Balancing Resource Utilization for Continuous Virtual Machine Requests in Clouds

تعدیل استفاده از منابع جهت درخواست های مداوم ماشین مجازی در Cloud

چکیده :

یکی از وظایف اصلی در Cloud ها ، قرار دادن ماشین های مجازی (VMs) در یک خوشه از ماشین های فیزیکی (PMs) است. ما می توانیم از سیاست های جایگذاری مناسبی بهره مند شویم ، به طور مثال : صرفه جویی در هزینه .

در این مقاله ما مسئله جایگذاری پیوسته یک ماشین مجازی را بررسی و یک الگوریتم برخط برای کاهش مصرف انرژی Cloud ها پیشنهاد میدهیم. روش های عمومی که با شیوه ی حرص و طمع با این مشکل روبرو می شوند ، منجر به ریزش (نشت) منابع و پدیده هدر رفتن منابع خواهند شد. تاثیر مستقیم این پدیده ، داشتن تعداد بیشتری از PM های در حال اجراست که باعث افزایش مصرف انرژی می شود. ما ثابت می کنیم در زمان جایگذاری VM ها برای به حداقل رساندن مصرف برق در حالی که درخواست های VM به طور مداوم به Cloud ها می رسد ، خود یک مسئله NP-Complete است . ما یک الگوریتم موازنه را با اصل اجتناب کردن از ریزش (نشت) منابع پیشنهاد می دهیم .

الگوریتم موازنه ، درخواستهای VM در زمان اجرا را مدیریت و بهره وری بالا از منابع را تضمین می کند.کاهش تعداد PMها و مصرف انرژی را می توان به عنوان نتیجه بهره وری بالا از منابع در نظر گرفت. ما الگوریتم موازنه را از طریق آزمایش شبیه سازی های مختلف ارزیابی می کنیم . نتایج تجربی نشان می دهد که الگوریتم موازنه می تواند به طور موثری تعداد PM های در حال اجرا را کاهش دهد ، به نحوی که کارایی قابل توجهی در صرفه جویی در مصرف انرژی داشته باشد.

۱) مقدمه :

اعتقاد بر این است که Cloud ها می توانند هزینه های زیربنایی فناوری اطلاعات (IT) را بطور قابل توجهی کاهش دهند و مجازی سازی مزایای قابل توجهی در رایانش ابری فراهم می کند . همراه با رشد محاسبات ابری و مجازی سازی، قرار دادن ماشین های مجازی (VMs) در یک خوشه از ماشین های فیزیکی (PMs) یکی از وظایف اصلی در Cloud ها از طریق مجازی سازی محسوب می شود. در عین حال ، این کار بسیار پیچیده ای است که ناشی از محدودیت های مختلف از جمله عملکرد ([۳]، [۲۴])، مقیاس پذیری ([۳]، [۲۲])، در دسترس بودن ([۲۰])، شبکه، ترافیک و یا پهنای باند ([۳]، [۱۱]، [۲۲]، [۲۱]) ، هزینه و قدرت ([۱۳])، قابلیت ارتجاعی ([۱۸]) و غیره ، می باشد . در این حوزه ، یکی از مسائل مهم برای جایگذاری VM، توان مصرفی PM های در حال اجرایی هستند که خود میزبان VM می شود. به طور کلی ، مصرف انرژی بخش قابل توجهی از هزینه Cloud ها است، شرکت ها میلیون ها دلار در این راه پرداخت می کنند ([۲۳]، [۲]) .

آثار موجود ([۹]،[۸]) ، نشان دهنده اهمیت مسئله جایگذاری VM هستند و ما می توانیم از سیاست های مناسبی در زمینه صرفه جویی در انرژی بهره مند شویم . به حداقل رساندن تعداد PM های مورد استفاده جهت کاهش شدت مصرف انرژی ([۸]) به شدت مفید است. کارهای قبلی ([۲۰]، [۳]، [۲۱]، [۲۲]) عمدتا در مورد جایگذاری استاتیک VM است، مجموعه ای از VMهای اختصاص یافته ، ابتدا در وضعیت ایستا قرار می گیرند . هرچند ، تخصیص مجموعه ای از VMها از قبل ، در همه موارد مناسب نیست . ما بر روی مشکل جایگذاری VM پیوسته تمرکز می کنیم، VMزمانی ایجاد می شود که Cloud یک درخواست VM دریافت کند و درخواست های VM به طور مداوم به Cloud برسد. یک درخواست VM به این معنی است که مشتری Cloud درخواست یک VM با منابع مورد نیاز را دارد، بطوریکه هر یک از نیازهای منابع شامل سه نوع منبع می شود : پردازنده ، حافظه و دیسک . آزمایشات نشان می دهند که قدرت PM هنوز به مقدار بالایی وجود دارد ، نزدیک به حداکثر قدرت، اگر چه بار کاری (Workload) کم است. بنابراین، از نقطه نظر صرفه جویی در انرژی، لازم است تعداد PM ها را به حداقل برسانیم مادامیکه تمام درخواست های VM در Cloud را می توان در زمان اجرا تضمین کرد.

مشکل جایگذاری VM را می توان به عنوان یک مسئله بسته بندی یکپارچه بردار چند بعدی (Ndimensional vector) در نظر گرفت که یک مسئله NPcomplete است. یک استراتژی ساده برای مقابله با درخواست های مداوم VM ، انتخاب یک PM برای میزبانی VM با شیوه ای حریصانه است ، شبیه به کاهش اولویت تناسب (FFD) است. اما، پدیده ای که "نشت (ریزش) منابع" نامیده می شود، ممکن است ناشی از استفاده نامتجانس از منابع باشد .اگر هر نوع از منابع تمام شده باشند، بار کاری بالایی برای PM به همراه خواهد داشت و PM دیگر نمی تواند بیش از این میزبان VM باشد.

برای یک PM ، اگر برخی از انواع منابع به اتمام برسند ، در حالی که انواع دیگر منابع هنوز مازاد هستند، منابع مازاد هدر رفته است زیرا بار کاری PM در حال حاضر به طور کامل پر شده است .ما این رخداد را " نشت منابع" می نامیم .بنابراین، این یک الگوریتم عالی برای اتخاذ شیوه ی حرص و طمع نیست، نشت منابع رخ می دهد و نتیجه آن بهره وری پایین از منابع خواهد بود . به غیر از الگوریتم شیوه حریصانه ، ما یک الگوریتم متوازن را پیشنهاد می کنیم که یک PM را با اصل اجتناب از نشت منابع در هنگام دریافت یک درخواست VM انتخاب و یا روشن می کند. با استفاده از روش موازنه بهره وری بالا از منابع می تواند حاصل شود. هنگامی که یک درخواست VM می رسد ، PM ای که محدودیت های زیر را رعایت کند، برای میزبانی VM جدید انتخاب می شود :

- منابع موجود برای میزبانی VM جدید کافی است.
- پس از میزبانی VM جدید، نشت منابع ایجاد نخواهد شد.
- اگر VM جدید را میزبانی می کند، دارای مناسب ترین الگوی استفاده از منابع باشد .

اگر هیچ PM ای به دلیل محدودیت منابع یا اجتناب از نشت منابع انتخاب نشده باشد ، یک PM جدید برای میزبانی این VM جدید شروع به کار خواهد کرد . در مقایسه با الگوریتم حریصانه ، الگوریتم موازنه در نتیجه کاهش تعداد PM ها و کاهش مصرف برق ، تضمین بهره وری بالاتری از منابع را به ما خواهد داد . در این مقاله ، ما مسئله جایگذاری مداوم VM را از نقطه نظر صرفه جویی در انرژی بررسی می کنیم.

مشارکت اصلی کار ما در این مقاله می تواند به شرح زیر خلاصه شود :

۱) ما مسئله جایگذاری مداوم مجازی ماشین را در Cloud ها بالا می بریم.

۲) ما یک الگوریتم برخط را با اصل اجتناب از نشت منابع برای افزایش بهره وری از منابع پیشنهاد می کنیم.

۳) آزمایشات نشان می دهد که الگوریتم ما کارایی قابل توجهی در صرفه جویی در مصرف انرژی دارد.

بقیه مقاله به شرح زیر است : بخش دوم بررسی کارهای مرتبط است. بخش سوم مقدمات را معرفی می کند. شرح مسئله در بخش چهارم آمده است و الگوریتم ما در بخش پنجم پیشنهاد شده است. بخش ششم نتایج تجربی را ارائه می دهد. سرانجام، مقاله در بخش هفتم نتیجه گیری می شود.

۲) کارهای مرتبط

همانطور که در بخش اول اشاره شد، جایگذاری ماشین های مجازی (VMs) در یک خوشه ای از ماشین های فیزیکی (PMs) یک وظیفه اصلی در Cloud ها تحت ظهور تکنولوژی مجازی سازی است . عوامل مختلفی در فرآیند جایگذاری VM مورد توجه قرار گرفتند. [۳] یک نمای کلی سطح بالا از جایگذاری VM را ارائه می دهد و یک طراحی معماری سیستم جایگذاری VM را با اتخاذ جایگذاری خودکار VM برای دستیابی به اهداف صرفه جویی بهینه در هزینه های منابع محاسباتی ارائه می دهد ، لیکن فاقد سیاست مدیریت منابع است.

شبکه ، ترافیک و یا پهنای باند عواملی هستند که بسیاری از محققان را مجذوب خود می کنند.[۲۱] جایگذاری VM ها را با روش آگاهانه ترافیک بهینه می کند. الگوهای ترافیکی در میان VM ها با فاصله ارتباطی بین آنها هماهنگ شده اند و VM ها با استفاده از پهنای باند دوطرفه به PM های مجاور اختصاص یافته اند. [۲۲] شرایط شبکه را در نظر می گیرد تا زمان صرف شده جهت انتقال اطلاعات را به حداقل برساند و عملکرد برنامه را حفظ کند. برای رسیدن به عملکرد بهتر ، مهاجرت زنده را در کار خود اتخاذ نموده اند . به همین ترتیب، [۱۱] تنظیم VM را با استفاده از محدودیت پهنای باند شبکه در قالب یک مسئله بسته بندی تصادفی تثبیت می کند.

راه حل بهینه ای ارائه شده است ، تعداد PM های مورد نیاز $(1+\epsilon)(\sqrt{2}+1)$ از PM های بهینه به ازای هر $\epsilon > 0$ است . مسئله نگران کننده در [۲۰] ، در دسترس بودن است ، نویسندگان یک ویژگی جدید دسترس پذیری در VM را تعریف می کنند. تا زمانی که تعداد خرابی PM ها به میزان K است ، تضمین می شود که یک VM را می توان بدون جابجایی سایر VM ها بر روی یک PM سالم منتقل کرد ، زمانی که یک VM به عنوان k-resilient مشخص شده است. در [۱۸] از تکنیک نمایه سازی (profiling technique) برای بهبود بهره وری در فرآیند جایگذاری VM استفاده شده است. بین کارایی نرم افزار و استفاده از منابع ، اختلاف نظر وجود دارد. عمل جایگذاری VM به طور گسترده تحت محدودیت های مختلف و اهداف متفاوت مورد بررسی قرار می گیرد. با این حال، در اکثر مواقع VM ها را به صورت استاتیک قرار می دهند، که کاملا متفاوت از جایگذاری مداوم VM در این مقاله است.

ارائه منابع انعطاف پذیر یکی دیگر از موضوعات مهم مورد بررسی در موضوع محاسبات ابری است. محققان ، کارهای بسیاری در این زمینه انجام داده اند، به عنوان مثال [۱۷]، [۱۶]، [۱۵]، [۱۳]. علاوه بر این ، پلت فرم تجاری مثل آمازون EC2 ([۱]) ظرفیت محاسباتی متغیری را در Cloud ارائه می دهد . جهت دستیابی به قابلیت ارتجاعی ، مهاجرت زنده VM همیشه در سیستم های Cloud استفاده می شود ([۲۲]، [۲۶]، [۲۷]، [۱۲]) .

در خصوص مفهوم این مقاله، VM های میزبانی شده در PM ها جدا شده اند و برنامه های کاربردی در VM ها با ویژگی های بسیار حساس تاخیر مستقر شده اند و برنامه ها کاملا مختل نیستند. به همین دلیل، مهاجرت زنده برای برآورده کردن QoS (کیفیت خدمات) و SLA (توافق سطح خدمات) ممنوع است. در روش ایستا (static) ، جایگذاری VM ها بر روی PM ها احتمالا یک نگاشت بین VM ها و PM ها است .

با وجود محدودیت های مختلف در مورد مسئله جایگذاری ، الگوریتم های جایگذاری پایه بر اساس همان پیش شرط اولیه هستند. در این مقاله ، ما با مسئله جایگذاری مداوم VM روبرو هستیم. به غیر از جایگذاری VM یه طوراستاتیک ، درخواستهای VM به طور مداوم به Cloud رسیده و یک PM باید برای میزبانی VM جدید در زمان اجرا انتخاب شود.

در این مقاله ، ما با مسئله جایگذاری مداوم VM روبرو هستیم که کاملا متفاوت از جایگذاری قبلی استاتیک VM است. هدف جایگذاری به حداقل رساندن تعداد PM ها برای میزبانی VM ها و سپس توانایی کاهش مصرف برق می باشد. ما یک الگوریتم بر خط را پیشنهاد می کنیم تا با بهینه کردن بهره وری منابع، مصرف انرژی کم شود.

۳) مقدمات

در این بخش، ما برخی از مقدمات مسئله جایگذاری VM به صورت ایستا و مسئله جایگذاری مداوم VM را ارائه می کنیم. در ابتدا ، ما پیش شرط مسئله و نشانه های فرموله شده را ارائه می دهیم و سپس محدودیت های منابع داده می شود. در نهایت، ما مسئله جایگذاری استاتیک VM را رسمیت می دهیم و دشواری آن را با کاهش مسئله بسته بندی بردار ۳ بعدی ثابت می کنیم .

3-1) پیش شرط

یک پلت فرم Cloud ای را تصور کنید که سرویس ارائه قابلیت محاسباتی را پیشنهاد می دهد. Cloud درخواست های VM از مشتریان را قبول می کند و VM هایی با منابع مورد نیاز به آنها اختصاص می دهد . مشتریان برنامه های خود را در VM ها نصب می کنند و برنامه های کاربردی حساس به تاخیر هستند و نمی توانند متوقف شوند . بنابراین ، مهاجرت زنده به منظور محافظت از VM ها از وقفه ممنوع است . این امر برای تضمین QoS (کیفیت خدمات) و SLA (موافقت نامه سطح خدمات) اهمیت دارد. از دیدگاه ارائه دهنده زیرساخت (یا مالک Cloud) ، کاهش تعدادی از PM های در حال اجرا جهت کاهش مصرف برق عمل شایسته ای است ، در حالی که درخواست های مداوم VM در زمان واقعی انجام می شود. برای هر درخواست VM، برخی از PM ها برای میزبانی درخواست VM جدید انتخاب شده اند.

به طور خلاصه، ما فرضهای زیر را ارائه می دهیم :

• PM ها در Cloud در ظرفیت محاسباتی هم ارز هستند .

• درخواست های VM به طور مداوم به Cloud می رسند، در هر شکاف زمانی (time slot) ، تنها یک درخواست وجود دارد.

• مهاجرت زنده برای اطمینان از QoS و SLA ممنوع است.

• VM های میزبان در PM ها برای ایمنی بیشتر ایزوله شده اند.

• پهنای باند شبکه داخلی Cloud برای ترافیک های شبکه بین VM ها کافی است.

• هر PM یا VM شامل سه نوع منبع است : پردازنده، حافظه و دیسک . در اینجا CPU به عنوان برش زمانی CPU قابل استفاده در نظر گرفته شده و منابع هر VM نیز از یکدیگر جدا و ایزوله شده است.

۲-۳) یادداشت ها (نشانه ها)

ما در اینجا نشانه های مربوط به نوع منبع را فرموله می کنیم . همانطور که در بالا ذکر شد ، سه نوع منبع در نظر گرفته شده است و ما از C ، M و D به ترتیب برای اشاره به CPU ، Memory و Disk استفاده می کنیم. همچنین یک عبارت انتزاعی R برای اشاره به هر نوع منبع معرفی شده است. به طور خلاصه، آن را می توان به صورت $R \in \{C, M, D\}$ فرموله کرد.

بر اساس این نشانه ها، ظرفیت منابع هر PM را به صورت $\{C_{cap}, M_{cap}, D_{cap}\}$ تعریف می کنیم که می تواند به صورت R_{cap} به طورخلاصه بیان شود.

برای هر PM ، منابع به دو بخش تقسیم می شوند :

هزینه مدیریت و منابع موجود برای VM ها . بخش مدیریت هزینه برای مدیریت PM استفاده می شود. منابع موجود به معنای منابعی است که می تواند به VM اختصاص یابد. هزینه مدیریت و منابع موجود به ترتیب به صورت R_{cost} و R_{avl} نشان داده شده اند . جایی که R دارای معنی مشابه با توضیح قبلی است. به عنوان مثال، M_{cost} به معنای "حافظه" بخشی از هزینه مدیریت است، و D_{avl} به معنای فضای دیسک قابل دسترس در حال حاضر است.

همانند PM ، هر VM شامل سه نوع از منابع است. همانطور که در بالا ذکر شد، درخواستهای VM به طور مداوم به Cloud می رسند ، در حالی که هر درخواست VM دارای نیازهای منابع متفاوت است. ما نیاز به منابع درخواست VM را به عنوان R_{req} در نظر می گیریم . به طور مثال ، درخواست *i* ام نیاز به منابع VM باید به صورت $\{C^i_{req}, M^i_{req}, D^i_{req}\}$ نمایش داده شود ، با این حال ، ما معمولا از عبارت خلاصه R^i_{req} استفاده می کنیم که نیاز به جایگزینی کامل دارد.

۳-۳) محدودیت منابع

اساسا ، جایگذاری VM فرآیند اختصاص منابع به VM است. با این حال، میزان این منابع برای همه PM ها محدود است. بنابراین، الگوریتم جایگذاری باید محدودیت منابع زیر را برآورده کند :

$$R^j_{cap} = R^j_{cost} + R^j_{avl} \qquad (1)$$

$$R^j_{cap} \geq R^j_{cost} \dotplus \sum_{i=1}^{n} R^i_{req} \qquad (2)$$

در معادله (۱) ، R^j_{cap} به معنای ظرفیت j امین PM است. به همین ترتیب، R^j_{cost} و R^j_{avl} به هزینه مدیریت و منابع قابل دسترس از j امین PM اشاره می کنند.

در معادله (۲) ، n تعداد VM های میزبان در j امین PM است . R^i_{req} به معنای منابع مورد نیاز i امین VM است ، در حالی که n تا از VM ها در j امین PM قرار گرفته بودند .

معادله (۱) بدین معنی است که منبع به دو قسمت تقسیم شده است که در بخش 2-3 ذکر شده است. معادله (۲) اشاره به این نکته دارد که منابع فراهم شده جهت VM ها نباید بیش از منابع موجود باشند . این دو معادله برای همه PM ها مناسب است.

۳-۴) جایگذاری VM به صورت ایستا

بر اساس نمادهای تعریف شده فوق وبا در نظر گرفتن محدودیتهای منابع ، بگذارید مسئله جایگذاری استاتیک VM را در اینجا تعریف کنیم :

Static VM Placement. (S-VMP)
Given the following input:

1) *A set of VMs* VM *with resource requirement* $\overrightarrow{VM_i} = \{C^i_{req}, M^i_{req}, D^i_{req}\}$, $1 \leq i \leq n$, *where* n *is the number of VMs.*
2) *A set of PMs* PM *with resource capacity* $\overrightarrow{PM_j} = \{C^j_{cap}, M^j_{cap}, D^j_{cap}\}$, $1 \leq j \leq m$, *where* m *is the number of PMs.*

Subject to:

- $R^j_{cap} = R^j_{cost} + R^j_{avl}$
- $R^j_{cap} \geq R^j_{cost} + \sum_{k=1}^{H_j} R^k_{req}$

Where H_j *is the number of VMs hosted in the* j^{th} *PM.*

Output:

Find a feasible placement $\mathbb{P}(V, P)$ *that satisfies the resource constrains.* $\mathbb{P}(V, P) = \{(i, j) | 1 \leq i \leq n, 1 \leq j \leq m\}$, *where* n *and* m *are the number of VMs and PMs respectively. The two-tuples* (i, j) *means that the* i^{th} *VM will be placed on the* j^{th} *PM. The objective of the placement is minimizing the number of PMs.*

مسئله جایگذاری استاتیک VM را می توان به عنوان یک در نظر گرفت . در این مقاله سه نوع منابع مورد توجه قرار گرفته است، در اینجا یک مسئله بسته بندی سه بعدی مطرح شده است. همانطور که مشخص است مسئله بسته بندی بردار *n* بعدی یک مسئله *NP-hard* است. همانند سختی مسئله بسته بندی ، مسئله جایگذاری استاتیک VM نیز یک مسئله *NP-hard* است . پیچیدگی محاسباتی بعدا ثابت خواهد شد ، اما اجازه دهید ابتدا مسئله بسته بندی سه بعدی را فرموله کنیم :

3-dimensional Vector Bin Packing Problem. *Given a set $\mathcal{C}$ of n cuboid with width w_j, height h_j, and depth d_j ($j \in J = \{1, 2, ..., n\}$), and an unlimited number of identical three dimensional containers (can be call "bins") having width W, height H, and depth D. Find a partition (packing) of $\mathcal{C}$ into A_1, A_2,..., A_m such that:*

$$\sum_{j=1}^{n} a_{ij} w_j \leq W$$

$$\sum_{j=1}^{n} a_{ij} h_j \leq H$$

$$\sum_{j=1}^{n} a_{ij} d_j \leq D$$

for all A_i, where $i \in I = \{1, 2, ..., m\}$, and

$$a_{ij} = \begin{cases} 1, & \text{if } j^{th} \text{ cuboid is placed in } i^{th} \text{ bin;} \\ 0, & \text{otherwise.} \end{cases}$$

The objective is to minimize the value of 'm', number of bins or partitions (packings).

همانطور که مشاهده شد پیدا کردن راه حل بهینه مسئله بسته بندی بردار ۳ بعدی به عنوان یک مسئله *NP-hard* شناخته شده است ، سپس ما ثابت خواهیم کرد که مسئله جایگذاری استاتیک VM نیز یک مسئله *NP-hard* است.

اصل ۱) مسئله جایگذاری استاتیک VM یک مسئله *NP-hard* است.

اثبات : ما برای اثبات سختی کار ، مسئله بسته بندی بردار ۳ بعدی را به مسئله جایگذاری استاتیک VM تقلیل می دهیم.

با توجه به یک مجموعه C از مکعب n با عرض w_j ، ارتفاع h_j و عمق d_j ، بطوریکه $(j \in J = \{1, 2, ..., n\})$ و تعداد نامحدودی از جعبه های سه بعدی یکسان با عرض W، ارتفاع H و عمق D.

ما عرض ، ارتفاع و عمق را به ترتیب با پردازنده ، حافظه و دیسک نگاشت می کنیم . بنابراین یک مکعب به عنوان یک VM در نظر گرفته شده است. ما مقدار R_{cost} را یک مقدار ثابت مثبت تعیین می کنیم، سپس منابع موجود PM به عنوان یک جعبه شامل مقادیر C_{cap} ، M_{cap} و D_{cap} در نظر گرفته می شود .

بدیهی است پیدا کردن راه حل بهینه مسئله بسته بندی جعبه سه بعدی معادل است با به حداقل رساندن تعداد PM ها. کاهش به صورت چندجمله ای است. پس با توجه به سختی مسئله بسته بندی جعبه سه بعدی ، ما مسئله جایگذاری استاتیک VM را داریم که خود یک مسئله *NP-hard* است .

۴) بیان مسأله

در این بخش، ما معیار مصرف انرژی را تعریف می کنیم و پس از آن فرمول مسئله جایگذاری مداوم VM در توان مصرفی تحت تنظیم VM تعریف خواهیم نمود که مهاجرت زنده در این شرایط ممنوع شده است. سپس ما ثابت می کنیم که مشکل جایگذاری مداوم VM خود یک مسئله *NP-hard* است .

۴-۱) مصرف برق

هدف ما از این کار کاهش مصرف برق است. برای هر PM ، مصرف انرژی عمدتا توسط استفاده از پردازنده تعیین می شود. این موضوع را می توان به شکل تقریبی به صورت زیر تعریف کرد :

$$power = a * U + b,$$

که در آن U نسبت مصرف CPU است. آزمایشات نشان می دهد که b یک مقدار بسیار بزرگتر از a است. این بدین معنی است که با وجود شباهت بسیار CPU در میزان مصرف انرژی ، مادامیکه PM فعال باشد ، استفاده از CPU متفاوت خواهد بود. بنابراین، در مسئله جایگذاری استاتیک VM ، مصرف برق متناسب با تعداد PM های در حال اجرا است ، که می توان آن را به صورت زیر تعریف نمود :

$$Power = \alpha * m,$$

جایی که m تعداد PM های در حال اجرا است ، α ضریب مصرف انرژی یک PM در یک اسلات زمانی است. جدای از جایگذاری استاتیک VM ، در سناریوی جایگذاری مداوم VM ، همراه با درخواست هایی که توسط VM به طور مداوم به Cloud می رسد ، تعداد PM ها متغیر خواهند بود . ما از معیار زیر را جهت اندازه گیری مصرف انرژی در جایگذاری مداوم VM تعریف می کنیم :

$$Power = \sum_{i=1}^{n} \alpha * m_i$$

جایی که m_i : تعداد PM های در حال اجرا است ، زمانی که i درخواست های VM است که به Cloud رسیده است ، n تعداد درخواست های VM است.

۲-۴)جایگذاری مداوم VM

بر اساس پیش بینی ها و معیار مصرف انرژی ، ما مسئله مصرف انرژی را در جایگذاری مداوم VM را به صورت زیر مطرح می کنیم :

بیان مسأله (جایگذاری مداوم VM)

در اینجا یک Cloud شامل PM های با ظرفیت منابع R_{cap} یکسان داده شده است . Cloud درخواستهای VM بر اساس نیاز به R_{req}. منابع را به طور مداوم پذیرش می کند. برخی از PM ها در زمان اجرا برای میزبانی VM جدید مطابق با برخی سیاست ها انتخاب شده اند ، VM جدید بر اساس نیاز منابع پذیرفته شده ایجاد شده است. محدودیت منابع شبیه به همان وضعیتی است که به ازای هر PM در جایگذاری استاتیک VM وجود داشت .

$$R_{cap} = R_{cost} + R_{avl}$$

$$R_{avl} \geq \sum_{i=1}^{n} a_{ij} R_{req}^{i}, \forall j \in J$$

جایی که n تعداد VM های جاری است و

$$a_{ij} = \begin{cases} 1, & \textit{if } i^{th} \textit{ VM is placed on } j^{th} \textit{ PM;} \\ 0, & \textit{otherwise.} \end{cases}$$

جایی که $j \in J = \{1, 2, ..., m_i\}$ و m_i تعداد PM های در حال اجرا است ، زمانی که i درخواست های VM است که به Cloud رسیده است ، n تعداد درخواست های VM است.

مصرف برق به شرح زیر است:

جایی که m_i : تعداد PM های در حال اجرا است ، زمانی که به i درخواست های VM رسیدگی می شود . توان مصرفی به شرح زیر نمایش داده می شود :

$$Power = \sum_{i=1}^{n} \alpha * m_i$$

هدف ، به حداقل رساندن مقدار توان مصرفی است. از شرح مسئله مشخص است که توان مصرفی $Power$ برابر است با تعداد PM های در حال اجرا هنگامی که n (تعداد VM ها) داده می شود.

ما ثابت خواهیم کرد که این یک مسئله *NP-hard* است که موجب صرفه جویی در مصرف انرژی در روال جایگذاری مداوم VM می شود.

فرضیه : مسئله جایگذاری مداوم VM یک مسئله *NP-hard* است.

اثبات: فرض کنید جمعاً تعداد $2n$ درخواست VM وجود دارد . آنگاه توان مصرفی را می توان به شرح زیر توصیف کرد :

$$\begin{aligned} Power &= \alpha m_1 + \alpha m_2 + ... + \alpha m_{2n} \\ &= \alpha(\sum_{k=1}^{n} m_k + \sum_{k=n}^{2n} m_k) \end{aligned}$$

جایی که α ضریب توان مصرفی یک PM است ، m_k تعداد PM هاست زمانی که به تعداد k درخواست VM رسیدگی شود. ما ثابت می کنیم که فرضیه نشان می دهد که این یک حالت خاص از مسئله *NP-hard* است. اجازه دهید که j امین درخواست VM به منابع مورد نیاز $(n \leq j < 2n)$ برابر با منابع موجود PM باشد ، به این معنی که $m_{j+1} = m_j + 1, (n \leq j < 2n)$. بنابراین توان مصرفی ($Power$) باید به صورت زیر باشد :

$$Power^* = \alpha(\sum_{k=1}^{n} m_k + n * m_n + \sum_{i=1}^{n} i)$$

برای به حداقل رساندن مقدار $Power^*$ لازم است که مقدار m_nرا به حداقل برسانیم . در معادله فوق ، m_n بهینه سازی شده ، راه حل مطلوبی جهت جایگذاری استاتیک VM است ، در حالی که به تعداد n تا VM داده شده باشد .

از اصل (۱) ، می دانیم که یافتن حداقل m_n ، یک مسئله NP-hard است. این نشان می دهد که به حداقل رساندن $Power^*$ نیز یک مسئله *NP-hard* است. $Power^*$ یک حالت خاص از $Power$ است و ما ثابت کرده ایم

این یک مسئله *NP-hard* برای به حداقل رساندن $Power^*$ است، به طوری که به حداقل رساندن $Power$ نیز یک مسئله *NP-hard* است. با توجه به سختی مسئله ، الگوریتم های ابتکاری (*heuristic algorithms*) در بخش بعدی پیشنهاد می شود.

۵) الگوریتم ما

در این بخش ، الگوریتم ما برای حل مسئله کاهش توان مصرفی در هنگام جایگذاری مداوم VM پیشنهاد می شود . با توجه به سختی کار که در بخش ۲-۴ ثابت شد ، دو الگوریتم ابتکاری پیشنهاد شده است. هنگامی که Cloud درخواست VM مبنی بر نیاز به منابع $R_{req.}$ دریافت کرد ، VM جدید روی PM در زمان اجرا ایجاد می شود . سیاست های مختلفی برای انتخاب PM برای میزبانی این VM وجود دارد. ما دو الگوریتم، الگوریتم حریصانه و الگوریتم موازنه را مطابق دو سیاست مختلف پیشنهاد می کنیم. الگوریتم حریصانه در ابتدا معرفی شده است ، که می تواند راه حل محلی مطلوبی را ارائه دهد ، در حالی که ممکن است راه حل خوبی برای دراز مدت نباشد . نشت منابع در فرآیند جایگذاری به روش حریصانه رخ می دهد، سپس یک الگوریتم موازنه برای جلوگیری از نشت منابع ارائه می شود .

۵-۱) الگوریتم حریصانه

ما یک الگوریتم بهینه محلی را به شیوه حریصانه برای جایگذاری VM در الگوریتم ۱ نشان می دهیم. هنگامی که یک درخواست VM به Cloud می رسد ، ابتدا یک اسکن جهت ازریابی PM ای که دارای منابع کافی موجود برای میزبانی VM باشد ، انتخاب خواهد شد. برای جلوگیری از اسکن غیر ضروری، نباید تمامی PM ها اسکن شوند. بنابراین ما برای هر PM ، دو حالت را معرفی می کنیم : کاملا بارگذاری شده (*fully loaded*) و آماده به کار (*standby*) .

برای یک PM ، اگر استفاده از منابع بیشتر از یک حد آستانه برای هر نوع منبعی باشد ، ما فکر می کنیم PM دیگر نمی تواند بیش از این یک VM جدید را میزبانی کند ، در این حالت PM به لیست کاملا بارگذاری شده ها (*fully loaded*) اضافه می شود. ما PM ها را فقط در لیست آماده به کار اسکن می کنیم.

در حالی که Cloud درخواست VM مبنی بر منبع R_{req} مورد نیاز را قبول می کند ، مجموعه P_{std} شامل PM های آماده به کار اسکن می شود. اگر منابع موجود در i امین PM (P_{std}) در دسترس باشد . R_{avl} برای میزبانی VM جدید کافی است وPM مورد نظر انتخاب و VM جدید روی PM ایجاد خواهد شد. برای PM انتخاب شده، استفاده از منابع μ_R پس از میزبانی VM جدید تغییر می کند؛ اگر مقدار جدید μ_R بیشتر از آستانه θ_{full} باشد ، PM در وضعیت کاملا بارگذاری شده است و به مجموعه کاملا بارگذاری شده ها P_{full} اضافه می شود و از مجموعه آماده به کار P_{std} حذف می شود.

اگر هیچ PM نمی تواند درخواست VM را پس از اسکن تمامی PM های $P_{std}.size$ در P_{std} برآورده کند ، یک PM جدید به نام P_{new} آغاز به کار کند و VM جدید در PM جدید ایجاد خواهد شد. به همین ترتیب ، استفاده از منابع μ_R به روز می شود و PM جدیدی که شروع به کار کرده به عنوان PM کاملا بارگذاری شده و یا PM آماده به کار براساس میزان استفاده از منابع طبقه بندی می شود.

بنابراین، توان مصرفی $Power$ در اسلات زمان بعدی ، توسط تعداد فعلی $(P_{all}.size)$ تمام PM ها یعنی P_{all} ، از جمله PM های کاملا بارگذاری شده و یا PM های آماده به کار تعیین می شود. بازای هر PM ، توان مصرفی در یک اسلات زمانی α است. توان مصرفی $Power$ در هر اسلات زمانی در هنگام رسیدگی به یک درخواست VM ذخیره می شود.

Algorithm 1 Greedy Algorithm

Input: Continuous VM requests REQ (one at each time slot) with resource requirement R_{req}.

Output: Power consumption $Power$ during the process of continuous VM placement.

$Power = 0$;
while $REQ \; != NULL$ **do**
  $selected = false$;
  for $i = 1; i \leq P_{std}.size; i = i + 1$ **do**
    if $P^i_{std}.R_{avl} \geq R_{req}$ **then**
      $selected = true$;
      create a new VM on P^i_{std};
      update μ_R of P^i_{std};
      if $\mu_R \geq \theta_{full}$ **then**
        $P_{full} = P_{full} \cup \{P^i_{std}\}$;
        $P_{std} = P_{std} - \{P^i_{std}\}$;
      end if
      $break$;
    end if
  end for
  if $selected = false$ **then**
    start a new PM P_{new};
    $P_{all} = P_{all} \cup \{P_{new}\}$;
    create a new VM on P_{new};
    update μ_R of P_{new};
    if $\mu_R \geq \theta_{full}$ **then**
      $P_{full} = P_{full} \cup \{P_{new}\}$;
    else
      $P_{std} = P_{std} \cup \{P_{new}\}$;
    end if
  end if
  $Power = Power + \alpha * P_{all}.size$;
end while
return $Power$;

۲-۵) الگوریتم موازنه

با توجه به استفاده کم از منبع ناشی از نشت منابع در الگوریتم حریصانه ، ما یک الگوریتم موازنه (که در الگوریتم ۲ نشان داده شده است) برای افزایش استفاده از منابع با اجتناب از نشت منابع پیشنهاد می کنیم. در حالی که یک درخواست VM وجود دارد ، چهار حالت بالقوه به شرح ذیل برای هر PM وجود دارد :

۱) *Resource Limited* : منابع موجود PM برای ایجاد یک VM جدید با توجه به نیاز به منابع R_{req} کافی نیست.

۲) *Fully Loaded PM* : بعد از تامین درخواست VM به حالت کاملا بارگذاری شده می رود.

۳) *Resource-Leak* : استفاده منابع غیر عادی است که منجر به نشت منابع می شود اگر این PM میزبان VM جدید ایجاد شده باشد.

۴) *Standby* : استفاده از منابع زمانی که VM جدید در این PM ایجاد می شود، هنوز هم صحیح و قانونی است.

در اینجا ، ما تعریف کمی از نشت منابع را ارائه می دهیم. یک PM در حالت نشت منابع وجود دارد اگر R ای وجود داشته باشد به نحوی که $(\mu_R > \theta_\Lambda)$ و $(|\max\{\mu_R\} - \min\{\mu_R\}| \geq \theta_\Delta)$، جایی که μ_R اشاره دارد به استفاده از منابع PM، θ_Λ و θ_Δ دو آستانه برای اندازه گیری مناسبت زمانی که نشت منابع رخ می دهد، $\max\{\mu_R\}$ و $\min\{\mu_R\}$ به ترتیب خلاصه عبارات $\max\{\mu_R|R \in \{C, M, D\}\}$ و $\min\{\mu_R|R \in \{C, M, D\}$ هستند.

به طور خلاصه ، اگر شکاف بزرگ (θ_Δ) بین برخی از بهره وری منابع وجود داشته باشد ، در حالیکه یکی از آنها برای تامین درخواست دیگری از VM کافی نیست . این وضعیت باعث نشت منابع می شود . به غیر از الگوریتم حریصانه، هر PM در PM_{std} اسکن شده است و در چهار حالت بالقوه طبقه بندی شده اند . ابتدا تعیین می شود که آیا منبع P^i_{std} در دسترس هست یا نه؟

Algorithm 2 Balanced Algorithm

Input: Continuous VM requests REQ (one at each time slot) with resource requirement R_{req}.

Output: Power consumption $Power$ during the process of continuous VM placement.

$Power = 0$;
while $REQ\ != NULL$ **do**
  $f = 0, \quad \overline{\mu_R^*} = 0$;
  $s = 0, \quad \sigma(\mu_R^*) = \infty$;
  for $i = 1; i \leq P_{std}.size; i = i + 1$ **do**
    if $P_{std}^i.R_{avl} \geq R_{req}$ **then**
      if $\overline{\mu_R'} \geq \theta_{full}$ **then**
        if $\overline{\mu_R'} > \overline{\mu_R^*}$ **then**
          $\overline{\mu_R^*} = \overline{\mu_R'}$;
          $f = i$;
        end if
      end if
      if $j = 0$ **then**
        if $(\exists R, \mu_R' > \theta_\Lambda)$ && $(|\max\{\mu_R'\} - \min\{\mu_R'\}| \geq \theta_\Delta)$ **then**
          continue;
        else if $\sigma(\mu_R') < \sigma(\mu_R^*)$ **then**
          $\sigma(\mu_R^*) = \sigma(\mu_R')$;
          $s = i$;
        end if
      end if
    end if
  end for
  if $j > 0$ **then**
    create a new VM on P_{std}^j;
    $P_{full} = P_{full} \cup \{P_{std}^i\}$;
    $P_{std} = P_{std} - \{P_{std}^i\}$;
  else if $s > 0$ **then**
    create a new VM on P_{std}^s;
  else
    start a new PM P_{new};
    $P_{all} = P_{all} \cup \{P_{new}\}$;
    create a new VM on P_{new};
    update μ_R of P_{new};
    if $\mu_R \geq \theta_{full}$ **then**
      $P_{full} = P_{full} \cup \{P_{new}\}$;
    else
      $P_{std} = P_{std} \cup \{P_{new}\}$;
    end if
  end if
  $Power = Power + \alpha * P_{all}.size$;
end while
return $Power$;

R_{avl} برای ایجاد VM درخواستی کافی است ، در غیر اینصورت ، منابع PM محدود است و در لیست مربوط به گروه [resource restricted] قرار دارد . اگر PM بتواند VM جدید را میزبانی کند ، الگوریتم تصمیم می گیرد در چه حالتی PM بر حسب μ'_R تعلق می گیرد ، که اشاره دارد به ترتیب استفاده از منبع ، اگر PM براساس درخواست VM منبع اختصاص دهد .

همانطور که در الگوریتم ۱ نشان داده شده است ، اگر μ'_R بیشتر از آستانه θ_{full} باشد ، PM به عنوان [کاملا لود شده] طبقه بندی می شود . الگوریتم در تصمیم گیری حالات [resource-leak] و [standby] عبور می کند و آن ها را نادیده می گیرد ، در صورتی که برخی از PM ها در حالت [fully loaded] قرار داشته باشند. اگر هنوز هیچ PM ای در حالت [fully loaded] وجود نداشته باشد ، الگوریتم قضاوت می کند که آیا این جایگذاری باعث نشت منابع خواهد شد یا نه ؟ طبق تعریف نشت منبع ، PM به دو حالت [resource-leak] و [standby] طبقه بندی می شود .

اگر تعداد PM های در حالت [fully loaded] صفر نباشد ، یکی از آنها که دارای حداکثر میانگین مقدار μ'_R از استفاده آتی از منابع می باشند ، به عنوان ارائه دهنده منبع انتخاب می شود . در غیر این صورت ، اگر هیچ PM ای در حالت [fully loaded] وجود نداشته باشد ، یکی از آنها که دارای حداقل واریانس $\sigma(\mu'_R)$ در حالت [standby] برای تامین منابع مورد نیاز انتخاب می شود . اگر هیچ PM ای یکی از دو حالت [resource-leak] و [standby] وجود نداشته باشد ، یک PM جدید شروع به کار خواهد کرد .

جزئیات در الگوریتم ۲ نشان داده شده است، دو نشانه f و s به شرح زیر توضیح داده شده است :

f ضبط موقعیت PM با حداکثر $\overline{\mu'_R}$ در P_{std} ، در حالی که متغیر μ^*_R برای ضبط حداکثر مقدار فعلی μ'_R استفاده می شود ، جایی که $\overline{\mu'_R}$ به مقدار میانگین $\{\mu'_R|R \in \{C, M, D\}\}$ اشاره می کند .

s ضبط موقعیت PM با حداقل $\sigma(\mu'_R)$ در P_{std} ، در حالی که متغیر $\sigma(\mu^*_R)$ برای ضبط حداقل مقدار فعلی $\sigma(\mu'_R)$ استفاده می شود ، جایی که $\sigma(\mu'_R)$ به مقدار میانگین $\{\mu'_R|R \in \{C, M, D\}\}$ اشاره می کند .

قابل توجه است که اگر $f > 0$ باشد ، مقدار s بی فایده است . علامت های دیگر مانند P_{all} ، P_{full} ، P_{std}، P_{new} و $Power$ در ظاهر از مفهوم یکسانی درالگوریتم ۱ برخوردار می باشند . نکته کلیدی الگوریتم موازنه ، متفاوت از الگوریتم حریصانه ، نوع نگرش آن در خصوص استفاده از منابع μ_R است . الگوریتم موازنه بهره برداری بیشتر از منابع را با شروع به کار یک PM جدید در لحظه مناسب برای جلوگیری از نشت منابع تضمین می کند .

۶) ارزیابی تجربی

در این بخش ، از آزمایشات شبیه سازی برای ارزیابی عملکرد الگوریتم استفاده می کنیم .الگوریتم حریصانه به عنوان یک معیار عمل می کند ، ما الگوریتم موازنه را با آن مقایسه می کنیم . ما فرض می کنیم درخواست های VM به طور منظم به Cloud می رسد و در هر اسلات زمانی ، تنها یک درخواست وجود دارد .در واقع ، توزیع ورود درخواست ، تأثیری بر روند تخصیص منابع و نشت منابع بالقوه آن ندارد . هزینه مدیریت R_{cost} مربوط به PM در شبیه سازی در نظر گرفته نمی شود ، درک این موضوع راحت است که هزینه فرض شده یک مقدار ثابت برای PM ها با ظرفیت یکسان است ، تمام منابع را می توان به VM ها اختصاص داد.

سه نوع منبع در نظر گرفته شده است : پردازنده ، حافظه و دیسک. ما بر روی دو سوال تمرکز می کنیم :

۱) تعداد PM ها زمانی که درخواست های VM به طور مداوم به Cloud می رسند ، تغییر می کند؟

۲) چگونه میزان توان مصرفی در شرایط یکسان متفاوت است؟

بنابراین دو روش برای اندازه گیری عملکرد الگوریتم ها استفاده می شود : تعداد PM ها و توان مصرفی .

۱-۶) نصب شبیه ساز

در شبیه سازی ها ، سه نوع منبع مستقل از یکدیگر هستند . ما دو نوع از توزیع مقدار منابع مورد نیاز را فرض می کنیم : توزیع یکنواخت و توزیع نرمال . سه مجموعه داده با پارامترهای مختلف برای توزیع یکنواخت مورد بررسی قرار گرفته و یک مجموعه داده برای توزیع نرمال وجود دارد . موارد در جدول ۱ نشان داده شده است. برای هر PM ، مقدار منابع موجود در R برابر ۱۵۰ است . برای هر یک از چهار مجموعه داده ها ، نمونه های زیادی با همان توزیع و تنظیم پارامتر وجود دارد . نتیجه تجربی به مقدار متوسط برای هر مجموعه داده اشاره دارد.

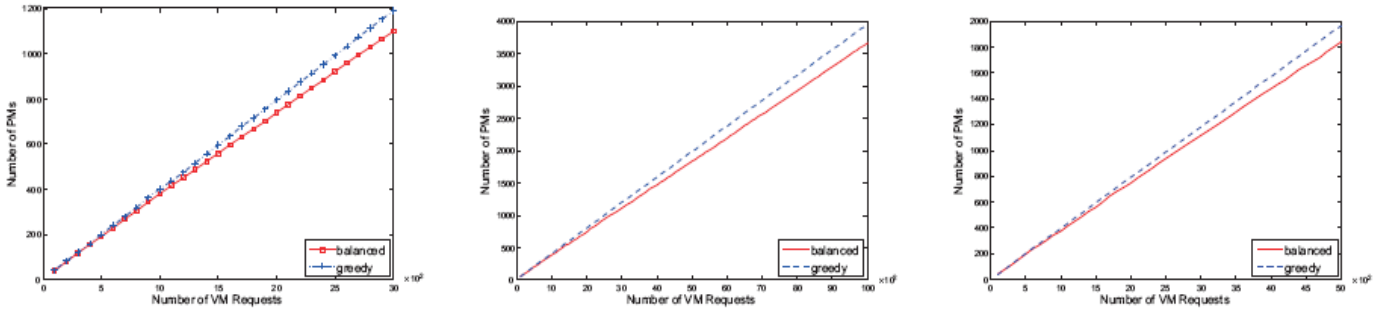
TABLE I
DISTRIBUTIONS OF DATA SETS

Data Set	Distribution	Sample Size
DS_1	*uniform distribution* $\min. = 25$ $\max. = 75$	3,000
DS_2	*uniform distribution* $\min. = 20$ $\max. = 80$	10,000
DS_3	*uniform distribution* $\min. = 15$ $\max. = 85$	5,000
DS_4	*normal distribution* $\mu = 50$ $\sigma = 12$	10,000

۲-۶) نتایج تجربی

شکل ۱ نتایج تعدادی از PM ها و مصرف انرژی را نشان می دهد زمانی که نیاز منابع با توزیع یکنواخت با سه پارامتر متفاوت مواجه می شود . محور افقی هر زیرمجموعه تعداد درخواست های VM است، خط بالا نتایج نتایج مربوط به تعداد PMها را نشان می دهد ، در حالی که نتایج مربوط به توان مصرفی با خط پایین نشان داده شده است .

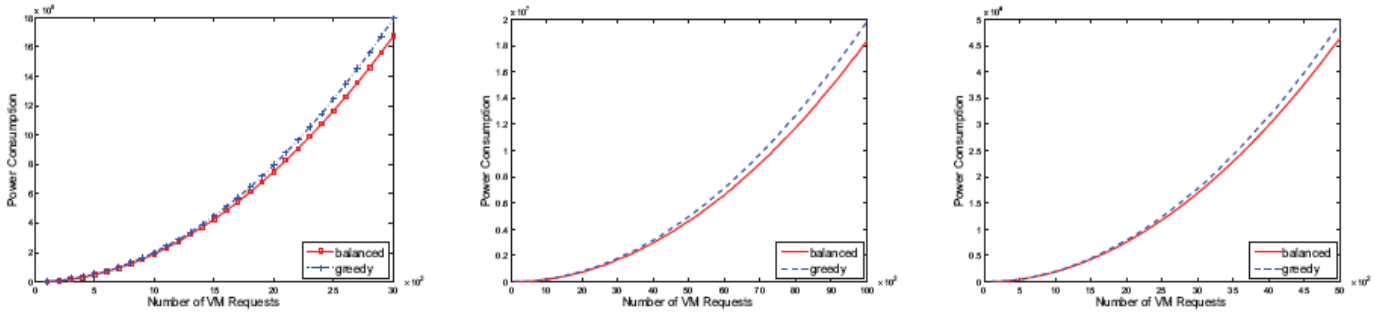
از روی شکل ها بدیهی است که از ارقام الگوریتم موازنه (خط قرمز) بهتر از الگوریتم حریصانه (خط آبی) در به حداقل رساندن تعداد PM ها و مصرف انرژی است .افزایش آهسته تر به این معنا است که برای پاسخگویی به درخواست های مداوم VM نیاز به قدرت پایین دارد . ما متوجه می شویم که تقریبا ۱۰٪ مصرف انرژی در بعضی موارد می تواند ذخیره شود .



(a) number of PMs when the distribution of resource requirement is uniform distribution, and $REQ.size = 3000, \min. = 25, \max. = 75$.

(b) number of PMs when the distribution of resource requirement is uniform distribution, and $REQ.size = 10000, \min. = 20, \max. = 80$.

(c) number of PMs when the distribution of resource requirement is uniform distribution, and $REQ.size = 5000, \min. = 15, \max. = 85$.



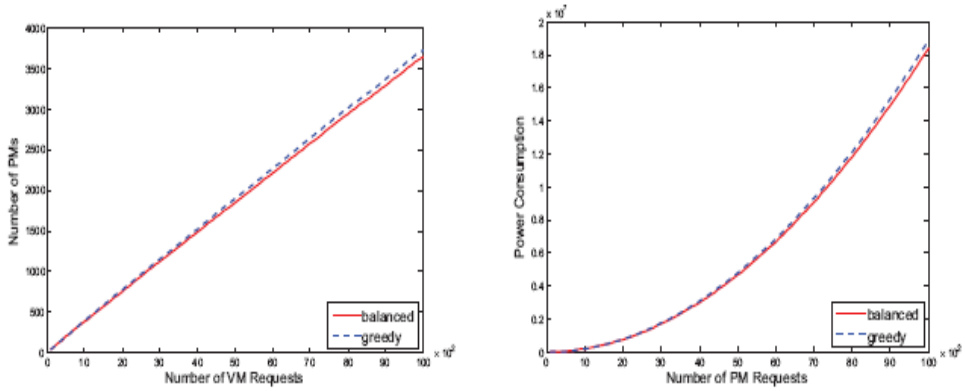
(d) power consumption when the distribution of resource requirement is uniform distribution, and $REQ.size = 3000, \min. = 25, \max. = 75$.

(e) power consumption when the distribution of resource requirement is uniform distribution, and $REQ.size = 10000, \min. = 20, \max. = 80$.

(f) power consumption when the distribution of resource requirement is uniform distribution, and $REQ.size = 5000, \min. = 15, \max. = 85$.

Fig. 1. Simulation results when the distribution of resource requirement is uniform distribution. The top row shows the number of PMs with different request size and range, while the bottom row shows the related power consumption with the same setting.

در حالت توزیع نرمال ، شکل ۲ نتایج را نشان می دهد . به علت تمرکز منابع مورد نیاز ، رخداد نشت منابع کمتر از توزیع یکنواخت وجود دارد . با این حال ، هنوز هم صرفه جویی قابل توجهی در توان مصرفی وجود دارد زمانی که تعداد درخواست های VM بالاست و هنوز هم تاثیر مستقیمی بر روی هزینه های صرفه جویی در مصرف انرژی خواهد داشت ، وقتی که ما بایستی میلیون ها دلار بابت توان مصرفی بپردازیم .



(a) number of PMs when the distribution of resource requirement is normal distribution, and $REQ.size = 10000, \mu = 50, \sigma = 12$.

(b) power consumption when the distribution of resource requirement is normal distribution, and $REQ.size = 10000, \mu = 50, \sigma = 12$.

Fig. 2. Simulation results when the distribution of resource requirement is normal distribution. The upper one shows the number of PMs, while the lower one shows the related power consumption with the same setting.

۷) نتیجه گیری

در این مقاله ، ما مسئله جایگذاری مداوم ماشین مجازی را از لحاظ صرفه جویی در مصرف انرژی مورد بررسی قرار می دهیم . ما آن را از جایگذاری استاتیک ماشین مجازی تشخیص می دهیم و توان مصرفی را برای فرآیند جایگذاری مداوم تعریف می کنیم . ثابت شده است که مسئله *NP-hard* جهت به حداقل رساندن توان مصرفی در Cloud است . ما پدیده ای به نام نشت منابع را که باعث تلف شدن منابع و افزایش مصرف انرژی می شود ، کشف کردیم . الگوریتم موازنه برای جلوگیری از نشت منابع و افزایش استفاده از منابع پیشنهاد شده است .

توان مصرفی را می توان به دلیل کاهش تعداد دستگاه فیزیکی به حداقل رساند . نتایج تجربی نشان می دهد که الگوریتم موازنه به خوبی عمل می کند و می تواند به میزان قابل توجهی مصرف انرژی را کاهش دهد .

منابع :

REFERENCES

[1] Amazon EC2, http://aws.amazon.com/ec2/.

[2] J. Hamilton, "Cost of Power in Large-scale Data Centers," http://perspectivvs.mvdirona.com/, Nov. 2009.

[3] C. Hyser, B. McKee, R. Gardner, and B. J. Watson, "Autonomic Virtual Machine Placement in the Data Center," *HP Labs Technical Report*, February 2008.

[4] M. R. Garey, R. L. Graham and D. S. Johnson, "Resource Constrained Scheduling as Generalized Bin Packing," *Journal of Combinatorical Theory*, vol. 21, no. 3, pp. 257-298, 1976.

[5] M. R. Garey and V. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," *W.H. Freeman and Co.* 1976.

[6] W. F. de la Vega and G. S. Lueker, "Bin Packing can be Solved with $1+\epsilon$ in Linear Time," *Combinatorica*, vol. 1, no. 4, pp. 349-355, 1981.

[7] G. J.Woeginger, "There is no asymptotic PTAS for two-dimensional vector packing," *Information Processing Letters*, vol. 64, no. 6, pp. 293-297, 1997.

[8] R. Bianchini and R. Rajamony, "Power and Energy Management for Server Systems," *IEEE Computer*, vol. 37, 2004.

[9] W. Vogels, "Beyond Server Consolidation," *ACM QUEUE*, vol. 6, no. 1, pp. 20-26, 2008.

[10] S. Martello, D. Pisinger, and D. Vigo, "The Three-Dimensional Bin Packing Problem," *INFORMS*, vol. 48, no. 2, pp. 256-267, March-April 2000.

[11] M. Wang, X. Meng, and L. Zhang, "Consolidating Virtual Machines with Dynamic Bandwidth Demand in Data Center," *In Porc. of INFOCOM 2011 (Mini-Conference).*

[12] H. Liu, H. Jin, X. Liao, C. Yu, and C.-Z. Xu, "Live Virtual Machine Migration via Asynchronous Replication and State Synchronization," *IEEE TPDS*, vol. 3, no. 22, pp. 426-438, 2011.

[13] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A Cost-aware Elasticity Provisioning System for the Cloud," *In Proc. of ICDCS 2011.*

[14] Y.-H. Han, C.-M. Kim, and J.-M. Gil, "A Greedy Algorithm for Target Coverage Scheduling in Directional Sensor Networks," *JoWUA*, vol. 1, no. 2/3, pp. 96-106, 2010.

[15] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "Kingfisher: Cost-aware Elasticity in the Cloud," *In Proc. of INFOCOM 2011 (Mini-conference).*

[16] Z. Gong, X. Gu, and J. Wilkes, "PRESS: PRedictive Elastic ReSource Scaling for cloud systems," *In Proc. of CNSM 2010.*

[17] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: Elastic Resource Scaling for Multi-Tenant Cloud Systems," *In Proc. of SOCC 2011.*

[18] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Y. Zomaya, and B. B. Zhou, "Profiling Applications for Virtual Machine Placement in Clouds," *In Proc. of Cloud Computing 2011.*

[19] J. Rao, X. Bu, K. Wang, and C.-Z. Xu, "Selt-adaptive Provisioning of Virtualized Resource in Cloud Computinng," *In proc. of SIGMETRICS 2011.*

[20] E. Bin, O. Biran, O. Boni, E. Hadad, E. K. Kolodner, Y. Moatti, and D. H. Lorenz, "Guaranteeing High Availability Goals for Virtual Machine Placement," *In Proc. of ICDCS 2011.*

[21] X. Meng, V. Pappas, and L. Zhang, "Improving the Scalability of Date Center Networks with Traffic-aware Virtual Machine Placement," *In Proc. of INFOCOM 2010.*

[22] J. T. Piao, and J. Yan, "A Network-aware Virtual Machine Placement and Migration Approach in Cloud Computing," *In Proc. of GCC 2010.*

[23] M. Lin, A. Wierman, L. H. Andrew, and E. Thereska, "Dynamic Right-sizing for Power-proportional Data Center," *In Proc. of INFOCOM 2011.*

[24] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and Performance Management of Virtualized Computing Environments via Lookahead Control," *Cluster Computing*, Vol. 12, pp. 1-15, 2009.

[25] Y. Shiraishi, M. Mohri, and Y. Fukuta, "A Server-Aided Computation Protocol Revisited for Confidentiality of Cloud Service," *JoWUA*, vol. 2, no. 2, pp. 83-94, 2011.

[26] V. Shrivastava, P. Zerfos, K. Lee, H. Jamjoom, Y-H Liu, and S. Banerjee, "Application-aware Virtual Machine Migration in Data Centers," *In Proc. of INFOCOM 2011 (Mini-conference).*

[27] W. Voorsluys, J. Broberg, S. Venugopal and R. Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," *In Proc. of CloudCom 2009.*