

*تخصیص منابع مبتنی بر مکانیسم Gossip برای محاسبات سبز در ابرهای بزرگ*

*چکیده*

*ما به مسئله تخصیص منابع در محیط ابر در مقیاس بزرگ می­پردازیم، که بهینه­سازی پیکربندی ابر به‌صورت پویا برای اهداف محاسبات سبز تحت محدودیت­های پردازنده و حافظه را رسمی می­کنیم. ما پروتکل عمومی gossip را برای تخصیص منابع پیشنهاد می­کنیم، که می­تواند برای اهداف خاص معرفی شود. ما نمونه­ای از این پروتکل عمومی را باهدف به حداقل رساندن مصرف انرژی از طریق تحکیم[[1]](#footnote-1) سرور توسعه می­دهیم، درحالی‌که رضایت­مندی تغییر الگوی بارگیری برآورده شود. این پروتکل، GRMP-Q نامیده شد، که یک راه­حل ابتکاری[[2]](#footnote-2) کارآمد فراهم می­کند که در اغلب موارد به خوبی عمل می­کند— در موارد خاص، بهینه است. تحت سربار[[3]](#footnote-3)، پروتکل یک تخصیص عادلانه از منابع پردازنده به خدمات گیرنده[[4]](#footnote-4) را می­دهد. نتایج شبیه­سازی نشان می­دهد که معیارهای کلیدی عملکرد، با افزایش اندازه سیستم تغییر نمی­کند، فرآیند تخصیص منابع مقیاس­پذیر برای بالاتر از 100000 سرور. به‌طورکلی، اثربخشی پروتکل در دستیابی به اهداف خود، با افزایش ظرفیت حافظه در سرورها، افزایش می­یابد.*

***کلمات کلیدی:*** *محاسبات ابر، محاسبات سبز، مدیریت توزیع، مدیریت انرژی، تخصیص منابع، پروتکل gossip، تحکیم سرور*

*1. مقدمه*

*مصرف انرژی در مراکز داده[[5]](#footnote-5) قابل توجه است و به سرعت در سال­های اخیر رشد کرده است و این رشد انتظار می­رود ادامه­دار باشد. چند مطالعه آن را نشان می­دهد [1]-[3]. یک روش مؤثر برای کاهش مصرف انرژی مراکز داده، تحکیم سرور [4[، [5] است که باهدف تمرکز حجم کار[[6]](#footnote-6) بر روی حداقل تعداد سرورها است. این کار مؤثر است، زیرا امروزه سطح بهره­برداری در مراکز داده اغلب پایین است، حدود 15% [6]. برای یک اجرا، سرور بیش از 60% از حداکثر مصرف انرژی خود را مصرف می­کند، حتی اگر آن بار حمل نکند ([4])، تعویض سرورها (به‌طور موقت) برای یک حالت ضروری نباشد که حداقل یا هیچ انرژی نیاز دارد که می­تواند به‌طور قابل‌توجهی مصرف انرژی را کاهش دهد.*

*امروزه همه تکنولوژی­های کلیدی موردنیاز برای تحکیم سرور، در دسترس هستند. تکنولوژی­های مجازی سازی[[7]](#footnote-7) و مهاجرت زنده[[8]](#footnote-8)، استحکام پویا حجم کار را تحت درخواست پشتیبانی می­کند. داشتن سطوح مختلف از حالت آماده‌به‌کار[[9]](#footnote-9) (مشخص شده با سطوح مختلف مصرف انرژی و زمان بیدار شدن[[10]](#footnote-10)) که پیشنهادهای تجهیزات مدرن، برای انطباق منابع مرکز داده برای نیازهای در حال تغییر را اجازه می­دهد.*

*در این کار، ما به مسئله مدیریت منابع برای محیط ابر در مقیاس بزرگ (محدوده بالاتر از 100000 سرور) باهدف خدمت یک حجم کار پویا با حداقل مصرف انرژی می­پردازیم. درحالی‌که سهم ما در زمینه کلی­تری است، ما بحث را از منظر مفهوم PaaS [[11]](#footnote-11)، با مورد استفاده[[12]](#footnote-12) خاص از یک ارائه‌دهنده خدمات ابر[[13]](#footnote-13) که سایت­های میزبان­ها[[14]](#footnote-14) در محیط ابر است، انجام می­دهیم. ذینفعان[[15]](#footnote-15) در این مورد استفاده در شکل 1(الف) نشان داده می­شود. ارائه‌دهنده خدمات ابر، زیر ساخت­های فیزیکی، که در آن خدمات ابر ارائه شده است را اداره می­کند. این، سرویس­های میزبانی[[16]](#footnote-16) به صاحبان سایت­ها از طریق یک واسط[[17]](#footnote-17) که در زیرساخت­هایش اجرا می­شود را ارائه می­دهد (شکل 2(ب) را ببینید). صاحبان سایت­ها خدمات را به کاربران مربوطه خود از طریق سایت­هایی که توسط ارائه‌دهنده خدمات ابر میزبان می­شوند ارائه می­دهند.*

**

*شکل 1. (الف) سناریو استقرار ذینفعان محیط ابر در نظر گرفته‌شده در این کار. (ب) معماری کلی محیط ابر. این کار بر روی مدیریت منابع انجام شده توسط لایه واسط تمرکز دارد [7].*

*نتایج حاصل از این کار به مهندسی یک لایه واسط که تخصیص منابع در یک محیط ابر را انجام می­دهد، با اهداف طراحی زیر کمک می­کند.*

*1) هدف عملکرد*

*a. وقتی سیستم در زیر بار است، هدف به حداقل¬رسانی مصرف انرژی است از طریق تثبیت سرور، درحالی‌که رضایت¬مندی تقاضا از سایت¬های میزبان باشد.*

*b. وقتی سیستم در سربار است، هدف تخصیص منابع موجود به‌طور عادلانه در تمامی سایت¬های میزبان است.*

*2) سازگاری: فرآیند تخصیص منابع باید به‌صورت پویا و کارآمد برای تغییرات در تقاضا سازگار باشد.*

*3) مقیاس¬پذیری: فرآیند تخصیص منابع باید در تعداد سرورهای موجود در ابر و تعداد سایت¬های میزبان ابر مقیاس-پذیر باشد. به‌طور خاص، منابع مصرف شده در هر سرور به‌منظور دستیابی به یک هدف عملکرد معین، باید با تعداد سرورها و تعداد سایت¬ها افزایش یابد.*

*رویکرد ما در اطراف یک طراحی غیرمتمرکز مرکزیت می­یابد که در آن اجزای لایه واسط در هر سرور از محیط ابر اجرا می­شود. (ما در ادامه این مقاله به یک سرور از ابر، به‌عنوان یک دستگاه رجوع می­کنیم). برای رسیدن به یک مقیاس­پذیری، ما تصور می­کنیم که تمام وظایف کلیدی لایه واسط، شامل برآورد حالات عمومی[[18]](#footnote-18)، قرار دادن ماژول­های سایت و محاسبه سیاست­ها برای درخواست ارسال، براساس الگوریتم­های توزیع‌شده هستند. بر خلاف نرم افزار مدیریت موجود برای ابرهای خصوصی، مانند OpenNebula [8]، OpenStack [9]، AppScale [10] و Cloud Foundry [11]، راه­حل پیشنهادی ما، در یک شکل ترکیب‌شده و یکپارچه فراهم می­شود. (الف) انطباق پویا از تخصیص منابع موجود در پاسخ به یک تغییر، (ب) مقیاس­گذاری پویا منابع برای یک کاربرد آنسوی یک دستگاه فیزیکی تک و (ج) مقیاس­پذیری آنسوی برخی از 100000 سرور.*

*این مقاله بر اساس کار قبلی ما در مورد مدیریت منابع مقیاس­پذیر برای محیط­های ابر است [7]. آن از معماری واسط از آن کار را استفاده می­کند، رسمی­سازی مسئله تخصیص منابع را سازگار می­کند و دوباره از مفهوم محاسبات سیاست­های تخصیص منابع از طریق پروتکل gossip استفاده می­کند. سهم کلیدی این مقاله به شرح زیر است. ابتدا ما پروتکل عمومی gossip را برای مدیریت منابع در محیط ابر معرفی می­کنیم که می­تواند برای اهداف خاص معرفی شود. سپس، ما مسئله حداقل سازی مصرف انرژی از طریق تثبیت سرور را رسمی می­کنیم و یک راه­حل ابتکاری در قالب یک نمونه از پروتکل عمومی ارائه می­دهیم. درنهایت ما از طریق شبیه­سازی اثربخشی پروتکل در مقایسه با یک سیستم ایده آل نشان می­دهیم، و ما نشان می­دهیم که پروتکل برای ابر بسیار بزرگ به خوبی مقیاس می­کند.*

*ساختار مقاله به‌صورت زیر است:*

*بخش 2 معماری یک لایه واسط که مدیریت منابع را برای یک محیط ابر در مقیاس بزرگ انجام می­دهد را شرح می­دهد. بخش 3 مدل ما برای مدیریت منابع در محیط ابر و راه­حل عمومی برای مسئله مدیریت منابع را ارائه می­دهد. بخش 4 مسئله خاص موردمطالعه در این مقاله و راه­حل ما را ارائه می­دهد. راه­حل از طریق شبیه­سازی در بخش 5 ارزیابی می­شود. بخش 6 کارهای مشابه را مرور می­کند و بخش 7 نتیجه­گیری از این مقاله و شرح کارهای آینده را شامل می­شود.*

*2. معماری سیستم*

*شکل 2 (سمت چپ) معماری واسط ابر را نشان می­دهد. مؤلفه­های لایه واسط روی تمام دستگاه­ها اجرا می­شود. منابع ابر در درجه اول توسط نمونه­های ماژول مصرف می­شوند که به موجب آن قابلیت­های یک سایت از یک یا چند ماژول ساخته شده است. واسط، شامل بخشی از منطق خدمات یک سایت (نشان داده‌شده با mi در شکل 2) یا مدیر سایت (نشان داده‌شده با SMi) است.*

*هر دستگاه یک مؤلفه مدیر دستگاه که سیاست تخصیص منابع را محاسبه می­کند را اجرا می­کند، که شامل تصمیم­گیری نمونه­های ماژول برای اجرا می­شود. سیاست تخصیص منابع توسط یک پروتکل (در این مقاله GRMP نامیده می­شود) محاسبه می­شود که در مؤلفه مدیر منابع اجرا می­شود. این مؤلفه تقاضای طرح­ریزی شده را برای هر ماژول به‌عنوان ورودی می­گیرد که دستگاه اجرا می­کند. سیاست تخصیص محاسبه‌شده، به زمان‌بند ماژول برای اجرا و همچنین مدیر سایت برای تصمیم­گیری درباره درخواست ارسال، ارسال می­شود. مدیر پوشش[[19]](#footnote-19)، یک الگوریتم توزیع‌شده را اجرا می­کند که یک نمودار پوشش[[20]](#footnote-20) از دستگاه در ابر را حفظ می­کند و هر مدیر منبع یک لیست دستگاه­ها برای تعامل فراهم می­کند.*

*معماری ما یک مدیر سایت را با هر سایت مرتبط می­سازد. هر مدیر سایت، درخواست کاربران به یک سایت خاص را کنترل می­کند. این دو مؤلفه مهم دارد: ایجادکننده مشخصات[[21]](#footnote-21) تقاضا و ارسال­کننده درخواست. ایجادکننده مشخصات تقاضا، تقاضای منابع از هر ماژول سایت بر اساس آمار درخواست، اهداف کیفیت سرویس[[22]](#footnote-22) و غیره را برآورد می­کند. (نمونه­هایی از چنین ایجادکننده مشخصات می­تواند در [12] و [13] یافت شود). برآورد برای تمامی مدیران دستگاه­ها که نمونه­هایی از ماژول­های متعلق به سایت را اجرا می­کنند، ارسال می­شود. به‌طور مشابه، ارسال­کننده درخواست، درخواست­های کاربران را برای پردازش برای نمونه­های ماژول­های متعلق به سایت ارسال می­کند. تصمیم درخواست ارسال، سیاست تخصیص منابع و محدودیت­هایی از قبیل وابستگی جلسه[[23]](#footnote-23) را به حساب می­آورد. شکل 2 (سمت راست) مؤلفه های یک مدیر سایت و چگونگی ارتباط با مدیران دستگاه را نشان می­دهد.*

**

*شکل 2. معماری برای واسط ابر (سمت چپ) و مؤلفه های برای رسیدگی درخواست­ها و تخصیص منابع (سمت راست) [7].*

*از نقطه نظر مصرف انرژی، ما یک دستگاه که دارای دو حالت فعال و آماده‌به‌کار است را در نظر می­گیریم. یک دستگاه فعال تمام لایه­های نرم­افزار و مؤلفه­های نشان داده‌شده در شکل 2 را اجرا می­کند و بنابراین سطح بالایی از انرژی را مصرف می­کند، درحالی‌که یک دستگاه آماده‌به‌کار، همه مؤلفه­های در شکل 2 را اجرا نمی­کند و در نتیجه مصرف انرژی آن کم یا ناچیز است. در این کار به خاطر دلایل آورده شده در [14]، ما خودمان را محدود به حالت آماده‌به‌کار می­کنیم، با دانستن اینکه استاندارد ACPI سطوح مختلف حالت آماده‌به‌کار را تعریف می­کند [15]. حالت آماده‌به‌کار در کار ما، می­تواند به‌عنوان حالت ACPI G2 در مشخصات ACPI تحقق یابد. دلیل این است که حالت، فعال­سازی یک دستگاه راه دور از طریق یک بسته شبکه[[24]](#footnote-24) را اجازه می­دهد. هر دستگاه در ابر، با خدمات مخزن[[25]](#footnote-25) دستگاه نشان داده‌شده در شکل 3 ثبت می­شود و ردیابی حالت انرژی دستگاه را نگه می­دارد، به‌عنوان مثال، فعال یا آماده‌به‌کار.*

**

*شکل 3. خدمات مخزن دستگاه*

*مؤلفه مدیر منابع تعیین می­کند که آیا یک دستگاه می­تواند به حالت آماده‌به‌کار برود یا یک دستگاه اضافی برای فعال شدن نیاز است. در مورد اول، آن یک پیام تعویض به حالت آماده‌به‌کار، به خدمات مخزن دستگاه می­فرستد که متعاقباً دستگاه را به حالت آماده‌به‌کار تعویض می­کند. در مورد دوم، آن پیام فعال کردن دستگاه را به خدمات می­فرستد، که شناسه یک دستگاه فعال را اگر یکی در دسترس باشد، بر می­گرداند. در ادامه این مقاله به قابلیت مؤلفه مدیر منابع تمرکز می­شود. برای دیگر مؤلفه­های معماری ما، مانند مدیر پوشش و ایجادکننده مشخصات تقاضا، ما به راه­حل­های شناخته شده تکیه می­کنیم. یک طراحی برای خدمات مخزن دستگاه، بخشی از کارهای آینده ما است.*

*3. مدل­سازی تخصیص منابع و راه­حل عمومی ما*

*برای این کار، ما یک ابر که دارای منابع محاسباتی (مانند پردازنده) و منابع حافظه است را در نظر می­گیریم که در دستگاه­های زیرساخت ابر موجود است. ما دستگاه­های موجود در ابر را همگن فرض می­کنیم، به این معنا که پردازنده و ظرفیت حافظه آن‌ها و نیز خواص مصرف انرژیشان یکسان هستند.*

*ما بحث را به موردی که در آن تمامی دستگاه­ها به یک دسته (خوشه) تک متعلق است محدود می­کنیم، و به‌عنوان همیار[[26]](#footnote-26) در وظیفه تخصیص منابع همکاری می­کنند. مسئله خاصی که ما به آن می­پردازیم، قرار دادن ماژول­ها (دقیق­تر، نمونه­های یکسان ماژول­ها) در دستگاه­ها و تخصیص منابع ابر به این ماژول­ها، به‌طوری‌که اهداف ابر به دست آید می­باشد.*

*ما مسئله مدیریت را به‌عنوان یک مسئله بهینه­سازی که در آن راه­حل یک ماتریس پیکربندی است که مؤلفه­های زمان‌بند ماژول و ارسال­کننده درخواست را کنترل می­کند، مدل می­کنیم. در زمان گسسته، رویدادهایی رخ می­دهد، مانند تغییرات بار، افزودن و حذف سایت یا دستگاه و غیره. در پاسخ به چنین رویدادهایی، مسئله بهینه­سازی به‌منظور حفظ پیکربندی بهینه، دوباره حل می­شود. ما مدلمان برای تخصیص منابع را در بخش 3-الف معرفی می­کنیم و الگوریتم عمومی برای مدیریت منابع را در بخش 3-ب ارائه می­هیم.*

**A*.*** *مدل*

*ما ابر را به‌عنوان یک سیستم با مجموعه­ای از سایت ها (S) و مجموعه­ای از دستگاه­ها (N) که سایت­ها را اداره می­کنند، مدل می­کنیم. هر سایت از مجموعه­ای از ماژول­ها که با Ms نشان داده می­شود، تشکیل شده است، و مجموعه­ای از تمام ماژول­های موجود در ابر، است.*

*ما تقاضا پردازنده را با بردار و تقاضا حافظه را با بردار مدل می­کنیم، با این فرض که تقاضا پردازنده وابسته به زمان است، درحالی‌که تقاضا حافظه وابسته به زمان نیست [16].*

*ما یک سیستم را در نظر می­گیریم که ممکن است بیش از یک نمونه از ماژول m که هر یک در دستگاه مختلف است را اداره کند. که در این صورت تقاضا پردازنده آن، در میان نمونه­هایش تقسیم می­شود. تقاضا از یک نمونه m در حال اجرا در دستگاه n با نشان داده می­شود که در آن و است. ما ماتریس A با عناصر را پیکربندی (ماتریس) سیستم می­نامیم. A یک ماتریس غیر منفی با است.*

*یک دستگاه در ابر، دارای ظرفیت پردازنده و ظرفیت حافظه است. ما و را برای نشان دادن بردارهای ظرفیت پردازنده و حافظه تمامی دستگاه­ها در سیستم استفاده می­کنیم. یک نمونه m در حال اجرا در دستگاه n، منبع پردازنده و منبع حافظه از n را تقاضا می­کند. دستگاه n برای ماژول m ظرفیت پردازنده (که ممکن است با متفاوت باشد) و ظرفیت حافظه را تخصیص می­دهد.*

*مقدار برای به سیاست تخصیص در ابر بستگی دارد. سیاست خاصی که ما در این استفاده کردیم، تخصیص است.*

**B*.* GRMP*: پروتکل مدیریت منابع عمومی[[27]](#footnote-27)***

*با توجه به مدل بالا، ماتریس پیکربندی A تعیین می­کند که چگونه منابع ابر به سایت­ها تخصیص داده می­شود. ما طرفدار استفاده از پروتکل gossip برای محاسبه کارآمد این ماتریس برای ابر مقیاس بزرگ هستیم. پروتکل­های gossip از پروتکل­های مبتنی بر دور[[28]](#footnote-28) هستند که در آن، در هر دور، یک گره[[29]](#footnote-29) یک زیرمجموعه از گره­های دیگر را برای تعامل انتخاب می­کند. انتخاب گره اغلب به‌صورت احتمالی است و به‌عنوان گره­هایی که با دور­های بیشتر عمل می­کند، حالت آن‌ها به یک حالت مطلوب همگرا می­شود. پروتکل gossip برای تعدادی وظایف مدیریت شامل انتشار اطلاعات با یک روش قوی، محاسبه تراکم[[30]](#footnote-30)، و نیز ایجاد و حفظ پوشش ارائه شده است.*

*در این زیر بخش، ما یک پروتکل gossip عمومی برای تخصیص منبع معرفی می­کنیم که می­تواند برای اهداف مدیریت مختلف با نمونه معرفی شود. ما این را پروتکل GRMP (پروتکل مدیریت منابع عمومی) می­نامیم. GRMP در مؤلفه مدیر منبع تمام دستگاه­ها موجود در ابر اجرا می­شود. (شکل 2 را ببینید). مجموعه­ای از دستگاه­های منتخب برای تعامل، توسط مؤلفه مدیر پوشش از مدیر دستگاه حفظ می­شود.*

*GRMP در زمان گسسته استناد شده است. بسته به استقرار خاص، فراخوانی[[31]](#footnote-31) ممکن است دوره­ای، در پاسخ به یک رویداد (مانند تغییر قابل توجه بار یا اضافه شدن دستگاه جدید)، یا ترکیبی از هر دو باشد. در طی هر فراخوانی GRMP، هر دستگاه rmax دور را انجام می­دهد و ماتریس پیکربندی A را تولید می­کند. مقدار برای rmax بستگی به نمونه خاص GRMP دارد. ماتریس A در سراسر دستگاه­های سیستم توزیع می­شود و شروع و توقف نمونه­های ماژول را کنترل می­کند و سیاست­های کنترل برای زمان‌بند ماژول و ارسال­کننده درخواست را تعیین می­کند. مؤلفه مدیر منبع تعیین می­کند که آیا ماتریس پیکربندی محاسبه‌شده، اجرا شده است یا نه. ما فرض می­کنیم که زمان آن برای GRMP به‌منظور محاسبه یک پیکربندی جدید، A، در مقایسه با زمان بین رویداهایی که باعث اجرای متوالی پروتکل می­شود، کم است. در زمان مقداردهی اولیه، GRMP یک پیکربندی امکان­پذیر سیستم را به‌عنوان ورودی می­خواند که می­تواند با استفاده از [7] و [17] محاسبه شود. در فراخوانی بعد، پروتکل، پیکربندی ماتریس تولیدشده در طی اجرای قبلی را به‌عنوان ورودی می­خواند.*

*شبه کد GRMP در الگوریتم 1 مشخص شده است. پروتکل، به اصطلاح الگوی تعامل فشار-کشش[[32]](#footnote-32) gossip را پیروی می­کند که ما با رشته[[33]](#footnote-33) فعال یا غیر فعال[[34]](#footnote-34) در هر دستگاه پیاده­سازی می­کنیم. برای حفظ سادگی ارائه، همزمانی رشته را حذف می­کنیم که از به روز رسانی متقارن حالت محلی، با رشته­های فعال و غیر فعال جلوگیری می­کند.*

***الگوریتم 1. پروتکل GRMP ماتریس پیکربندی A را محاسبه می­کند. کد برای دستگاه n است.***

******

 *GRMP یک پروتکل عمومی است، به این معنا که سه روش انتزاعی (غیر عملی) باید به‌منظور محاسبه یک ماتریس پیکربندی برای یک هدف مدیریت منابع خاص اجرا شود.*

*1) initInstance() روش مقداردهی اولیه برای پروتکل gossip خاص است.*

*2) choosePeer() روشی برای انتخاب یک همیار برای تعامل gossip است.*

*3) updatePlacement() روشی برای محاسبه دوباره حالت محلی در طی یک تعامل gossip است.*

*در زیر بخش 4-ب، ما یک نمونه­ای از GRMP که GRMP-Q نامیده می­شود را ارائه می­دهیم، که تخصیص منابع را باهدف کاهش مصرف انرژی انجام می­دهد. پروتکل gossip که ما در کار قبلیمان توسعه دادیم، می­تواند به‌عنوان یک نمونه از GRMP تفسیر شود [7]. آن پروتکل هدف تخصیص عادلانه منابع پردازنده به سایت­ها را انجام می­دهد. درحالی‌که روش­های initInstance() و choosePeer() در یک روش مشابه مانند اونهایی که برای GRMP-Q بود اجرا می­شوند، معنی updatePlacement() متفاوت است. این، حالت­های محلی دستگاه­های در حال تعامل را در چنین راهی که درخواست­های نسبی پردازنده به روز رسانی می­کند، به‌صورت برای n دستگاه محاسبه می­شود، برابر هستند. این پروتکل تحت شرایط خاصی بهینه است، به این معنی که پروتکل دنباله­ای از پیکربندی­ها تولید می­کند، هنگامی­که دورهای اجرا، به حالت بهینه به‌صورت نمایی تند همگرا می­شود.*

1. *مسئله و راه­حل ما*

**A.** *مدیریت منابع به‌عنوان یک مسئله بهینه­سازی*

*هدف اول، برآوردن درخواست کاربر است، اگر با منابع خوشه در دسترس ممکن است (برای مثال، زیر بار) و تخصیص عادلانه منابع اگر با منابع خوشه در دسترس ممکن نیست (برای مثال، سربار). ما این را با استفاده از مفهوم سودمندی[[35]](#footnote-35) رسمی می­کنیم. ما سودمندی تولیدشده توسط یک نمونه از ماژول m روی دستگاه n را به‌عنوان نسبت ظرفیت پردازنده تخصیص‌یافته به درخواست نمونه در آن دستگاه خاص تعریف می­کنیم. یعنی . (نمونه‌ای با ، سودمندی بی‌نهایت تولید می­کند). سودمندی تولیدشده توسط یک سایت به‌صورت تعریف می­شود. سودمندی ابر به‌صورت تعریف می­شود. هدف اول می­تواند به‌عنوان حداکثرسازی بیان شود، که تضمین می­کند که تمام درخواست­های سایت، در صورت زیر بار، برآورده می­شود. در صورت سربار، حداکثرسازی ، حداکثر-حداقل عادلانه[[36]](#footnote-36) در مورد تخصیص منابع پردازنده برای سایت­ها را تضمین می­کند.*

*هدف دوم، حداقل سازی مصرف انرژی ابر است. ما مصرف انرژی دستگاه n را با تابع زیر مدل می­کنیم.*

 *به این معنی است که دستگاه می­تواند به حالت آماده­به­کار تعویض شود، و به این معنی است که دستگاه باید فعال بماند. ما مصرف انرژی ابر را با بیان می­کنیم. بنابراین هدف دوم حداقل سازی است.*

*مسئله تخصیص منابع تطبیق پیکربندی A(t) به پیکربندی جدید A(t+1) است، به‌طوری‌که هدف سیستم مدیریت منابع برای تقاضا جدید به دست آید. هدف سوم شناسایی یک پیکربندی است که تابع هزینه را حداقل کند. این تابع هزینه جریمه مرتبط با تغییر پیکربندی A(t) به A(t+1) را حساب می­کند. چنین جریمه­ای ممکن است منعکس‌کننده باشد. برای مثال، سطح بالایی از مصرف پهنای باند شبکه یا یک‌زمان وقفه سرویس طولانی در طول پیکربندی دوباره. ( تابع هزینه­ای که ما در این کار در نظر می­گیریم، تعداد نمونه­هایی از ماژول­ها را می­شمارد که برای پیکربندی مجدد سیستم از پیکربندی فعلی به پیکربندی جدید شروع می­شوند).*

*حالا ما مسئله بهینه­سازی را با استفاده از سه هدف بحث شده در بالا رسمی می­کنیم. یک ابر با ظرفیت پردازنده و ظرفیت حافظه را در نظر بگیرید. بنابراین با توجه به پیکربندی A(t)، بردار تقاضا پردازنده و بردار تقاضا حافظه ، مسئله، یافتن یک پیکربندی A(t+1) است که مسائل بهینه­سازی زیر را حل کند.*

*(OP)*

*:به قسمی که*

*این مسئله بهینه­سازی اهداف را اولویت­بندی کرده است. این به این معنی است که، بین تمام پیکربندی­های A که سودمندی ابر را حداکثر می­کند، ما آن پیکربندی­هایی که مصرف انرژی را حداقل می­کند انتخاب می­کنیم. خارج این پیکربندی­ها، ما آن را که تابع هزینه را حداقل می­کند انتخاب می­کنیم. محدودیت­های (OP) مربوط به (1) تقسیم تقاضا پردازنده هر ماژول به تقاضا از نمونه­های ماژول، و (2) تضمین اینکه منابع پردازنده و حافظه تخصیص‌یافته در هر دستگاه نمی­تواند بزرگ‌تر از ظرفیت در دسترس آن باشد.*

 *اجازه دهید به‌طور خلاصه در مورد سختی (OP) توضیح دهیم. تقاضا حافظه برای یک ماژول قابل‌تقسیم نیست، به این معنی که تقاضا حافظه یک ماژول نمی­تواند در بین نمونه­هایش که در دستگاه­های مختلف اجرا می­شوند، تقسیم شود. این باعث NP-سخت[[37]](#footnote-37) شدن (OP) می­شود. بااین‌حال، در بسیاری موارد عملی که در آن‌ها تقاضا حافظه ترکیب‌شده، به‌طور قابل‌توجهی کوچک­تر از ظرفیت حافظه ابر است، یک راه­حل برای (OP) به‌راحتی می­تواند یافت شود.*

**B***. راه­حل* ***GRMP-Q*** *ما: یک راه­حل ابتکاری برای (****OP****)*

*به‌عنوان یک نمونه از GRMP، GRMP-Q سه روش انتزاعی GRMP، نشان داده‌شده در الگوریتم 2 را پیاده­سازی می­کند. در روش initInstance، دستگاه n مقداردهی اولیه می­کند Nn را، که Nn مجموعه­ای از دستگاه­ها است که ماژول­های مشترک با n را اجرا می­کنند. یک دستگاه n ترجیح می­دهد به اجرای مرحله gossip با دیگر دستگاه­های . دلیل این است که، بار می­تواند بین دو دستگاه بدون نیاز به حافظه اضافی و بدون هیچ هزینه پیکربندی دوباره، حرکت کند. بااین‌حال، همیشه انتخاب j از Nn، ممکن است در ابر منجر به تقسیم شدن در مجموعه­های متلاشی دستگاه­های در حال تعامل شود. برای جلوگیری از این وضعیت، n گاهی اوقات با یک دستگاه خارج از مجموعه Nn جفت می­شود. تابع انتخاب همسایه choosePeer()، این حالت را به‌صورت زیر پیاده­سازی می­کند: آن، یک دستگاه که به‌صورت یکنواخت تصادفی از مجموعه Nn با (قابل پیکربندی) احتمال p و از مجموعه N-Nn با احتمال 1-p را برمی­گرداند.*

*هسته پروتکل در تابع updatePlacement پیاده­سازی می­شود که نمونه­های ماژول را از یک دستگاه به دستگاه دیگر حرکت می­دهد. هدف از این حرکت، توسط تقاضا پردازنده نسبی دستگاه­های شرکت­کننده مشخص می­شود که برای دستگاه n به‌عنوان تعریف می­شود. به‌طور خاص، برای دستگاه­های n و j، اگر ، پروتکل تخمین می­زند که ابر در حالت سربار است و یک تابع باهدف دستیابی به عدالت برای منابع پردازنده فراخوانده می­شود. این تابع (که در [7] شرح داده‌شده) ماژول­ها را از دستگاه با تقاضای نسبی بالاتر به دستگاه با تقاضای نسبی پایین­تر، باهدف تساوی و حرکت می­دهد.*

*اگر ، پروتکل تخمین می­زند که ابر در حالت زیر بار است و توابع باهدف کاهش مصرف انرژی ابر فراخوانده می­شوند، درحالی‌که تقاضای سایت­ها برآورده شود. این توابع، packNonShared که همیشه فراخوانده می­شود، و packshared که تنها اگر دو دستگاه ماژول­ها را به اشتراک بگذارند فراخوانده می­شود، هستند. توابع بر مبنای دو مفهوم زیر هستند.*

*مفهوم اول، که با تابع pickSrcDest پیاده­سازی می­شود، تضمین می­کند که پروتکل در درجه اول ماژول­ها را از دستگاه دارای سربار، به دستگاه دارای زیر بار حرکت می­دهد، باهدف برآوردن تقاضا نمونه­های ماژول بر روی دستگاه دارای سربار. از سوی دیگر، اگر هر دو دستگاه دارای زیر بار هستند، پروتکل ماژول­ها را از دستگاه با بار کمتر به دستگاه با بار بالاتر حرکت می­دهد، در تلاش برای بسته کردن[[38]](#footnote-38) کامل یک دستگاه یا آزاد کردن[[39]](#footnote-39) دستگاه دیگر.*

*مفهوم دوم مربوط به بهره­وری بسته­بندی پروتکل است. به‌طور خاص، آن تلاش می­کند برای جلوگیری از شرایطی که در آن یک نوع تک از منبع (مانند فقط پردازنده یا حافظه) یک دستگاه استفاده می­شود، درحالی‌که دیگری استفاده نمی­شود. چنین شرایطی بهره­وری بسته­بندی پروتکل را کاهش می­دهد و ازاین‌رو باعث کاهش مصرف انرژی می­شود. بنابراین، در طول تعامل، پروتکل منابع غالب در مقصد را شناسایی می­کند (مانند نوع منبعی که دارای تقاضای نسبی بزرگ‌تر است)، و ماژول­ها در دستگاه منبع را انتخاب می­کند، به‌طوری‌که آن‌ها منابع غالب کمتر دارند. (در شبه کد، تقاضای حافظه نسبی به‌عنوان تعریف می­شود). توصیف کامل تابع در [18] موجود است.*

***الگوریتم 2. پروتکل* GRMP-Q*. یک نمونه از* GRMP *برای حل* (OP). *کد برای دستگاه n است.***

******

***C****. خصوصیات* **GRMP-Q**

*ازآنجاکه GRMP-Q یک راه­حل ابتکاری است، پیکربندی که آن تولید می­کند، به‌طورکلی برای (OP) بهینه نیست. برای درک خصوصیات پروتکل، ما مفاهیم مفید را معرفی می­کنیم: ضریب بار پردازنده[[40]](#footnote-40) و ضریب بار حافظه[[41]](#footnote-41) . ابر در حالت سربار است، هرگاه . به این معنا که تقاضای کل برای منابع پردازنده بیش از ظرفیت موجود در ابر است. (این مقاله مورد را شامل نمی­شود، زیرا مکان اولیه برای چنین بار در ابر ممکن نیست و تقاضاهای حافظه ثابت فرض شده است).*

*الف) ابر در حالت سربار ( و ): پروتکل در یک راهی که تمام دستگاه­های موجود در ابر، درنهایت سربار می­شوند. هنگامی‌که این مورد است، پروتکل به همان صورت پروتکل منصفانه شرح داده‌شده در [7] عمل می­کند، به این معنی که آن برای تخصیص منبع پردازنده در سراسر سایت­ها با استفاده از سیاست منصفانه حداکثر-حداقل تلاش می­کند.*

*ب) تقاضا حافظه بسیار کمتر از ظرفیت (): بعد از هر تعامل gossip، دستگاه­های در حال تعامل در یکی از حالت­های زیر هستند: (1) هر دو دستگاه دارای بار برابر هستند. (2) یک دستگاه حداکثر بار پردازنده حمل می­کند. (3) یک دستگاه هیچ باری حمل نمی­کند. تحت این شرایط، پیکربندی محاسبه‌شده توسط پروتکل، به یک راه­حل بهینه از (OP) همگرا می­شود—اگر ما هزینه پیکربندی دوباره را نادیده بگیریم. اگر ، راه­حل بهینه اشاره دارد به اینکه، تعداد دستگاه حداکثر بار را حمل می­کنند و تعداد دستگاه هیچ باری را حمل نمی­کنند، درحالی‌که تمام تقاضای سایت­ها برآورده می­شود.*

*ج) حالت کلی ( و ): با طراحی، پروتکل اولویت را به حرکت بار، دور از دستگاه سربار شده روی بار در حال انتقال، به‌منظور کاهش مصرف انرژی می­دهد. به‌عنوان نتیجه، ما می­توانیم بگوییم که اگر پیکربندی جدید، پروتکل دستگاه­هایی که بار حمل نمی­کنند را تولید کند، دستگاه­های با بار، به‌طور کامل تقاضا را برآورده می­کنند.*

*5. ارزیابی از طریق شبیه­سازی*

*ما GRMP-Q را از طریق شبیه­سازی گسترده با استفاده از شبیه­ساز رویداد گسسته شبیه­سازی کرده­ایم که ما در سازمان[[42]](#footnote-42) گسترش دادیم. ما یک سیستم توزیع‌شده را شبیه­سازی می­کنیم که مؤلفه­های مدیر دستگاه تمام دستگاه­های موجود در ابر را اجرا می­کند. به‌طور خاص، این مدیرهای دستگاه­ها، پروتکل GRMP-Q را اجرا می­کنند، که تخصیص ماتریس A را محاسبه می­کند، و همچنین پروتکل CYCLON، که تابع انتخاب یک همسایه تصادفی را برای GRMP-Q فراهم می­کند. شبیه­ساز همچنین الگوریتم شرح داده‌شده در [7] برای محاسبه پیکربندی امکان­پذیر اولیه ابر را پیاده­سازی می­کند. رویدادهای خارجی برای این شبیه­سازی، تغییرات در بردار تقاضای هستند.*

*معیارهای ارزیابی: ما کاهش مصرف انرژی را به‌عنوان کسری از دستگاه­های موجود در ابر اندازه می­گیریم که توسط پروتکل آزاد می­شوند. دوم، ما عدالت تخصیص منابع را از طریق ضریب تغییرات سودمندی سایت، که به‌عنوان نسبت انحراف معیار به متوسط سودمندی­ها محاسبه می­شود، اندازه می­گیریم. سوم، ما تقاضای برآورده شده را به‌عنوان کسری از سایت­ها که سودمندی بزرگ‌تر یا مساوی 1 تولید می­کند، اندازه می­گیریم. درنهایت، ما هزینه پیکربندی دوباره را به‌عنوان نسبت نمونه­های ماژول شروع‌شده به نمونه­های ماژول در حال اجرا، در هر دستگاه، اندازه می­گیریم.*

*تولید بردار تقاضای و : تعدادی از ماژول­های یک سایت، با استفاده از توزیع گسسته پواسون با میانگین 1، افزایش‌یافته با 1، انتخاب می­شود. تقاضای حافظه یک ماژول، به‌طور یکنواخت تصادفی از مجموعه انتخاب می­شود. {128 مگابایت، 256 مگابایت، 512 مگابایت، 1 گیگابایت، 2 گیگابایت}. برای سایت s، در هر تغییر در تقاضا، ایجادکننده مشخصات تقاضا، تقاضاهای پردازنده انتخاب‌شده از توزیع نمایی با میانگین را تولید می­کند. ما توزیع را برای ، در میان تمام سایت­ها، برای توزیع زیپف[[43]](#footnote-43) شدن با انتخاب می­کنیم. اثبات در [19] موجود است. حداکثر مقدار برای توزیع، است. 500 گیگ واحد پردازنده و اندازه جمعیت استفاده‌شده، 20000 است. برای ماژول m از سایت s، ما فاکتور تقاضا با ، انتخاب‌شده به‌صورت یکنواخت تصادفی را انتخاب می­کنیم، که سهم ماژول m در تقاضای سایت s را توصیف می­کند. و فاکتورهای مقیاس هستند (پایین را ببینید).*

*پارامترهای سناریو: ما عملکرد تخصیص منابع پروتکل GRMP-Q را تحت نمونه­های متفاوت بار پردازنده و حافظه، ارزیابی می­کنیم، که ما با تغییر و دگرگون می­کنیم. تمام دستگاه­های موجود در ابر دارای ظرفیت پردازنده یکسان 34.513 گیگ واحد پردازنده، و ظرفیت حافظه 36.409 گیگابایت هستند. این مقادیر، MLF=CLF=0.5 برای را می­دهد. ما پارامترهای زیر را استفاده می­کنیم، مگر اینکه غیر این‌ها ذکرشده باشد:*

* *|N|=10000، |S|=24000، rmax=30،*
* *حداکثر تعداد نمونه­ها/ماژول: 100 و تعداد تغییرات بار در طول اجرا: 100*

**A***. عملکرد* GRMP-Q *تحت* CLF *و* MLF *مختلف*

*در این سناریو، ما عملکرد GRMP-Q را برای CLF={0.1,0.4,0.7,1,1.3} و MLF={0.1,0.3,0.5,0.7,0.9}، توسط اندازه­گیری معیارهای لیست­شده در بالا، ارزیابی می­کنیم. ما نتایجمان را با یک سیستم ایده آل که دارای تراکم ظرفیت حافظه و پردازنده ابر است و اینکه انرژی را طبق تابع مصرف می­کند، مقایسه می­کنیم. ( یک کران پایین برای است که یک تخمین خوب از مقدار بهینه برای ، مقدار پایین MLF است). در این مقاله، نتایج مربوط به کاهش انرژی و تقاضای برآورده شده را گزارش می­کنیم. ارزیابی کامل پروتکل، در [18] موجود است.*

*1) کاهش مصرف انرژی: شکل 4الف، کاهش در مصرف انرژی به‌دست‌آمده توسط GRMP-Q برای مقادیر مختلف CLF و MLF را نشان می­دهد. همان‌طور که انتظار می­رفت، این مقدار برای افزایش CLF و MLF، کاهش می­یابد. برای نمونه، کاهش در مصرف انرژی، از 85% برای CLF=MLF=0.1 به 0 برای و ، را کاهش می­دهد. این ازآنجا مورد انتظار است که تعداد دستگاه­های موردنیاز برای اجرا و برآورده کردن تقاضاهای همه سایت­ها، با هر دو CLF و MLF افزایش می­یابد. این کاهش همچنین به 0 کاهش می­یابد برای .*

*2) تقاضای برآورده شده: شکل 4ب نشان می­دهد که تقاضای برآورده شده به هر دو CLF و MLF بستگی دارد. برای سیستم ایده آل، تقاضای برآورده شده فقط به CLF بستگی دارد. به‌طور خاص، تقاضای همه سایت­ها، وقتی CLF کمتر از 1 است، برآورده می­شود، و در غیر این صورت برآورده نمی­شود. پروتکل ما بیشتر از 99% تقاضاهای سایت را در سناریو زیر بار برآورده می­کند، به‌جز مورد (CLF,MLF)=(0.7,0.7) و (CLF,MLF)=(0.7,0.9). همان‌طور که دیده می­شود، برای مقادیر CLF بزرگ‌تر از 1، پروتکل ما یک تقاضای برآورده شده بزرگ‌تر از سیستم ایده آل، در مصرف ناعادلانه تخصیص پردازنده، به دست می­آورد.*

**

**

*ب. کسری از سایت­ها با تقاضای برآورده شده*

*شکل 4. عملکرد پروتکل تخصیص منبع GRMP-Q، در تابعی از ضریب بار پردازنده (CLF) و ضریب بار حافظه (MLF) از ابر (10000 دستگاه و 24000 سایت)*

*در این سناریو، ما وابستگی معیارهای ارزیابی خود را روی اندازه ابر، اندازه می­گیریم. برای رسیدن به این، ما شبیه­سازی را برای ابر با (2500، 5000، 10000، 20000، 40000، 160000) دستگاه و (6000، 12000، 24000، 48000، 96000، 384000) سایت انجام می­دهیم. (حفظ نسبت سایت­ها به دستگاه­ها در 2.4). در تنظیمات، ما دو مجموعه متفاوت از CLF و MLF که {(0.5,0.5),(0.25,0.25)} هستند را ارزیابی می­کنیم. شکل 5 نتایج به‌دست‌آمده را نشان می­دهد، که نشان می­دهد که تمام معیارهای در نظر گرفته‌شده، وابسته به‌اندازه سیستم هستند. به‌عبارت‌دیگر، اگر تعداد دستگاه­ها با نرخ یکسان از تعداد سایت­ها رشد کند، (درحالی‌که ظرفیت­های پردازنده و حافظه یک دستگاه، و همچنین تمام پارامترهای مشخص‌کننده یک سایت، مانند تقاضا، تعداد ماژول‌ها و غیره، یکسان بماند) ما انتظار داریم که تمام معیارهای در نظر گرفته‌شده ثابت باقی بماند. توجه داشته باشید که نتیجه­گیری ما به‌طور انحصاری به مقیاس­پذیری پروتکل GRMP-Q مربوط است. سیستم مدیریت منابع کامل، شامل بسیاری توابع بیشتر است که در اینجا مورد ارزیابی قرار نگرفته است، برای نمونه، مقیاس­پذیری انتخاب مؤثر یک همیار تصادفی.*

**

تعداد دستگاه­ها × 1000

*شکل 5. مقیاس­پذیری با توجه به تعداد دستگاه­ها و ماشین­ها.*

**6.** *کارهای مشابه*

*مسئله کاهش مصرف انرژی یک مرکز داده تحت محدودیت­های عملکرد، به‌طور گسترده موردمطالعه قرارگرفته است [5]، [20]-[27] و همچنین تولید راه­حل­هایی که یک راه­حل را برای چنین مسئله­ای ترکیب می­کند وجود دارد [4]. فاکتور کلیدی تفاوت کار ما، در مقایسه با تمام‌کارهای دیگر، استفاده از یک الگوریتم غیرمتمرکز برای محاسبه سیاست­های تخصیص منبع در ابر است. این، در تضاد تند با راه­حل­های موجود، اجازه می­دهد تا سیستم مدیریت منبع ما به 100000 دستگاه مقیاس کند، و به‌صورت پویا با تغییرات در تقاضای سایت­های در حال اجرا انطباق یابد. ارائه کامل کارهای مشابه، در [18] موجود است.*

**7.** *بحث و نتیجه­گیری*

*ما سه همکاری را با این مقاله ایجاد کردیدم. اول، ما مسئله حداقل سازی مصرف انرژی را از طریق تحکیم سرور، وقتی‌که سیستم در حالت زیر بار است و تخصیص منبع عادلانه در حالت سربار، معرفی و رسمی کردیم. دوم، ما GRMP را ارائه کردیم، که یک پروتکل gossip عمومی برای مدیریت منبع است که می­تواند برای اهداف مختلف معرفی شود. (یک پروتکل برای تخصیص منابع عادلانه از کار قبلی ما، درواقع یک نمونه از این پروتکل است). درنهایت، ما یک نمونه از GRMP را ارائه دادیم که یک راه­حل ابتکاری برای مسئله حداقل سازی مصرف انرژی فراهم می­کند، که ما برای مؤثر شدن و مقیاس­پذیر شدن نشان دادیم.*

*مطالعات شبیه­سازی GRMP-Q نشان می­دهد که پروتکل طبق اهداف طراحی‌اش که در بخش 1 گفته شد، برای محدوده پارامترهای بررسی‌شده، انجام می­دهد. برای نمونه، برای سناریو حالت زیر بار با CLF=MLF=0.1، پروتکل یک پیکربندی محاسبه می­کند که در آن کمتر از 20% دستگاه­ها، بار حمل می­کنند، درحالی‌که تقاضای کاربر هنوز برآورده می­شود. در سناریو حالت سربار، پروتکل منابع را به‌صورت عادلانه به سایت­ها تخصیص می­دهد، تا زمانی که حافظه کافی موجود است [18]. علاوه بر این، نتایج نشان می­دهند که پروتکل مقیاس­پذیر است، به این معنا که معیارهای کلیدی عملکردش با افزایش اندازه سیستم، تغییر نمی­یابد.*

*با توجه به کار آینده، ما طرح­ریزی می­کنیم برای (1) تعیین نرخ همگرایی GRMP-Q و وابستگی آن به تقاضاهای پردازنده و حافظه، (2) توسعه یک نسخه از پروتکل برای یک محیط ابر ناهمگن که در آن ظرفیت پردازنده و حافظه در سراسر دستگاه­ها متفاوت است، (3) توسعه یک مکانیسم توزیع‌شده که به‌صورت کارآمد سایت­های جدید را قرار می­دهد، (4) ایجاد پروتکل قوی برای خرابی دستگاه­ها، (5) توسعه نسخه­های GRMP اهداف بیشتری را پشتیبانی می­کند (مانند تمایز خدمات) و محدودیت­ (مانند اشتراک مکانی و ضد اشتراک مکانی[[44]](#footnote-44))، (6) توسعه یک اجرای مقیاس­پذیر از خدمات مخزن دستگاه که انرژی مصرفی را برای خنک‌کننده در نظر بگیرد.*

**REFERENCES**

[1] U.S. EPA, “Report to congress on server and data center energy efficiency public law 109-431,” 2007.

[2] The Climate Group, “Smart 2020: Enabling thelow carbon economy in theinformation age,” June 2008.

[3] Open Data Center Alliance, “Open data center alliance usage: carbon footprint values,” June 2011.

[4] VMWare, “VmwareR distributed power management white paper,” http://www.vmware.com/files/pdf/DPM.pdf.

[5] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, “Server workload analysis for power minimization using consolidation,” in USENIX’09. Berkeley, CA, USA: USENIX Association, 2009, pp. 28–28.

[6] U.S. EPA, “Working group notes from the EPA technical workshop on energy efficient servers and datacenters,” 2007.

[7] F. Wuhib, R. Stadler, and M. Spreitzer, “Gossip-based resource management for cloud environments,” in CNSM 2010, October, pp. 1 –8.

[8] OpenNebula Project Leads, “http://opennebula.org/.”

[9] OpenStack, “http://openstack.org/.”

[10] UC Santa Barbara, “http://appscale.cs.ucsb.edu/.”

[11] VMWare, “http://www.cloudfoundry.com/.”

[12] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi, “Dynamic estimation of CPU demand of web traffic,” in ValueTools 2006. New York, NY, USA: ACM, p. 26.

[13] Z. Gong, X. Gu, and J. Wilkes, “PRESS: PRedictive Elastic ReSource Scaling for cloud systems,” in CNSM 2010, October, pp. 9 –16.

[14] D. Meisner, B. T. Gold, and T. F. Wenisch, “PowerNap: eliminating server idle power,” SIGPLAN Not., vol. 44, pp. 205–216, March 2009.

[15] Hewlett-Packard, Intel, Microsoft, Phoenix Technologies Ltd., Toshiba Corporations, “Advanced configuration and power interface specification,” 2010.

[16] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguade, “Utility-based placement of dynamic web applications with fairness goals,” in IEEE NOMS, April 2008, pp. 9 –16.

[17] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, “A scalable application placement controller for enterprise data centers,” in WWW2007. New York, NY, USA: ACM, 2007, pp. 331–340.

[18] R. Yanggratoke, F. Wuhib, and R. Stadler, “Gossip-based resource allocation for green computing in large clouds (long version),” KTH Royal Institute of Technology, <https://eeweb01.ee.kth.se/upload/publications/reports/2011/> TRITA-EE 2011 036.pdf, Tech. Rep. TRITA-EE 2011:036, April 2011.

[19] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and Zipf-like distributions: evidence and implications,” in INFOCOM, vol. 1, 1999, pp. 126 –134.

[20] V. Petrucci, O. Loques, and D. Moss´e, “Dynamic optimization of power and performance for virtualized server clusters,” in ACM SAC 2010, 2010, pp. 263–264.

[21] G. Jung, M. Hiltunen, K. Joshi, R. Schlichting, and C. Pu, “Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures,” in ICDCS2010, 2010, pp. 62 –73.

[22] B. Speitkamp and M. Bichler, “A mathematical programming approach for server consolidation problems in virtualized data centers,” IEEE TSC, vol. 3, no. 4, pp. 266 –278, 2010.

[23] N. Tolia, Z. Wang, P. Ranganathan, C. Bash, M. Marwah, and X. Zhu, “Unified thermal and power management in server enclosures,” ASME Conference Proceedings, vol. 2009, no. 43604, pp. 721–730, 2009.

[24] M. Cardosa, M. Korupolu, and A. Singh, “Shares and utilities based power consolidation in virtualized server environments,” in IM 2009, pp. 327 –334.

[25] D. Gmach, J. Rolia, L. Cherkasova, G. Belrose, T. Turicchi, and A. Kemper, “An integrated approach to resource pool management: Policies, efficiency and quality metrics,” in Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on, june 2008, pp. 326 –335.

[26] C. Subramanian, A. Vasan, and A. Sivasubramaniam, “Reducing data center power with server consolidation: Approximation and evaluation,” in HiPC 2010, 2010, pp. 1 –10.

[27] J. Choi, S. Govindan, J. Jeong, B. Urgaonkar, and A. Sivasubramaniam, “Power consumption prediction and power-aware packing in consolidated environments,” Computers, IEEE Transactions on, vol. 59, no. 12, pp. 1640 –1654, 2010.

1. consolidation [↑](#footnote-ref-1)
2. Heuristic [↑](#footnote-ref-2)
3. Overload [↑](#footnote-ref-3)
4. Client [↑](#footnote-ref-4)
5. Datacenters [↑](#footnote-ref-5)
6. Workload [↑](#footnote-ref-6)
7. Virtualization [↑](#footnote-ref-7)
8. live migration [↑](#footnote-ref-8)
9. standby [↑](#footnote-ref-9)
10. Wakeup [↑](#footnote-ref-10)
11. Platform-as-a-Service [↑](#footnote-ref-11)
12. Use case [↑](#footnote-ref-12)
13. Cloud service provider [↑](#footnote-ref-13)
14. Host [↑](#footnote-ref-14)
15. Stakeholder [↑](#footnote-ref-15)
16. Hosting [↑](#footnote-ref-16)
17. middleware [↑](#footnote-ref-17)
18. Global [↑](#footnote-ref-18)
19. Overlay manager [↑](#footnote-ref-19)
20. Overlay graph [↑](#footnote-ref-20)
21. Profiler [↑](#footnote-ref-21)
22. QoS [↑](#footnote-ref-22)
23. Session affinity [↑](#footnote-ref-23)
24. Wake-on-LAN packet [↑](#footnote-ref-24)
25. Pool [↑](#footnote-ref-25)
26. Peer [↑](#footnote-ref-26)
27. Generic Resource Management Protocol [↑](#footnote-ref-27)
28. Round-based [↑](#footnote-ref-28)
29. Node [↑](#footnote-ref-29)
30. Aggregates [↑](#footnote-ref-30)
31. Invocation [↑](#footnote-ref-31)
32. Push-pull [↑](#footnote-ref-32)
33. Thread [↑](#footnote-ref-33)
34. Passive [↑](#footnote-ref-34)
35. Utility [↑](#footnote-ref-35)
36. Max-min fairness [↑](#footnote-ref-36)
37. NP-hard [↑](#footnote-ref-37)
38. Pack [↑](#footnote-ref-38)
39. Freeing up [↑](#footnote-ref-39)
40. CPU load factor [↑](#footnote-ref-40)
41. Memory load factor [↑](#footnote-ref-41)
42. In-house [↑](#footnote-ref-42)
43. Zipf [↑](#footnote-ref-43)
44. Colocation and anti-colocation [↑](#footnote-ref-44)