود.

$$DTT = \sum_{m=1}^{K} Comp_m \times \left(\frac{Req_m.Data \times 8}{BW_{S_{m+},S_{m-1}}} + Latency_{S_{m+},S_{m-1}}\right) \quad (٤)$$

در رابطه ٤ S_0 به‌عنوان محل شروع درخواست جایابی، $Comp_m$ مقدار سازگاری نهایی، $BW_{S_{m+},S_{m-1}}$ پهنای باند، $\mathrm{Req_m.Data}$ میزان داده انتقالی، ضریب ۸ در نظر گرفتیم چون مبنا بر بایت لحاظ شد و تبدیل به بیت کردیم، $Latency_{S_{m+},S_{m-1}}$ میزان تاخیر می‌باشد.

پس از محاسبه زمان انتقال، زمان پردازش محاسبه می‌شود. در رابطه ٥ $\mathrm{Comp_m}$ سازگاری نهایی، $\mathrm{Req_m.MIPS}$ میزان پردازش مورد نیاز درخواست، $\mathrm{VM_m.MIPS}$ میزان توان پردازشی ماشین، $Max(DTT)$ بیشترین زمان مورد نیاز انتقال داده که از رابطه ٤ حاصل شد، می‌باشد.

$$PT = \sum_{m=1}^{K} Comp_m \times \left(\frac{Req_m.MIPS}{VM_m.MIPS} + Max(DTT)\right) \quad (٥)$$

در ادامه مسیری که برای اجرای کلیه درخواست‌ها باید طی شود با رابطه ٥ محاسبه می‌شود.

$$Path = set\ i\ \epsilon\ \{1..k\} |\ Comp_i = 1 \quad (٦)$$

در نهایت با توجه به مسیر طی شده،زمان اجرا به‌صورت زیر محاسبه می‌شود. در رابطه ۷ PT زمان پردازش، $Distance_{S_m,S_{m-1}}$ فاصله بین دو سرویس‌دهنده، L طول مسیر یا تعداد سرویس‌دهنده‌های طی شده برای اجرا می‌باشد.

$$TET = \sum_{m=1}^{L} PT \times Distance_{S_m, S_{m-1}} \quad (۷)$$

۳-۵ هدف دوم: هزینه

$$Cost = \sum_{m=1}^{K} Comp_m \times ((DTT_m \times VM_m.CostBW) + (PT_m \times (VM_m.CostMIPS + VM_m.CostMem + VM_m.CostDisk)) \quad (۸)$$

برای محاسبه هزینه از رابطه ۸ استفاده می‌شود. در رابطه ۸، $Comp_m$ سازگاری نهایی، DTT_m زمان انتقال داده، $VM_m.CostB$ هزینه پهنای باند، PT_m زمان پردازش، $VM_m.CostMIPS$ هزینه پردازش، $VM_m.CostMem$ هزینه حافظه، $VM_m.CostDisk$ هزینه دیسک مورد نیاز می‌باشد.

الگوریتم۳، شبه کد نحوه تخصیص منابع را با استفاده از الگوریتم Hype نشان می‌دهد. در این بخش هر مجموعه نرم‌افزار به‌عنوان خدمت همانند از لیست نرم‌افزار به‌عنوان خدمت مورد نیاز خوانده شده(خط ۱) و در ادامه توسط الگوریتم تخصیص بر پایه الگوریتم Hype منبع مورد نظر با این مجموعه جهت جایابی انتخاب می‌گردد(خط ۲). سرویس‌دهنده‌های انتخاب شده جهت جایابی از لیست منابع موجود حذف می‌گردد(خط۳). سپس سرویس‌دهنده‌های انتخاب شده به لیست سرویس‌دهنده‌های تخصیص یافته جهت معرفی به ابر اضافه می‌شود(خط ٤). در نهایت پس از اتمام درخواستها لیست منابع انتخاب شده جهت تخصیص به‌عنوان خروجی این بخش استخراج می‌گردد (خط ٦).

```
Algorithm3: Pseudo code for Hype_Placement (R_Cl , MS );
----------
1: foreach(Ser in Queue)
2: Ra=Hype_Select( R_Cl, Ser );
3: R_Cl.Remove(Ra);
4: Fin_RA.Add(Ra);
5: end for
6: return Fin_Ra;
```

جدول۷: مشخصات مراکز داده

معماری	X64
سیستم عامل	Cloud Linux
مدیریت ماشین‌های مجازی	XEN

جدول۸: مشخصات میزبان(Host)

پهنای باند	حافظه اصلی (GB)	فرکانس (MIPS)	تعداد هسته
Gbit/s 10	۳۲	۴۰۹۶	۸

۴ - ارزیابی عملکرد و نتایج

در این بخش نتایج حاصل از پیاده‌سازی الگوریتم HYPE برای بهبود جایابی مؤلفه‌های نرم‌افزار به‌عنوان خدمت مرکب در ابر با اهداف بهینه‌سازی در هزینه و زمان اجرای کل مورد بررسی قرار می‌گیرد. محیط مورد استفاده شبیه‌سازی NetBeans است. به‌منظور شبیه‌سازی دقیق و قابل توسعه ابر از ابزار Cloud Sim[۲٦] استفاده شده است. در ادامه به بررسی پارامترهای شبیه‌سازی، مورد استفاده قرارگرفته در این پژوهش و نتیجه شبیه‌سازی پرداخته می‌شود.

٤-۱- پارامترهای شبیه‌سازی

ساختار مرکز داده مورد استفاده در شبیه‌سازی در جدول ۷مشخص شده است.

در مرکز داده ۱۰ گره یا میزبان وجود دارد. در جدول ۷ مشخصات میزبان یا ساختار فیزیکی به‌صورت جامع آمده است. به هر میزان سخت افزار میزبان قوی‌تر و از سطح بالاتری برخوردار باشد میزان هزینه دسترسی به منابع مربوط به ماشین مجازی موجود در میزبان افزایش می‌یابد.

در جدول ۹ مهم‌ترین پارامترهای الگوریتم بهینه‌سازی HYPE را نمایش داده شده است.

در جدول ۱۰و ۱۱ به ترتیب مشخصات دو الگوریتم ازدحام ذرات چندهدفه و فاخته نشان داده شده است.

جدول ۱۲ میزان تاخیر بین سه عامل اصلی در روش پیشنهادی را نشان می‌دهد.

جدول۹: مشخصات الگوریتم بهینه‌سازی HYPE

تعداد جمعیت اولیه	تعداد نسل	نرخ جهش	نرخ ترکیب	شرط خاتمه
۵۰	۲۰۰	۰,۱	۰,۶	تعداد نسل

جدول۱۰: مشخصات الگوریتم بهینه‌سازی ازدحام ذرات چندهدفه

تعداد جمعیت اولیه	تعداد نسل	ضریب جابه‌جایی	شرط خاتمه
۵۰	۲۰۰	۰,۶	تعداد نسل

جدول ۱۱: مشخصات الگوریتم بهینه‌سازی فاخته

تعداد جمعیت اولیه	تعداد نسل	نرخ مهاجرت	شرط خاتمه
۵۰	۲۰۰	۰,۷	تعداد نسل

جدول ۱۲: میزان تاخیر در ارتباط بین سه عامل اصلی

ارتباط	میزان تاخیر (میلی ثانیه)
درخواست و سرویس‌دهنده‌ها	تصادفی توزیع نرمال بین ۵ تا ۲۰
سرویس‌دهنده‌ها با همدیگر	تصادفی توزیع نرمال بین ۳ تا ۱۰

در جدول ۱۳ مقادیر نیاز مولفه‌های نرم‌افزار به‌عنوان خدمت به پردازنده، حافظه اصلی، فضای ذخیره سازی و پهنای باند مشخص شده است. مقادیر در نظر گرفته شده برای سرویس‌دهنده‌ها هم به همین ترتیب است.

۲-٤ معیارهای کارآیی

شاخص های ارزیابی کل ساختار عبارتست از: هزینه و زمان اجرا.

- **هزینه**: مجموع هزینه اختصاص ماشین مجازی و هزینه ناشی از جریمه به ازای رخداد تخطی از شرایط نرم‌افزار به‌عنوان خدمت را میزان هزینه می گویند.
- **زمان اجرا**: زمان اجرا، تفاوت زمانی دقیق بین زمان درخواست جایابی و نیاز به اجرای نرم‌افزار به‌عنوان خدمت و زمان تحویل نرم‌افزار به‌عنوان خدمت میباشد.
- **میزان تابع ارزش‌گذاری**: مقدار حاصل شده از تابع ارزش‌گذاری در الگوریتم چندهدفه که متشکل از کاهش زمان اجرا و کاهش هزینه جایابی مولفه‌های نرم‌افزار به‌عنوان خدمت می‌باشد.

جدول ۱۳: مقادیر سخت افزار مورد نیاز مولفه‌های نرم‌افزار به‌عنوان خدمت و سرویس‌دهنده‌ها

پردازنده	تصادفی بین ۵۰ تا ۲۰۰ MIPS
حافظه اصلی	تصادفی بین ۱۲۸ تا ۵۱۲ MB
حافظه ذخیره سازی	تصادفی بین ۲ تا ۴ GB
پهنای باند	تصادفی بین ۲۰۰ تا ۵۰۰ Mbit/S
هزینه بابت مولفه سخت افزاری	تصادفی بین ۰,۱ تا ۰,۳ دلار

۵- نتایج شبیه‌سازی

در این بخش نتایج شبیه‌سازی مورد بررسی می گردد و نتایج روش پیشنهادی با دو الگوریتم فاخته و بهینه‌سازی ازدحام ذرات چند هدفه مقایسه می شود.

۵-۱ ارزیابی و نتایج

جهت ارزیابی روش پیشنهادی ۲ سناریو با ساختار جدول ١٤تشکیل شد. در شبیه‌سازی هر سناریو سه معیار مهم ارزیابی در سه روش مورد بررسی قرار می‌گیرد.

۵-۲ سناریو اول

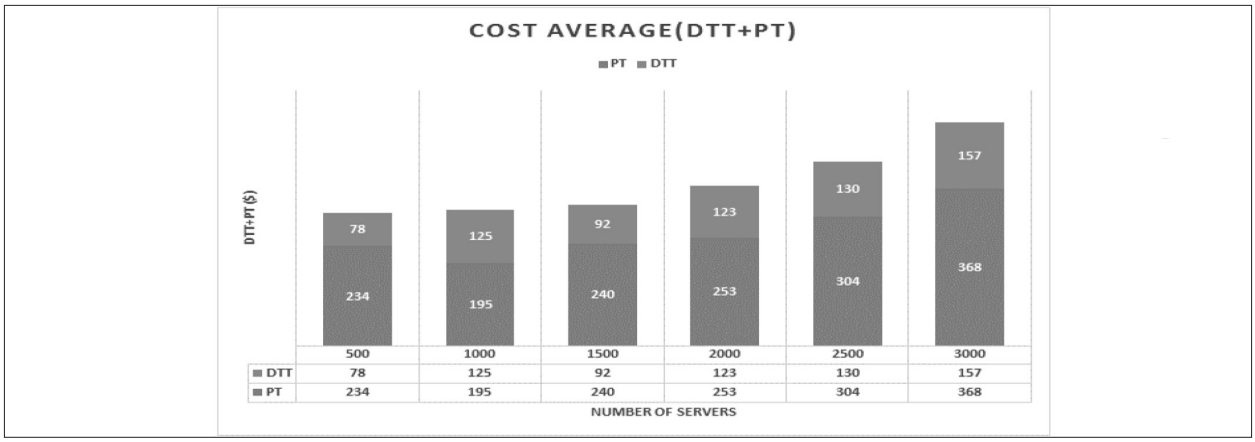
زمان اجرا به‌عنوان یک از موثرترین اهداف شرایط نرم‌افزار به‌عنوان خدمت یا توافقنامه سطح خدمات نقش موثری در تخصیص منبع دارد. نمودار ۱ مدت زمان دو شاخص زمان انتقال داده[44] و زمان پردازش[45]را در زمان اجرا در روش پیشنهادی را نمایش می‌دهد.

در صورتی‌که زمان اجرا مورد نظر درخواست که با نام حد مجاز زمان پاسخگویی شناسایی می‌شود حاصل نشود، تخطی از شرایط پذیرش نرم‌افزار به‌عنوان خدمت رخ داده است و کیفیت سرویس‌دهی از دید کاربر کاهش داشته است. در این بخش میانگین زمان پاسخگویی در روش پیشنهادی بررسی می‌شود و با دو الگوریتم فاخته و بهینه‌سازی ذرات چند هدفه مقایسه می‌شود. نمودارهای۲ میانگین زمان اجرا در طول شبیه‌سازی با تغییر سرویس‌دهنده‌ها از ۵۰۰ تا ۳۰۰۰ و تعداد ٤۰ مولفه نرم‌افزار به‌عنوان خدمت را نمایش می‌دهد.

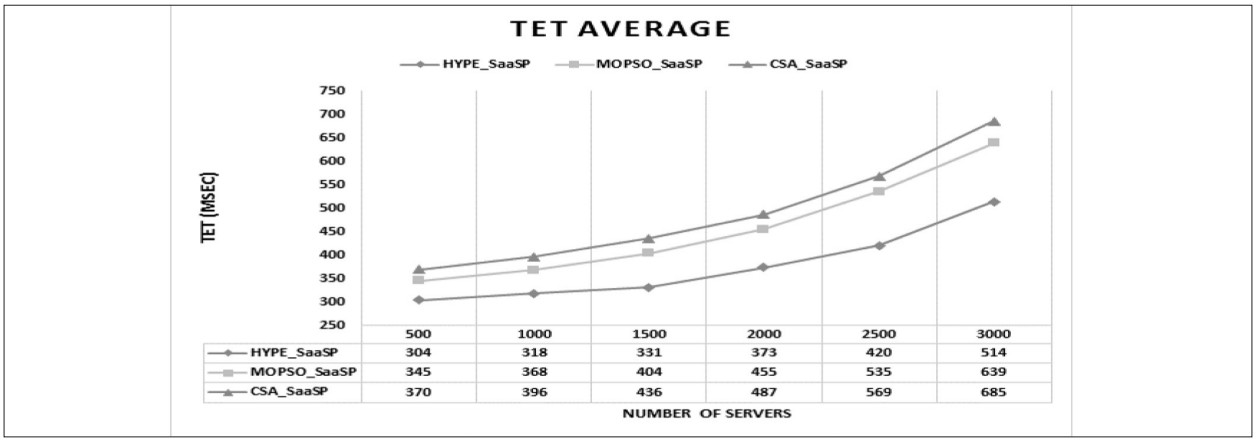
44- DTT
45- PT

جدول ۱۴: سناریوهای مورد ارزیابی

سناریو	تعداد سرویس‌دهنده‌ها	تعداد مولفه‌های نرم‌افزار به‌عنوان خدمت	هدف
یک	۵۰۰ تا ۳۰۰۰ (۲۵ درصد ذخیره سازی، ۷۵ درصد محاسباتی)	۴۰	اندازه‌گیری سه شاخص: مقدار تابع برازش، زمان پاسخگویی، مقایسه هزینه و میزان بهره‌وری منابع
دو	۳۰۰۰	۵۰ تا ۲۰۰ (۲۰ درصد DC و ۸۰ درصد AC)	



نمودار ۱: میانگین زمان انتقال و محاسبه در روش پیشنهادی

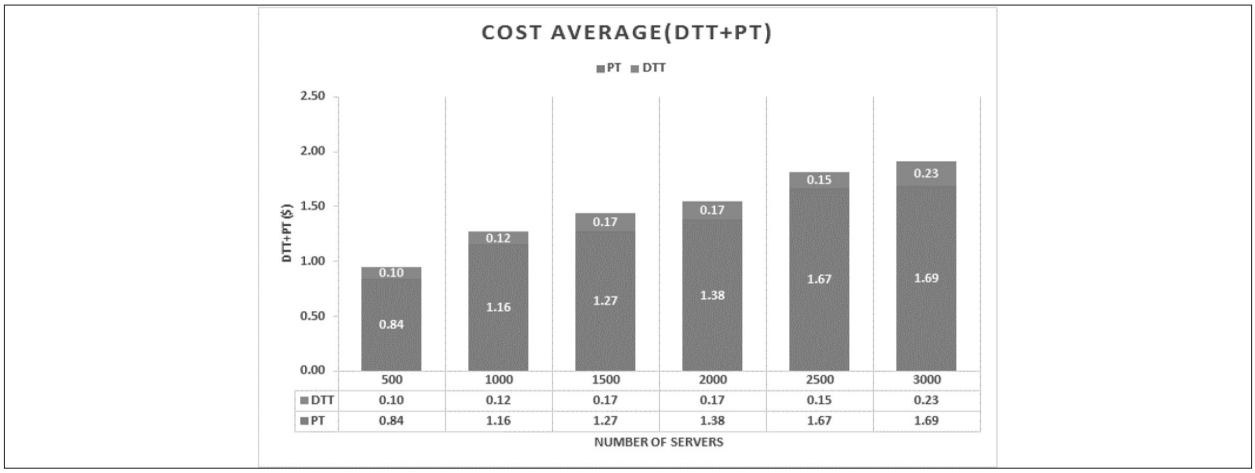


نمودار ۲: میانگین زمان اجرا در سناریوی اول

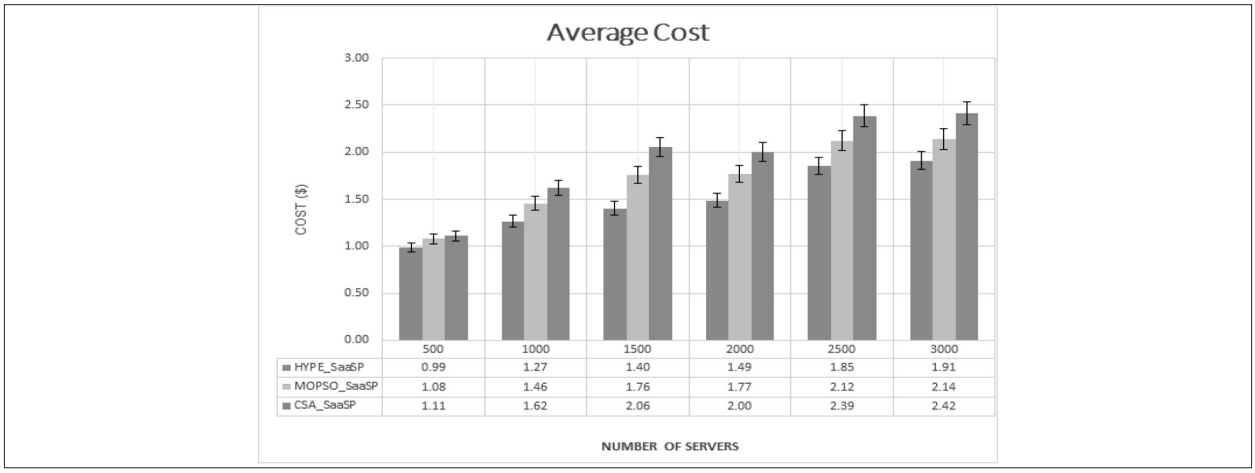
بـــا توجه به نمـــودار ۲ میانگین زمان اجـــرا در روش پیشنهادی کمتر است.

اصلی‌ترین شرط ســـرویس میزان هزینه مورد توافق کاربر اســـت. در صورتی که هزینه سرویس‌دهی افزایش یابد، به‌طور مشخص سود سرویس‌دهنده کاهش و به همان میزان موجب نارضایتی کاربر از ســـاختار سرویس‌دهی می‌شود. در این بخش میانگین هزینه در روش پیشنهادی بررسی می‌شـــود و با دو الگوریتم فاخته و ازدحام ذرات چند هدفه مقایسه می‌شود. نمودار ۳میانگین هزینه را برای حالت انتقال(پهنای باند) و حالت محاسبه را به تفکیک نشان می‌دهد. نمودار ٤میانگین هزینه را درشبیه‌ســـازی سناریو دوم نمایش می‌دهد.

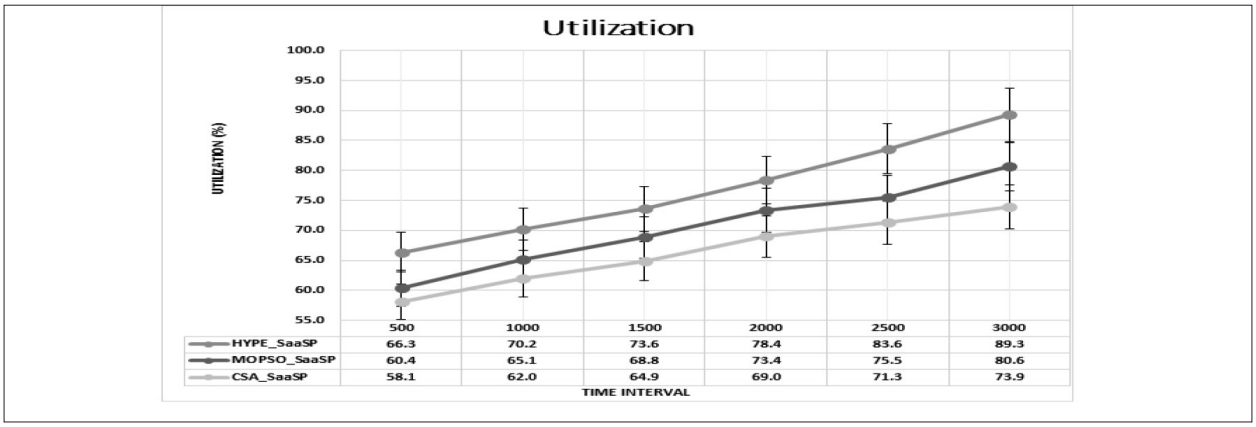
بـــا توجه به نتایـــج، روش پیشـــنهادی کارآیی بهتری در مـــورد میانگین هزینه دارد. در الگوریتم پیشـــنهادی به دلیل در نظر گرفتن شـــرایط درخواست و قرار دادن هزینه به‌عنوان یکی از اهداف الگوریتم HYPE بیشترین دقت را در مورد هزینه تخصیص سرویس‌دهنده‌ها و جایابی نرم‌افزار به‌عنـــوان خدمـــت اعمال می‌کنـــد. اســـتفاده از دو مرحله ســـخت‌گیری در مورد زمان ارتباط و زمان محاسبه دقت در انتخاب و تخصیص ســـرویس‌دهنده را افزایش داده و



نمودار ۳: میانگین هزینه زمان انتقال و محاسبه



نمودار ۴: میانگین هزینه در سناریو اول

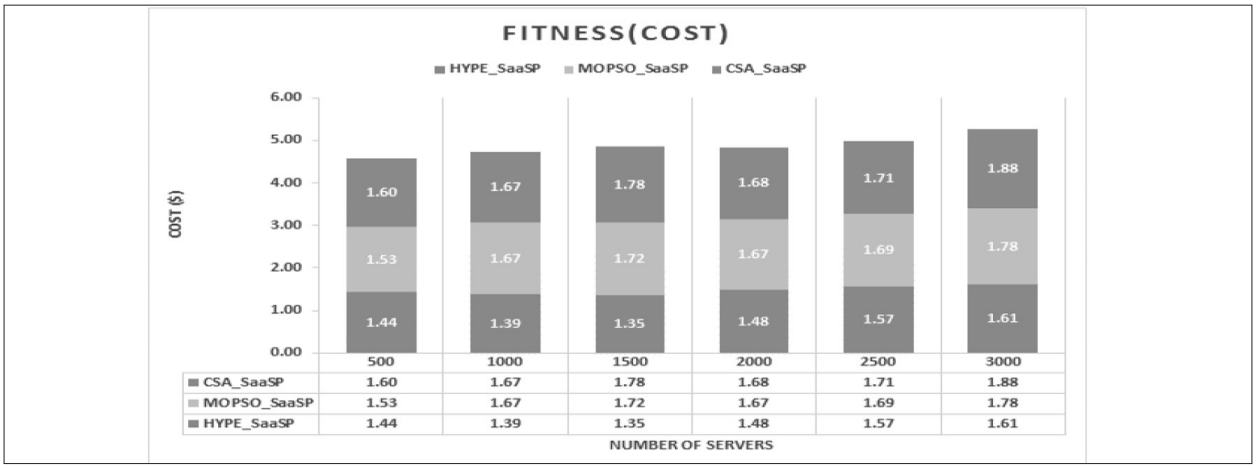


نمودار ۵: میانگین بهره‌وری از منابع در سناریو اول

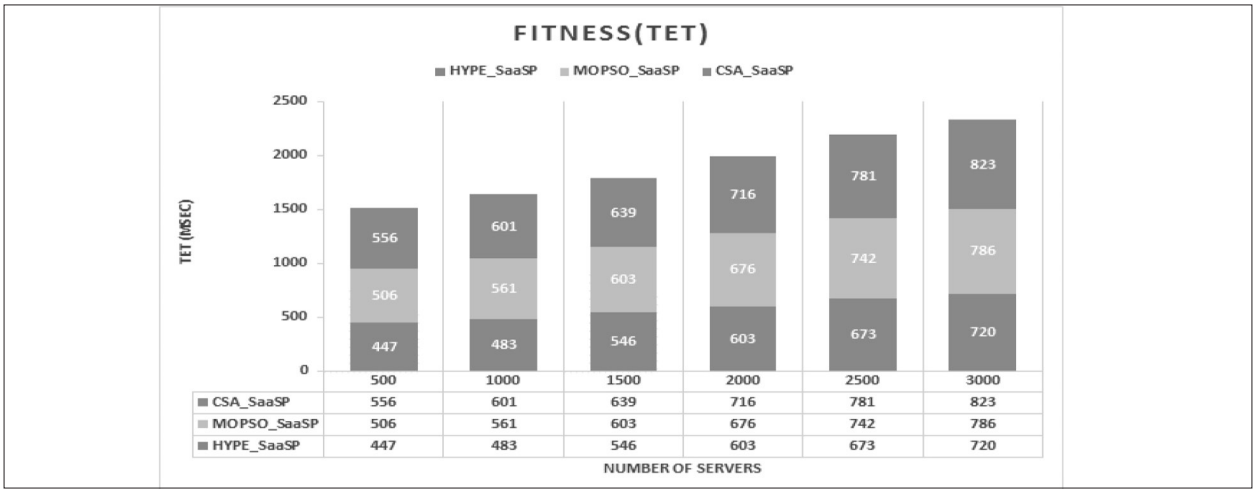
به همین ترتیب هزینه مورد نظر کاربران را برآورده نماید.

نمودار ۵ میزان بهره‌وری(میزان اســتفاده از منابع) را در سه روش مورد مقایسه نمایش می‌دهد.

به‌طور مشــخص به دلیل تخصیص صحیح وظایف به مراکز داده، منابع پردازشگر با بهترین وضعیت در اختیار درخواســت ها قرار می گیرد. البته ممکن اســت در تعداد درخواســت بالا میزان بهره وری به آستانه یک برسد که دلیل آن تعداد درخواســت های بالا اســت کــه نیاز به در



نمودار ۶: میانگین تابع برازش در مورد هزینه



نمودار ۷: میانگین تابع برازش در مورد زمان اجرا

اختیـــار گرفتن پردازنـــده دارند. اســـتفاده از HYPE، تاثیر بســـیار زیادی در کنترل مناســـب درخواست ها و تصمیم گیری در مورد تخصیص مراکز داده شده است.

نمودار٦ و ۷ میانگین میزان تابع برازش در مورد هدف هزینه و هدف زمان اجرا را نشان می‌دهد.

جدول١٤ میـــزان بهترین، میانگین و بدترین مقدار تابع برازش در تغییرات تعداد سرویس‌دهنده‌ها را در روش‌های مختلف نشان می‌دهد.

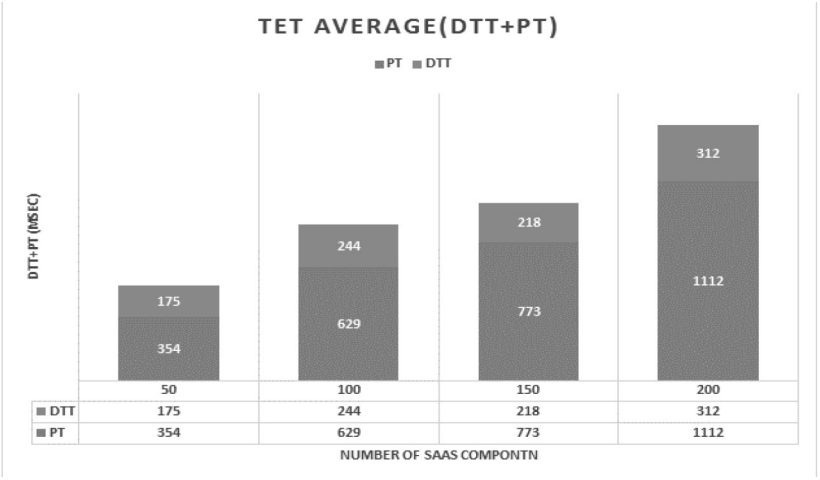
۳-۵ سناریو دوم

زمان اجـــرا به‌عنوان یک از موثرترین اهداف شـــرایط نرم‌افزار به‌عنوان خدمت یا توافقنامه ســـطح خدمات نقش موثری در تخصیص منبـــع دارد. نمودار ۸ مدت زمان دو شـــاخص زمان انتقال داده و زمان پـــردازش را در زمان اجرا در روش پیشـــنهادی را نمایـــش می‌دهد. در صورتی کـــه زمان اجرا مورد نظر در خواســـت که با نام حد مجاز زمان پاسخگویی شناسایی می شود حاصل نشود، تخطی از شـــرایط پذیرش نرم‌افزار به‌عنوان خدمت رخ داده است و کیفیت سرویس‌دهی از دید کاربر کاهش داشته است. در این بخش میانگین زمان پاســـخگویی در روش پیشنهادی بررســـی می‌شـــود و با دو الگوریتم فاخته و بهینه‌سازی ازدحام ذرات چند هدفه مقایســـه می‌شـــود. نمودارهای ۸ میانگین زمان اجرا در طول شبیه‌سازی با تغییر مولفه‌های نرم‌افـــزار به‌عنوان خدمـــت از ۵۰ تـــا ۲۰۰ و تعداد ۳۰۰۰ سرویس‌دهنده را نمایش می‌دهد.

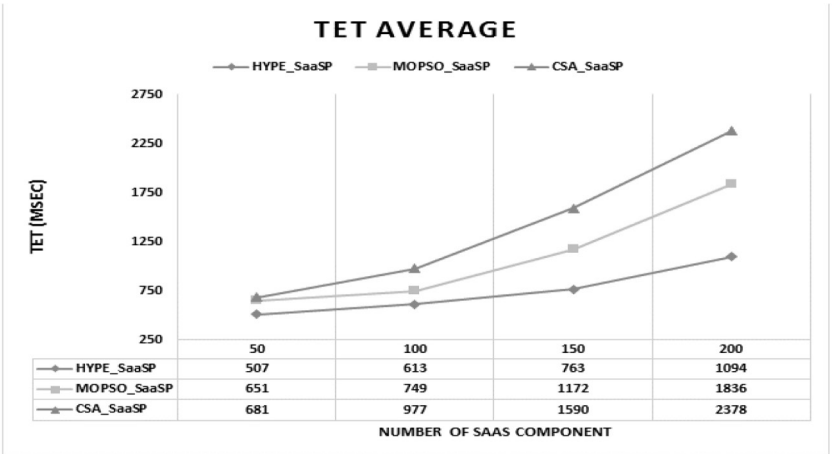
بـــا توجه به نمـــودار ۹ میانگین زمان اجـــرا در روش پیشنهادی کمتر است.

جدول ۱۴: جدول تغییرات تابع برازش

تعداد سرویس‌دهنده		۵۰۰		۱۰۰۰		۱۵۰۰		۲۰۰۰		۲۵۰۰		۳۰۰۰	
ه= هزینه، ز= زمان اجرا		ه	ز	ه	ز	ه	ز	ه	ز	ه	ز	ه	ز
HYPE	بهترین	۰,۹۹	۳۰۴	۱,۲۷	۳۱۸	۱,۴۰	۳۳۱	۱,۴۹	۳۷۳	۱,۸۵	۴۲۰	۱,۹۱	۵۱۴
	بدترین	۱,۲۴	۳۸۸	۱,۳۳	۶۰۳	۱,۴۷	۶۴۱	۱,۷۷	۶۷۸	۲,۱۳	۷۷۲	۱,۹۷	۸۱۲
	میانگین	۱,۱۴	۳۵۴	۱,۳۴	۴۶۹	۱,۴۸	۴۹۶	۱,۶۴	۵۳۴	۲,۰۷	۶۰۱	۲,۰۲	۶۶۸
MOPSO	بهترین	۱,۲۱	۳۷۲	۱,۲۹	۴۳۸	۱,۳۲	۴۶۱	۱,۴۰	۵۵۷	۱,۴۷	۵۶۹	۱,۶۷	۶۲۴
	بدترین	۱,۸۱	۵۸۴	۱,۹۰	۶۱۸	۱,۹۲	۶۵۸	۱,۸۹	۷۱۲	۲,۰۷	۷۸۲	۲,۲۱	۸۴۵
	میانگین	۱,۵۳	۵۰۶	۱,۶۷	۵۶۱	۱,۷۲	۶۰۳	۱,۶۷	۶۷۶	۱,۶۹	۷۴۲	۱,۷۸	۷۸۶
CSA	بهترین	۱,۲۹	۳۸۸	۱,۴۱	۴۵۱	۱,۴۹	۴۷۳	۱,۵۸	۵۹۲	۱,۶۶	۶۰۱	۱,۷۳	۶۸۹
	بدترین	۱,۸۴	۶۰۱	۱,۹۷	۶۲۷	۱,۹۸	۷۰۴	۱,۹۳	۷۷۹	۲,۱۶	۸۲۷	۲,۳۳	۹۱۹
	میانگین	۱,۶	۵۵۶	۱,۶۷	۶۰۱	۱,۷۸	۶۳۹	۱,۶۸	۷۱۶	۱,۷۱	۷۸۱	۱,۸۸	۸۲۳

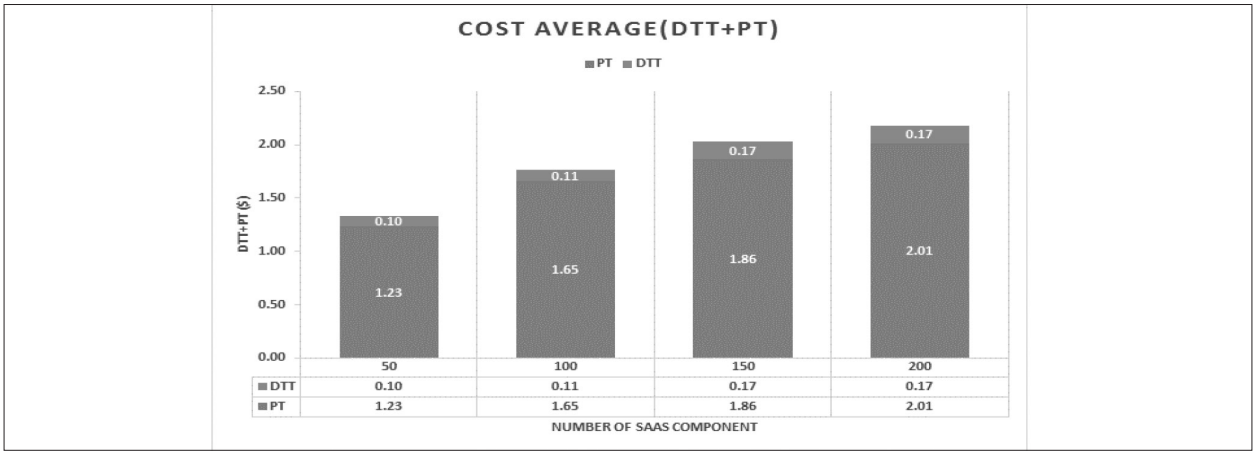


نمودار ۸: میانگین زمان انتقال و محاسبه در روش پیشنهادی

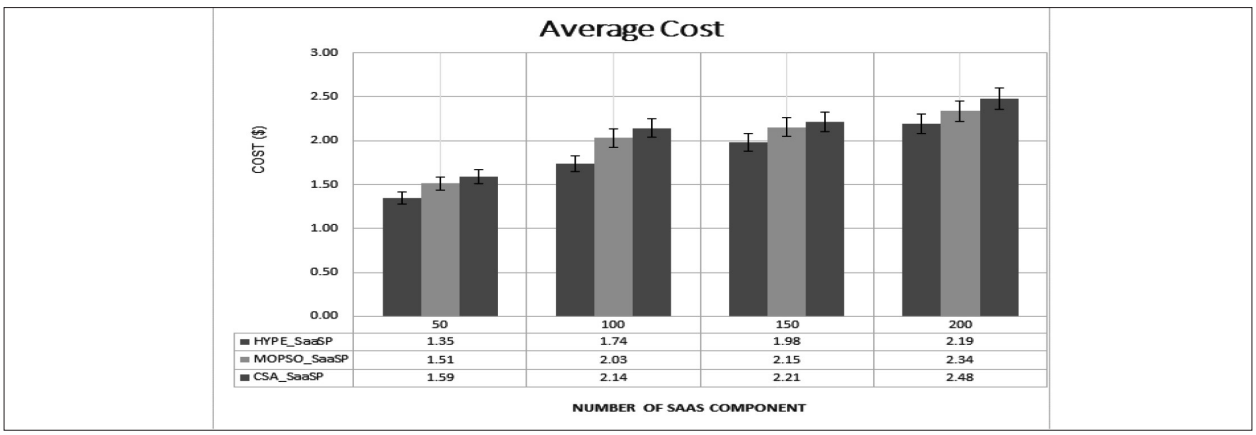


نمودار۹: میانگین زمان اجرا در سناریوی دوم

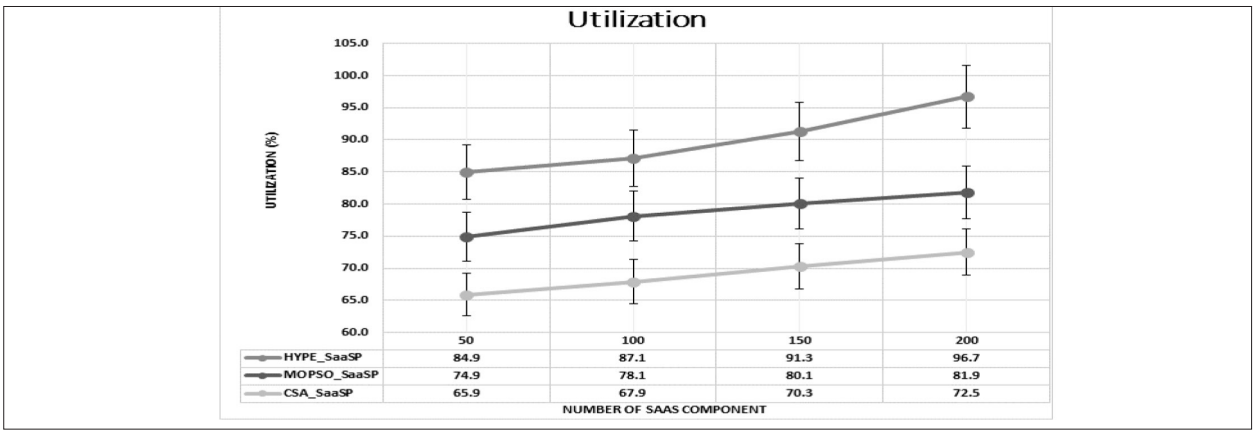
اصلی‌ترین شرط نرم‌افزار به‌عنوان خدمت میزان هزینه مورد توافق کاربر است. در صورتی که هزینه سرویس‌دهی افزایش یابد، به‌طور مشخص سود سرویس‌دهنده کاهش و به همان میزان موجب نارضایتی کاربر از ساختار سرویس‌دهی می شود. در این بخش میانگین هزینه در روش پیشنهادی بررسی می‌شود و با دو الگوریتم فاخته



نمودار ۱۰: میانگین هزینه زمان انتقال و محاسبه



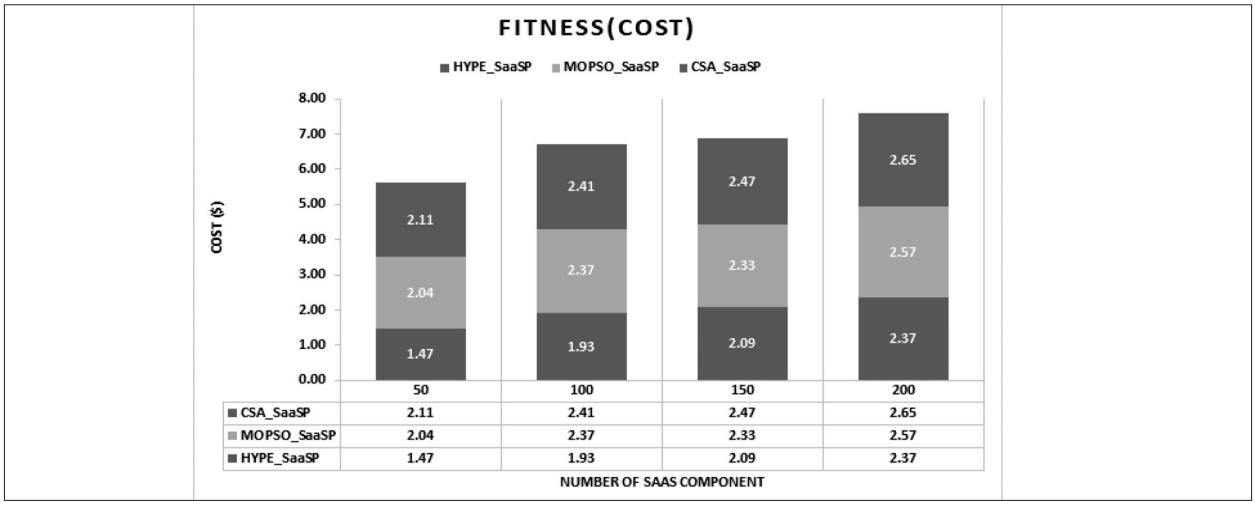
نمودار ۱۱: میانگین هزینه در سناریو دوم



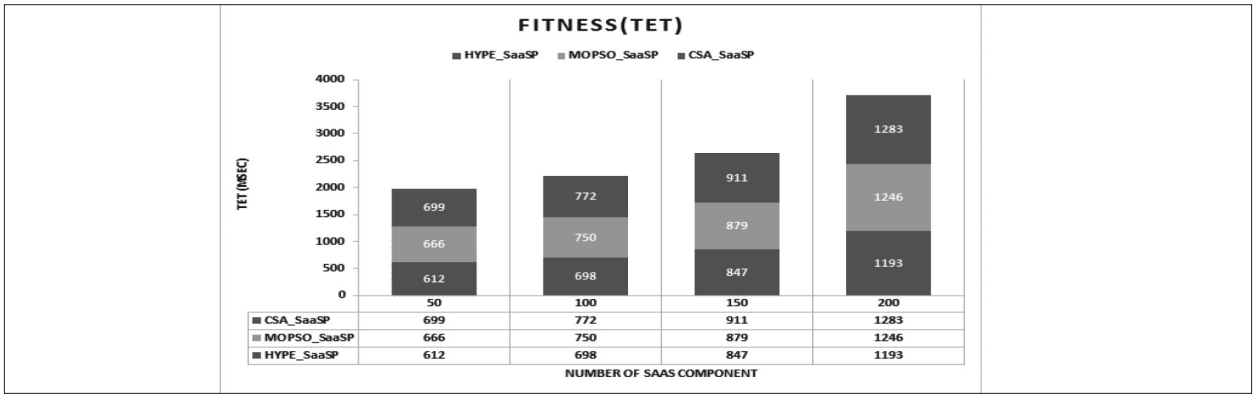
نمودار ۱۲: میانگین بهره‌وری از منابع در سناریو دوم

و ازدحـــام ذرات چند هدفه مقایســـه می‌شـــود. نمودار ۱۰ میانگیـــن هزینه را برای حالـــت انتقال(پهنای باند) و حالت محاســـبه را به تفکیک نشـــان می‌دهد. نمودار ۱۱میانگین هزینه را درشبیه‌سازی سناریو دوم نمایش می‌دهد.

بـــا توجه به نتایـــج، روش پیشـــنهادی کارآیی بهتری در مـــورد میانگین هزینه دارد. در الگوریتم پیشـــنهادی به دلیل در نظر گرفتن شـــرایط درخواست و قرار دادن هزینه به‌عنـــوان یکی از اهداف الگوریتم HYPE بیشـــترین دقت را در مـــورد هزینه تخصیـــص ســـرویس‌دهنده‌ها و جایابی نرم‌افـــزار به‌عنوان خدمـــت اعمال می‌کند. اســـتفاده از دو مرحله سخت‌گیری در مورد زمان ارتباط و زمان محاسبه دقت در انتخاب و تخصیص سرویس‌دهنده را افزایش داده



نمودار۱۳: میانگین تابع برازش در مورد هزینه



نمودار ۱۴: میانگین تابع برازش در مورد زمان اجرا

جدول ۱۵: جدول تغییرات تابع برازش

تعداد مولفه‌ها		۵۰		۱۰۰		۱۵۰		۲۰۰	
ه= هزینه، ز= زمان اجرا		ه	ز	ه	ز	ه	ز	ه	ز
HYPE	بهترین	۱,۳۵	۵۲۷	۱,۷۴	۵۸۱	۱,۹۸	۶۲۷	۲,۱۹	۱۱۵۸
	بدترین	۱,۳۹	۵۷۷	۱,۸۰	۶۳۳	۲,۳۷	۶۹۴	۲,۳۲	۱۲۱۶
	میانگین	۱,۴۲	۵۶۷	۱,۸۷	۶۱۸	۲,۲۷	۶۷۲	۲,۲۷	۱۱۹۸
MOPSO	بهترین	۱,۵۸	۵۴۹	۱,۷۸	۵۹۳	۱,۸۵	۶۹۸	۲,۰۷	۱۱۷۷
	بدترین	۲,۱۳	۷۲۸	۲,۸۷	۷۹۲	۲,۹۱	۹۴۸	۳,۱۸	۱۳۸۶
	میانگین	۲,۰۴	۶۶۶	۲,۳۷	۷۵۰	۲,۳۳	۸۷۹	۲,۵۷	۱۲۴۶
CSA	بهترین	۱,۷۲	۵۸۲	۱,۸۳	۶۱۰	۱,۹۲	۷۱۵	۲,۲۱	۱۱۹۲
	بدترین	۲,۲۱	۷۵۹	۲,۹۶	۸۱۴	۳,۰۸	۹۸۷	۳,۲۷	۱۴۰۵
	میانگین	۲,۱۱	۶۹۹	۲,۴۱	۷۷۲	۲,۴۷	۹۱۱	۲,۶۵	۱۲۸۳

و به همین ترتیب هزینه مورد نظر کاربران را برآورده نماید. نمودار ۱۲میزان بهره‌وری (میزان استفاده از منابع) را در سه روش مورد مقایسه نمایش می‌دهد. استفاده از HYPE، تاثیر بسیار زیادی در کنترل مناسب درخواست ها و تصمیم گیری در مورد تخصیص مراکز داده شده است.

نمودار۱۳و ۱۴ میانگین میزان تابع برازش در مورد هدف هزینه و هدف زمان اجرا را نشان می‌دهد.

جدول میزان بهترین، میانگین و بدترین مقدار تابع

برازش در تغییـــرات تعداد مولفه‌های نرم‌افـــزار به‌عنوان خدمت را در روش‌های مختلف نشان می‌دهد.

۶-نتیجه‌گیری

در این مقاله الگوریتـــم HYPE برای جایابی مؤلفه‌های نرم‌افزار به‌عنوان خدمت در محیط ابر با اهداف کاهش هزینه وکاهش زمان اجرا برای بهینه‌ســـازی جایابی ارائه شد که در الگوریتم پیشنهادی مؤلفه‌های نرم‌افزار به‌عنوان خدمت به‌طوری که مؤلفه‌های داده بر روی ماشین مجازی مستقر بر روی سرویس‌دهنده‌های ذخیره و مؤلفه‌های برنامه بر روی ماشین مجازی مســـتقر بر روی سرویس‌دهنده‌های محاســـباتی اســـتقرار یابند به‌گونه‌ای که ظرفیت منابع (از جمله ظرفیت منابع پردازشی، ظرفیت منابع حافظه، ظرفیت ذخیـــره) با میـــزان ظرفیت درخواســـت مورد نیـــاز برای پردازش، ذخیره و حافظه ســـازگاری داشـــته باشد تا این جایابی بتواند انجام شـــود و دو هدف زمان اجرای کلی و هزینه ما را به حداقل برساند. این جایابی در ماژول جایابی سیستم مدیریت منابع قرار دارد که بین مؤلفه‌های نرم‌افزار به‌عنـــوان خدمت و زیرســـاخت ابر ارائه‌دهنـــده نرم‌افزار به‌عنـــوان خدمـــت مدیریت می‌کند اطلاعـــات این دو بخش بـــه بخش نظارت رفته و در نهایـــت به بخش جایابی که با الگوریتم HYPE کار می‌کند ارسال می‌شود اهداف الگوریتم HYPE کاهش در زمان اجرای کلی و هزینه می‌باشـــد که با محاسبه این دو معیار به بهترین جایابی با بهینه‌سازی در زمان اجرای کلی و هزینه حاصل می‌گردد. به‌منظور تکمیل و توسعه این تحقیق استفاده ازروش‌های ترکیبی، به‌عنوان نمونه ترکیب برنامه‌نویسی پویا با الگوریتم‌های تکاملی دیگر، در مسئله‌های بزرگ در صورتی‌که تعداد فراهم‌کننده‌ها بالا باشد و به نسبت آن تعداد درخواست‌های زیادی نیز وجود داشته باشـــد. می‌توان با استفاده از برنامه‌نویسی پویا آن مســـئله را به زیرمسئله‌ها شکســـت و هر کدام از آن‌ها را با الگوریتم‌های تکاملی به جواب بهینه رســـاند و همچنین استفاده از الگوریتم‌های تکاملی دیگر برای جایابی بهینه از جمله GREA ، NPSO پیشنهاد می‌شود.

۷.مراجع

1. Mell P, Timothy G The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology.2011
2. Foster, I., Zhao, Y., Raicu, I. and Lu, S. Cloud computing and grid computing 360-degree compared. arXiv preprint arXiv:0901.0131. ., 2008
3. Motahari-Nezhad, H.R., Stephenson, B. and Singhal, S., Outsourcing business to cloud computing services: Opportunities and challenges. IEEE Internet Computing, 10(4), pp.1-17. 2009.
4. Vaquero, L.M., Rodero-Merino, L., Caceres, J. and Lindner, M. A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), pp.50-55. ., 2008
5. Bhardwaj, S., Service level agreement aware SaaS placement in cloud (Doctoral dissertation).2015
6. Kumar, A., Placement of software-as-a-service components in cloud computing environment (Doctoral dissertation).2014
7. Yusoh ZIM Composite SaaS resource management in cloud computing using evolutionary computation. PhD thesis, Science and Engineering Faculty Queensland University of Technology Brisbane, Australia.2013.
8. Candan KS, Li W-S, Phan T, Zhou M At the frontiers of information and software as services. In: New Frontiers in information and software as services. Springer, pp 283–300,2011.RightScaleIRightscale2016stateofthecloudreport.Technical report, RightScale Inc.2016.
9. Statista I (2016) Software as a service (SaaS) subscription revenue from 2012 to 2016 by category (in billion u.s. dollars). http:// www.statista.com/statistics/468649/saas-software-subscriptionrevenue-by-category/. Accessed 22 March 2016.
10. Cisco (2008) Cisco service-oriented network architecture: support and optimizesoaandweb2.0applications. Technicalreport,CiscoInc
11. TalbiE-G,GuzekM,BouvryP(2015)Asurveyofevolutionarycomputationforresource managementofprocessingincloudcomputing [review article]. IEEE Comput Intell Mag 10(2):53–67. 2015
12. Chainbi, W. and Sassi, E.,. A multiswarm for composite SaaS placement optimization based on PSO. Software: Practice and Experience, 48(10), pp.1847-1864.2018
13. Altmann, J. and Kashef, M.M., Cost model based service placement in federated hybrid clouds. Future Generation Computer Systems, 41, pp.79-90.2014
14. Huang, K.C. and Shen, B.J., Service deployment strategies for efficient execution of composite SaaS applications on cloud platform. Journal of Systems and Software, 107, pp.127-141.2015

15. Bhardwaj, S., Service level agreement aware SaaS placement in cloud (Doctoral dissertation).2015
16. Kumar, A., Placement of software-as-a-service components in cloud computing environment (Doctoral dissertation).2014
17. Yusoh, Z.I.M. and Tang, M., July. A penalty-based genetic algorithm for the composite SaaS placement problem in the cloud. In IEEE Congress on Evolutionary Computation (pp. 1-8). IEEE.2010
18. Yusoh, Z.I.M. and Tang, M., November. A cooperative coevolutionary algorithm for the composite SaaS placement problem in the cloud. In International Conference on Neural Information Processing (pp. 618-625). Springer, Berlin, Heidelberg.2010
19. Yusoh, M. and Izzah, Z., Composite SaaS resource management in cloud computing using evolutionary computation (Doctoral dissertation, Queensland University of Technology).2013
20. Ni, Z.W., Pan, X.F. and Wu, Z.J., An ant colony optimization for the composite SaaS placement problem in the cloud. In Applied mechanics and materials (Vol. 130, pp. 3062-3067). Trans Tech Publications.2012
21. Huang, K.C. and Shen, B.J., Service deployment strategies for efficient execution of composite SaaS applications on cloud platform. Journal of Systems and Software, 107, pp.127-141.2015
22. Liu, Z., Hu, Z. and Jonepun, L.K., Research on Composite SaaS Placement Problem Based on Ant Colony Optimization Algorithm with Performance Matching Degree Strategy. Journal of Digital Information Management, 12(4).2014
23. Mezni, H., Sellami, M. and Kouki, J., Security-aware SaaS placement using swarm intelligence. Journal of Software: Evolution and Process, 30(8), p.e1932.2018
24. Mousa, A., Bentahar, J. and Alam, O., Context-aware composite SaaS using feature model. Future Generation Computer Systems, 99, pp.376-390.2019
25. Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar 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