
بسم الله الرحمن الرحيم

دانشکده فنی و مهندسی
کارشناسی ارشد مهندسی کامپیوتر نرم‌افزار
گروه مهندسی کامپیوتر و فناوری اطلاعات

**گزارش درس سمینارعادی**

موضوع:

# کاهش مصرف انرژی (ابر سبز) در مراکز داده های ابری با کمک الگوریتم های فرا ابتکاری و تنظیم کاهش مهاجرت ماشین مجازی

نگارش:

**سید علی محترمی**

استاد راهنما:

**دکتر سید علی رضوی ابراهیمی**

بهمن-اسفند ۱۳۹۹

## چکیده

رشد صعودی دانش و ارتباطات سبب شده است سبک محاسباتی نوینی با عنوان رایانش ابری پدید آید. یکی از بزرگترین مزیتهای مورد توجه تأمین کنندگان زیرساخت ابر، بحث مالی با به حداقل رسانیدن هزینه ها، تضمین قرارداد سطح سرویس و حداکثر سوددهی میباشد. در این راستا مدیریت انرژی در مراکز داده ابری، موضوعی مهم و قابل توجه برای رسیدن به هدف مذکور است. یکی از راه های کاهش مصرف انرژی، آزاد سازی میزبان های بیکار و راه دیگر کاهش تعداد مهاجرت ماشین های مجازی می باشد. بدین منظور، یکی از چالشهای مطرح، انتخاب روش جایگذاری ماشین های مجازی مهاجرت داده شده، بر روی میزبان مناسب است. در این جزوه، با معرفی الگوریتم کاهش تعداد مهاجرت و الگوریتم جایگذاری بر اساس حد آستانه مناسب، به کاهش مصرف انرژی در مراکز داده ابری پرداخته شده است .این الگوریتم با ارث بری از الگوریتم اولین انتخاب و مرتب کردن نزولی میزبان ها و ماشین های مجازی براساس رتبه آنها، که این رتبه براساس بار کاری بدست می آید، عمل جایگذاری را انجام میدهد. راهکار ارائه شده در نرم افزار کلودسیم شبیه سازی شده است. نتایج حاصل از شبیه سازی حاکی از کاهش تعداد مهاجرت ماشین مجازی، افزایش بهره وری مهاجرت وکاهش مصرف انرژی می باشد.



**کلمات کلیدی:** محاسبات ابری سبز، کاهش انرژی مصرفی مراکز داده، تجمیع میزبان ها، کاهش تعداد مهاجرت

عنوان ..... صفحه

صفحه

# فهرست اشکال

**عنوان ..... صفحه**



# فهرست جداول

**عنوان ..... صفحه**



# فهرست علائم اختصاری

| |
|---|
| NIST: National Institute of Standards and Technology |
| IaaS: Infrastructure as a Service |
| PaaS: Platform as a Service |
| SaaS: Software as a Service |
| SLA: Service Level Agreement |
| CDC: Cloud Data Centers |
| ICT: Information and Communication Technologies |
| EPA: US Environmental Protection Agency |
| DPM: Dynamic Power Management |
| SPM: Static Power Management |
| DVFS: Dynamic Voltage and Frequency Scaling |
| DCD: Dynamic Component Deactivation |
| DPS: Dynamic Performance Scaling |
| VMM: Virtual Machine Monitor |
| DFS: Dynamic Frequency Scaling |
| LLC: Limited Lookahead Control |
| FFD: First-Fit Decreasing |
| PABFD: Power Aware Best Fit Decreasing |
| BFD: Best Fit Decreasing |
| MMT: Minimization of Migrations Time |
| HPG: Highest Potential Growth |
| NAS: network attached storage |

# فصـــل اول

# مقـــدمـــه

## ۱-۱ مقدمه

در دنیای امروز غیرممکن است بتوان تصور کرد فردی بدون رایانه زندگی را سپری کند. درواقع همه افراد و در همه سنین روزانه از رایانه استفاده میکنند و رایانه ها در همه مشاغل و مراودات نقش کلیدی بازی میکنند. رشد صعودی دانش و ارتباطات و از همه مهمتر شبکه گسترده اینترنت باعث شده است کارشناسان فناوری اطلاعات پس از سالها کار با شبکه های اینترنت و ایجاد سبکهای محاسباتی متفاوت، تصمیم به ساخت سبک محاسباتی نوینی به عنوان پردازش ابری [۱]، رایانش ابری و یا محاسبات ابری بگیرند (۱)

مفاهیم ابتدایی این فناوری در سال ۱۹۶۰ توسط پروفسور جان مک کارتی بیان شد و در سال ۲۰۰۶ برای اولین بار توسط آمازون ارائه گردید و بدین ترتیب در سالهای بعد شرکتهای دیگر به ارائه خدمات خود بر مبنای این فناوری روی آوردند(۲)

تعاریف بسیاری از این روش محاسباتی شده است. تعریف موسسه ملی استاندارد و فناوری اطلاعات [۲] که رایان ابری را مدلی برای فراهم کردن دسترسی آسان بر اساس تقاضای کاربر از طریق شبکه به مجموعه ای از منابع رایانشی  قابل تغییر و پیکربندی (مانند شبکه، گره ها ، فضای ذخیره سازی،برنامه های کاربردی و...)که این دسترسی بتواند با کمترین نیاز به مدیریت منابع و یا نیاز به دخالت مستقیم فراهمکننده سرویس به سرعت فراهمشده، معرفی کرده است(۳) .فلسفه استفاده از ابر بدین دلیل است که اینترنت مانند ابر در همه جا وجود دارد و نماد انتزاعی برای نمایش نقاط مرزی بین فراهم کنندگان ابرمی باشد. استفاده از رایانش ابری سبب کاهش هزینه ها، ظرفیت ذخیره سازی نامحدود، سهولت در استفاده، مقیاس پذیری[۳] ، قابلیت حمل آسان وغیره شده است(۱).

[1] Cloud computing
[2] National Institute of Standarads and Technology(NIST)
[3] Scalability

با ظهور ابر و رشد درخواست از ساختار آن باعث افزایش چشمگیر مصرف انرژی در مراکز داده شده است. این مصرف بالای انرژی سبب افزایش هزینه ها و کاهش میزان سود برای فراهم کننده ابر شده است. با افزایش دما، کاهش عمر تجهیزات، کاهش قابلیت اطمینان و افزایش دی اکسید کربن، نقض کیفیت سرویس و بدنبال آن نقض توافق نامه سطح سرویس بوجود خواهد آمد و دلایل فوق از نتایج افزایش مصرف انرژی در مراکز داده بوده است (۱و۸)بنابراین در جهت تحقق رایانش ابری سبز[1] باید با کاهش انرژی در مراکز داده تأثیر مخرب عوامل فوق را در محاسبات ابری کاهش داد. یکی از بهترین روش های به کارگرفته شده برای کاهش مصرف انرژی، تکنیک تجمیع ماشین های مجازی و گره ها است (۴و۸)در این روش که در محیط های ابری و مجازی شده به کار می رود، بار کاری (ماشین مجازی) چندین گره فیزیکی بر روی یک گره قرار گرفته و گره کم بار خاموش یا هایبرنت می شود.

بمنظور تجمیع دو چالش وجود دارد: ۱ -انتخاب بهترین ماشین فیزیکی برای قرار دادن ماشین مجازی بر روی آن بگونه ای که حداکثر منابع ماشین فیزیکی و کمترین هدر رفتگی منابع را داشته باشیم. ۲ -اگر عمل نگاشت به درستی انجام نشود منجر به افزایش تعداد مهاجرت ماشین مجازی خواهد شد که افزایش مصرف انرژی را بدنبال دارد چرا که در عمل مهاجرت، پردازنده و پهنای باند درگیر انتقال صفحات حافظه از گره مبدأ به گره مقصد خواهند شد (۴)

بنابراین انتخاب بهترین ماشین فیزیکی برای تخصیص منابع موجود و همچنین کاهش تعداد مهاجرت ماشین مجازی به منظور کاهش مصرف انرژی امری ضروری و هدف این تحقیق می باشد .هدف از تحقیق حاضر کاهش مصرف انرژی با رویکرد حفظ کیفیت سرویس دهی در محیط های رایانش ابری در لایه زیر ساخت به عنوان سرویس با کاهش تعداد مهاجرت می باشد. این تحقیق سعی دارد با استفاده ازبهینه سازی برنامه ریزی تخصیص ماشین های مجازی به ماشین های میزبان، میزبان های اضافی را خاموش و یا هایبرنت کند تا در مصرف انرژی

[1] Grid Computing

صرفه جویی گردد و همین طور با کاهش تعداد مهاجرت ماشین مجازی نه تنها در مصرف انرژی کاهش صورت می گیرد بلکه در کیفیت سرویس دهی و جلوگیری از نقض قرارداد سطح سرویس نیز تأثیر بسزایی خواهد داشت.

### ۱-۲ روش تجزیه و تحلیل اطلاعات

برای آزمایش اهداف کاربردی در محیط ابر مانند کاهش تعداد مهاجرت ماشین مجازی و کاهش انرژی مصرفی در مراکز داده ابر، به دلیل صرف هزینه و زمان زیاد، نمی توان از محیط ابر واقعی استفاده نمود. بنابراین بهترین راه حل، استفاده از ابزار شبیه سازی است که امکان تست را در محیط کاملا کنترل شده و تکرار پذیر و در زمان اندک و هزینه کم امکان پذیر می سازد. یکی از شبیه سازهای رایج در این زمینه، نرم افزار کلودسیم[1] می باشد که تحقیقات تیم توسعه در زمینه ابر تأثیرهای مهمی بر قابلیتهای این شبیه ساز داشته است (۵،۴،۳،۲)از جمله امکانات کلودسیم می توان به پشتیبانی از انواع مدل سازی سیاست تخصیص منابع و شبیه سازی زیر ساخت ابر و مدیریت سرویس های مجازی اشاره نمود.

برتری این شبیه ساز نسبت به شبیه سازهای دیگر مانند گریدسیم و سیم گرید تفکیک لایه های مورد نیاز که در س رایانش ابری استفاده می شود (مانند لایه های زیر ساخت به عنوان سرویس ، بستر به عنوان سرویس و نرم افزار به عنوان سرویس). علاوه بر این در شبیه سازهای مذکور پشتیبانی چندانی از منابع مجازی و مدل سازی مراکز داده با گره های محاسباتی فراوان صورت نگرفته است. با توجه به موارد فوق، ابزار استفاده شده در این تحقیق نرم افزار کلودسیم است که چارچوب شبیه ساز عمومی را بمنظور مدلسازی، شبیه سازی و ارزیابی راهکار پیشنهادی در زیر ساخت ابر فراهم می کند

ساختار تحقیق سایر بخش های تحقیق حاضر بدین شکل تنظیم شده است: ابتدا در فصل دوم به تعریف اصول و مفاهیم پایه ی موضوع می پردازد. سپس در فصل سوم به راهکارهای موجود جهت کاهش مصرف انرژی پرداخته

[1] Cloud sim

می شود و آنها از جنبه های مختلف با هم مقایسه می شوند. فصل پنجم به راه اندازی ابزار کلودسیم می پردازد و درفصل آخر به کدهای به کارگرفته شده در شبیه ساز پرداخته می شود.

# فصـــل دوم

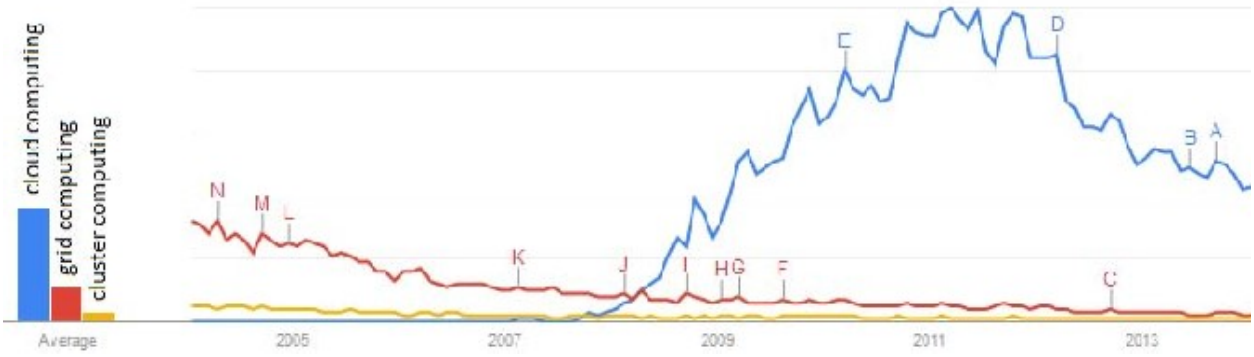
# ادبیات تحقیق

در این فصل تعاریف و مفاهیم پیش زمینه ای را که برای شناخت دقیق مسئله در تحقیق ضروری است، توضیح میدهیم

## ۲-۱مقدمه

سیرتکاملی محاسبات به گونه ای است که میتوان آن را پس از آب، برق، گاز و تلفن به عنوان عنصراساسی پنجم فرض نمود. درچنین حالتی، کاربران سعی میکنند، بر اساس نیازهایشان و بدون توجه به اینکه یک سرویس در کجا قرار دارد و یا چگونه تحویل داده میشود، به آن دسترسی یابند. نمونه های متنوعی از سیستمهای محاسباتی ارائه شده است، که سعی دارند چنین خدماتی را به کاربرانارئه دهند(۳)

برخی ازاین سیستم های محاسباتی عبارتنداز: محاسبات خوشه ای[1] ، محاسبات توری[2] واخیراًمحاسبات ابری[3] که از آن به عنوان رایانش ابری نیز یاد میشود. محبوبیت این سه رویکرد محاسباتی، از دید موتور جستجوی گوگل مورد ارزیابی قرارگرفته است، که نتیجه ی آن در شکل۲-۱ نمایش داده شده است، حاکی از آن است که محبوبیت رایانش ابری، پس از ظهور مفاهیم اولیه آن در سال ۲۰۰۷ ،بافاصله زیادی نسبت به سایر رویکردهای محاسباتی در حال افزایش است.(۵۴)




شکل ۲-۱: بررسی گوگل از مقبولیت سیستمهاي کلاستري، توري و ابري تا سال۲۰۱۳

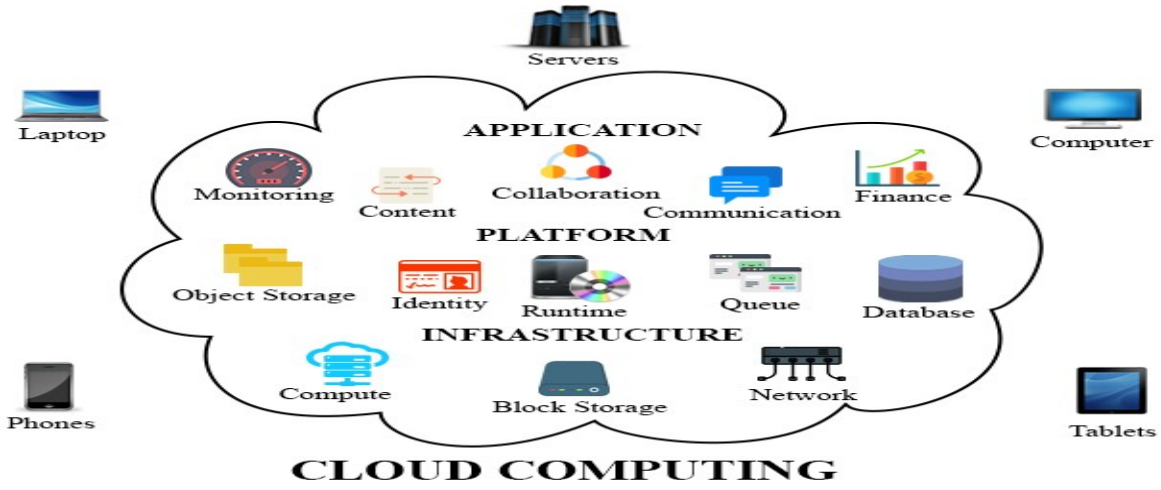
دنیای محاسبات به سرعت به سمت توسعه نرم افزارهایی پیش میرود که به جای اجرا بر روی رایانه های منفرد، به عنوان یک سرویس در دسترس میلیونها مصرف کننده قرار داده میشوند. از این نقطه نظر، رایانش ابری از دید

[1] Cluster Computing
[2] Grid Computing
[3] Grid Computing

کاربران نهایی ساختاری شبیه به یک توده ابر دارد که به واسطه آن میتوانند به برنامه های کاربردی از هرجایی از دنیا دسترسی داشته باشند.(۵۵)

اما رایانش ابری از دید فراهم کنندگان منابع زیرساخت، میتواند با کمک ماشین های مجازی شبکه شده، به عنوان یک روش جدید برای ایجاد پویای نسل جدید مراکز داده، مورداستفاده قرار گیرد تا بتوانند یک زیرساخت قابل انعطاف برای ارائه انواع مختلف خدمات محاسباتی و ذخیره سازی در اختیار داشته باشند(۱۰)

درراپانش ابری از دید ارائه سرویس و برنامه های کاربردی تلاش بر این است، که خدمات اینترنتی به صورت یک رایانه واحد در اختیار تمام کاربرانی که به آن متصل هستند، قرار بگیرد و فنّاوری هایی نظیر وب دو از عوامل مهم در نیل به این هدف هستند. در این رویکرد جدید، میتوان از لایه های مختلف و قابل انعطاف ارائه شده در ابر استفاده کرد.(۱۰) در حقیقت حرکت در ابر به سمتی پیش میرود، که دیگر برای کاربر فرقی نمیکند، در حال استفاده از کدام سایت است. کاربر کل اینترنت را همانند یک رایانه شخصی در خدمت خود مشاهده میکند. شکل زیر یکپارچه سازی خدمات ابر از دید کاربر است(۵).




شکل ۲-۲ :یکپارچه سازي کلیه خدمات ارائه شده ابر از دید کاربر

کاملاً واضح است که در چنین محیطی کار با سایت های مختلف، همانند کار با برنامه های کاربردی مختلف در یک رایانه شخصی میباشد و گرایش به سمتی است که دیگر کاربر مرزی را بین این خدمات احساس نکند و بتواند با همان درجه آزادی که در رایانه شخصی خود دارد، در اینترنت نیز به فعالیت بپردازد. اما اگر بخواهیم به چنین سطحی از انعطاف پذیری در ارائه خدمات کاربردی به کاربران دست پیدا کنیم، نیاز به زیرساختی قابل انعطاف خواهیم داشت که توسط فراهم کنندگان منابع زیرساخت میتوان به آن دست پیدا کرد(۵۵).

### ۲-۲رایانش ابری

برای شناخت بهتر رایانش ابری از دید زیرساخت، ابتدا نگاهی به سیرتکاملی سیستم های محاسباتی از ابتدا تا کنون می اندازیم، تا بتوانیم جایگاه آن را در بین دیگر سیستم ها تشخیص دهیم. اگر به نسل اول سیستم های محاسباتی نگاه کنیم، ما در ابتدا با یک سیستم بسیار بزرگ مواجه بودیم، که کاربران از طریق یک ترمینال واحد به آن دسترسی داشته اند. کم کم این سیستم ها کوچک تر شدند و با توان پردازشی بیشتر و قیمت کمتر، به صورت رایانه های شخصی در اختیار همه کاربران قرار گرفتند. سپس این امکان فراهم شد که با اتصال مجموعه ای از این سیستم های کوچک، شبکه ای با توان پردازشی بیشتر فراهم شود، تا پاسخگوی نیاز های کاربران شود. اما نیاز های پردازشی در حال افزایش بود و نیاز به سیستم های محاسباتی قویتر، بنابراین تعداد بسیاری از این شبکه ها بصورت اختصاصی در سرتاسر اینترنت به یکدیگر متصل شدند و شبکه های محاسبات توری را بوجود آوردند.(۵۴) در این بین مشاهده شد، که میلیون ها کاربر در اینترنت وجود دارند که در اکثر اوقات از تمام توان رایانه خود استفاده نمی کنند و سیستم محاسباتی دیگری شکل گرفت، تا کاربرانی که تمایل داشته باشند، زمان های بیکار سیستم خود را برای کارهای محاسباتی عام المنفعه هدیه کنند. بنابراین تعداد بسیارزیادی منبع محاسباتی کوچک در شبکه ای تحت عنوان محاسبات داوطلبانه به هم پیوستند و توان پردازشی عظیمی را بوجود آوردند.( ۵۴)

اما هنوز منابع بسیار زیاد دیگری در سازمان ها و مراکز داده اینترنتی وجود داشت که تمام ظرفیت آنها بطورکامل بکارگرفته نشده بود. این منابع نمی توانستند در شبکه محاسبات توری بصورت اختصاصی بکارگرفته شوند، زیرا برای آنها وظیفه دیگری تعریف شده بود. در عین حال امکان استفاده از آنها در شبکه داوطلبانه هم وجود نداشت، چون فلسفه وجودی آنها، کاربردهای تجاری بود. به این ترتیب رویکرد جدیدی شکل گرفت که بتوان با استفاده از فناوری های مجازی سازی این منابع را بصورت قابل انعطاف و پویا برای کاربردهای مختلف مورد استفاده قرار داد و از تمام ظرفیت آن ها بطور موثر استفاده کرد. این فناوری رایانش ابری در لایه زیرساخت نام داشت که امکان استفاده از منابع محاسبات و ذخیره سازی را بصورت یک سرویس بر حسب نوع نیاز فراهم می آورد. در حقیقت با ایجاد یک لایه انتزاعی بر روی کلیه منابع فیزیکی خود( به کمک مجازی سازی) امکان مدیریت پویای منابع فیزیکی حاصل می شود(۵۵).

بنابراین رایانش ابری از دید زیرساخت، به گونه ای به سیستم های توزیع شده و موازی اطلاق می گردد که مجموعه ای از رایانه های مجازی را که به یکدیگر متصل هستند، شامل می شود. این رایانه ها بطور پویا عرضه شده و به عنوان یک یا چند منبع محاسباتی یکپارچه براساس توافقات سطح سرویس ارائه می شوند.(۵۴)

این توافقات در طول انجام مذاکرات سرویس دهندگان و مصرف کنندگان برقرار میگردند. رایانش ابری سعی دارد نسل جدیدی از مراکز داده ای را، با ارائه کردن سرویس ها و خدمات در ماشین های مجازی شبکه شده به صورت پویا، به گونه ای ممکن سازد که ارائه دهندگان خدمات کاربردی بتوانند، سرویس ها و برنامه های کاربردی را با انعطاف پذیری و سهولت بیشتری ارائه کنند و کاربران نیز بتوانند از هر جایی از دنیا به برنامه های کاربردی دسترسی داشته باشند(۱و۲و۳)

در تکنولوژی رایانش ابری، کاربران می توانند از طریق ابزارهای مختلف( نظیر رایانه های شخصی، رایانه های همراه و تلفن همراه و...) به برنامه ها، فضاهای ذخیره سازی، پردازش، از طریق سرویس های ارائه شده توسط رایانش ابری، دسترسی داشته باشند. به این ترتیب منابع به جای قرار گیری در سمت کاربر، در سمت سرورها

قرار می گیرند و خدمات فراهم شده از طریق رایانش ابری همانند خدمات عمومی (آب، برق و تلفن و ...) بر اساس سطح استفاده  افراد پرداخت می شود(۵۳)

### ۲-۳عناصر زیرساخت رایانش ابری

مجازی سازی[1] این فناوری که مربوط به دهه ۷۰ هست جهت جلوگیری از افزایش تعداد سرویس دهنده ها توسط کارشناسان به منظور کاهش هزینه ها به وجود آمد و در دستگاههای کامپیوتری به معنی پیاده سازی یک طرح واقعی با شبیه سازی تعریف میشود و به طورکلی امکان تخصیص منابع سختافزاری برای چندین سیستم عامل (مختلف) را امکانپذیر میکند(۶و۸)

·ماشین مجازی[2] ماشین مجازی به دستگاه ی گفته میشود که اجرای سیستم مجازی را بر عهده دارد و هر ماشین مجازی خود یک کامپیوتر است با همان مشخصات و با یک سیستم عامل میهمان ولی جدا از کامپیوتر والد هست و به همین دلیل عملکرد یک ماشین مجازی بر فعالیت دیگر ماشینها تأثیری نخواهد داشت(۸،۶)

فوق ناظر[3] یک برنامه سطح پایین است که برای فراهم کردن دسترسی منابع سیستم به ماشینهای مجازی، مورداستفاده قرار میگیرد و باعث میشود که ماشینهای مجازی از دید یکدیگر پنهان بمانند، به طوری که هر ماشین مجازی تصور میکند، تمام منابع را به تنهایی در اختیار خود گرفته است(۵۳)

پردازش شبکه ای[4] این فناوری با استفاده از زیرساختهای ارتباطی و شبکه های کامپیوتری، امکان دسترسی به انواع منابع از راه دور را میدهد. این شبکه دارای چندین ریزپردازنده، هزاران کامپیوتر شخصی و ایستگاه کاری در سراسر دنیا هست و بااتصال این منابع امکان اجرای پردازشهای بسیار عظیم را امکان میسازد، اما با پردازش ابری تفاوتهای عمدهای دارد.(۳۲) در پردازش شبکه ای تخصیص منابع محدود و از پیش بر اساس قراردادی بین سازمان و مدیر دامنه تعیین شده است .بنابراین منابع محدودی را در اختیار دارد. اما در پردازش ابری

[1] Virtualization
[2] Virtual Machines
[3] Hypervisor
[4] Grid Computing

تخصیص منابع بر اساس تأمین در صورت درخواست هست یعنی برای کاربر پرداخت درازای استفاده صورت میپذیرد.(۳۲) در پردازش شبکه ای تمامی دستگاههای شبکه باید از سیستم عامل ها و نرم افزارهای یکسان و مشابهی استفاده کنند، اما در پردازش ابری محدودیتی در این زمینه وجود ندارد(۳۲)

پردازش شبکه ای برای تعداد اندکی از کاربران که پردازش های بزرگی دارند ولی پردازش ابری برای هر تعداد کاربر با هر نوع درخواستی هست. اما وجه تشابه هردو استفاده از فنّاوری مجازی سازی هست .

وب دو

وب دو پدیدهای است که در نحوه استفاده از فناوری و طراحی سایت ها در اینترنت مطرح شده است. سایت ها یا خدمات اینترنتی که امکان تبادل اطلاعات را بین کاربران فراهم میکنند، یا به آنها اجازه تولید یا دستکاری در اطلاعات را میدهند معمول وب دو تلقی میشوند(۵۴).

### ۲-۴ مدل های استقرار ابر

ابر عمومی[1] در این مدل فراهم کننده ابر منابع را به صورت اشتراکی به کاربران اجاره میدهد و مشابه صنعت برق و تلفن برای کاربران صورتحساب میفرستد. مانند خدمات مایکروسافت و گوگل(۵۴و۴۷)

ابر خصوصی ابر خصوصی برای استفاده انحصاری یک سازمان به کار میرود. مزیت ابرهای خصوصی امنیت بیشتر هست مانند ابر خصوصی آمازون.

ابر ترکیبی[2] این ابر همانطور که از اسمش پیداست، تشکیلشده از یک یا چند ابر عمومی و خصوصی، که هرکدام با حفظ هویتش با ترکیب شدن به عنوان یک واحد عمل میکند.

شکل زیر نمای کلی از مدل های ابررا نشان می دهد:

[1] Public Cloud

[2] Hybrid Cloud


شکل ۲-۳ :مدل هاي استقرار ابر

## ۲-۵ معماری ابر

زیر ساخت به عنوان سرویس[۱]

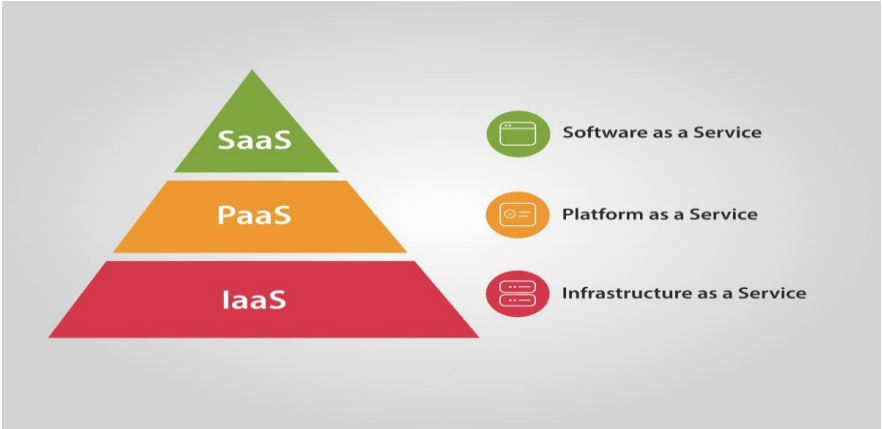
ارائه یک بستر مجازی شده در قالب سرویس توسط فراهم کننده ابر هست که این بستر میتواند فضای ذخیره سازی گره ها، اجزای شبکه و ماشین مجازی باشد. برخی سازمانها که این خدمات را ارائه میدهند مانند آمازون و گوگل(۸)

سکو به عنوان سرویس[۲] این لایه در بالای لایه زیرساخت قرار دارد. در این سرویس امکان ساخت لایه های بالاتر به وجود میآید. این لایه شامل میان افزار، امکانات تبادل پیام و تنظیم اتصال هست. درواقع محیطی را برای توسعه دهنده ی وب برای ایجاد یک برنامه کاربردی ایجاد میکند. مانند گوگل اپ (۸)

نرم افزار به عنوان سرویس[۳] در این مدل نرم افزار و سخت افزار محاسباتی به عنوان سرویس در اختیار مشتری قرار میگیرد. یعنی میتوان به طورکلی اینطور تعریف کرد که تمامی نرم افزارهایی که تاکنون از مغازه ها تهیه میکردیم اکنون آنها را در ابر و با استفاده از اینترنت و با یک واسط، که در سراسر دنیا به آنها دسترسی خواهیم داشت. مانند سرویس جیمیل از شرک گوگل(۸و۶۰)

در شکل زیر معماری چندلایه ای ابر نشان داده شده است

[1] Infrastructure as a Service(Iaas)
[2] Platform as a Service(Paas)
[3] Software as a Service(Saas)




شکل ۲-۴ : معماري لايه هاي ابر

## ۲-۶ مزایای رایانش ابری از دید زیر ساخت

به منظور استفاده حداکثر از رایانش ابری، توسعه دهندگان باید بتوانند ساختار برنامه های خود را بگونه ای اصلاح کنند که به بهترین شکل توسط رویکردمعماری رایانش ابری پشتیبانی شود. مزایای استقراربرنامه ها با استفاده از رایانش ابری شامل کاهش زمان اجرا و زمان پاسخ، حداقل کردن ریسک آماده سازی و ارائه زیرساخت فیزیکی، کاهش هزینه ورود و افزایش امکان نوآوری می باشد(۱۸و۱)

شاید بزرگترین مزیت، مورد توجه تأمین کنندگان زیرساخت ابر، بحث مالی باشد که در برگیرنده هزینه های سرمایه گذاری است. برای مثال اگر شما یک سرویس دهنده خریداری کنید، یک هزینه سرمایه گذاری انجام داده اید، زیرا در ابتدا هزینه را پرداخت کرده اید و سپس ممکن است پس از ۲ تا ۳ سال نتیجه سرمایه گذاری خود را بدست آورید(۲)

از دیدگاه یک ارائه دهنده ابر مسأله مهم این است که با به حداقل رساندن هزینه های عملیاتی و تضمین قرارداد سطح سرویس، به حداکثر سود دهی برسند که در همین راستا مدیریت انرژی در مراکز داده ابری تبدیل به یک موضوع بسیار مهم و قابل توجه برای رسیدن به این هدف می باشد(۲).

## ۲-۷مراکز داده ابری[1]

مراکز داده ابر با نوع سنتی خود کاملا متفاوت است. این مراکز تشکیل شده از صدها یا هزاران شبکه کامپیوتری که به منظور ذخیره سازی و توزیع قدرت و دارای تجهیزات شبکه، تهویه و زیر ساخت خنک کننده ها می باشد. ابرها نیاز به مصرف انرژی بالایی دارند، برای مثال یک مرکز داده معمولی با ۱۰۰۰قفسه نیاز به ۱۰ مگاوات قدرت دارد که از هزینه های عملیاتی آنهم بیشتر خواهد شد. از سال ۱۹۹۰ تا به امروز مصرف برق دوبرابر شده است و پیش بینی می شود تا سال ۲۰۴۰ تقریبا افزایش ۲،۲ ٪را در هر سال بدنبال داشته باشد (۱) این افزایش دما کاهش عمر تجهیزات، تولید گازهای گلخانه ای، کاهش قابلیت اطمینان، نقض کیفیت سرویس و بدنبال آن نقض توافق نامه سطح سرویس بدنبال خواهد داشت(۱) از دیگر مسائل جدی که تأمین کننده ابر باید به آن توجه نماید این است که، با مصرف بسیار انرژی الکتریکی در مراکز داده ابری با این مقیاس بزرگ، دی اکسید کربن تولید می شود که اثر مخرب بر اتمسفر (تأثیر گازهای گلخانه ای) و همچنین اثر مخرب بر محیط زیست دارد(۱)

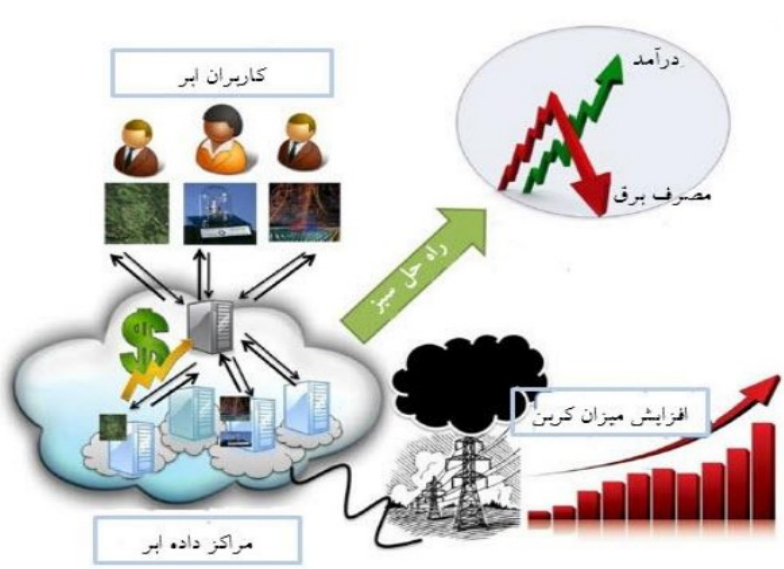
در ماه آوریل ۲۰۰۷ گارتنر تخمین زده است که صنعت فناوری اطلاعات و ارتباطات حدود ۲ ٪از کل گازهای گلخانه ای جهان را تولید می کند که با صنعت حمل و نقل هوایی برابر است و طبق گزارش منتشر شده توسط اتحادیه اروپا کاهش در حجم انتشار گاز دی اکسید کربنتا ۱۵-٪۳۰ ٪قبل از سال ۲۰۲۰ برای حفظ افزایش دمای جهانی لازم و ضروری می باشد(۱۱،۱).

براساس گزارش آژانس حفاظت ازمحیط زیست آمریکا سال ۲۰۰۷ در مراکز داده ابری آمریکا در حدود ۵،۱درصد ۱ از کل انرژی، که هزینه ای حدود ۵،۴میلیارددلار است، مصرف می شود که این مصرف بالای انرژی منجر به انتشار کربن بسیار بالا که درحدود۸۰-۱۱۶ مگاتن در هر سال برآورد شده، می شود(۱) با توجه به محبوبیت روبه رشد استفاده از محاسبات ابری و افزایش آگاهی مردم در سراسر جهان به سمت استفاده از منابع

[1] Cloud Data Centers (CDC)

سازگار با محیط زیست محققان را وادار کرده است به تدبیر در مورد مفاهیم ابر سازگار با محیط زیست به نام محاسبات ابری سبز[1] با کاهش مصرف انرژی و کاهش انتشار گاز دی اکسید کربن(۴،۳،۲)

شکل زیرتأثیر ابر بر محیط زیست را نشان داده است:

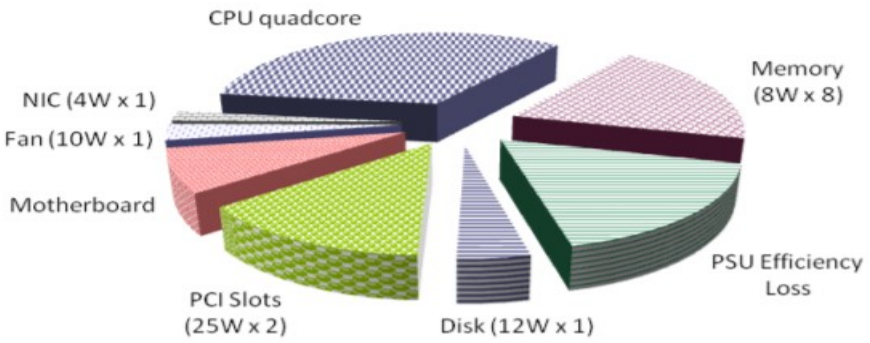



شکل ۲-۵ :ابرو تأثیر آن بر محیط زیست

## ۲-۸ مدل های کاهش مصرف انرژی

در شکل زیر میزان مصرف انرژی در یک گره نشان داده شده است، همان طور که نمایان است میزان مصرف انرژی در مراکز داده ابری ارتباط مستقیم یا خطی با میزان مصرف پردازنده دارد و این دلیلی می شود تا بیشترین تمرکز بر این قطعه باشد.

[1] Green cloud computing




شکل ۲-۶ :مصرف انرژي در یک گره

برای درک هرچه بهتر مدیریت انرژی لازم است ابتدا دو اصطلاح توان(برق) و انرژی را تعریف کنیم: توان و انرژی با تعریف کارانجام شده توسط سیستم معرفی می شوند. توان نرخ انجام کار در یک دوره زمانی وانرژی کل کارانجام گرفته که بیش از یک دوره زمانی است. توان و انرژی در(۲-۱)تعریف شده اند:

$$P = \frac{W}{T},$$
$$E = P \cdot T,$$

تفاوت این دو مهم است، به عبارتی کاهش مصرف توان از طریق کاهش هزینه های زیرساخت ابر، هزینه های ژنراتوربرق و سیستم خنک کننده صورت می گیرد، درحالی که کاهش انرژی کلی و در برگیرنده کاهش قبوض برق می باشد

مصرف انرژی می تواند به طور موقت از طریق مدیریت پویای توان[1] کاهش بیابد با تکنیک DPM ،یا به طور دائم تکنیک SPM ،که این تکنیک با استفاده از قطعات سخت افزاری بسیارکارآمد با استفاده از مدیریت استاتیک توان[2] مانند پردازنده، ذخیره سازی دیسک و منابع تغذیه مصرف انرژی را بهینه سازی می کند در مقابل تکنیک پویا با استفاده از منابع نرم افزاری به این هدف دست پیدا می کند(۵،۴)

[1] Dynamic Power Management(DPM)
[2] Static Power Management (SPM)

- ۲-۸-۱ مدیریت استاتیک و مدیریت پویای توان

مصرف استاتیک توان یا نشت توان ، جریانی که در حال حاضر در هر مدار فعال ایجاد می شود و مستقل از نرخ ۱ ساعت است. کاهش این توان نیاز به بهبود طراحی سیستم های سطح پایین دارد[۵،۴ ]. مصرف پویای توان عمدتا نیاز به یک سناریوی کاربردی خاص دارد و وابسته به ساعت می باشد و آنرا به صورت زیرمعرفی می کنیم:

$$P_{dynamic} = a \cdot C \cdot V^2 \cdot f ,$$

aیک فعالیت سوئیچینگ است وc خازن فیزیکی وv ولتاژ تغذیه وf فرکانس ساعت است. ارزش هایa,c با طراحی سطح پایین سیستم تعیین می شوند و کاهش پویا ترکیبی از ولتاژ و فرکانس، که به این تکنیک DVFS[1]گفته می شود. ایده اصلی این روش در مقیاس پایین عملکرد پردازنده می باشد، زمانی که از آن به طور کامل استفاده نمی شود پس تکنیک DVFS یعنی کاهش ولتاژ و فرکانس پردازنده می باشد(۵و۴)

- ۲-۸-۲مدل مصرف انرژی

یک رابطه خطی بین استفاده از پردازنده و مصرف کل یک گره وجود دارد. گره در حالت غیر فعال و زمانی که به طورکامل استفاده شده است، در زیر رابطه خطی نشان داده شده است:

$$P(u) = P_{idle} + (P_{busy} - P_{idle}) * u ,$$

که در آن P برآورد مصرف برق می باشد و Pidle مصرف یک گره بیکارو pbusy انرژی مصرف شده توسط یک گره زمانی که به طور کامل استفاده شود و U استفاده از پردازنده می باشد و همچنین در(۲-۴ (مدل غیر خطی نیز نشان داده شده است(۴و۶)

$$P(U) = P_{idl} + (P_{Busy} - P_{idle})(2U - U^r)$$

[1] Dynamic Voltage and Frequency Scaling (DVFS)

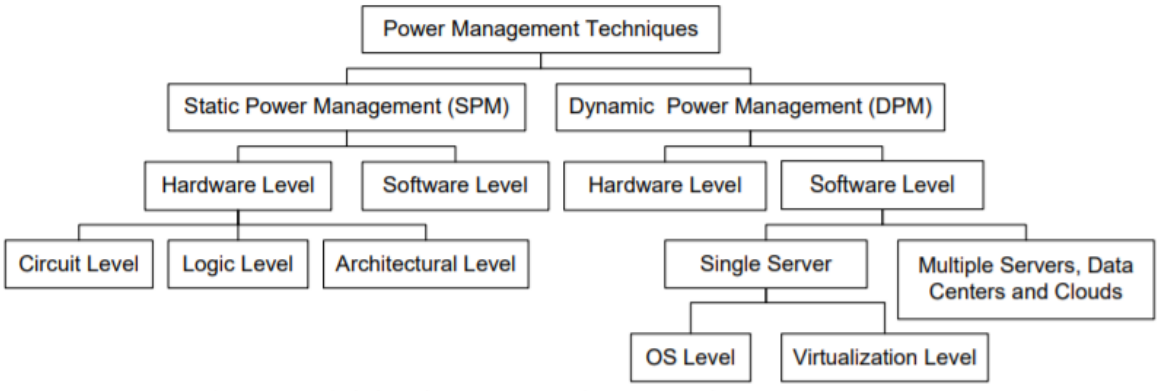
rیک پارامتر کالیبراسیون  است که مربع خطا را به حداقل می رساند و به صورت تجربی بدست می آید .یا در در رابطه زیر(۴)و این رابطه که در آن K ،کسری از توان مصرف شده توسط گره بیکار و pMax،ماکزیمم توان مصرف شده هنگامی که گره به طور کامل استفاده می شود، در ۲۵۰ فرض می شود(۵).

$$P(U) = kP_{max} + (1-k)P_{max}U$$

و تابع انرژی کل مصرفی توسط گره های فیزیکی در(۲-۶ (تعریف می شود که t(U) تابعی از بهره برداری پردازنده در واحد زمان می باشد(۴)

$$E = \int_t P(U(t))$$

شکل زیرمدل های مدیریت توان درسطح نمودار درختی نشان می دهد




شکل ۲-۷ : تکنیک هاي مدیریت قدرت

همانطور که در شکل فوق نشان داده شده است تکنیک های DPM در سطح مجازی سازی، در سطح مراکز داده و همینطوردر سطح سیستم عامل اعمال می شوند که در سطح مراکز داده با تکنیک های غیر فعال کردن قطعات پویا  و عملکرد مقیاس پذیری پویا  عمل می کند .

غیر فعال کردن قطعات پویا[1]DCD : قطعاتی از سیستم که عملکرد مقیاس پذیری را پشتیبانی نمی کنند نیاز به تکنیکی دارند تا زمانی که بیکار هستند، غیر فعال شوند. به طور مثال برای صرفه جویی در توان در مواقعی نیاز به خاموش کردن منبع تغذیه می باشد(۴،۶،۵).

این تکنیک براساس روش های تطبیقی و پیش بینی رفتار سیستم در گذشته و آینده عمل می کند یا بر اساس تعیین پارامتراستاتیک آستانه زمان بیکاری قطعات به منظور کاهش توان و انرژی عمل می کند.

عملکرد مقیاس پذیری پویا[2] DPSبرخی قطعات مانند پردازنده این اجازه را می دهند تا با کاهش یا افزایش تدریجی فرکانس با ولتاژ به منظور کاهش توان تنظیم شوند و این رویکرد وقتی مفید واقع می شود که منابع به طور کامل استفاده نشده باشد. روش DVFS نمونه ای از تکنیکDPS می باشد.

- ۲-۸-۳ سطح سیستم عامل

این بخش به بحث مدیریت منابع در سطح سیستم عامل می پردازد. همانطور که در شکل زیر نشان داده شده ما در این سطح ویژگی هایی داریم که ابتدا تعاریفی ازآنها خواهیم داشت۰ :

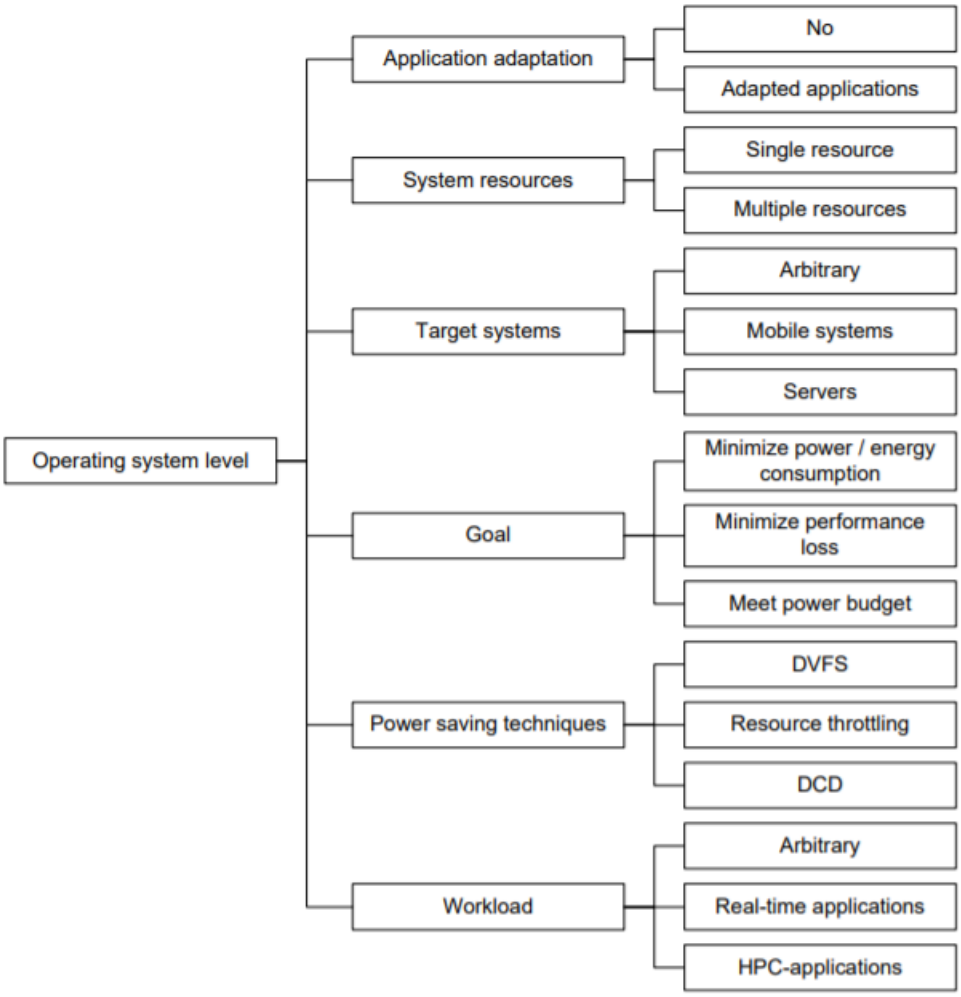
انطباق نرم افزار : آیا سیستم به منظور مدیریت توان، نیاز به اعمال تغییرات در نرم افزاردارد یا نه؟

منابع سیستم: آیا سیستم در بهینه سازی منابع مانند پردازنده تکی عمل می کند یا به صورت چندتایی؟ هدف قرار دادن سیستم : آیا هدف کلی است یا مختص سیستمی خاص می باشد؟

هدف : آیا سیستم مصرف برق یا انرژی را کاهش خواهد داد؟

تکنیک های صرفه جویی انرژی مانند DCD,DPS(DVFS)

[1] Dynamic Component Deactivation (DCD)
[2] Dynamic Performance Scaling (DPS)




شکل ۲-۸ : مدیریت توان در سطح سیستم عامل

- ۲-۸-۴ سطح مجازی سازی

فن آوری که استفاده از منابع سیستم رابهبود می بخشد ودر نتیجه کاهش توان را بدنبال دارد تکنیک مجازی سازی می باشد. مجازی سازی یک لایه انتزاعی است بین سیستم عامل و سخت افزار که به آن مانیتور(فوق ناظر) ماشین مجازی گفته می شود که این ناظر منابع فیزیکی را تقسیم می کند به برش های منطقی به نام ماشین مجازی که هر ماشین مجازی می تواند یک سیستم عامل منحصر به فرد برای کاربر خود ایجاد نماید (۴) مجازی سازی این امکان را می دهد تا چندین ماشین مجازی روی یک گره فیزیکی ایجاد شود و از مزیت های این روش، توانایی حرکت ماشین مجازی از یک گره فیزیکی به گره دیگر می باشد. به این حرکت مهاجرت

ماشین مجازی گفته می شود حال این حرکت می تواند آنلاین یا گرم و در زمان اجرا باشد و یا آفلاین یا سرد باشد.(۴)

مهاجرت ماشین مجازی در زمان اجرا از روش های صرفه جویی در انرژی می باشد که با تجمیع ماشین های مجازی برروی یک گره فیزیکی و کاهش تعداد گره های فیزیکی بیکارو تبدیل آنها به حالت خاموش یا هایبرنت به این هدف می رسد. لازم به ذکر است که یک گره بیکار در حدود ۷۰٪از کل انرژی را مصرف می کند(۵،۴) نکته مهم این است که ناظر ماشین مجازی می تواند به عنوان یک سیستم عامل آگاه به انرژی عمل نموده و با نظارت به عملکرد کلی سیستم و اعمال تکنیک هایDVFS وDCDبر اجزای سیستم مصرف انرژی را کاهش دهد(۴).

- ۲-۸-۵ سطح مراکز داده

یک گره بیکار حدود ۷۰ ٪از مصرف انرژی را در مراکز داده به خود اختصاص می دهد. در حالت استاتیک با خاموش کردن یا به حالت خواب بردن گره در مصرف برق صرفه جویی می شود ولی در حالت پویا با تجمیع حجم کار بر یک گره و غیر فعال کردن گره های بیکار، این عمل سبب بهبود استفاده از انرژی خواهد شد. تجمیع یا تثبیت حجم کار ممکن است منجربه تخریب عملکرد برنامه های کاربردی کاربران شود و درنتیجه نقض قرارداد سطح سرویس که خود جای بحث دارد را بدنبال داشته باشد(۴ )

ویژگی های این سطح درشکل زیراینگونه تعریف می شوند :

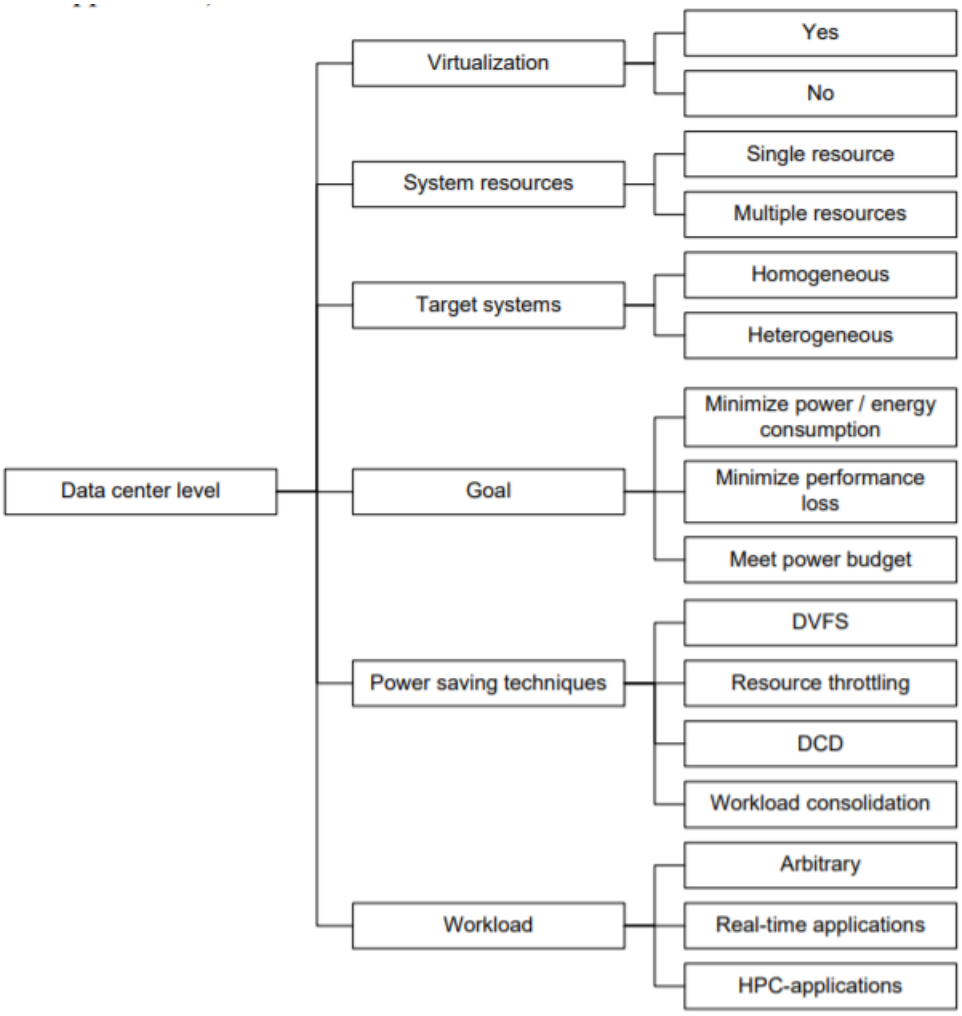
مجازی سازی: آیا این رویکرد به مجازی سازی منابع در مراکز داده می پردازد یا نه؟

هدف سیستم: روی محیط همگن عمل می کند یا محیط ناهمگن ؟

هدف: آیا مصرف انرژی یا برق را کاهش می دهد؟

تکنیک های به کارگرفته شده برای رسیدن به این هدف کدامند؟آیا سیستم از تکنیک هایDPS مانند DVFSاستفاده می کند یا از تکنیک هایDCD و یا تجمیعVM به منظور به حداقل رساندن انرژی استفاده می کند(۴)

ساختار : آیا مدیریت منابع متمرکز است یا به صورت توزیع شده می باشد.




شکل ۲-۹ : مدیریت توان در سطح مراکز داده

## ۲-۹ نتیجه گیری

در این فصل مفاهیم پایه، اصول و تعاریف مورد نیاز برای درک مفاهیم رایانش ابری مانند مفاهیم مربوط به مرکز داده ابری، تکنیک مجازی سازی، مدلهای کاهش مصرف انرژی و تکنیک های مدیریت توان را در سطح سیستم عامل، در سطح مجازی سازی و سطح مراکز داده مورد بررسی قرارگرفت. برای مثال در سطح مرکز داده مشخص شد که در حالت استاتیک با خاموش کردن یا به حالت خواب بردن گره در مصرف برق صرفه جویی می شود ولی در حالت پویا با تجمیع حجم کار بر یک گره و غیر فعال کردن گره های بیکار، این عمل سبب بهبود استفاده از انرژی می شود.

# فصـــل سوم

# پیشینه تحقیق

این فصل به مرور کارهای انجام شده جهت حل مشکل، مصرف انرژی و راهکارهای انجام شده روی این موضوع می پردازیم.

## ۳-۱ مقدمه

به دلیل اهمیت مصرف انرژی در محاسبات در مقیاس بزرگ مانند ابرها و وجود نگرانی های بسیار از انتشارکربن، ازبین رفتن قابلیت اطمینان سیستم در بخش های زیر پژوهش های انجام شده برای مقابله با مشکلات ذکر شده، نشان داده شده است .

- ۳-۱-۱ کاهش مصرف انرژی در گره ها در محیط ناهمگن

هیت و همکاران در(۲۰)مسأله توزیع درخواست کاربران به گره ها مانند وب سرورها از لحاظ کارآمدی انرژی در محیط ناهمگن را مورد بررسی قرار داده اند و برای اولین بار است که در محیط ناهمگن، کاری تحقیقاتی که توزیع حجم کار با در نظر گرفتن انرژی کارآمد، در نظر گرفته می شود .ایده این کار براساس تعیین عملکرد گره های ناهمگن براساس توان مصرف شده توسط منابع مانند پردازنده، دیسک سخت برآورد شده است و در مرحله بعد نرم افزارهای مستقر در گره و نوع درخواستشان از منابع و عملکرد هرکدام و میزان مصرف برق در نظرگرفته می شود و با غیر فعال کردن گره ای که هیچ درخواستی به آن نشده است و هیچ حجم کاری روی آن نمی باشد. پژوهش هاحاکی ازآن است که روش ارائه شده، قادر به کاهش انرژی به میزان ۴۲٪می باشند .

- ۳-۱-۲ تجمیع انرژی آگاه برای محاسبات ابری

سریکانتایاه و همکاران د(۴۲) به تجمیع ماشین مجازی در مراکز داده ابری به منظور بهینه سازی انرژی می پردازند و به عنوان نخستین گام در جهت تجمیع به روابط متقابل بین مصرف انرژی و استفاده از منابع می پردازند. میزان مصرف انرژی حاکی ازآن است که وقتی منابع به درستی استفاده نشود منجر به هزینه های بالاتر و زمان اجرای طولانی تر ودرنتیجه مصرف انرژی بیشتر خواهد شد و با توجه به نتایج بدست آمده و اظهارداشت نویسندگان پژوهش، هدف از انرژی آگاه با تجمیع حجم کار برای حفظ گره های خوب می باشد .

- ۳-۱-۳ تخصیص بهینه گره

گاندی و همکاران در(۱۵) میزان مصرف برق گره ها در ایالات متحده را مورد بررسی قرار دادند و مصرف بیش از ۱/۵از کل برق در ایالات متحده را و هزینه نزدیک به ۴/۵ میلیارد دلاری را در مزارع گره ها می بینند و با توجه به افزایش قیمت انرژی بسیاری از صنایع به دنبال راهکاری برای بهترین استفاده از انرژی می باشند. ایده آنها این بود که برای رسیدن به بهترین عملکرد، یک گره باید در بالاترین سطح قدرت خود قرار گیرد و طرح تخصیص بهینه به عوامل بسیاری بستگی دارد.

آنها با استفاده از تکنیک های مقیاس پذیری فرکانس پویا[1] ، مقیاس پذیری ولتاژ - فرکانس پویا[2] و ترکیب این دو قدرت گره را به حداکثر رسانده و آزمایشات حاکی از آن است، با توجه به رابطه قدرت با فرکانس و با استفاده از طرح تخصیص بهینه می توان به طور قابل توجهی به کاهش متوسط زمان پاسخ و کاهش مصرف انرژی دست یافت.

- ۳-۱-۴ مدیریت برق

ناتوجی و همکاران در (۳۵)به بررسی مشکل مدیریت منابع در مراکز داده مجازی در مقیاس بزرگ پرداختند و این اولین باربود که تکنیک مدیریت انرژی و توان در زمینه سیستم های مجازی استفاده شد. علاوه براین تکنیک مقیاس پذیری سخت افزار وتجمیع ماشین های مجازی اولین بار در کنار هم و اعمال روش مدیریت انرژی توسط نویسندگان به نام روش مقیاس پذیری منابع نرم افزاری صورت گرفت .هدف از این روش بهره برداری از ماشین های مجازی مهمان[3] می باشد و نویسندگان مدیریت منابع را در دوسیاست محلی و جهانی تقسیم کرده اند. در سطح محلی مدیریت توان ماشین مجازی مهمان در هر دستگاه فیزیکی می باشد و در سیاست جهانی که مدیریت چندین ماشین مجازی را به عهده دارد به تجمیع ماشین های مجازی با استفاده از تکنیک مهاجرت به منظور آزاد سازی گره سبک بار و متقابلا صرفه جویی در انرژی می پردازد. آزمایشات حاکیاز

[1] Dynamic Frequency Scaling (DFS)
[2] Dynamic Voltage and Frequency Scaling (DVFS)
[3] Guest

آن است که استفاده از روش ارائه شده منجر به هماهنگی مؤثر از ماشین های مجازی و سیاست های مدیریت انرژی و کاهش مصرف انرژی تا ۳۴ ٪خواهد شد

رمیا راگاوندرا و همکاران در (۳۶)به مصرف برق و گرمای حاصل شده، به عنوان چالش های کلیدی در محیط های مرکز داده اشاره می کنند و طبقه بندی آنان نیز در هردو سطح محلی و جهانی می باشد. روش آنها از یک کنترلرماشین مجازی[1] استفاده می کند تا بر کنترل بهینه سازی مصرف برق، بهره برداری از منابع، پیش بینی تقاضای آینده و تنظیم پردازنده نظارت نماید. این کنترلر، مصرف برق Vms را در سراسرگره فیزیکی کنترل می کند و با تجمیع ماشین های مجازی و تعویض گره ها از آماده به کار یا بالعکس به صرفه جویی در انرژی می رسد.

- ۳-۱-۵ قدرت و عملکرد مدیریت

دارا کیوزیک و همکاران در(۲۴ )هدف از کار خود را به حداقل رساندن مصرف برق و جلوگیری از نقض قراردادسطح سرویس می نامند و اینکار را با استفاده از یک کنترلر انجام می دهند به نام کنترلر محدود به نگاه جلو که مدیریت منابع را مانند به اشتراک گذاری پردازنده به ماشین های مجازی وتجمیع و مهاجرت آنلاین ماشین های مجازی به همراه تغییر حالت گره از روشن به خاموش یا بالعکس به عنوان مکانیزمی در صرفه جویی قدرت اعمال می کنند. نویسندگان این مقاله با ارائه یک مدل ریاضی برای حل مشکل بهینه سازی قدرت نشان دادند کهLLC ، میزان ۲۶ ٪در هزینه های مصرف برق برای یک دوره ۲۴ ساعته صرفه جویی کرده و به میزان ۶،۱ ٪از درخواست ها با نقضSLA مواجه شده اند.

[1] Limited Lookahead Control (LLC)

- ۳-۱-۶ تخصیص منبع با استفاده از خوشه های مجازی

مارک استیل ول و همکاران در (۴۳)به مقایسه الگوریتم های تخصیص منبع پرداخته و الگوریتم بین پکینگ[1] در محیط همگن مجازی را به عنوان بهترین الگوریتم تخصیص نام می برند. روند کاری این الگوریتم بدین گونه است که با مرتب کردن نزولی وظایف و پویا گرفتن حجم کار و استفاده از یک ناظر با عنوان مدیر ماشین های مجازی به تخصیص منابع می پردازد. مدیر ماشین مجازی امکان مهاجرت ماشین های مجازی را درمیان گره های فیزیکی می دهد

.سانگ و همکاران در (۴۱)در جهت بهبود مصرف انرژی و کاهش آن استفاده درست و کارآمد از منابع را در مراکز داده مجازی مورد مطالعه قرار داده اند. منابعی که در این کار در نظر گرفته می شود پردازنده و حافظه اصلی می باشند. منابع برای اطمینان از کیفیت سرویس دهی براساس الویت های برنامه در اختیارش قرار می گیرد. یعنی زمانی که منابع محدود می باشد عملکرد یک برنامه با الویت پایین را عمدا تخریب کرده و منابع را به برنامه ای مهم و با الویت بالا اختصاص می دهد. نویسندگان برنامه ریزی را در سه سطح انجام می دهند: زمانبندی در سطح برنامه برای فرستادن درخواست به تمام ماشین های مجازی، زمانبندی در سطح محلی منابع ماشین مجازی در حال اجرا، زمانبندی در سطح جهانی و کنترل زمانبندی بین تمام برنامه های کاربران.

- ۳-۱-۷چارچوب آگاه از توان و انرژی

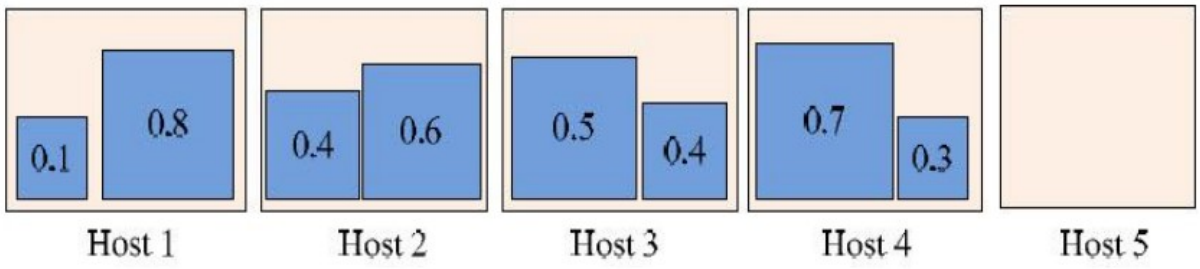
ورما و همکاران در(۴۴)به بررسی و طراحی چارچوبی آگاه از توان در یک محیط مجازی ناهمگن می پردازد و با استفاده از تکنیک های سخت افزاری مانند DVFS و تکنیک مجازی سازی به مدیریت انرژی می پردازند و با تعریف یک مدیر جهانی به تخصیص ماشین مجازی جدید و همچنین تخصیص دوباره ماشین مجازی مهاجرت کرده می پردازند و آنان با استفاده از اندازه ماشین های مجازی هزینه مهاجرت آنان را بدست می آورند . نویسندگان به مقایسه چندین الگوریتم تعریف شده برای حل مشکل بهینه سازی توان پرداخته و مشکلاتی که این الگوریتم ها دارند مانند الگوریتم بین پکینگ که دارای اشکالاتی بود مانند اندازه متغیر بین و هزینه بسته

[1] Bin Packing Algorithm

بندی بین  ها، که برای حل مشکل بسته بندی بین، نویسندگان از الگوریتم اولین انتخاب به صورت کاهشی استفاده کردند .نتایج نشان می دهند که استفاده از این چارچوب در حدود ۲۵ ٪در توان صرفه جویی شده است .

### ۳-۱-۷-۱اولین انتخاب به صورت کاهشی

در این روش ابتدا کالاها به صورت نزولی مرتب شده، سپس با انتخاب بزرگترین کالا، بسته های موجود یکی یکی بررسی می شوند تا به اولین بسته ای که بتواند کالای انتخابی را در خود جای دهد برسد، سپس کالادر آن قرار می گیرد. در شکل زیر چگونگی این روش نشان داده شده است.




شکل۳-۱ :تخصیص ماشین هاي میزبان به ماشین هاي مجازي با استفاده از روشFFD

- ۳-۱-۸ابر سبز : انرژی کارآمد و مدیریت منابع

بویا و همکاران در (۵)به چالش ها و معماری برای مدیریت انرژی کارآمد در محاسبات ابری سبز و به ارائه راه حل هایی برای محاسبات ابری سبز پرداخته اند که نه تنها می تواند با کاهش انرژی تولید گازهای گلخانه ای را کاهش دهد و محیط زیست رانجات دهد، بلکه کاهش هزینه های عملیاتی را نیز بدنبال خواهد داشت .منابع در چند سطح ارائه می شوند: در اولین سطح، کاربر درخواست خود را به کارگزار[1] برای تأمین منبع مانند  پردازنده می دهد و پس از تعیین یک مهلت زمانی[2]  از طرف کاربر، کارگزار درخواست کاربر را به تعدادی از مراکز داده

[1] Broker
[2] Deadlin

می دهد برای تأمین ماشین مجازی، مراکز داده با در اختیار کذاشتن ماشین مجازی براساس مهلت مورد نیاز کاربر این درخواست را برآورده می نمایند .

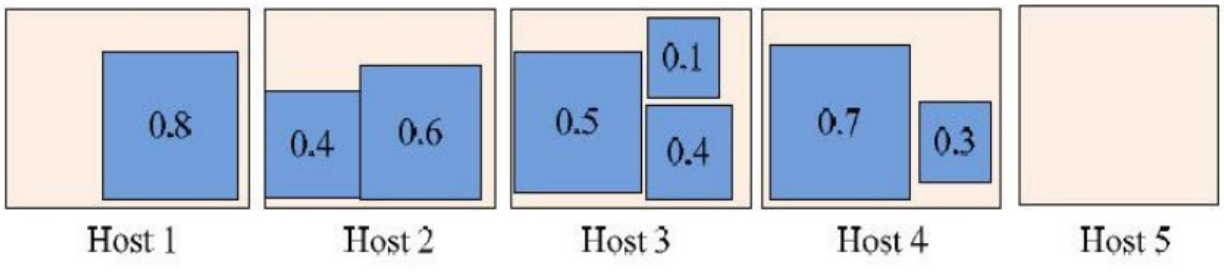
### ۳-۱-۸-۱ سیاست تخصیص

در این پژوهش(۵)سیاست تخصیص ماشین مجازی به دو بخش تقسیم می شود: بخش اول، پذیرش درخواست های جدید برای ارائه ماشین مجازی و قراردادن ماشین مجازی بر روی گره و در بخش دوم بهینه سازی تخصیص ماشین های مجازی فعلی بعد از مهاجرت می باشد .

بدلیل مشکل بسته بندی الگوریتم بین پکینگ، آنان ازاصلاح الگوریتم بهترین انتخاب به صورت کاهشی به منظور جایگذاری ماشین های مجازی به گره استفاده کردند، بدین صورت که تخصیص براساس حداقل افزایش مصرف انرژی بعد از این تخصیص، صورت می گیرد. پیچیدگی این الگوریتم m.n می باشد که n تعداد ماشین های مجازی و m تعداد میزبان های فیزیکی می باشد .

### ۳-۱-۸-۲بهترین انتخاب به صورت کاهشی

در این روش ابتدا پاکت ها به صورت نزولی مرتب می شوند، سپس با انتخاب بزرگترین کالا به جایگذاری آن درون بسته ای با بهترین انتخاب اقدام می نماید، بهترین انتخاب جعبه ای است کهکمترین فضای باقی مانده پس از جایگذاری را داشته باشد.. در شکل ۳-۲ چگونگی این روش نشان داده شده است.(۳۲).




شکل ۳-۲ :تخصیص ماشین هاي میزبان به ماشین هاي مجازي با روشBFD

بر اساس الگوریتم جایگذاری ماشین مجازی، اصلاح الگوریتم بهترین انتخاب به صورت کاهشی ابتدا ماشین های مجازی به صورت نزولی و براساس میزان بهره وری پردازنده شان مرتب می شوند و این ماشین های مرتب شده به گره هایی تخصیص داده خواهند شد که کمترین افزایش توان را پس از تخصیص ایجاد نمایند(۶،۵،۴)

### ۳-۱-۸-۳سیاست انتخاب ماشین های مجازی

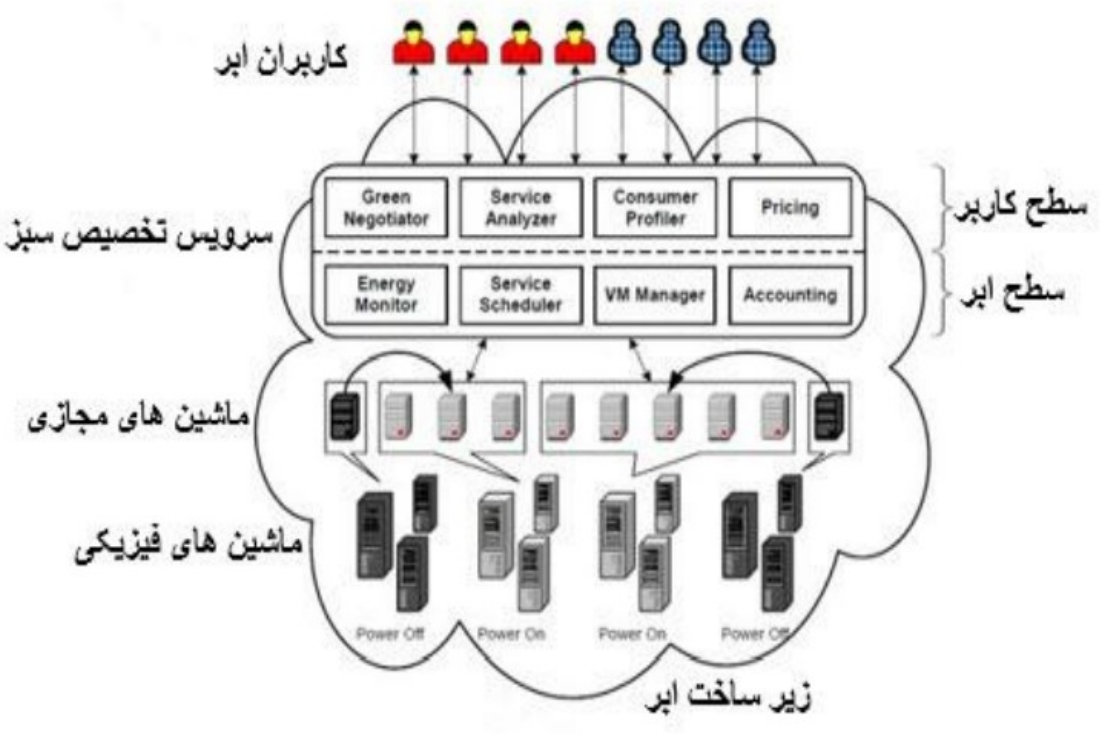
نویسندگان(۵)سه سیاست برای انتخاب ماشین های مجازی که مجبور به مهاجرت می شوند تعریف می کنند(۲)

سیاست کمترین زمان مهاجرت :کاهش زمان مهاجرت با انتخاب ماشین های مجازی براساس کمترین میزان حافظه آنها و به حداقل رساندن سربار مهاجرت .

بالاترین رشد بالقوه[1] :مهاجرت ماشین های مجازی که دارای کمترین میزان استفاده از پردازنده هستند به منظور به حداقل رسانیدن نقض قرارداد سطح سرویس .

انتخاب تصادفی :انتخاب تعداد لازم از ماشین های مجازی برای مهاجرت با توجه به متغیر توزیع تصادفی . آنان در سیاست انتخاب از روش های کران بالا و پایین بهره وری میزبان ها به عنوان معیار انتخاب استفاده کرده به گونه ای که بهره وری کلی پردازنده از مقادیر معین تجاوز نکند .شکل ساختار سیستم کارآمد برای انرژی در زیر ساخت ابر نشان داده شده است:

[1] Highest Potential Growth (HPG)




شکل ۳-۳ : ساختار سیستم انرژي کارآمد براي محاسبات ابري سبز

بلاگلازوف در(۲ )یک روش فرااکتشافی برای تخصیص منابع انرژی آگاه و تجمیع ماشین های مجازی پیشنهاد داد .آنها یک مجموعه آستانه بالا برای استفاده از پردازنده در یک گره تعیین کردند و سپس در طی یک دوره ثابت آنها را بررسی کردند، اگر استفاده از گره بیش از حد آستانه باشد، گره مشخص می شود و سپس ماشین مجازی برای مهاجرت از گره مذکور انتخاب می شود. بلاگلازوف در کار بعدی خود(۵)به جای آستانه ثابت از یک متغیر استفاده نمود.

زیانگ فو در (۱۴)کارهای انجام گرفته توسط تیم ابر (۳و۵)مورد مطالعه قرار داده است و به منظور کاهش مصرف انرژی از سیاست بهبود انتخاب ماشین مجازی که بر اساس استفاده از پردازنده و سیاست تخصیص ماشین مجازی مهاجرت داده شده به یک میزبان با استفاده از روش حداقل ضریب همبستگی (یعنی اینکه آیا با قرار دادن این ماشین مجازی مهاجرت داده شده عملکرد این میزبان دچار تخریب شده و بر روی عملکرد ماشین های مجازی موجود در این میزبان اختلال ایجاد می شود) بهره برده است .

عزیز مرتضائو و سانگ یون اه در(۳۴)به تجمیع گره ها با استفاده از الگوریتم مهاجرت ماشین مجازی در محاسبات ابری سبز پرداخته اند و ازآنجاکه مهاجرت یک عمل پر هزینه برای تأمین کننده ابر می باشد، هدف دوم این نویسندگان به حداقل رسانیدن تعداد مهاجرت ها می باشند و با بکارگیری الگوریتم خودشان و مقایسه آن با الگوریتم های اکتشافی بین- پکینگ مانند اولین انتخاب به صورت کاهشی و بهبود آن به این مهم دست یافته اند، اما چالش های این کار در این است که در حالی که تعداد ماشین مجازی در حال افزایش می باشد، تعداد مهاجرت نیز بسیار افزایش پیدا می کنند و به چگونگی دوباره تخصیص ماشین مجازی مهاجرت داده شده نمی پردازد، همچنین دراین کار به کاهش انرژی پرداخته نشده است. درجدول زیر تحقیقات پیشین در سطح مراکز داده آورده شده است

جدول۳-۱:مقایسه تحقیقات پیشین در سطح مراکز داده

| تکنیک ذخیره انرژی | هدف | منابع | مجازی سازی | نویسنده |
|---|---|---|---|---|
| تجمیع بار کاری، تعویض حالت گره DCD | کاهش انرژی ، بالا بردن بهره وری | Cpu,disk,network | خیر | [20] هیت و همکاران |
| تجمیع بارکاری، تعویض حالت گره، DVFS DCD | کاهش انرژی | Cpu,disk | خیر | [42]اسریکاتایه |
| DVFS | کاهش هزینه زمان اجرا | CPU | خیر | [15]گاندی |
| تجمیع ماشین مجازی- تعویض حالت گره- DVFS, DCD | کاهش انرژی | CPU | بله | [35]ناتوجی |
| تجمیع ماشین مجازی- تعویض حالت گره- DVFS ,DCD | کاهش انرژی و هزینه | CPU | بله | [36]راگاوندرا |
| تجمیع ماشین مجازی- تعویض گره- DCD | کاهش انرژی | CPU | بله | [24]کیوزیک |
| تجمیع ماشین مجازی – گلوگاه منبع DCD RESOURCE THROTTLING | کاهش انرژی | CPU | بله | [43]استیل ول |
| گلوگاه منبع RESOURCE THROTTLING | کاهش انرژی | CPU,RAM | بله | [41]سانگ |
| تجمیع ماشین مجازی- تعویض حالت گره- DVFS DCD | کاهش انرژی | CPU | بله | [44]ورما |
| DVFS | کاهش انرژی | CPU | بله | [4]یویا |
| تجمیع ماشین مجازی و DVFS DCD | کاهش قدرت و صرفه جویی در بودجه | CPU,RAM,NETWORK | بله | [8]یویا |
| تجمیع ماشین مجازی ، تعویض حالت گره DCD | کاهش هزینه های عملیاتی، صرفه جویی در انرژی،افزایش کیفیت سرویس دهی | CPU | بله | [2]بلگلازوف ۲۰۱۰ |

| | | | | |
|---|---|---|---|---|
| تجمیع ماشین مجازی ، تعویض حالت گره DCD | کاهش هزینه های عملیاتی، صرفه جویی در انرژی،افزایش کیفیت سرویس دهی،بهینه سازی استفاده از منابع، جلوگیری از نقض SLA | CPU | بله | [3]بلاگلازوف۲۰۱۲ |
| تجمیع ماشین مجازی ، تعویض حالت گره DCD | کاهش هزینه های عملیاتی، صرفه جویی در انرژی،افزایش کیفیت سرویس دهی،بهینه سازی استفاده از منابع، جلوگیری از نقض SLA | CPU,RAM,NETWORK | بله | [5]بلاگلازوف۲۰۱۲ |
| تجمیع ماشین مجازی ، تعویض حالت گره DCD | کاهش انرژی، کاهش تعداد مهاجرت ماشین مجازی و کاهش نقض SLA | CPU | بله | [14]زیانگ فو |
| تجمیع ماشین مجازی ، تعویض حالت گره DCD | کاهش تعداد مهاجرت ماشین مجازی و کاهش انرژی | RAM,CPU,Network | بله | [34]مرتضایو |

## ۳-۲ نتیجه گیری

در این فصل با توجه به اهمیت مصرف انرژی در محاسبات ابری در مقیاس بزرگ، به کاهش مصرف انرژی در گره ها در محیط ناهمگن پرداخته شد و روش های حل شده در این محیط مورد بحث و بررسی قرار گرفت.

# فصـــل چهارم

# روش کار

این فصل به بیان نوآوری این پژوهش می پردازد.

## ۴-۱مقدمه

در این فصل چگونگی روش به کار برده شده، به منظور کاهش تعداد مهاجرت و کاهش انرژی در مراکز داده ابر می پردازد .

## ۴-۲روش کار

در محیط ابر مجموعه ای گره فیزیکی وجود دارد، که برنامه های کاربردی، در حال اجرا بر روی آنها می باشند . فرض شده است، در این مجموعه پس از مدتی در نحوه چیدمان ماشین های مجازی بر روی گره ها پراکندگی دیده شود، به عبارتی تعداد گره های بیکار روبه افزایش باشند و برای صرفه جویی در انرژی باید روندی بکارگرفته شود تا با مهاجرت ماشین های مجازی از این گره های بیکار و تخصیص دوباره این ماشین های مجازی به گره های مناسب به هدف موردنظر رسید. پس اولین هدف این است که از طریق فناوری مهاجرت ماشین مجازی، با مهاجرت ماشین های مجازی، حداقل گره ها استفاده شود. از آنجا که مهاجرت عملی پر هزینه است، دومین هدف کاهش تعداد مهاجرت است. از نتایج اهداف ذکر شده، هدف اصلی یعنی کاهش انرژی مصرفی خواهد بود .

ابتدا لازم است برای استفاده از گره ها، میزان آستانه، استفاده از پردازنده گره مشخص شود. چراکه استفاده ۱۰۰٪ از پردازنده به تخریب کارائی منجر خواهد شد. انتخاب این مقدار بسیار مهم است، به دلیل آنکه اگر مقدار آستانه بسیار بالا انتخاب شود، برعملکرد ماشین های مجازی در حال اجرا در آن گره اثر گذاشته و کارایی آن گره را کاهش می دهد و اگر بسیار پایین انتخاب شود دیگر تجمیعی رخ نخواهد داد. دراین تحقیق با توجه به (۵و۲۴)این مقدار بین ۵۰ ٪تا ۹۰ ٪انتخاب می شود .

راهکارپیشنهادی از سه کاراکتر پردازنده، حافظه و پهنای باند برای ماشین های مجازی و گره ها استفاده می کند

اندازه دیسک درنظر گرفته نمی شود و فرض می شود، که یک شبکه ذخیره سازی متصل به عنوان یک ذخیره ساز اصلی در امتداد خوشه وجود دارد.

ciظرفیت پردازنده (گیگاهرتز) و mi ظرفیت حافظه به مگابایت است، معرفی می ، که گره ی i با دو بعد ci و miشود. برای ماشین های مجازی j،نیز دوبعد ظرفیت پردازنده Vcj و ظرفیت حافظه ماشین Vmj درنظر گرفته می شود

.اگر فرض شود، Mتا گره و K تا ماشین مجازی وجود دارد عمل تجمیع اینگونه عمل می کند که : {Sm,……,S3,S2,S1}از خوشه Si ،که نشان دهنده حالت گرهi ام می باشد و مجموعه { ip,…..i2,i1}که درآن ip……i2,i1عضو بازه[k,1] که نشان دهنده شماره شناسایی vm که در حال حاضر در گره i ساکن است.

باید در نظر گرفته شود، که برای هر گره i ، ظرفیت کلی ماشین های مجازی روی یک گره نمی تواند از ظرفیت خود گره( ni)بیشتر باشد:

$$\sum_{j \epsilon Si} V\mathrm{i} < n_{\mathrm{i}}$$

سه معیاری که بر تجمیع گره ها تأثیر گذار می باشند، استفاده از تعداد گره ها، تعداد گره های آزاد شده، تعداد مهاجرت ماشین مجازی هستند، که هدف بر معیار اول و سوم می باشد. کاهش معیارها، با کمک ارث بری از الگوریتم اولین انتخاب به صورت کاهشی، با تخصیص ماشین مجازی مهاجرت داده شده و چگونگی مدل تجمیع گره های فیزیکی بدست خواهد آمد.

- ۱-۲-۴مدل تجمیع گره های فیزیکی

Mتا گره براساس رتبه شان با استفاده از رابطه زیربه صورت نزولی مرتب می شوند:

$$score(Si) = \lambda.\,cli + (1 - \lambda).\,mli$$

که به ترتیب λ از cliو mli ازروابط زیر بدست خواهد آمد:

$$\lambda = \frac{\sum_{i\epsilon m} cli}{\sum_{i\epsilon m}(cli+mli)}$$

$$Cli = \frac{\sum_{j\epsilon Si} Vcj}{Ci}$$

$$Mli = \frac{\sum_{j\epsilon Si} Vmj}{mi}$$

گره ها براساس رتبه شان به صورت نزولی مرتب می شوند و آخرین گره از این لیست انتخاب می شود ، یعنی کوچکترین گره از لحاظ رتبه، و بررسی می شود که آیا مهاجرت امکان پذیر است یا نه. این منطق در این الگوریتم گنجانده می شود که اگرحتی یک ماشین مجازی از این گره نتواند مهاجرت داده شود، این گره نمی تواند آزاد شود

به عبارتی اگر m ، لیست تعداد ماشین درحال مهاجرت از یک گره باشد، منطق ایجاب می کند که m تا ماشین باید مهاجرت داده شوند، یا هیچ کدام از ماشین ها ی مذکور نمی توانند مهاجرت داده شوند .بنابراین

گره های دارای حداقل بارکاری آزاد خواهند شد و از مهاجرت های متعدد جلوگیری می شود، چراکه با این روش گره ای انتخاب خواهدشد، که دارای کمترین بارکاری و کم ترین تعداد ماشین مجازی ساکن برآن است . در نتیجه کاهش تعداد مهاجرت را بدنبال خواهد داشت و این مراحل تکرار خواهد شد تا دیگر هیچ مهاجرتی امکان پذیر نباشد

- ۴-۲-۲ مدل مهاجرت ماشین مجازی

ماشین های مجازی که روی گره آخر (کمترین بار کاری) در این لیست می باشند، انتخاب می شوند وکاندیدای مهاجرت می شوند. سپس همین لیست انتخابی حاوی ماشین های مجازی نیز به صورت نزولی براساس رتبه شان طبق رابطه زیرمرتب می شوند.

$$score(Vi) = \lambda.\mathrm{vci} + (1 - \lambda).\mathrm{vmi}$$

برای تخصیص ماشین مجازی از این لیست استفاده می شود. شرایط توقف این الگوریتم به شرح زیر خواهد بود : با تعیین متغیر کل مهاجرت و امکان پذیر بودن مهاجرت تمام ماشین های مجازی از گره انتخاب شده و شرط دیگر توقف حلقه، ME0که حداقل بهره وری می باشد و مقدار آن۱،۰ درنظرگرفته می شود. یعنی اگر میزان بهره وری مهاجرت طبق رابطهکمتر از ۱،۰ بدست آمد، آنگاه الگوریتم پایان پذیرد.

- ۴-۳-۲مدل تخصیص ماشین مجازی

ماشین های مجازی براساس رتبه شان در لیست مهاجرت قرار می گیرند و منتظر قرار گرفتن بر گره مورد نظر می باشند .این الگوریتم به صورت دوره ای لیستی از ماشین های مجازی در حال مهاجرت را و لیستی از گره های موجود (به غیر از گره های آزاد شده) را می گیرد. برای معرفی پارامترهایش ابتدا تابع هدفی تعریف می شود، بدین صورت که ماشین مجازی بهگره ای داده شود که با اضافه شدن آن ماشین به آن گره، بار کل گره موردنظر از حد بالای آستانه(یعنی ۹۰٪)بیشتر نشود.

در پایان با گرفتن توان از گره های فیزیکی، میزان انرژی مصرفی بر حسب کیلووات در ساعت با استفاده از رابطه زیربدست آورده می شود.

انرژی = توان مصرفی مراکز داده بر اساس نوع مدل پاور / (۱۰۰۰*۳۶۰۰

بهره وری مهاجرت = (تعداد گره های آزاد شده / تعداد مهاجرت ماشین مجازی) * ۱۰۰

## ۳-۴ نتیجه گیری

در این فصل روش کار ارائه شد. که با ارث بری از الگوریتم اولین انتخاب به صورت کاهشی و با بدست آوردن رتبه گره های موجود ورتبه ماشین های مجازی و با در نظر گرفتن حداکثر حد آستانه عمل تخصیص ماشین های مجازی مرتب شده در لیست مهاجرت را انجام داد. در پایان میزان مصرف انرژی و میزان بهره وری بدست آمد.

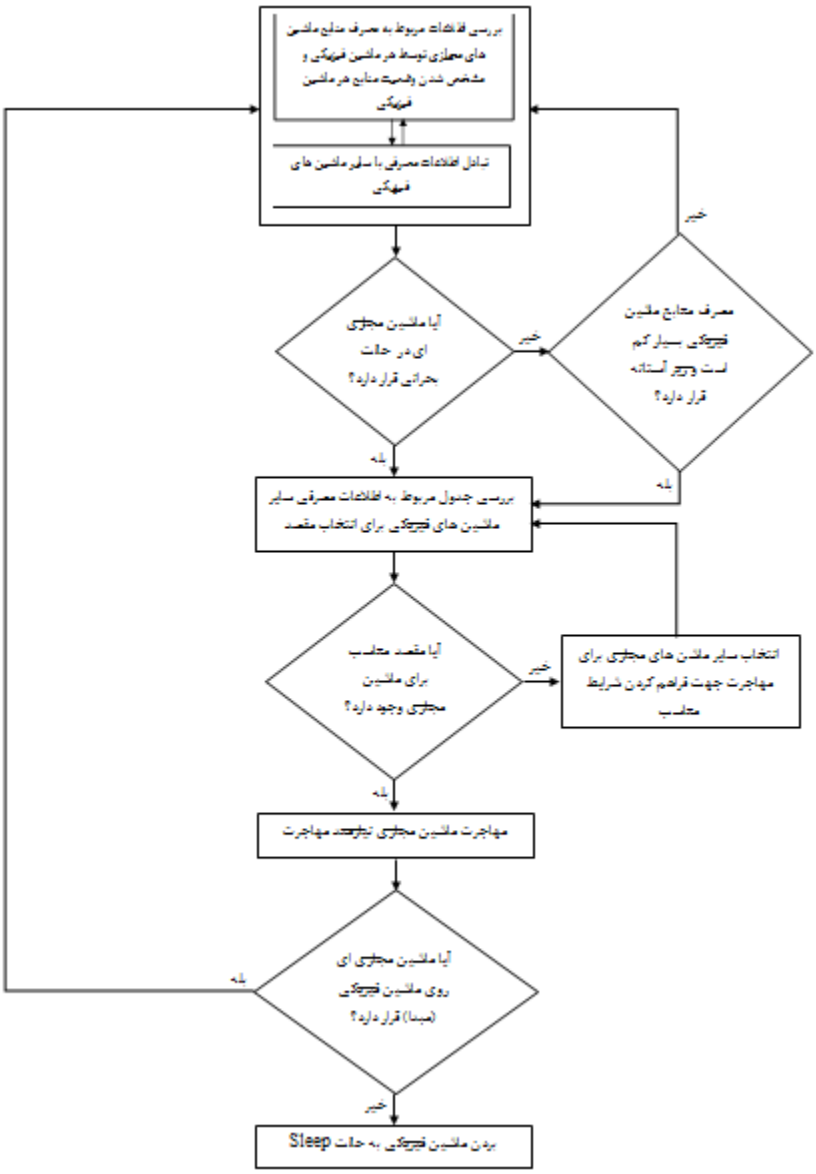
# فصـــل پنجم

# کدهاشبیه سازی ها

در این فصل کدهای شبیه سازی شده به طور کامل شرح داده می شوند.

## ۵-۱مقدمه

در این بخش به تشریح کامل کد نویسی های مربوط به پیاده سازی مهاجرت زنده ماشینهای مجازی در رایانش ابری خواهیم پرداخت. مهاجرت زنده ماشین های مجازی بر اساس انتقال لاگ ها انجام می شود. بـا انقـال لاگ ها، مصرف انرژی ماشین های مجازی مدیریت شده و در حالتی که هر ماشین مجازی از جنبه مصرف باطری یـا انرژی در پایین تر از حد آستانه قرار گیرد، کلیه ماشین های مجازی همسایه که دارای بیشترین میـزان مصـرف انرژی و کمترین فاصله با گره فعلی کاوش شده و سپس ماشین مجازی مناسب انتخاب و مابقی مهـاجرت ادامـه پیدا می کند. در کل سناریوی پیشنهادی مطابق با فلوچارت زیر می باشد.



با توجه به فلوچارت بالا، کد های پیاده سازی شده به شرح ذیل می باشد:

در تکه کدهای زیر فایلهای کتابخانه ای مربوط به زبان برنامه نویسی جاوا و همچنین کلود سیم جهت بار گذاری مراکز داده، هاست ها و ماشینهای مجازی در ابتدای شبیه سازی بارگذاری می گردد.

```
package RM_Balancing_vms;
import org.cloudbus.cloudsim.examples.power.*;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
```

یکی از مهمترین علت های استفاده از فایلهای کتابخانه کلود سیم در محیط جاوا فراهم نمودن امکان لازم جهت مدیریت و اجرای ایده هایی در محیط رایانش ابر است. در این کلیه فایل های کتابخانه ای مربوط به هاست ها، مراکز داده، ماشین های مجازی، تخصیص منابع به هر ماشین مجازی، بروکر ها، زمانبندی کارها، تعادل بار پویا جهت انتخاب ماشین مجازی بهینه با مصرف انرژی لازم و غیره ایمپورت شده اند.

```
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerDynamicWorkload;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.HostDynamicWorkload;
import org.cloudbus.cloudsim.HostStateHistoryEntry;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicy;
import org.cloudbus.cloudsim.VmSchedulerTimeSharedOverSubscription;
import org.cloudbus.cloudsim.VmStateHistoryEntry;
import org.cloudbus.cloudsim.power.PowerDatacenter;
import org.cloudbus.cloudsim.power.PowerDatacenterBroker;
import org.cloudbus.cloudsim.power.PowerHost;
import org.cloudbus.cloudsim.power.PowerHostUtilizationHistory;
import org.cloudbus.cloudsim.power.PowerVm;
import org.cloudbus.cloudsim.power.PowerVmAllocationPolicyMigrationAbstract;
```

```
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
import org.cloudbus.cloudsim.util.MathUtil;
```

در این قسمت تمامی توابع و متدهایی که برای عملیات مختلفی نوشته شده اند

```
public class MainCode {

	/**
	 * Creates the vm list.
	 *
	 * @param brokerId the broker id
	 * @param vmsNumber the vms number
	 *
	 * @return the list< vm>
	 */
```

در این قسمت مجموعه ای از ماشین های مجازی در هر هاست موجود در یک دیتاسنتر تولید می گردند.

```
	public static List<Vm> createVmList(int brokerId, int vmsNumber) {
		List<Vm> vms = new ArrayList<Vm>();
		for (int i = 0; i < vmsNumber; i++) {
			int vmType = i / (int) Math.ceil((double) vmsNumber /
Constants.VM_TYPES);
```

در قسمت زیر نیز هر ماشین مجازی که تولید می شود به آن یک میزان انرژی مصرفی، یک میزان حافظه رم، سیستم عامل مورد نظر و سایر پیش نیاز های لازم اختصاص داده شده و در لیست ماشین های مجازی اضافه می شود.

```
			vms.add(new
PowerVm(i,brokerId,Constants.VM_MIPS[vmType],Constants.VM_PES[vmType],Constants.V
M_RAM[vmType],Constants.VM_BW,Constants.VM_SIZE,1,"Xen",new
CloudletSchedulerDynamicWorkload(Constants.VM_MIPS[vmType],
Constants.VM_PES[vmType]),Constants.SCHEDULING_INTERVAL));
		}
		return vms;
	}
```

در این قسمت یک تعداد هاست تعریف شده و با توجه به تعداد هاست های تعیین شده، میزبانهای مربوطه تولید شده و به لیست میزبان ها اضافه می گردد.

```
	public static List<PowerHost> createHostList(int hostsNumber) {
		List<PowerHost> hostList = new ArrayList<PowerHost>();
		for (int i = 0; i < hostsNumber; i++) {
			int hostType = i % Constants.HOST_TYPES;

			List<Pe> peList = new ArrayList<Pe>();
			for (int j = 0; j < Constants.HOST_PES[hostType]; j++) {
```

```
                        peList.add(new Pe(j, new
PeProvisionerSimple(Constants.HOST_MIPS[hostType])));
                    }
                    hostList.add(new PowerHostUtilizationHistory(i,new
RamProvisionerSimple(Constants.HOST_RAM[hostType]),new
BwProvisionerSimple(Constants.HOST_BW),Constants.HOST_STORAGE,peList,new
VmSchedulerTimeSharedOverSubscription(peList),Constants.HOST_POWER[hostType]));
            }
            return hostList;
    }
```

در این قسمت سرویس دهنده های بروکر که عملیات را به صورت غیره متمرکز انجام می دهند تعریف می گردد. مراکز داده در واقع بین تولید کنندگان و مصرف کنندگان منابع یک کارگزار (Broker) قرار گرفته است. پس از احراز هویت، مصرف کنندگان (کاربران) منابع با کارگزاران برای اجرای برنامه های کاربردی خودشان و دسترسی به منابع راه دور اتصال برقرار می کنند. کارگزاران وظیفه کشف، انتخاب و تجمیع منابع، انتقال منابع و برنامه ها، اجراء عملیات، دسترسی راه دور به منابع و گردآوری نتایج حاصل را بر عهده دارد.

```
    public static DatacenterBroker createBroker() {
        DatacenterBroker broker = null;
        try {
            broker = new PowerDatacenterBroker("Broker");
        } catch (Exception e) {
            e.printStackTrace();
            System.exit(0);
        }
        return broker;
    }

    /**
     * Creates the datacenter.
     *
     * @param name the name
     * @param datacenterClass the datacenter class
     * @param hostList the host list
     * @param vmAllocationPolicy the vm allocation policy
     * @param simulationLength
     *
     * @return the power datacenter
     *
     * @throws Exception the exception
     */
```

در این قسمت دیتاسنترهای لازم تعریف و تولید می گردد. هر دیتاسنتر یا مرکز داده نیز شامل یک سری ویژگی ها و خصوصیات می باشد که عبارتند زا : نام دیتاسنتر، کلاس دیتاسنتر، لیست هاست های موجود در دیتاسنتر،

سیاست تخصیص منابع، مدت اجرای مرکز داده. در نهایت میزان مصرف انرژی توسط دیتاسنتر محاسبه و بر گردانده می شود.

```
public static Datacenter createDatacenter(String name,Class<? extends Datacenter>
datacenterClass,List<PowerHost> hostList,VmAllocationPolicy vmAllocationPolicy)
        throws Exception
    {
```

در قسمت زیر برای دیتاست تعیین شده پارامترهای مربوطه مشخص شده است. جلوی هر معیار مقدار و توضیح آن نشان داده شده است.

```
        String arch = "x86"; // system architecture
            String os = "Linux"; // operating system
            String vmm = "Xen";
            double time_zone = 10.0; // time zone this resource located
            double cost = 3.0; // the cost of using processing in this resource
            double costPerMem = 0.05; // the cost of using memory in this resource
            double costPerStorage = 0.001; // the cost of using storage in this resource
            double costPerBw = 0.0; // the cost of using bw in this resource

            DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(arch,os,vmm,hostList,time_zone,cost,costPerMem,costPerStorage,cost
PerBw);
            Datacenter datacenter = null;
            try {
                datacenter = datacenterClass.getConstructor(String.class,

DatacenterCharacteristics.class,VmAllocationPolicy.class,List.class,Double.TYPE).newInstance
(name,characteristics,vmAllocationPolicy,
                    new LinkedList<Storage>(),Constants.SCHEDULING_INTERVAL);
            } catch (Exception e) {
                System.exit(0);
            }
            return datacenter;
    }
```

این تابع قبل از اینکه هر هاست یا میزبان را در حالت sleep یا خاموش شدن قرار دهد، زمانهای انجام کار و پردازش ها را دریافت می کند. در واقع کلیه میزبان های موجود در یک دیتاسنتر کاوش شده و مدت زمان اجرای مربوط به هر هاست را محاسبه و ارزیابی می کند. در این قسمت شرایط بحرانی برای ماشین های مجازی از لحاظ مصرف انرژی بررسی میگردد. در صورتی که از یک حد آستانه کمتر باشد ماشین مجازی در حالت شات دون میرود.

```
    public static List<Double> getTimesBeforeHostShutdown(List<Host> hosts) {
        List<Double> timeBeforeShutdown = new LinkedList<Double>();
        for (Host host : hosts) {
            boolean previousIsActive = true;
```

```
                        double lastTimeSwitchedOn = 0;
                        for (HostStateHistoryEntry entry : ((HostDynamicWorkload)
host).getStateHistory()) {
                                if (previousIsActive == true && entry.isActive() == false &&
entry<=10 ) {
                                        timeBeforeShutdown.add(entry.getTime() -
lastTimeSwitchedOn);
                                }
                                if (previousIsActive == false && entry.isActive() == true) {
                                        lastTimeSwitchedOn = entry.getTime();
                                }
                                previousIsActive = entry.isActive();
                        }
                }
                return timeBeforeShutdown;
        }

        /**
                        * با استفاده از این متد زمان شروع عملیات در دیتاسنتر‌دریافت مگردد
         *
         * @param vms the vms
         * @return the times before vm migration
         */
```

با استفاده از این متد زمانی که عملیات مهاجر زنده و غیره متمرکز ماشین های مجازی صورت می گیرد دریافت شده و در واقع مدت زمان انجام مهاجرت زنده ماشین های مجازی از یک سرور به سرور دیگر را محاسبه می کند. انتقال لاگ ها در این قسمت انجام می شود.

```
        public static List<Double> getTimesBeforeVmMigration(List<Vm> vms) {
                List<Double> timeBeforeVmMigration = new LinkedList<Double>();
                for (Vm vm : vms) {
                        boolean previousIsInMigration = false;
                        double lastTimeMigrationFinished = 0;
```

در این قسمت اطلاعات سایر ماشین های همجوار به روش حریصانه بررسی شده و در هر حالت ماشینی انتخاب می شود که بیشترین انرژی لازم را داشته و در دسترس نیز باشد.

```
                        for (VmStateHistoryEntry entry : vm.getStateHistory()) {
                                if (previousIsInMigration == true && entry.isInMigration() ==
false) {
                                        timeBeforeVmMigration.add(entry.getTime() -
lastTimeMigrationFinished);
                                }
                                if (previousIsInMigration == false && entry.isInMigration() ==
true) {
                                        lastTimeMigrationFinished = entry.getTime();
                                }
```

```
                        previousIsInMigration = entry.isInMigration();
                    }
                }
                return timeBeforeVmMigration;
        }
```

این تابع کلیه نتایج و عملیاتی را که در هنگام مهاجرت مجازی، تخصیص منابع و کلیه پردازش های انجام شده را نمایش و چاپ می کند.

```
        public static void printResults(PowerDatacenter datacenter,List<Vm> vms,double
lastClock,String experimentName,boolean outputInCsv,String outputFolder)
        {
                Log.enable();
                List<Host> hosts = datacenter.getHostList();

                int numberOfHosts = hosts.size();
                int numberOfVms = vms.size();
```

در این قسمت زیر معیار های انرژی و زمان مهاجرت و سایر موارد دریافت، محاسبه و چاپ می شود.

```
                double totalSimulationTime = lastClock;
                double energy = datacenter.getPower() / (3600 * 1000);
                int numberOfMigrations = datacenter.getMigrationCount();

                Map<String, Double> slaMetrics = getSlaMetrics(vms);
```

در این قسمت که حدود ۱۵ خط است کلا زمانهای مربوط به فعال و غیره فعال شدن یک ماشین مجازی در میزبان، یک میزبان در دیتاسنتر ودر کل یک دیتاسنتر است.

```
                double slaOverall = slaMetrics.get("overall");
                double slaAverage = slaMetrics.get("average");
                double slaDegradationDueToMigration = slaMetrics.get
        ("underallocated_migration");
                double slaTimePerActiveHost = getSlaTimePerActiveHost(hosts);
                double sla = slaTimePerActiveHost * slaDegradationDueToMigration;
                List<Double> timeBeforeHostShutdown = getTimesBeforeHostShutdown(hosts);
                int numberOfHostShutdowns = timeBeforeHostShutdown.size();
                double meanTimeBeforeHostShutdown = Double.NaN;
                double stDevTimeBeforeHostShutdown = Double.NaN;
                if (!timeBeforeHostShutdown.isEmpty()) {
                        meanTimeBeforeHostShutdown =
MathUtil.mean(timeBeforeHostShutdown);
                        stDevTimeBeforeHostShutdown =
MathUtil.stDev(timeBeforeHostShutdown);
                }
```

در این قسمت کلیه زمان ها بررسی می شوند و در صورتی که زمانی مقدار دهی نشده است یا اینکه زمان شروع ماشین مجازی یا سروری مقدار دهی نشده است آن را مقدار دهی می کنند.

```
                List<Double> timeBeforeVmMigration = getTimesBeforeVmMigration(vms);
```

```
		double meanTimeBeforeVmMigration = Double.NaN;
		double stDevTimeBeforeVmMigration = Double.NaN;
		if (!timeBeforeVmMigration.isEmpty()) {
			meanTimeBeforeVmMigration =
MathUtil.mean(timeBeforeVmMigration);
			stDevTimeBeforeVmMigration =
MathUtil.stDev(timeBeforeVmMigration);
		}
		if (outputInCsv) {
			File folder = new File(outputFolder);
			if (!folder.exists()) {
				folder.mkdir();
			}
			File folder1 = new File(outputFolder + "/stats");
			if (!folder1.exists()) {
				folder1.mkdir();
			}
			File folder2 = new File(outputFolder + "/time_before_host_shutdown");
			if (!folder2.exists()) {
				folder2.mkdir();
			}
			File folder3 = new File(outputFolder + "/time_before_vm_migration");
			if (!folder3.exists()) {
				folder3.mkdir();
			}
			File folder4 = new File(outputFolder + "/metrics");
			if (!folder4.exists()) {
				folder4.mkdir();
			}

			StringBuilder data = new StringBuilder();
			String delimeter = ",";
```

در این قسمت کلیه زمانهای قبل و بعد از عملیات مهاجرت مقدار دهی، محاسبه و نمایش داده می شوند.

```
			data.append(experimentName).append(delimeter);
			data.append(parseExperimentName(experimentName));
			data.append(String.format("%d", numberOfHosts)).append(delimeter);
		StringBuilder append = data.append(String.format("%d",
numberOfVms)).append(delimeter);
			data.append(String.format("%.2f",
totalSimulationTime)).append(delimeter);
		StringBuilder append1 = data.append(String.format("%.5f",
energy)).append(delimeter);
			data.append(String.format("%d",
numberOfMigrations)).append(delimeter);
			data.append(String.format("%.10f", sla)).append(delimeter);
```

```
                StringBuilder append2 = data.append(String.format("%.10f",
slaTimePerActiveHost)).append(delimeter);
                StringBuilder append3 = data.append(String.format("%.10f",
slaDegradationDueToMigration)).append(delimeter);
                StringBuilder append4 = data.append(String.format("%.10f",
slaOverall)).append(delimeter);
                StringBuilder append5 = data.append(String.format("%.10f",
slaAverage)).append(delimeter);
                        // data.append(String.format("%.5f", slaTimePerVmWithMigration) +
delimeter);
                        // data.append(String.format("%.5f", slaTimePerVmWithoutMigration) +
delimeter);
                        // data.append(String.format("%.5f", slaTimePerHost) + delimeter);
                        data.append(String.format("%d", numberOfHostShutdowns) + delimeter);
                        data.append(String.format("%.2f", meanTimeBeforeHostShutdown) +
delimeter);
                        data.append(String.format("%.2f", stDevTimeBeforeHostShutdown) +
delimeter);
                        data.append(String.format("%.2f", meanTimeBeforeVmMigration) +
delimeter);
                        data.append(String.format("%.2f", stDevTimeBeforeVmMigration) +
delimeter);
```

در این قسمت نیز سیاست های مهاجرت و تخصیص منابع برای کلیه ماشین های مجازی مقدار دهی می شود.

```
                        if (datacenter.getVmAllocationPolicy() instanceof
PowerVmAllocationPolicyMigrationAbstract) {
                                PowerVmAllocationPolicyMigrationAbstract vmAllocationPolicy
= (PowerVmAllocationPolicyMigrationAbstract) datacenter
                                                .getVmAllocationPolicy();
```

زمانهای اجرا در این قسمت مدیریت می شود.

```
                                double executionTimeVmSelectionMean =
MathUtil.mean(vmAllocationPolicy.getExecutionTimeHistoryVmSelection());
                                double executionTimeVmSelectionStDev =
MathUtil.stDev(vmAllocationPolicy.getExecutionTimeHistoryVmSelection());
                                double executionTimeHostSelectionMean =
MathUtil.mean(vmAllocationPolicy.getExecutionTimeHistoryHostSelection());
                                double executionTimeHostSelectionStDev =
MathUtil.stDev(vmAllocationPolicy.getExecutionTimeHistoryHostSelection());
                                double executionTimeVmReallocationMean =
MathUtil.mean(vmAllocationPolicy.getExecutionTimeHistoryVmReallocation());
                                double executionTimeVmReallocationStDev =
MathUtil.stDev(vmAllocationPolicy.getExecutionTimeHistoryVmReallocation());
                                double executionTimeTotalMean =
MathUtil.mean(vmAllocationPolicy.getExecutionTimeHistoryTotal());
                                double executionTimeTotalStDev =
MathUtil.stDev(vmAllocationPolicy.getExecutionTimeHistoryTotal());
```

```
                                data.append(String.format("%.5f",
executionTimeVmSelectionMean)).append(delimeter);
                                data.append(String.format("%.5f",
executionTimeVmSelectionStDev)).append(delimeter);
                        StringBuilder append6 = data.append(String.format("%.5f",
executionTimeHostSelectionMean)).append(delimeter);
                        StringBuilder append7 = data.append(String.format("%.5f",
executionTimeHostSelectionStDev)).append(delimeter);
                        StringBuilder append8 = data.append(String.format("%.5f",
executionTimeVmReallocationMean)).append(delimeter);
                                data.append(String.format("%.5f",
executionTimeVmReallocationStDev)).append(delimeter);
                                data.append(String.format("%.5f",
executionTimeTotalMean)).append(delimeter);
                                data.append(String.format("%.5f",
executionTimeTotalStDev)).append(delimeter);

                                writeMetricHistory(hosts, vmAllocationPolicy, outputFolder +
"/metrics/" + experimentName
                                                + "_metric");
                        }

                        data.append("\n");

                        writeDataRow(data.toString(), outputFolder + "/stats/" + experimentName
+ "_stats.csv");
                        writeDataColumn(timeBeforeHostShutdown, outputFolder +
"/time_before_host_shutdown/"
                                        + experimentName + "_time_before_host_shutdown.csv");
                        writeDataColumn(timeBeforeVmMigration, outputFolder +
"/time_before_vm_migration/"
                                        + experimentName + "_time_before_vm_migration.csv");

                } else {
                        Log.setDisabled(false);
                        Log.printLine();
                        Log.printLine(String.format("Experiment name: " + experimentName));
                        Log.printLine(String.format("Number of hosts: " + numberOfHosts));
                        Log.printLine(String.format("Number of VMs: " + numberOfVms));
                        Log.printLine(String.format("Total simulation time: %.2f sec",
totalSimulationTime));
                        Log.printLine(String.format("Energy consumption: %.2f kWh", energy));
                        Log.printLine(String.format("Number of VM migrations: %d",
numberOfMigrations));
                        Log.printLine(String.format("SLA: %.5f%%", sla * 100));
                        Log.printLine(String.format(
```

```
                                "SLA perf degradation due to migration: %.2f%%",
                                slaDegradationDueToMigration * 100));
                Log.printLine(String.format("SLA time per active host: %.2f%%",
slaTimePerActiveHost * 100));
                Log.printLine(String.format("Overall SLA violation: %.2f%%",
slaOverall * 100));
                Log.printLine(String.format("Average SLA violation: %.2f%%",
slaAverage * 100));

                Log.printLine(String.format("Number of host shutdowns: %d",
numberOfHostShutdowns));
                Log.printLine(String.format(
                                "Mean time before a host shutdown: %.2f sec",
                                meanTimeBeforeHostShutdown));
                Log.printLine(String.format(
                                "StDev time before a host shutdown: %.2f sec",
                                stDevTimeBeforeHostShutdown));
                Log.printLine(String.format(
                                "Mean time before a VM migration: %.2f sec",
                                meanTimeBeforeVmMigration));
                Log.printLine(String.format(
                                "StDev time before a VM migration: %.2f sec",
                                stDevTimeBeforeVmMigration));

                if (datacenter.getVmAllocationPolicy() instanceof
PowerVmAllocationPolicyMigrationAbstract) {
                        PowerVmAllocationPolicyMigrationAbstract vmAllocationPolicy
= (PowerVmAllocationPolicyMigrationAbstract) datacenter
                                        .getVmAllocationPolicy();

                        double executionTimeVmSelectionMean =
MathUtil.mean(vmAllocationPolicy
                                        .getExecutionTimeHistoryVmSelection());
                        double executionTimeVmSelectionStDev =
MathUtil.stDev(vmAllocationPolicy
                                        .getExecutionTimeHistoryVmSelection());
                        double executionTimeHostSelectionMean =
MathUtil.mean(vmAllocationPolicy
                                        .getExecutionTimeHistoryHostSelection());
                        double executionTimeHostSelectionStDev =
MathUtil.stDev(vmAllocationPolicy
                                        .getExecutionTimeHistoryHostSelection());
                        double executionTimeVmReallocationMean =
MathUtil.mean(vmAllocationPolicy
                                        .getExecutionTimeHistoryVmReallocation());
```

```
                                double executionTimeVmReallocationStdDev =
MathUtil.stdDev(vmAllocationPolicy
                                                .getExecutionTimeHistoryVmReallocation());
                                double executionTimeTotalMean =
MathUtil.mean(vmAllocationPolicy
                                                .getExecutionTimeHistoryTotal());
                                double executionTimeTotalStDev =
MathUtil.stdDev(vmAllocationPolicy
                                                .getExecutionTimeHistoryTotal());

                                Log.printLine(String.format(
                                                "Execution time - VM selection mean: %.5f sec",
                                                executionTimeVmSelectionMean));
                                Log.printLine(String.format(
                                                "Execution time - VM selection stDev: %.5f sec",
                                                executionTimeVmSelectionStDev));
                                Log.printLine(String.format(
                                                "Execution time - host selection mean: %.5f sec",
                                                executionTimeHostSelectionMean));
                                Log.printLine(String.format(
                                                "Execution time - host selection stDev: %.5f sec",
                                                executionTimeHostSelectionStDev));
                                Log.printLine(String.format(
                                                "Execution time - VM reallocation mean: %.5f sec",
                                                executionTimeVmReallocationMean));
                                Log.printLine(String.format(
                                                "Execution time - VM reallocation stDev: %.5f
sec",
                                                executionTimeVmReallocationStDev));
                                Log.printLine(String.format("Execution time - total mean: %.5f
sec", executionTimeTotalMean));
                                Log.printLine(String
                                                .format("Execution time - total stDev: %.5f sec",
executionTimeTotalStDev));
                        }
                        Log.printLine();
                }

                Log.setDisabled(true);
        }

        /**
         * Parses the experiment name.
         *
         * @param name the name
         * @return the string
```

*/

این تابع خروجی ها را در واقع دریافت نموده و در یک فایل ذخیره سازی میکند. زیاد مورد استفاده قرار نمیگیرد این تابع.

```
public static String parseExperimentName(String name) {
	Scanner scanner = new Scanner(name);
	StringBuilder csvName = new StringBuilder();
	scanner.useDelimiter("_");
	for (int i = 0; i < 4; i++) {
		if (scanner.hasNext()) {
		StringBuilder append = csvName.append(scanner.next()).append(",");
		} else {
			csvName.append(",");
		}
	}
	scanner.close();
	return csvName.toString();
}
```

این تابع زمان فعالیت های میزبان ها در مراکز داده را محاسبه نموده و بر میگرداند.

```
protected static double getSlaTimePerActiveHost(List<Host> hosts) {
	double slaViolationTimePerHost = 0;
	double totalTime = 0;

	for (Host _host : hosts) {
		HostDynamicWorkload host = (HostDynamicWorkload) _host;
		double previousTime = -1;
		double previousAllocated = 0;
		double previousRequested = 0;
		boolean previousIsActive = true;

		for (HostStateHistoryEntry entry : host.getStateHistory()) {
			if (previousTime != -1 && previousIsActive) {
				double timeDiff = entry.getTime() - previousTime;
				totalTime += timeDiff;
				if (previousAllocated < previousRequested) {
					slaViolationTimePerHost += timeDiff;
				}
			}

			previousAllocated = entry.getAllocatedMips();
			previousRequested = entry.getRequestedMips();
			previousTime = entry.getTime();
			previousIsActive = entry.isActive();
		}
	}
```

```
        return slaViolationTimePerHost / totalTime;
}
```

این تابع نیز زمان های اجرای کلیه میزبان ها اعم از هاست های فعال و غیره فعال دریافت می گردد.

```
protected static double getSlaTimePerHost(List<Host> hosts) {
        double slaViolationTimePerHost = 0;
        double totalTime = 0;

        for (Host _host : hosts) {
                HostDynamicWorkload host = (HostDynamicWorkload) _host;
                double previousTime = -1;
                double previousAllocated = 0;
                double previousRequested = 0;

                for (HostStateHistoryEntry entry : host.getStateHistory()) {
                        if (previousTime != -1) {
                                double timeDiff = entry.getTime() - previousTime;
                                totalTime += timeDiff;
                                if (previousAllocated < previousRequested) {
                                        slaViolationTimePerHost += timeDiff;
                                }
                        }

                        previousAllocated = entry.getAllocatedMips();
                        previousRequested = entry.getRequestedMips();
                        previousTime = entry.getTime();
                }
        }

        return slaViolationTimePerHost / totalTime;
}
```

این تابع خروجی های تولید شده را به صورت ستونی چاپ می کند. ورودی های این تابع نیز مسیر ذخیره سازی و داده های دریافتی است.

```
public static void writeDataColumn(List<? extends Number> data, String outputPath) {
        File file = new File(outputPath);
        try {
                file.createNewFile();
        } catch (IOException e1) {
                e1.printStackTrace();
                System.exit(0);
        }
        try {
```

```
                BufferedWriter writer = new BufferedWriter(new FileWriter(file));
                for (Number value : data) {
                        writer.write(value.toString() + "\n");
                }
                writer.close();
        } catch (IOException e) {
                e.printStackTrace();
                System.exit(0);
        }
}
```

این تابع خروجی های تولید شده را به صورت سطری چاپ می کند. ورودی های این تابع نیز مسیر ذخیره سازی و داده های دریافتی است.

```
public static void writeDataRow(String data, String outputPath) {
        File file = new File(outputPath);
        try {
                file.createNewFile();
        } catch (IOException e1) {
                e1.printStackTrace();
                System.exit(0);
        }
        try {
                BufferedWriter writer = new BufferedWriter(new FileWriter(file));
                writer.write(data);
                writer.close();
        } catch (IOException e) {
                e.printStackTrace();
                System.exit(0);
        }
}
```

این تابع نیز معیار های مربوطه را در خروجی چاپ می کند.

```
public static void writeMetricHistory(
                List<? extends Host> hosts,
                PowerVmAllocationPolicyMigrationAbstract vmAllocationPolicy,
                String outputPath) {
        // for (Host host : hosts) {
        for (int j = 0; j < 10; j++) {
                Host host = hosts.get(j);

                if (!vmAllocationPolicy.getTimeHistory().containsKey(host.getId())) {
                        continue;
                }
                File file = new File(outputPath + "_" + host.getId() + ".csv");
                try {
                        file.createNewFile();
                } catch (IOException e1) {
```

```
                        e1.printStackTrace();
                        System.exit(0);
                    }
                    try {
                        BufferedWriter writer = new BufferedWriter(new FileWriter(file));
                        List<Double> timeData =
vmAllocationPolicy.getTimeHistory().get(host.getId());
                        List<Double> utilizationData =
vmAllocationPolicy.getUtilizationHistory().get(host.getId());
                        List<Double> metricData =
vmAllocationPolicy.getMetricHistory().get(host.getId());

                        for (int i = 0; i < timeData.size(); i++) {
                            writer.write(String.format(
                                    "%.2f,%.2f,%.2f\n",
                                    timeData.get(i),
                                    utilizationData.get(i),
                                    metricData.get(i)));
                        }
                        writer.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                        System.exit(0);
                    }
                }
            }
```

این تابع در واقع نتایج نهایی شبیه سازی را مدل و در خروجی چاپ می کند.

```
        public static void printCloudletList(List<Cloudlet> list) {
            int size = list.size();
            Cloudlet cloudlet;

            String indent = "\t";
            Log.printLine();
            Log.printLine("========== resource management with load balancing in
Datacenters OUTPUT ==========");
            Log.printLine("Cloudlet ID" + indent+ indent + "STATUS" + indent + "Resource
ID" + indent + "VM ID" + indent
                    + "Time" + indent + "Start Time Sim" + indent + "Finish Time");

            DecimalFormat dft = new DecimalFormat("###.##");
            for (int i = 0; i < size; i++) {
                cloudlet = list.get(i);
                Log.print(indent + cloudlet.getCloudletId());

                if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                    Log.printLine(indent + "RM: " + indent + indent
```

```
                                + cloudlet.getResourceId() + indent
                                        + cloudlet.getVmId() + indent +
dft.format(cloudlet.getActualCPUTime()) + indent
                                        + dft.format(cloudlet.getExecStartTime()) + indent
+ indent
                                        + dft.format(cloudlet.getFinishTime()));
                        }
                }
            Log.printLine(" Resource Management In Cloud Is Finished ...");

        }
```

### ۵-۲نتیجه گیری:

بنابراین با اجرای شبیه سازی فوق می توان به این نتیجه رسید که استفاده از لاگ هـا در انتقـالات و مهـاجرت های ماشین های مجازی ضمن اینکه با سرعت بیشتری انجام می شود، با ایجاد اختلال در هر ماشـین مجـازی، سایر ماشین های موجود در هاست می توانند جای گزین شده و بدون اختلال عملیات مهاجرت را انجام دهند.

## منابع :


[1]A.Atrey, N. Jain , Iyengar ,"A Study on Green Cloud Computing" International Journal of Grid and Distributed Computing,vol. 6,p. 93-102, (2013).

[2]A.Beloglazov, J. Abawajy , R. Buyya ,"Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing" vol. 28,p.755–768, (2012).

[3]A.Beloglazov, R. Buyya ,"Adaptive Threshold-Based Approach for Energy Efcient Consolidation of Virtual Machines in Cloud Data Centers" ,p.1-6, (2010).

[4]A.Beloglazov, R. Buyya ,"Energy Efficient Resource Management in Virtualized Cloud Data Centers",p.826-831, (2010).

[5]A.Beloglazov, R. Buyya ,"Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Center",vol. 24,p.1397–1420,(2012).

[6]A.Beloglazov, R. Buyya, Y. C. Lee , A. Zomaya ,"A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems" ,vol. 82,p.1-50, (2011).

[7]R.Buyya, A. Beloglazov,"OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds" vol.27(5),p.1310–1333, (2015).

[8]R.Buyya, A. Beloglazov , J. Abawajy ,"Energy-efficient management of data center resources for Cloud computing" ,in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA),p. 1–12, (2010).

[9]R.N.Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose , R. Buyya ,"CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software:Practice and Experience,vol. 41,p.23-50, (2011).

[10]M.Cardosa, M. Korupolu ,A. Singh ,"Shares and utilities based power consolidation in virtualized server environments" ,in Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM), p.327-334, (2009).

[11]P.company, www.gartner.com, (2015).[12]S.E.Dashti, A. M. Rahmani ,"Dynamic VMs placement for energy efficiency by PSO in cloud computing",Experimental & Theoretical Artificial,Intelligence,p.1-16, (2015).

[13]H.Duan, Q. Luo, Y. Shi , G. Ma ,"Hybrid Particle Swarm Optimization and Genetic Algorithm for MultiUAV Formation Reconfiguration" , IEEE Computational Intelligence Society,vol. 8(3),p.16 – 27, (2013).

[14]X.FU, C. ZHOU ,"Virtual machine selection and placement for dynamic consolidation in Cloud computingenvironment" ,vol. 9(2),p.322-330, (2015).

[15]A.Gandhi, M. Harchol-Balter, R. Das , C. Lefurgy ,"Optimal power allocation in server farms" ,in Proceedings of the 11th Joint International Conference on Measurement .and Modeling of Computer Systems (SIGMETRICS/Performance),p. 157–168, (2009).

[16]Y.Gaoa, H. Guana, Z. Qia, Y. Houb ,L. Liu ,"A multi-objective ant colony system algorithm for virtual machine placement in cloud computing", Journal of Computer and System Sciences ,vol.79(8),p.1230–1242,(2013).

[17]S.K.Garg, R. Buyya ,"Green Cloud computing and Environmental Sustainability",p. 1-27, (2012).

[18]D.Gmach, J. Rolia, L. Cherkasova ,A. Kemper ,"Resource pool management: Reactive versus proactive or let's be friends" ,p.2905–2922, (2009).

[19]I.Goiri, J. L. Berral, J. O. Fitó, F. Julià, R. Nou, J. Guitart, R. Gavaldà , J. Torres ,"Energy-efficient and multifaceted resource management for profit-driven virtualized data centers",vol. 28,p.718–731, (2012).

[20]T.Heath, B. Diniz, E. V. Carrera, W. Meira , R. Bianchini ,"Energy conservation in heterogeneous server clusters" ,PPoPP '05 Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming,p. 186-195, (2005).

[21]A.Khosravi, S. K. Garg, R. Buyya ,"Energy and Carbon-Efficient Placement of Virtual Machines in Distributed Cloud Data Centers" ,p.317-328, (2013).

[22]K.H.Kim, A. Beloglazov , R. Buyya ," Power-aware provisioning of Cloud resources for real-time services",in Proceedings of the 7th International Workshop on Middleware for Grids,p. 1–6, (2009).

[23]M.R.V.Kumar, S. Raghunathan ,"Heterogeneity and thermal aware adaptive heuristics for energy efficientconsolidation of virtual machines in Infrastructure clouds", Journal of Computer and System Sciences ,vol.82(2),p. 1-30, (2015).

[24]D.Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy , G. Jiang ,"Power and performance management ofvirtualized computing environments via lookahead control" ,Cluster Computing,vol. 12,p.1–15, (2009).

[25]B.Li, J. Li, J. Huai, T. Wo, Q. Li , L. Zhong ,"EnaCloud: An Energy-saving Application Live Placement Approach for Cloud Computing Environments " ,IEEE,p. 17 – 24, (2009).

[26]C.Lin, "A novel Green Cloud Computing FrameWork For Improving System Efficiency" ,p.2326-2333, (2012).

[27]W.Lin, J. Z. Wang, C. Liang ,D. Qi ,"A Threshold-based Dynamic Resource Allocation Scheme for Cloud Computing" ,vol. 23,p.695–703, (2011).

[28]T.Mastelic, A. Oleksiak, H. Claussen, I. Brandic , J.-M. Pierson ,"Cloud Computing: Survey on Energy Efficiency" Journal ACM Computing Surveys (CSUR),vol. 47(2),p. 33, (2015).

[29]K.Maurya, R. Sinha ,"Energy Conscious Dynamic Provisioning of Virtual Machines using Adaptive Migiration Thresholds in Cloud Data Center",p. 74-82, (2013).

[30]K.maurya, R. sinha ,"Energy Conscious Dynamic Provisioning Of Virtual Machines Using Adaptive Migration Thresholds in Cloud Data center " ,vol. 2(3),p. 74-82, (2013).

[31]Ms.V.Srimathi, Ms.D.Hemalatha, Mr.R.Balachander ,"Green Cloud Environmental Infrastructure", Engineering And Computer Science ,vol.1(3),p. 168-177, (2012).

[32]R.Nathuji, C. Isci , E. Gorbatov, "Exploiting platform heterogeneity for power efficient data centers", in Proceedings of the 4th International Conference on Autonomic Computing (ICAC),p. 5-10,(2007). [33]R.Nathuji, K. Schwan ,"VirtualPower: Coordinated power management in virtualized enterprise systems" ,ACM SIGOPS Operating Systems Review,vol. 41,p. 265–278, (2007).

[34]A. Murtazaev,S.Oh,"Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing " ,IETE TECHNICAL REVIEW ,vol. 28, p.1-20, (2012).

[35]N.Quang-Hung, P. D. Nien, N. H. Nam, N. H. Tuong , N. Thoai ,"A Genetic Algorithm for Power-Aware Virtual Machine Allocation in Private Cloud", Springer Berlin Heidelberg ,vol.7804, p. 183-191, (2013). [36]R.Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, X. Zhu ,"No “Power” Struggles: Coordinated Multilevel Power Management for the Data Center" ,SIGARCH Computer Architecture News , vol.36, p.48-59, (2008).

[37]S.Rathore, "Efficient Allocation of Virtual Machine in Cloud Computing Environment" , p.59-62, (2012). [38]S.R.Suraj , R.Natchadalingam ,"Adaptive Genetic Algorithm for Efficient Resource Management in Cloud Computing" ,International Journal of Emerging Technology and Advanced Engineering , vol.4(2), p.350-356, (2014).

[39]K.Sammy, R. Shengbing , C. Wilson ,"Energy Efficient Security Preserving VM Live Migration In Data Centers For Cloud Computing", vol.9, (2012).

[40]J.Sekhar, G. Jeba ,"Energy Efficient VM Live Migration in Cloud Data Centers." IJCSN International Journal of Computer Science and Network, vol.2(2),p. 71-75, (2013).

[41]Y.Song, H. Wang ,B. Feng , Y. Li and Y. Sun ,"Multi-tiered on-demand resource scheduling for VM-based data center", in Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid ,p:148-155, (CCGrid 2009).

[42]S.Srikantaiah, A. Kansal , F. Zhao ,"Energy aware consolidation for Cloud computing" ,in Proceedings of the 2008 USENIX Workshop on Power Aware Computing and Systems (HotPower), p. 1-5, (2008). [43]M.Stillwell, D. Schanzenbach , F. Vivien , H. Casanova , "Resource allocation using virtual clusters" ,in Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, p.260–267, (2009). [44]A.Verma , P. Ahuja ,A. Neogi , "pMapper: power and migration cost aware application placement in virtualized systems", n Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, p. 243-264, (2008).

[45]W.Shi, B.Hong , "Towards Profitable Virtual Machine Placement in the Data Center", p. 138-145, (2011). [46]Sh.Wang , Z. Liu, Z. Zheng, Q. Sun , F. Yang , "Particle Swarm Optimization for Energy-AwareVirtual Machine Placement Optimization in Virtualized Data Centers", 19th IEEE International Conference on Parallel and Distributed Systems,p. 102 - 109, (2013). [47]www.cloudbus.org/papers/Cloud-EnvSustainability, "Green Cloud computing and Environmental Sustainability., (2011).

[48] V.Suresh Kumar, M. Aramudhan" Trust Based Resource Selection in Cloud Computing Using Hybrid Algorithm", I.J. Intelligent Systems and Applications, 2015, p.59-64 .Published Online July 2015 in MECS DOI: 10.5815/ijisa.2015.08.08

[49] Yongqiang Gao , Haibing Guan , Zhengwei Qi , Yang Hou, Liang Liu," A multi-objective ant colony system algorithm for virtual machine placement in cloud computing", 2013, http://dx.doi.org/10.1016/j.jcss.2013.02.004,p.1-13

[50] Akhil Goyal, Navdeep S. Chahal," A Proposed Approach for Efficient Energy Utilization in Cloud Data Center", International Journal of Computer Applications (0975 – 8887) Volume 115 – No. 11, April 2015,p.24- 27

[51] Elina Pacini, Cristian Mateos, Carlos Garc´ıa Garino," Dynamic Scheduling based on Particle Swarm Optimization for Cloud-based Scientific Experiments", CLEI ELECTRONIC JOURNAL, VOLUME 14, NUMBER 1, PAPER 2, APRIL 2014.

[52] Gao, Y., Guan, H., Qi, Z., Hou, Y., & Liu, L. (2013). A multi-objective ant colony system algorithm for 85 virtual machine placement in cloud computing. Journal of Computer and System Sciences, 79, 1230–1242. doi:10.1016/j.jcss.2013.02.004

## Abstract

he upward growth of knowledge and communication has led to the emergence of a new computing style called cloud computing. One of the biggest advantages of cloud infrastructure suppliers is the financial discussion with minimizing costs, guaranteeing the level of the service contract and maximum profitability. In this regard, energy management in cloud data centers is an important and significant issue to achieve this goal. One way to reduce energy consumption is to free unemployed hosts and the other way is to reduce the migration of virtual machines. To this end, one of the challenges is choosing the method of placing the migrated virtual machines on the appropriate host. In this booklet, by introducing the migration reduction algorithm and the placement algorithm based on the appropriate threshold, the energy consumption in cloud data centers is reduced. This algorithm inherits from the algorithm of the first selection and descending order of hosts and virtual machines based on their rank, which is obtained based on workload, performs the placement operation. The solution provided in CloudSim software is simulated. The simulation results show a decrease in the number of virtual machine migrations, an increase in migration productivity, and a reduction in energy consumption.

## Keywords

Green cloud computing, reducing data center power consumption, host aggregation, reducing migration

Payam Noor University

Faculty of Engineering

**Seminar Report**

**Department of Computer Engineering and Information Technology**

**Title**

# Reduce energy consumption (green cloud) in cloud data centers with the help of meta-heuristic algorithms and adjust the reduction of virtual machine migration

**Seyed Ali Mohtarami**

**Supervisor:**

**Dr. Seyed Ali Razavi Ebrahimi**

February 2021