علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۵، شماره ۲، ۱۳۹۶
صفحات ۵۶- ۶۹
مقاله عادی

نشریه علوم رایانش و فناوری اطلاعات

بهبود توان مصرفی و زمان اجرا در رایانش ابری رهیافت تخصیص منبع و زمانبندی وظایف

عاطفه یکتا اول محمود فتحی

دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران

چکیده

جریان‌های کاری علمی یکی از راه‌های معروف به منظور مدلسازی برنامه‌های کاربردی ابر می‌باشند. یکی از چالشی‌ترین موضوعات تحقیق در این زمینه، زمانبندی وظایف این جریان کاری بر روی منابع در دسترس است به گونه‌ای که بتوانیم به حداقل زمان تکمیل اجرای وظایف دست یابیم. از طرفی، یکی از مسائل چالش‌برانگیز دیگر در رایانش ابر کاهش میزان توان مصرفی مراکز داده می‌باشد. مشکل موجود مغایرت این معیارهای کیفی است که الگوریتم‌های بسیاری سعی در بهینه‌سازی آنها دارند. در این مقاله به منظور کاهش توان مصرفی و زمان اجرا، تخصیص منبع را در دو سطح انجام داده‌ایم: ابتدا در سطح زیرساخت به گونه‌ای منابع فیزیکی را به ماشین‌های مجازی تخصیص می‌دهیم، که با افزایش بهره‌وری منابع کمترین میزان توان مصرفی را در مرکز داده داشته باشیم. در مرحله‌ی بعدی، برنامه‌های کاربردی را به گونه‌ای بر روی این ماشین‌های مجازی زمانبندی می‌کنیم که به حداقل زمان تکمیل وظایف دست یابیم. نتایج آزمایش‌ها بیانگر موفقیت این روش می‌باشد.

کلمات کلیدی: رایانش ابر، گراف وظایف، زمانبندی وظایف، تخصیص منابع، مدیریت توان.

۱- مقدمه

تاکنون الگوریتم‌های بسیاری در رابطه با نحوه‌ی تخصیص و یا زمانبندی منابع در رایانش ابر ارائه شده‌اند که هدف برخی از این الگوریتم‌ها بهینه‌سازی عاملی از کیفیت سرویس مانند کاهش زمان تکمیل وظایف و یا عواملی مانند کاهش توان مصرفی مراکز داده[۱] بوده است. جریان‌های‌کاری علمی[۲] یکی از راه‌های بسیار معروف و مرسوم به منظور مدلسازی برنامه‌های کاربردی[۳] می‌باشند به گونه‌ای که این برنامه‌ها بتوانند بر روی سیستم‌های توزیع شده نظیر ابر اجرا شوند. زمانی که یک جریان کاری تشکیل می‌شود، یکی از چالشی‌ترین موضوعات تحقیق در این زمینه این است که چگونه وظایف مختلف این جریان کاری را بر روی منابع در دسترس زمانبندی کنیم. زمانبندی موثر برنامه‌های کاربردی یک راه حیاتی برای دستیابی به کارایی بالا در محیط‌های محاسباتی ناهمگن می‌باشد. از طرفی یکی از مسائل عمده و چالش برانگیز دیگر در رایانش ابر کاهش میزان توان مصرفی مراکز داده است. براساس [۱]، گزارشات نشان می‌دهند که حدود ۵۰ درصد از بودجه‌ی مدیریتی مرکز داده‌ی آمازون صرف هزینه‌هایی برای خنک‌سازی ماشین‌های فیزیکی می‌شود. در این میان با وجود بهبودها و بهینه‌سازی‌هایی که در بهره‌وری[۴] انرژی سخت‌افزار صورت گرفته است، مصرف انرژی مراکز داده به خاطر افزایش نیاز به منابع رایانشی روز به روز در حال افزایش است و این افزایش مصرف انرژی به دلیل تاثیری که بر روی هزینه‌های عملیاتی می‌گذارد یکی از نگرانی‌های عمده برای صاحبان مراکز داده محسوب می‌شود. مسلما به همراه داشتن پارامترهای کیفیت سرویس مانند کاهش زمان اجرای برنامه‌های کاربردی می‌تواند باعث افزایش توان مصرفی شود. مشکل موجود مغایرت این معیارهای کیفی است که این الگوریتم‌ها سعی در بهینه‌سازی آنها دارند. آن‌چه مسلم است این است که تخصیص منابع قسمت مهمی از مدیریت منابع است و انتخاب متدهای مناسب برای اختصاص یافتن بار کاری[۵] یا ماشین مجازی به میزبان مناسب راه رسیدن به هدف تخصیص بهینه منابع است.

در بخش بعدی مسئله‌ی زمانبندی وظایف، زمانبندی پویا و ایستا و نمونه‌هایی از الگوریتم‌های ارائه شده در این زمینه را معرفی می‌کنیم. در بخش ۳ تکنیک‌های کاهش توان مصرفی را بیان کرده و در بخش ۴ کارهای مرتبط با هدف کاهش زمان اجرا و کاهش توان مصرفی را مورد بررسی قرار داده و در بخش ۵ به مدلسازی مسئله می‌پردازیم. در بخش ۶ به تخصیص منبع در سطح زیرساخت با هدف کاهش توان مصرفی پرداخته و الگوریتم پیشنهادی آگاه به توان را معرفی می‌کنیم. در بخش ۷ زمانبندی برنامه‌ی کاربردی بر روی ماشین‌های مجازی و الگوریتم زمانبندی HEFT را ارائه داده و در بخش ۸ نتایج ارزیابی و شبیه‌سازی‌های خود را نشان می‌دهیم و در نهایت در بخش ۹ نتیجه‌ی بررسی‌های خود را بیان می‌کنیم.

۲- زمانبندی وظایف

به‌طور کلی مسئله‌ی زمانبندی وظایف شامل نگاشت وظایف یک برنامه‌ی کاربردی بر روی منابع مناسب و ترتیب دادن به اجرای وظایف بر روی هر منبع می‌باشد. مدل یک سیستم زمانبندی شامل یک برنامه‌ی کاربردی، یک محیط محاسباتی و یک معیار کارایی برای زمانبندی است. زمانی که مشخصات یک برنامه‌ی کاربردی شامل زمان اجرای وظایف، حجم داده‌ی انتقال داده شده بین وظایف و وابستگی بین وظایف، شناخته شود، ما آن را با یک مدل ایستا نشان می‌دهیم. در یک مسئله‌ی زمانبندی ایستا، یک برنامه‌ی کاربردی با یک گراف جهت‌دار بدون دور (DAG) [۶] نشان داده می‌شود، G(V,E)، که V مجموعه‌ای شامل n گره و E مجموعه‌ای از m یال می‌باشد که در آن گره‌ها[۷] نشان‌دهنده‌ی وظایف برنامه‌ی کاربردی و یال‌ها[۸] نشان دهنده‌ی وابستگی داده‌ای میان وظایف می‌باشند.

هر گره دارای برچسبی است که نشان دهنده‌ی هزینه‌ی محاسباتی (زمان محاسبه) آن وظیفه است و هر یال نیز دارای برچسبی است که نشان‌دهنده‌ی هزینه‌ی انتقال (زمان انتقال) داده‌های بین وظایف می‌باشد. هر یال $(i,j)\in E$ نشان‌دهنده‌ی محدودیت‌های اولویت است به گونه‌ای که وظیفه‌ی n_i می‌بایست اجرایش را قبل از اینکه وظیفه‌ی n_j شروع به کار کند، به پایان رساند. $Data_{ij}$ نیز نشان‌دهنده‌ی مقدار داده‌ی مورد نیاز برای انتقال از وظیفه‌ی n_i به وظیفه‌ی n_j می‌باشد. در یک گراف از وظایف، وظیفه‌ای که هیچ والدی نداشته باشد، وظیفه‌ی ورودی و وظیفه‌ای که هیچ فرزندی نداشته باشد، وظیفه‌ی خروجی نامیده می‌شود [۲].

۲-۱- تکنیک‌های زمانبندی وظایف

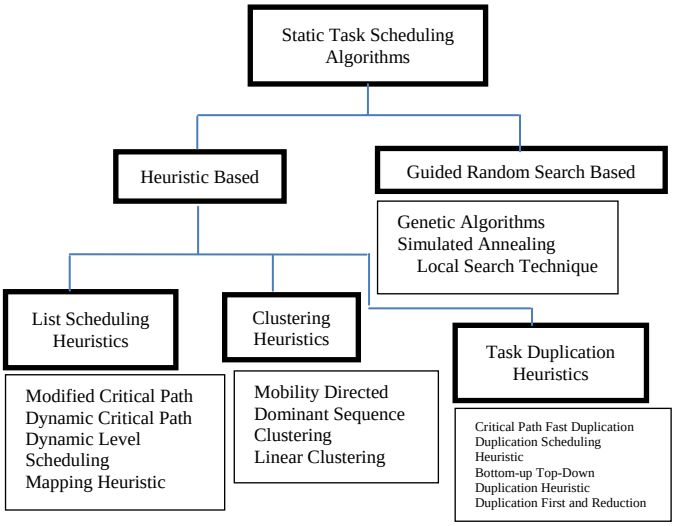
تکنیک‌های زمانبندی وظایف در سیستم‌های توزیع شده با هدف استفاده‌ی کارا و موثر از سیستم مورد مطالعه قرار گرفته‌اند. الگوریتم‌های زمانبندی وظایف عموماً به دو دسته تقسیم می‌شوند [۳] و [۴]:

۱- زمانبندی ایستای وظایف[۹]

۲- زمانبندی پویای وظایف[۱۰]

۲-۱-۱- زمانبندی ایستای وظایف

در زمانبندی ایستا وظایف به‌طور همزمان به پردازنده می‌رسند و بر روی منابع دردسترس زمانبندی می‌شوند. تصمیمات زمانبندی قبل از اینکه وظایف برسند گرفته می‌شود و زمان پردازش بعد از اجرای کامل یک وظیفه به‌روز می‌شود. این نوع زمانبندی اغلب برای وظایفی که به‌طور متناوب ارسال می‌شوند به کار گرفته می‌شود [۳]. اطلاعات در مورد هزینه‌ی اجرای وظایف و زمان انتقال وظایف در زمان کامپایل شناخته و مشخص می‌شود. الگوریتم‌های زمانبندی ایستا غیرانحصاری[۱۱] هستند و همواره بر روی منبعی که تخصیص داده شده‌اند اجرا می‌شوند. در برخی مواقع الگوریتم‌های زمانبندی ایستا نسبت به الگوریتم‌های زمانبندی پویا زمان اجرای کوتاهتری دارند و فقط اجازه‌ی پردازش بر روی یک پردازنده را می‌دهند [۴]. الگوریتم‌های زمانبندی ایستای وظایف می‌توانند به دو گروه اصلی دسته‌بندی شوند که هر یک از این دو گروه نیز خود شامل دسته‌بندی‌های دیگری می‌شوند [۲].



شکل ۱- دسته‌بندی الگوریتم‌های زمانبندی ایستای وظایف [۲]

۲-۱-۱-۱- زمانبندی ایستا مبتنی بر مکاشفه [۲]

مکاشفه‌های زمانبندی مبتنی بر لیست:

یک مکاشفه‌ی زمانبندی مبتنی بر لیست، لیستی از وظایف یک گراف داده شده را براساس اولویت میان آنها نگهداری می‌کند. این زمانبندی شامل دو فاز می‌شود: فاز اولویت‌بندی وظایف به منظور انتخاب وظیفه‌ی آماده با بالاترین اولویت و فاز انتخاب پردازنده به منظور انتخاب یک پردازنده‌ی مناسب که یک تابع هزینه‌ی از پیش تعیین شده را حداقل کند (که این تابع می‌تواند زمان آغاز اجرا باشد.) برخی از الگوریتم‌های ارائه شده در این زمینه در شکل بالا مشخص شده است.

اکثر الگوریتم‌های مبتنی بر لیست برای تعداد محدودی از پردازنده‌های ناهمگن متصل به یکدیگر می‌باشند. مکاشفه‌های زمانبندی مبتنی بر لیست به‌طور کلی سودمندتر بوده و نتایج کارایی بهتری را در زمان کمتری نسبت به سایر گروه‌ها فراهم می‌کنند.

مکاشفه‌های خوشه‌بندی:

الگوریتم‌های موجود در این دسته، وظایف یک گراف داده شده را به تعداد نامحدودی از خوشه‌ها نگاشت می‌دهند. در هر گام وظیفه‌ی انتخاب شده برای خوشه‌بندی می‌تواند هر وظیفه‌ای باشد و لزوما آن وظیفه، وظیفه‌ی آماده نیست. هر تکرار، خوشه‌بندی قبلی را با ترکیب کردن برخی خوشه‌ها تصحیح می‌کند. اگر دو وظیفه به یک خوشه تخصیص داده شوند، آن دو بر روی یک پردازنده اجرا خواهند شد. یک مکاشفه‌ی خوشه‌بندی، به گام‌های اضافی برای تولید یک زمانبندی نهایی احتیاج دارد: ۱- گام ترکیب خوشه به منظور ترکیب کردن خوشه‌ها به گونه‌ای که تعداد خوشه‌های باقی‌مانده مساوی با تعداد پردازنده‌ها باشد. ۲- گام نگاشت خوشه به منظور نگاشتن خوشه‌ها بر روی پردازنده‌های در

دسترس و ۳- گام مرتب کردن وظایف به منظور مرتب کردن وظایف نگاشت شده درون هر پردازنده.

مکاشفه‌های تکثیر وظیفه[۱۲]

ایده‌ی پشت این الگوریتم‌های زمانبندی این است که گرافی از وظایف را با نگاشت برخی از وظایفش زمانبندی کند که این کار سربار ارتباط میان پردازنده‌ها را کاهش می‌دهد. الگوریتم‌های تکثیر وظیفه مطابق با استراتژی انتخاب وظیفه برای عمل تکثیر متفاوتند. الگوریتم‌های این دسته معمولا برای تعدادی از پردازنده‌های همانند بوده و نسبت به الگوریتم‌های دیگر دسته‌ها پیچیدگی بیشتری دارند.

۲-۱-۱-۲- تکنیک‌های جستجوی تصادفی هدایت شده[۱۳] [۲]

این دسته از تکنیک‌ها از انتخاب تصادفی استفاده می‌کنند تا در میان فضای مسئله هدایت شوند. البته عملکرد این جستجو صرفا با یک جستجوی تصادفی که از طریق متدهای جستجو انجام می‌گیرد یکسان نمی‌باشد. این تکنیک‌ها دانش به دست آمده از نتایج جستجوی قبلی را با برخی از ویژگی‌های تصادفی ترکیب می‌کنند تا نتایج جدید تولید کنند. الگوریتم‌های ژنتیک مشهورترین و گسترده‌ترین تکنیک‌های استفاده شده برای مسائل زمانبندی وظایف می‌باشند. الگوریتم‌های ژنتیک کیفیت خوبی از خروجی زمانبندی تولید می‌کنند. اگرچه که دفعات زمانبندی آنها معمولا بسیار بیشتر از تکنیک‌های مبتنی بر مکاشفه است. علاوه بر این در یک الگوریتم ژنتیک، چندین پارامتر کنترلی باید به درستی تعیین شوند. مجموعه‌ی بهینه از پارامترهای کنترلی مورد استفاده برای زمانبندی یک گراف ممکن است برای گرافی دیگر جواب بهینه را ندهد. علاوه بر الگوریتم‌های ژنتیک، متدهای simulated annealing و جستجوی محلی، متدهای دیگر در این دسته هستند.

۲-۱-۲- زمانبندی پویای وظایف

در زمانبندی پویا، تعداد وظایف، مکان ماشین‌های مجازی و شیوه‌ی تخصیص منبع ثابت نیست و دفعات ورود وظایف قبل از ارسال آنها مشخص نیست [۲]. زمانبندی پویا، وظایف را بر روی منابع در زمان اجرا زمانبندی می‌کند تا به یک تعدیل بار میان المان‌های پردازشی دست یابد [۴].

در مقاله‌ی [۵] یک ایده‌ی زمانبندی پویا به نام DGS[۱۴] معرفی شده که به صورت پویا منابع مجازی را به وظایف محاسباتی تخصیص می‌دهد و با استفاده از استراتژی حریصانه‌ی[۱۵] بهبودیافته فرآیند زمانبندی و اجرا را تکمیل می‌کند.

در ابتدا اگر ناظر ماشین مجازی تشخیص داد که ماشین مجازی در مرکز داده وجود ندارد یا توان محاسباتی آن محدود است، ماشین‌های مجازی جدید ایجاد خواهند شد. بعد از اینکه اولین زمانبندی توسط استراتژی حریصانه صورت گرفت، اگر ناظر ماشین مجازی دید که برخی از ماشین‌های مجازی در حال اجرا در وضعیت بحرانی به سر می‌برند، ماشین‌های مجازی جدید ایجاد کرده و دومین زمانبندی را شکل خواهد داد. در حقیقت در بازه‌های زمانی از پیش تعیین شده مقدار بار هر ماشین مجازی محاسبه شده و این مقدار با مقدار بار آستانه مقایسه می‌شود. در صورتی که این مقدار از مقدار آستانه‌ی تعیین شده بیشتر باشد، ماشین مجازی دچار سربار شده و ناظر ماشین مجازی، ماشین مجازی دیگری دقیقا مشابه با ماشین مجازی قبلی ایجاد خواهد کرد. سپس ماژول زمانبندی وظایف، وظایفی را که در صف ماشین مجازی سربار شده قرار داشتند، به دو ماشین مجازی روشن تخصیص می‌دهد. در صورتی که باز هم سربار وجود داشت این روال تکرار خواهد شد.

۳- تکنیک‌های کاهش توان مصرفی

امروزه برنامه‌های کاربردی مدرن که کاربرد تجاری یا علمی دارند نیاز به زیرساخت‌های رایانشی با کارایی بالایی دارند. این منجر به مراکز داده رایانشی با مقیاس بسیار بالا می‌شود که مصرف توان الکتریکی بالایی دارند. با وجود بهبودها و بهسازی‌هایی که در بهره‌وری انرژی سخت‌افزار صورت گرفته است، مصرف انرژی کلی مرکز داده به خاطر افزایش نیاز به منابع رایانشی روز به روز در حال افزایش است. جدای از هزینه‌های سرسام‌آور کاربردی، ساختن مرکز داده‌ای که برای پاسخگویی به اوج بار یا حجم کاری درخواستی طراحی می‌شوند در حالی که بهره‌وری متوسط پایینی دارند از دلایل دیگر مصرف توان بالای مراکز داده می‌باشند [۶]. علاوه بر آن کمبود سیستم‌های خنک‌کننده یا خرابی آنها می‌تواند منجر به گرم شدن شدید منابع مرکز داده شده که باعث پایین آمدن قابلیت اطمینان سیستم و کاهش طول عمر دستگاه‌ها و تجهیزات داخل مرکز داده شود. علاوه بر آن مصرف توان بالای زیرساخت‌ها باعث انتشار قابل توجهی گاز کربن دی اکسید منجر به پدیده‌ی گلخانه‌ای می‌شود [۷].

تکنیک‌های سخت‌افزاری: از نقطه‌نظر سخت‌افزاری مدیریت ایستای انرژی شامل روش‌های بهینه‌سازی است که در زمان طراحی در سطوح مختلف سخت‌افزار شامل مدار، لاجیک و معماری سیستم به کار می‌روند. تکنیک‌های سخت‌افزاری مدیریت توان به صورت پویا متناسب با اجزای سخت‌افزاری فرق می‌کنند. اما این روش‌ها معمولا به دو دسته‌ی مقیاس‌پذیری پویای ولتاژ یا فرکانس و خاموش کردن اجزا به‌طور کامل یا جزیی در زمان‌های بیکاری یا عدم فعالیت سیستم تقسیم‌بندی می‌شوند. مقیاس‌پذیری پویای ولتاژ و فرکانس[۱۶] در بعضی از اجزای سخت‌افزاری نظیر پردازنده‌ها تعبیه شده است. با بهره‌گیری از این قابلیت پردازنده‌ها در زمان‌های کاهش بار کاری می‌توانند در وضعیت‌های عملیاتی با مصرف پایین توان قرار بگیرند. مصرف انرژی در مراکز داده با استفاده از این قابلیت فقط می‌تواند حدود ۱۰ تا ۲۰ درصد کاهش داده شود [۸].

تکنیک‌های نرم‌افزاری: روش‌های نرم‌افزاری مدیریت ایستای انرژی شامل استفاده از تکنیک‌های برنامه‌نویسی موازی، الگوریتم‌های چندنخی[۱۷]، بهینه‌سازی کد برنامه و بهینه‌سازی فرآیند کامپایل کد هستند که به هنگام توسعه‌ی برنامه‌های کاربردی استفاده می‌شوند. تکنیک‌های نرم‌افزاری مدیریت توان به صورت پویا شامل مکانیزم‌ها و سیاست‌هایی هستند که از دانش درباره‌ی وضعیت جاری سیستم استفاده می‌کنند. این تکنیک‌های نرم‌افزاری مطابق با سیاست‌های خود از روش‌های سخت‌افزاری مدیریت پویای توان استفاده می‌کنند.

به دلیل محدودیت‌های موجود بسیاری از کارهای مرتبط برای کاهش انرژی مصرفی از رویکرد متمرکزسازی بارهای کاری بر روی تعداد کمتری از سرورها و خاموش کردن سرورهای بیکار یا قرار دادن آنها در مدهای خواب یا خواب زمستانی و یا از ترکیب رویکرد متمرکزسازی با قابلیت مقیاس‌پذیری ولتاژ و فرکانس استفاده کرده‌اند [۸].

فناوری مجازی‌سازی: در کنار تکنیک‌های نرم‌افزاری و سخت‌افزاری برای مدیریت توان، فناوری مجازی‌سازی می‌تواند با بهبود درصد استفاده از منابع، توان مصرفی سرورها را کاهش دهد. فناوری مجازی‌سازی این امکان را فراهم می‌سازد که یک یا بیشتر ماشین‌های مجازی بتوانند بر روی یک سرور فیزیکی واحد اجرا شوند. به این ترتیب ضمن افزایش میزان استفاده از منابع، مقدار سخت‌افزار مورد نیاز برای اجرای برنامه‌های کاربردی کاهش می‌یابد. با بهره‌گیری از قابلیت مهاجرت[۱۸] ماشین‌های مجازی، امکان مدیریت پویای منابع سخت‌افزاری فراهم می‌شود. ماشین‌های مجازی می‌توانند در زمان اجرا بر روی حداقل سرور فیزیکی یکپارچه‌سازی شوند. در این حالت سرورهای بیکار می‌توانند در وضعیت خاموشی و یا مدهای عملیاتی کم‌مصرف قرار بگیرند. از آنجایی که انرژی مصرفی ماشین‌های

فیزیکی در حالت بیکاری، به عنوان مثال در حالت ۱۰ درصد استفاده از پردازنده، بیشتر از ۵۰ درصد انرژی مصرفی نسبت به حالت حداکثر بهره‌وری آنها می‌باشد [۹]، با استفاده از قابلیت متمرکزسازی ماشین‌های مجازی بر روی تعداد کمتری از ماشین‌های فیزیکی می‌توان تعداد ماشین‌های بیکار را کاهش و در نتیجه انرژی مصرفی را کمینه کرد.

۴- کارهای مرتبط با هدف کاهش زمان اجرا و کاهش توان مصرفی

الگوریتم‌های زمانبندی مبتنی بر لیست، مشهورترین الگوریتم‌ها در زمینه‌ی زمانبندی ایستای وظایف می‌باشند. این الگوریتم‌ها به هر وظیفه اولویتی نسبت داده و بر همین اساس وظایف را به ترتیب نزولی مرتب می‌کنند. سپس وظایف براساس این اولویت زمانبندی می‌شوند. سه مکاشفه‌ی زمانبندی مبتنی بر لیست که به منظور کاهش توان مصرفی ارائه شده‌اند عبارتند از:

1- Power aware list scheduling (PALS)
2- Power aware task clustering (PATC)
3- Low power heterogeneous-earliest-finish-time algorithm (LPHEFT)

- **الگوریتم PALS**

الگوریتم PALS که توسط لیزا وانگا[۱۹] و همکارانش [۴] معرفی شد، یک الگوریتم زمانبندی مبتنی بر لیست برای یافتن بهترین زمان پاسخگویی و یک مکاشفه‌ی زمانبندی آگاه از توان برای وظایف موازی ارائه می‌کند. در این الگوریتم وظایف از طریق الگوریتم زمانبندی ETF[۲۰] زمانبندی می‌شوند. همچنین به منظور کاهش توان مصرفی، ولتاژ پردازنده برای وظایفی که حساس نیستند و اهمیت زیادی ندارند با استفاده از الگوریتم Non-critical time slot voltage scaling مقیاس‌بندی شده و کاهش می‌یابد. این الگوریتم حداقل به میزان ۴۴.۳٪ منجر به کاهش توان مصرفی می‌شود.

- الگوریتم زمانبندی ETF:

الگوریتم ETF یک الگوریتم زمانبندی مبتنی بر لیست برای یافتن بهترین زمان پاسخگویی برای هر وظیفه می‌باشد. این الگوریتم به هر وظیفه اولویتی نسبت داده سپس وظایف آماده با بالاترین اولویت را انتخاب کرده و آنها را بر روی پردازنده‌ای که زودترین زمان شروع را دارد زمانبندی می‌کند.

- الگوریتم Non-Critical Time slot Voltage scaling:

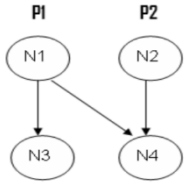
این الگوریتم نشان می‌دهد که چطور برای وظایف غیر حساس مقیاس‌بندی ولتاژ انجام می‌شود. برای هر پردازنده، تمامی بازه‌های زمانی جستجو می‌شود. زمانی که پردازنده بیکار است یا داده‌ای در یک بازه‌ی زمانی انتقال می‌یابد، الگوریتم فرکانس پردازنده را کاهش می‌دهد. هنگامی که در یک بازه‌ی زمانی یک وظیفه‌ی غیر حساس اجرا می‌شود، میزان زمان جای افتاده را محاسبه می‌کند، زمان اجرای آن وظیفه را به اندازه‌ی این زمان محاسبه شده گسترش می‌دهد و فرکانس پردازنده را به مقدار مناسب کاهش می‌دهد.

- **الگوریتم PATC**

الگوریتم PATC [۴] یک مکاشفه‌ی زمانبندی برای وظایف موازی است که با هدف کاهش توان مصرفی توسط لیزا وانگا و همکارانش معرفی شد. این الگوریتم در ابتدا تمام یال‌ها را به عنوان امتحان نشده علامت‌گذاری کرده و هر وظیفه را به یک شاخه‌ی جداگانه تخصیص می‌دهد. بعد از مرتب کردن تمامی یال‌ها به ترتیب نزولی زمان انتقال، الگوریتم به صورت تکراری وظایفی را که هزینه‌ی انتقال میان آنها بالاست با همدیگر ترکیب می‌کند البته این کار را در صورتی انجام می‌دهد که مصرف کلی توان افزایش پیدا نکند. این الگوریتم حداقل به میزان ۳۹٪ منجر به بهبود توان مصرفی می‌شود.

- **الگوریتم LPHEFT**

این الگوریتم [۱۰] که نسخه‌ی کم‌مصرف الگوریتم HEFT می‌باشد زمان اجرای یکسان و میزان توان مصرفی کمتری نسبت به الگوریتم HEFT نتیجه می‌دهد (الگوریتم HEFT در بخش ۶-۱ به‌طور کامل توضیح داده خواهد شد). در این الگوریتم ولتاژ پردازنده در طول زمان‌های بیکاری پردازنده مقیاس‌بندی می‌شود. زمان بیکاری برای یک پردازنده زمانی ایجاد می‌شود که آن پردازنده منتظر اجرای وظیفه‌ای است که آن وظیفه وابسته به وظایف دیگری است که بر روی پردازنده‌های دیگر اجرا می‌شود. زمان بیکاری با کاهش سرعت پردازنده حذف می‌شود که این کار باعث می‌شود که مدت زمانی که پردازنده وظایف قبلی را انجام می‌دهد، افزایش یابد. توجه به این نکته ضروری است که تمام وابستگی‌های بین هر وظیفه با اجدادش می‌بایست برآورده شود.



با استفاده از این الگوریتم اگر EFT(N ,P1) + C(P1,P2) > EFT(N2,P2) باشد، EFT(N2,P2) افزایش پیدا می‌کند بدلیل اینکه P2 مجبور است منتظر بماند تا اجرای N1 تمام شده و سپس N4 را اجرا کند. اگر زمان بیکاری روی P2 جستجو شود، در صورتی که EST(N4 , P2) کمتر از EFT(N2 , P2) باشد، زمان‌های بیکاری را بر روی این پردازنده یافت خواهیم کرد. به همین دلیل ما به دنبال یافتن زمان‌های بیکار دردسترس بر روی تمامی پردازنده‌ها هستیم.

۵- مدلسازی مسئله

۵-۱- مدل جریان کاری

جریان کاری یک برنامه‌ی کاربردی معمولا به صورت یک گراف بدون دور جهت‌دار (DAG) مدلسازی شده و به صورت W=(A , D) نشان داده می‌شود. این گراف شامل n وظیفه $A=\bigcup_{1}^{n} A_i$ می‌باشد که این وظایف از طریق جریان‌های کنترلی[۲۱] و جریان‌های داده[۲۲] به همدیگر وابسته می‌باشند. $D=\{(A_i,A_j,Data_{ij})\mid (A_i,A_j)\in A\times A\}$ که $Data_{ij}$ نشان‌دهنده‌ی سایز داده‌ی مورد نیاز برای انتقال از وظیفه‌ی A_i به وظیفه‌ی A_j می‌باشد. مجموعه‌ی وظایف والد[۲۳] A_i (وظایفی که قبل از اجرای وظیفه‌ی A_i می‌بایست اجرا شوند) را به صورت $Pred(A_i)=\{A_k \mid (A_k,A_i,Data_{ij})\in D\}$ نشان می‌دهیم. ما فرض می‌کنیم که بار کاری محاسباتی هر وظیفه به صورت طول هر وظیفه داده شده است که در واقع نشان‌دهنده‌ی تعداد دستورالعمل‌های ماشین موردنیاز برای اجرای آن وظیفه می‌باشد.

۵-۲- مدل زمان اجرای وظایف در گراف

برای محاسبه‌ی زمان اجرای یک جریان کاری ابتدا می‌بایست زمان اجرای وظیفه‌ی A_i بر روی ماشین مجازی VM_j را بدست آوریم که ما این زمان را به صورت

مجموع زمان مورد نیاز برای انتقال بزرگترین داده ورودی از هر $A_P \in pred(A_P)$ و زمان مورد نیاز برای اجرای وظیفه‌ی A_i بر روی ماشین مجازی VM_j تعریف می‌کنیم:

$$t_{(A_i,VM_j)} = \max_{Ap \in pred(A_i)} \left\{ \frac{Data_{pi}}{b_{pi}} \right\} + \frac{workload(A_i)}{s_j} \quad (۱)$$

که $Data_{pi}$ سایز داده‌ای است که باید بین A_P و A_i انتقال داده شود، b_{pi} پهنای باند بین ماشین مجازی که وظیفه‌ی A_P را اجرا کرده با VM_j که قرار است وظیفه‌ی A_i را اجرا کند، می‌باشد. $workload(A_i)$ ، طول وظیفه‌ی A_i (تعداد دستورالعمل‌های ماشین برای اجرای A_i) و s_j سرعت ماشین مجازی VM_j می‌باشد. بنابراین زمان تکمیل اجرای وظیفه‌ی A_i (T_{Ai}) با بررسی زمان اجرای وظیفه‌ی A_i و اجداد این وظیفه به صورت زیر تعریف می‌شود:

$$T_{Ai} = \begin{cases} t_{(A_i,Vm_j)} \rightarrow pred(A_i) = \phi \\ \max\limits_{Ap \in pred(A_i)} \{T_{Ap} + t_{(A_i,Vm_j)}\} \rightarrow pred(A_i) \neq \phi \end{cases} \quad (۲)$$

بنابراین زمان اجرای یک جریان کاری به صورت زیر محاسبه می‌شود [۱۱]:

$$T_W = \max_{i \in [1,n]} \{T_{(A_i,VM_j)}\} \quad (۳)$$

۵-۳- مدل توان

در اینجا به جای استفاده از مدل توان مصرفی در پردازنده‌های چند هسته‌ای، از داده‌های واقعی برای مصرف توان استفاده می‌کنیم. این داده‌ها به وسیله‌ی نتایج SPEC power benchmark فراهم شده است که در جدول ۴ در بخش پیوست آمده است. مقادیر درون این جدول، میزان توان مصرفی هر یک از سرورها را به ازای افزایش ۱۰ درصدی میزان بهره‌وری نشان می‌دهد. براساس [۹] بهره‌وری پردازنده‌ی یک میزبان را به صورت مجموع بهره‌وری پردازنده‌های مختلف آن میزبان تعریف می‌کنیم.

$$U_{cpu}(t) = \sum_{c=1}^{PE} \sum_{i=1}^{r} \frac{MIPS_{i,c}}{MIPS_C} \quad (۴)$$

در رابطه‌ی بالا PE نشان‌دهنده‌ی تعداد پردازنده‌های میزبان و r نشان‌دهنده‌ی تعداد ماشین‌های مجازی هستند که بر روی میزبان قرار دارند. همچنین $MIPS_{i,c}$ نشان‌دهنده‌ی MIPS تخصیص داده شده‌ی پردازنده‌ی c میزبان به ماشین مجازی i و $MIPS_C$ نشان‌دهنده‌ی مجموع MIPS مربوط به پردازنده‌ی c میزبان می‌باشد.

۶- تخصیص منبع در سطح زیرساخت با هدف کاهش توان مصرفی

در این قسمت شیوه‌ی تخصیص جدیدی را مطرح می‌کنیم تا ماشین فیزیکی که از نظر میزان توان مصرفی کارآمدتر است را برای نگاشت ماشین‌های مجازی انتخاب کنیم. مسئله‌ی زمانبندی ماشین‌های مجازی را به صورت زیر فرمول‌بندی می‌کنیم:

مجموعه‌ای از n ماشین مجازی داده شده است تا بر روی m ماشین فیزیکی قرار داده شود.

$$\{VM_i(pe_i, mips_i, ram_i, bw_i) \mid i = 1,2,3,...,n\}$$
$$\{M_j(PE_j, MIPS_j, RAM_j, BW_j) \mid j = 1,2,3,...,n\}$$

هر ماشین مجازی به pe_i المان پردازشی، $mips_i$ ، ram_i ، bw_i (پهنای باند) بدون توقف یا بدون مهاجرت در طول زمان اجرا نیاز دارد. در اینجا ما سه نوع منبع پردازنده، حافظه و پهنای باند را مورد بررسی قرار می‌دهیم. هدف این زمانبندی مصالحه‌ای بین کاهش مصرف انرژی در تامین ساختن بیشترین نیازمندی‌های ماشین‌های مجازی است، بدین معنی که بتوانیم میزبانی را انتخاب نماییم که در عین حال که میزان توان مصرفی کمتری نسبت به سایر میزبان‌ها دارد، ظرفیت و کارایی بالاتری داشته و بتواند تعداد بیشتری ماشین مجازی را میزبانی کند، تا بدین ترتیب از تعداد کمتری ماشین فیزیکی استفاده شود و توان مصرفی مرکز داده کاهش یابد. الگوریتم تعدیل بار که الگوریتم پیش‌فرض استفاده شده برای تخصیص منابع فیزیکی به ماشین‌های مجازی در CloudSim و WorkflowSim می‌باشد به صورت زیر است:

```
Input: a VM (vm) and a set of Host (H)
Output: a host for vm
For each host in host list do
Select the host with less processor in use
CloudSim.allocationMap.put (vm.getId (), host.getID ());
```

شکل ۲- الگوریتم تعدیل بار برای تخصیص منابع فیزیکی به ماشین‌های مجازی

این الگوریتم هر ماشین مجازی را بر روی میزبانی قرار می‌دهد که نسبت به سایر میزبان‌ها تعداد کمتری پردازنده‌ی در حال استفاده داشته باشد به عبارت دیگر بهره‌وری کمتری داشته باشد.

۶-۱- الگوریتم تخصیص منبع پیشنهادی آگاه به توان

این الگوریتم هر ماشین مجازی را بر روی میزبانی قرار می‌دهد که بزرگترین مقدار $G_h = \frac{h.TotalMips}{h.GetPower(100\%)}$ را داشته باشد. برای هر میزبان، G_h می‌تواند به عنوان نسبتی از مجموع MIPSهای تمامی پردازنده‌های ماشین فیزیکی به بیشترین توان مصرفی پردازنده زمانی که میزان بهره‌وری ۱۰۰٪ است، در نظر گرفته شود. با این کار ما در واقع میزبانی را انتخاب می‌نماییم که در عین حال که میزان توان مصرفی کمتری نسبت به سایر میزبان‌ها دارد، سرعت، کارایی و ظرفیت بالاتری داشته، چرا که هر چقدر $h.TotalMips$ بیشتر باشد، تعداد هسته‌های آن میزبان بیشتر بوده و برای تخصیص ماشین‌های مجازی به تعداد کمتری میزبان نیازمند می‌شویم و بدین ترتیب از تعداد کمتری ماشین فیزیکی استفاده شده و توان مصرفی مرکز داده کاهش می‌یابد.

در الگوریتم پیشنهادی ما شکل ۳، ورودی به صورت مجموعه‌ای از میزبان‌ها و ماشین مجازی، که قرار است بر روی میزبان مناسب قرار داده شود، داده شده است. در ابتدا متغیر MaxValue را برابر با یک مقدار کمینه قرار می‌دهیم سپس برای هر میزبان، میزان بهره‌وری پردازنده‌های آن را با استفاده از رابطه ۴ محاسبه می‌کنیم. در صورتی که این مقدار (میزان بهره‌وری پردازنده‌ها) کمتر از ۱ باشد، مقدار G_h برای آن میزبان و ماشین مجازی محاسبه می‌شود، در صورتی که مقدار G_h میزبان از MaxValue بیشتر باشد و سایر نیازمندی‌های دیگر آن ماشین مجازی را (مانند حافظه و پهنای باند) فراهم کند مقدار G_h به عنوان MaxValue و آن میزبان به عنوان مناسب‌ترین میزبان انتخاب می‌شود.

این روند برای تمامی میزبان‌های موجود ادامه یافته و هر بار پس از بررسی بهره‌وری پردازنده‌های میزبان، درصورتی که مقدار G_h از مقدار MaxValue قبلی بیشتر باشد، جایگزین MaxValue قبلی می‌شود و آن میزبان به عنوان مناسب‌ترین میزبان انتخاب می‌شود. برای تخصیص سایر ماشین‌های مجازی، میزبان مناسب با استفاده از این الگوریتم محاسبه می‌شود.

```
Find Host for VM by Green Metric
Energy aware VM Allocation Policy
Output: best Host (a best host for allocation of VM)
Input: a VM (vm) and a set of Host (H)
{
   MaxValue=Integer.MIN_VALUE
   For all (Power Host h in Host)
   {
   Calculate utilization of CPU:
   U_cpu(t) = Σ_{c=1}^{PE} Σ_{i=1}^{r} MIPS_{i,c} / MIPS_c
   If (utilization of CPU<1)
   {
   G_h = h.TotalMips / h.GetPower(100%)
   If (G_h > MaxValue && h.IsSutableForVM(vm))
   {
   MaxValue = G_h
   bestHost = h
   }
  }
}
Return best Host
CloudSim.AllocationMap.put (vm.getId, host.getId)
}
```

شکل ۳- الگوریتم تخصیص منبع پیشنهادی آگاه به توان

۷- زمانبندی برنامه‌ی کاربردی بر روی ماشین‌های مجازی

تابع هدف ما در این مسئله زمانبندی، نگاشت وظایف بر روی ماشین‌های مجازی و ترتیب بخشیدن به اجرای آنهاست به گونه‌ای که نیازمندی‌های اولویت بین وظایف برآورده شده و کمترین زمان اجرای کل گراف حاصل شود.

فرض می‌کنیم که محیط محاسباتی شامل مجموعه‌ای از q ماشین مجازی ناهمگن می‌باشد که به صورت کامل به یکدیگر متصل می‌باشند. در این مدل فرض شده است که وظایف یک برنامه‌ی کاربردی به صورت غیرانحصاری اجرا می‌شوند. W ماتریس هزینه‌ی محاسباتی با اندازه‌ی n*q است به گونه‌ای که هر $w_{i,j}$ زمان تخمینی از زمان تکمیل وظیفه‌ی n_i بر روی ماشین q_j را نشان می‌دهد. قبل از زمانبندی، وظایف با میانگین زمان اجرای‌شان بر روی ماشین‌های مختلف برچسب‌گذاری می‌شوند. میانگین زمان اجرای یک وظیفه‌ی n_i به صورت زیر تعریف می‌شود:

$$\overline{w}_i = \frac{\sum_{j=1}^{q} w_{i,j}}{q} \quad (۵)$$

نرخ انتقال داده بین ماشین‌های مجازی نیز در ماتریس B با اندازه ی q*q ذخیره می‌شود. هزینه‌ی انتقال یک یال (I,k) که برای انتقال داده از وظیفه‌ی n_i (که بر روی ماشین vm_m زمانبندی شده است) به وظیفه‌ی n_j (که بر روی ماشین vm_n زمانبندی شده است) استفاده می‌شود، به صورت زیر تعریف شده است:

$$c_{i,k} = \frac{data_{i,k}}{B_{m,n}} \quad (۶)$$

زمانی که هم n_i و هم n_k بر روی ماشین‌های مجازی یکسان زمانبندی شوند، $c_{i,k}$ صفر خواهد بود. قبل از زمانبندی، میانگین هزینه‌ی انتقال محاسبه شده و بر روی هر یال برچسب‌گذاری می‌شود. بنابراین میانگین هزینه‌ی انتقال یک یال (I, k) که بین دو وظیفه‌ی n_i و n_k می‌باشد به صورت زیر محاسبه می‌شود:

$$\overline{c}_{i,k} = \frac{data_{i,k}}{\overline{B}} \quad (۷)$$

که $\overline{B}$ میانگین نرخ انتقال میان ماشین‌های مجازی (میانگین پهنای باند میان ماشین‌های مجازی) می‌باشد.

قبل از ارائه‌ی تابع هدف، ابتدا مقادیر EST و EFT را تعریف می‌کنیم.

EST (n_i, q_j) و EFT (n_i, q_j) به ترتیب نشان‌دهنده‌ی زودترین زمان شروع اجرا و زودترین زمان پایان اجرای وظیفه‌ی n_i بر روی ماشین مجازی q_j می‌باشد. برای یک وظیفه‌ی ورودی $EST(n_{entry}, q_j) = 0$ می‌باشد. برای وظایف دیگر موجود در گراف، مقادیر EST و EFT به صورت بازگشتی و با شروع از وظیفه‌ی ورودی محاسبه می‌شود. برای محاسبه‌ی EFT وظیفه‌ی n، تمامی وظایف پیشین (والد) آن می‌بایست زمانبندی شود.

$$EST(n_i, q_j) = \max\{avail[j], \max_{n_m \in pred(n_i)}(AFT(n_m) + c_{m,i})\} \quad (۸)$$

$$EFT(n_i, q_j) = w_{i,j} + EST(n_i, q_j) \quad (۹)$$

که در روابط بالا $pred(n_i)$ مجموعه‌ی وظایف پیشین (والد) وظیفه‌ی n_i و $avail[j]$ زودترین زمانی‌ست که ماشین j آماده‌ی اجرای n_i است. اگر n_k وظیفه‌ی آخر زمانبندی شده بر روی ماشین q_j باشد، $avail[j]$ زمانی است که ماشین q_j اجرای وظیفه‌ی n_k را تمام کرده است و آماده است تا وظیفه‌ی دیگری را اجرا کند. بلاک max داخل تساوی EST، زمان آماده را برمی‌گرداند یعنی زمانی که تمامی داده‌های مورد نیاز برای اجرای وظیفه‌ی n_i به ماشین مجازی q_j رسیده است. بعد از اینکه وظیفه‌ی n_i بر روی ماشین مجازی q_j زمانبندی شد، زودترین زمان شروع و زودترین زمان پایان وظیفه‌ی n_i بر روی ماشین q_j به ترتیب برابر با زمان شروع واقعی، $AST(n_i)$، و زمان پایان واقعی، $AFT(n_i)$، آن وظیفه نامیده می‌شود. بعد از اینکه تمامی وظایف در گراف زمانبندی شد، طول این زمانبندی (زمان اجرای کل وظایف گراف) زمان پایان واقعی وظیفه‌ی خروجی خواهد بود. اگر چندین وظیفه‌ی خروجی وجود داشته باشد این زمان به صورت زیر محاسبه می‌شود [2]:

$$makespan = \max\{AFT(n_{exit})\} \quad (۱۰)$$

۷-۱- الگوریتم زمانبندی HEFT

الگوریتم زمانبندی HEFT (Heterogeneous Earliest Finish Time Algorithm) یک الگوریتم معروف زمانبندی برای کمینه ساختن زمان اجرای وظایف در یک جریان کاری می‌باشد که توسط هالوک[۲۴] و همکارانش معرفی شد [۲]. این الگوریتم شامل دو فاز می‌باشد: فاز اولویت‌بندی وظایف و فاز انتخاب

ماشین مجازی. در فاز اولویت‌بندی وظایف، اولویت هر وظیفه براساس معیار B-rank به آنها داده می‌شود که این معیار در واقع نشان‌دهنده‌ی فاصله‌ی آن وظیفه تا انتهای جریان کاری می‌باشد. بعد از اینکه اولویت و مرتبه‌ی هر وظیفه که در واقع نشان‌دهنده‌ی ترتیب اجرای وظایف هستند، مشخص شد، در فاز دوم وظایف به ترتیب اولویت‌شان برای اجرا بر روی منابع مجازی قرار می‌گیرند و هر وظیفه بر روی منبعی قرار می‌گیرد که زمان پایان کمتری را نسبت به منابع مجازی دیگر برای آن وظیفه فراهم کند.

Require: $W=(A,D)$, ▷ Workflow Application
Require: ▷ Set of resources
Ensure: $sched\,w=\{(A_i, sched(A_i))\,|\,\forall A_i \in A\}$
▷ Workflow Schedule
1: Function HEFT (W, R)
2: $Rank \leftarrow B-Rank\,(A)$ ▷ Order the tasks according to B – Rank
3: $sched\,w \leftarrow \phi$ ▷ Initialize workflow schedule with empty set
4: **for** $i \leftarrow 1,n$ **do** ▷ Iterate over the ranked tasks
5: $T_{min} \leftarrow \infty$
6: for $j \leftarrow 1,m$ *do* ▷ Iterate over all resources
7: $T_{Rank_i} \leftarrow \max_{A_p \in pred(Rank_i)} \{T_{A_p} + t_{(Rank_i, R_j)}\}$
▷ Compute completion time of $Rank_i$
8: if $T_{Rank_i} < T_{min}$ then ▷ Save the minimum completion time
9: $T_{min} \leftarrow T_{Rank_i}$
10: $R_{min} \leftarrow R_j$
11: end if
12: end for
13: $sched\,w \leftarrow sched\,w \cup (Rank_i, R_{min})$
▷ Schedule the task
14: end for
15: return $sched\,w$
16: end function

شکل ۴- الگوریتم HEFT [۱۱]

در این الگوریتم ابتدا هزینه‌های محاسباتی وظایف و هزینه‌های انتقال یال‌ها به ترتیب براساس روابط (۵) و (۷) محاسبه می‌شوند. سپس تمامی وظایف جریان کاری براساس معیار B-rank اولویت‌بندی می‌شوند. در این اولویت‌بندی، اولویت هر وظیفه براساس رابطه‌ی زیر محاسبه می‌شود که این رتبه‌بندی براساس هزینه‌های محاسباتی و انتقال آن وظیفه مشخص می‌شود. سپس با مرتب کردن وظایف براساس مرتبه‌یشان به صورت نزولی لیستی از وظایف تولید می‌شود.

$$rank_u(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} \left(\overline{c_{i,j}} + rank_u(n_j) \right) \quad (۱۱)$$

که در این رابطه $succ(n_i)$ مجموعه‌ای از وظایفی هستند که بلافاصله بعد از وظیفه‌ی n_i اجرا می‌شوند. $\overline{c_{i,j}}$ میانگین هزینه‌ی انتقال یال (i,j) و $\overline{w_i}$ میانگین هزینه‌ی محاسباتی وظیفه‌ی n_i می‌باشد. به دلیل اینکه این اولویت به صورت بازگشتی و با پیمایش گراف به سمت بالا محاسبه می‌شود و ابتدا از وظیفه‌ی خروجی (آخرین وظیفه) شروع می‌کنیم، این مرتبه‌بندی، مرتبه‌بندی رو به بالا نامیده می‌شود. برای وظیفه‌ی خروجی n_{exit} این مقدار برابر است با:

$$rank_u(n_{exit}) = \overline{w_{exit}} \quad (۱۲)$$

در فاز انتخاب ماشین مجازی برای هر وظیفه n_i، پردازنده p_j انتخاب شده و مقدار $EFT(n_i, p_j)$ (زودترین زمان پایان) براساس رابطه‌ی (۹) محاسبه می‌شود. در نهایت وظیفه n_i به پردازنده‌ای اختصاص داده می شود که EFT کمتری برای آن وظیفه داشته باشد.

۸- ارزیابی

در این بخش به جزئیات شبیه‌سازی الگوریتم‌های معرفی شده خواهیم پرداخت. سپس از طریق رسم نمودار به ارزیابی روش‌های مذکور می‌پردازیم. محیط مورد بررسی ما یک محیط زیرساخت به عنوان سرویس است و همچنین ابزار ما برای شبیه‌سازی نرم‌افزار Workflowsim [۱۲] و [۱۳] است که توسعه‌یافته‌ی شبیه‌ساز CloudSim می‌باشد. این شبیه‌ساز، به ما اجازه‌ی ایجاد یک محیط مجازی‌سازی شده را می‌دهد و از تخصیص منابع براساس تقاضا پشتیبانی می‌کند. همچنین امکان ارسال جریان‌های کاری از طرف کاربر را فراهم می‌آورد.

۸-۱- مشخصات مرکز داده و جریان‌های کاری مورد استفاده

مرکز داده‌ی ما از ۱۰۰ گره فیزیکی ناهمگون و ۲۲۰ ماشین مجازی ناهمگون تشکیل شده است. مشخصات سرورهای مورد استفاده در جدول ۱ آمده است:

جدول ۱- مشخصات و تنظیمات سرورهای مورد نظر

سرور	نوع پردازنده	کارایی پردازنده MIPS	تعداد هسته	حافظه GB	پهنای باند شبکه
HP ProLiant ML110 G3	PentiumD930	۳۰۰۰	۲	۴	۱GB/s
HP ProLiant ML110 G4	Intel Xeon 3040	۱۸۶۰	۲	۴	۱GB/s
HP ProLiant ML110 G5	Intel Xeon 3075	۲۶۶۰	۲	۴	۱GB/s
IbmX3250	XeonX3470	۲۹۳۳	۴	۸	۱GB/s
IbmX3250	XeonX3480	۳۰۶۷	۴	۸	۱GB/s

ماشین‌های مجازی مورد استفاده نیز در ۴ نوع مختلف ایجاد شده‌اند، مشخصات انواع ماشین‌های مجازی به نمونه‌های Amazon EC2 مربوط می‌شود که در جدول ۲ آمده است.

جدول ۲- مشخصات و تنظیمات ماشین‌های مجازی موردنظر

نوع ماشین مجازی	کارایی پردازنده MIPS	تعداد هسته	حافظه MB	پهنای باند MB/s
High-CPU Medium	۲۵۰۰	۱	۸۷۰	۱۰۰
Extra Large	۲۰۰۰	۱	۱۷۴۰	۱۰۰
Small	۱۰۰۰	۱	۱۷۴۰	۱۰۰
Micro	۵۰۰	۱	۶۱۳	۱۰۰

ما در این شبیه‌سازی پهنای باند میان ماشین‌های مجازی را یکسان در نظر گرفته‌ایم. به منظور محاسبه‌ی توان مصرفی سرورها، از داده‌های واقعی توان مصرفی که از نتایج استفاده از SPEC power benchmark فراهم شده است استفاده می‌کنیم. مقادیر درون جدول ۴ در قسمت پیوست، میزان توان مصرفی هر یک از سرورها را به ازای افزایش ۱۰ درصدی میزان بهره‌وری نشان می‌دهد.

ما از ۴ نوع گراف یا جریان کاری متفاوت برای نشان دادن برنامه‌ی کاربردی ارسال شده از طرف کاربر استفاده می‌کنیم. که هر یک از این گراف‌ها با یک جریان کاری از برنامه‌ی کاربردی در دنیای واقعی مطابقت می‌کند. این جریان‌های کاری عبارتند از جریان کاری Montage با تعداد ۲۵، ۵۰، ۱۰۰ و ۱۰۰۰ وظیفه، Sipht با تعداد ۳۰، ۶۰، ۱۰۰و ۱۰۰۰ وظیفه، CyberShake با تعداد ۳۰، ۵۰، ۱۰۰ و ۱۰۰۰ وظیفه و Inspiral با تعداد ۳۰، ۵۰، ۱۰۰ و ۱۰۰۰ وظیفه [۱۴].

۸-۲- شبیه‌سازی و ارزیابی الگوریتم HEFT

در این قسمت به مقایسه و ارزیابی زمان تکمیل وظایف جریان‌های کاری در الگوریتم HEFT با الگوریتم‌های MCT، MAX-MIN و MIN MIN [۱۵] می‌پردازیم. الگوریتم MCT [۱۵] جز الگوریتم‌های حریصانه‌ی معروفی است که برای زمانبندی پویای وظایف به کار می‌رود. در این الگوریتم بر خلاف الگوریتم HEFT اولویت بین وظایف مورد بررسی قرار نمی‌گیرد و هر وظیفه با ترتیبی دلخواه، بر روی منبعی قرار می‌گیرد که نسبت به سایر منابع زمان تکمیل کمتری را برای آن وظیفه داشته باشد.

این امر باعث می‌شود برخی از وظایف بر روی ماشینی قرار بگیرند که الزاما کمترین زمان اجرا را به همراه نداشته باشند. الگوریتم HEFT یک الگوریتم معروف زمانبندی به منظور کمینه ساختن زمان اجرای وظایف در یک جریان کاری می‌باشد. همانطور که نتایج شبیه‌سازی شکل ۶ نشان می‌دهند الگوریتم HEFT به‌طور میانگین نسبت به سه الگوریتم زمانبندی دیگر کاهش زمان تکمیل جریان کاری را داشته است که این کاهش زمان در ۴ جریان کاری مورد استفاده ما متفاوت است. این میزان کاهش در جدول ۳ نشان داده شده است. به همین منظور ما برای زمانبندی جریان‌های‌کاری ورودی از الگوریتم HEFT استفاده نموده‌ایم.

جدول ۳- درصد کاهش زمان اجرای جریان‌های کاری در الگوریتم HEFT نسبت به الگوریتم‌های زمانبندی دیگر

جریان کاری/الگوریتم زمانبندی	MIN MIN	MAX MIN	MCT
Montage	۵0.03	۶0.0	۰.۰۳
Sipht	14.67	۹0.002	۱۱.۸۲
Cybershake	0.61	۶0.3	۱.۲۳
Inspiral	۱3.	۲3.2	2.23

۸-۳- شبیه‌سازی و ارزیابی الگوریتم پیشنهادی آگاه به توان

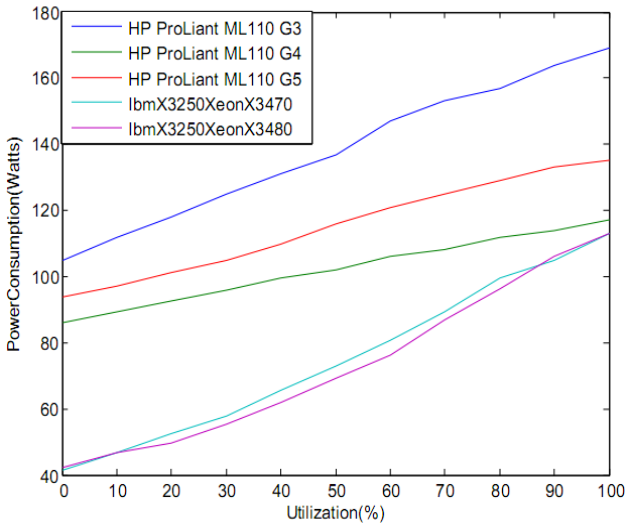
ما در این شبیه‌سازی از ورژن WorkflowSim-1.0-master برای مدلسازی و شبیه‌سازی مرکز دادیمان استفاده کردیم. در اینجا الگوریتم آگاه به توان معرفی شده خود را با الگوریتم‌های تخصیص منبع زیر مقایسه می‌کنیم:

الگوریتم PABFD: این الگوریتم که در مرجع [۱۶] معرفی شده است، تمام ماشین‌های مجازی را براساس EST (زودترین زمان شروع) آنها مرتب می‌کند و برای جایگذاری هر ماشین مجازی، میزبانی را انتخاب می‌کند که کمترین افزایش توان مصرفی را داشته باشد. (در پیوست این الگوریتم معرفی شده است).

الگوریتم تعدیل بار: این الگوریتم که در بخش ۶ مقاله نیز معرفی شد، هر ماشین مجازی را بر روی میزبانی قرار می‌دهد که نسبت به سایر میزبان‌ها تعداد کمتری پردازنده‌ی در حال استفاده داشته باشد به عبارت دیگر بهره‌وری کمتری داشته باشد.

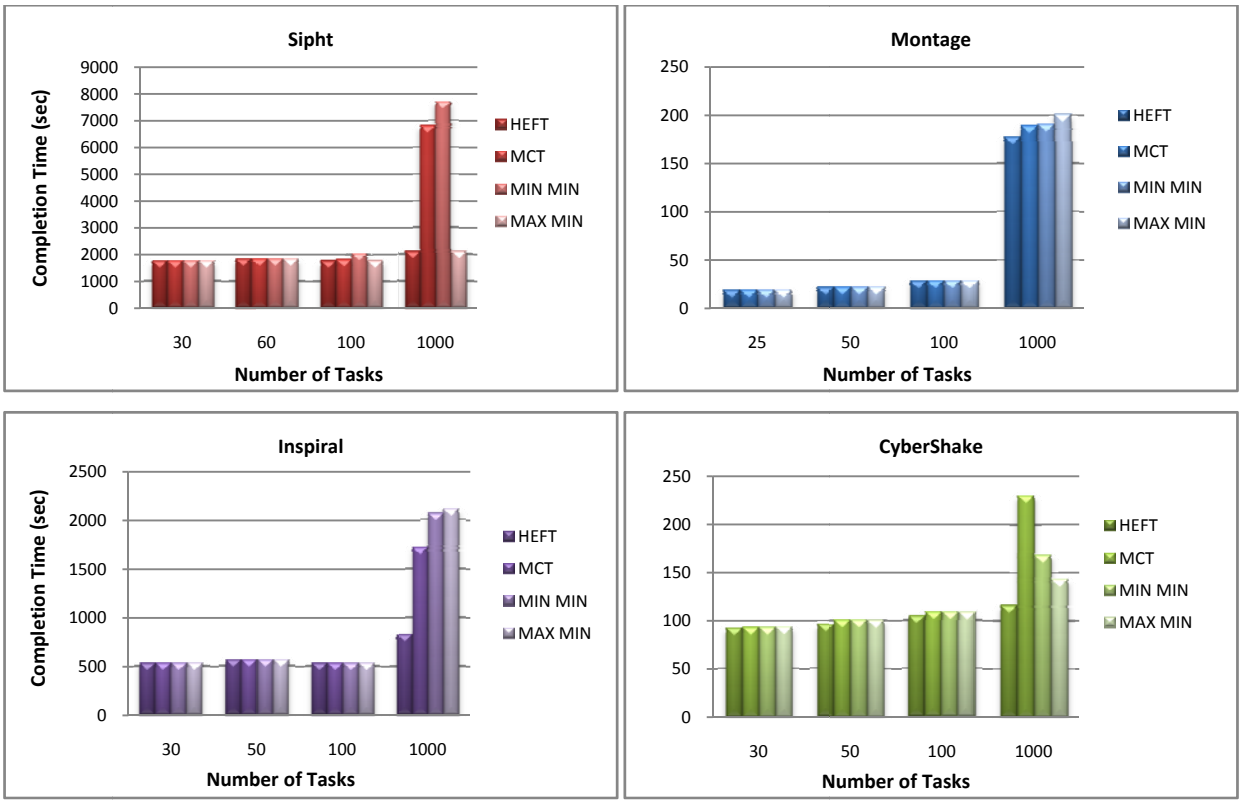
الگوریتم پیشنهادی آگاه به توان ما میزبانی را انتخاب می‌کند که بیشترین مقدار G_h را داشته باشد و الگوریتم PABFD ترجیح می‌دهد که یک ماشین مجازی جدید را به میزبانی تخصیص دهد که کمترین افزایش در توان مصرفی را داشته باشد. برای مثال مجموعه‌ای از ۱۲ ماشین مجازی (که هر یک به ۱ هسته نیاز دارد) داده شده است. الگوریتم PABFD براساس شکل ۵ که روند افزایش توان مصرفی را نشان می‌دهد، ۶ عدد سرور نوع HP ProLiant ML110 G4 را استفاده می‌کند که هر یک دارای ۲ هسته است به دلیل اینکه با افزایش بهره‌وری، کمترین میزان افزایش توان را در بین دیگر میزبان‌ها دارد.

در مقابل الگوریتم پیشنهادی ما از ۳ سرور نوع IbmX3250XeonX3480 استفاده می‌کند که هر یک دارای ۴ هسته است چرا که بیشترین میزان MIPS و کمترین توان مصرفی در بهره‌وری ۱۰۰ درصد را دارد. بنابراین چنین انتخابی می‌تواند هم تعداد سرورهای مورد استفاده و هم توان مصرفی مرکز داده را کاهش دهد. الگوریتم تعدیل بار نیز یک ماشین مجازی جدید را به میزبانی تخصیص می‌دهد که بهره‌وری کمتری داشته باشد و این امر باعث می‌شود که تعداد میزبان‌های مورد استفاده افزایش یافته و توان مصرفی مرکز داده افزایش یابد.



شکل ۵- میزان توان مصرفی سرورها بر حسب وات در سطوح متفاوت بهره‌وری

براساس نتایج شبیه‌سازی شکل ۷ که بر روی ۴ نوع گراف متفاوت با تعداد وظایف مختلف صورت گرفته است مشاهده می‌کنیم که الگوریتم آگاه به توان معرفی شده نسبت به الگوریتم تعدیل بار، در تمامی حالات کاهش چشمگیری در میزان توان مصرفی مرکز داده داشته است. که البته این میزان کاهش بسته به نوع گراف و تعداد وظایف گراف متفاوت است. در حقیقت در تمامی الگوریتم‌های تخصیص منبع مورد استفاده، با به کار بردن مثلا ۵ ماشین مجازی زمان اجرا به کمترین مقدار خود می‌رسد ولی با این تفاوت که بسته به مکان ماشین مجازی، تعداد سرورهای مورد نیاز متفاوت هستند.



شکل ۶- نمودار زمان تکمیل وظایف (ثانیه) در جریان‌های کاری مختلف

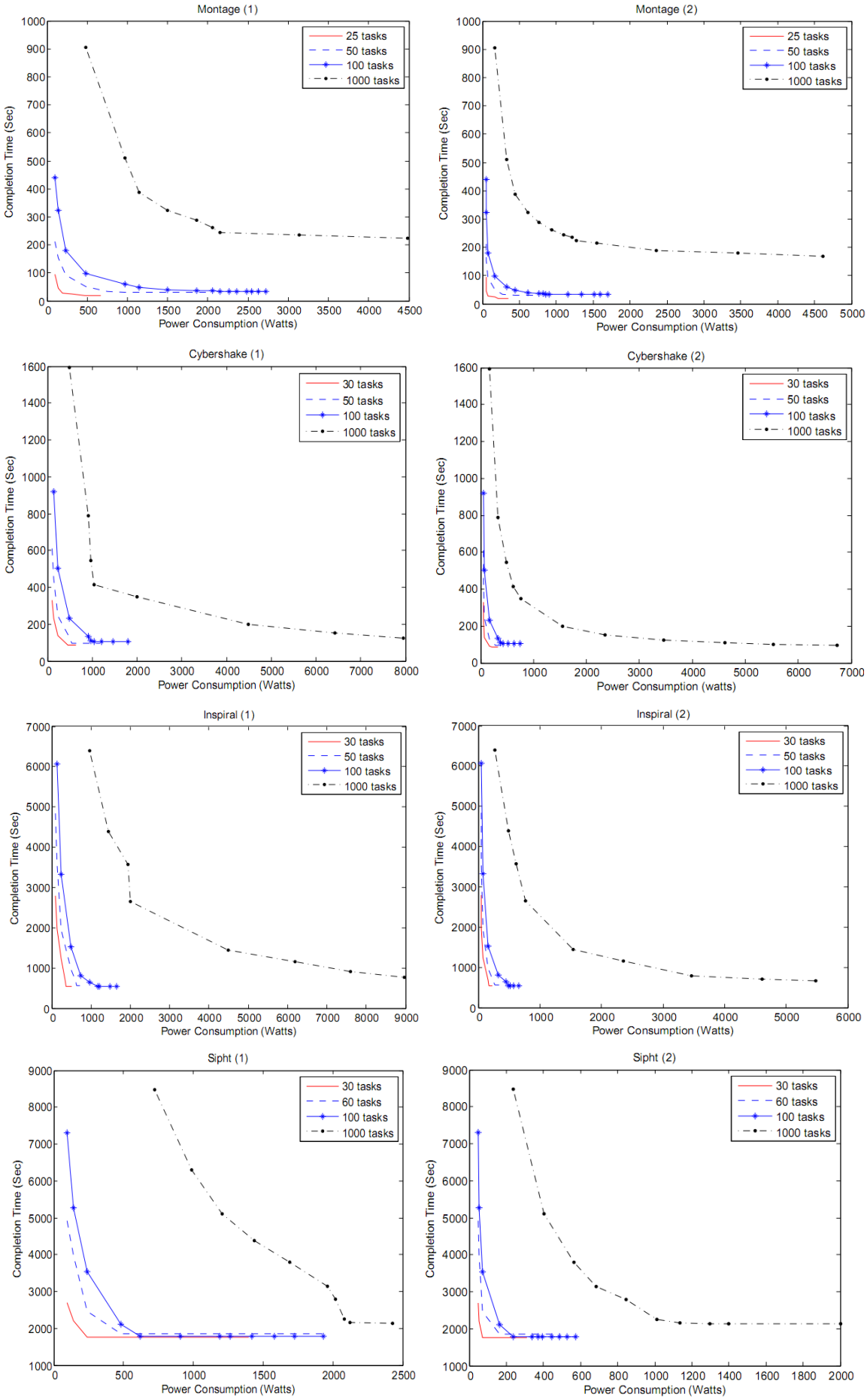
همانطور که در نمودار شکل ۷ مشاهده می‌شود، در هر دو الگوریتم، زمان تکمیل وظایف یکسان است، چرا که برای زمانبندی وظایف، در هر دو مورد از الگوریتم HEFT استفاده کرده‌ایم. به علاوه براساس رابطه ۷، برای محاسبه‌ی زمان انتقال وظایف بین ماشین‌های مجازی، میانگین پهنای باند میان ماشین‌های مجازی محاسبه می‌شود و ما در این شبیه‌سازی، پهنای باند میان ماشین‌های مجازی را یکسان در نظر گرفته‌ایم.

از طرفی هر ماشین مجازی MIPS مخصوص خود را دارد و در الگوریتم HEFT، انتخاب ماشین مجازی برای زمانبندی وظایف (همانطور که در بخش ۷ مطرح شد) به الگوریتم تخصیص ماشین‌های مجازی بر روی میزبان‌ها بستگی ندارد، بنابراین الگوریتم‌های تخصیص متفاوت که در سطح زیرساخت به کار می‌روند تاثیری بر زمان تکمیل جریان‌های کاری ندارند.

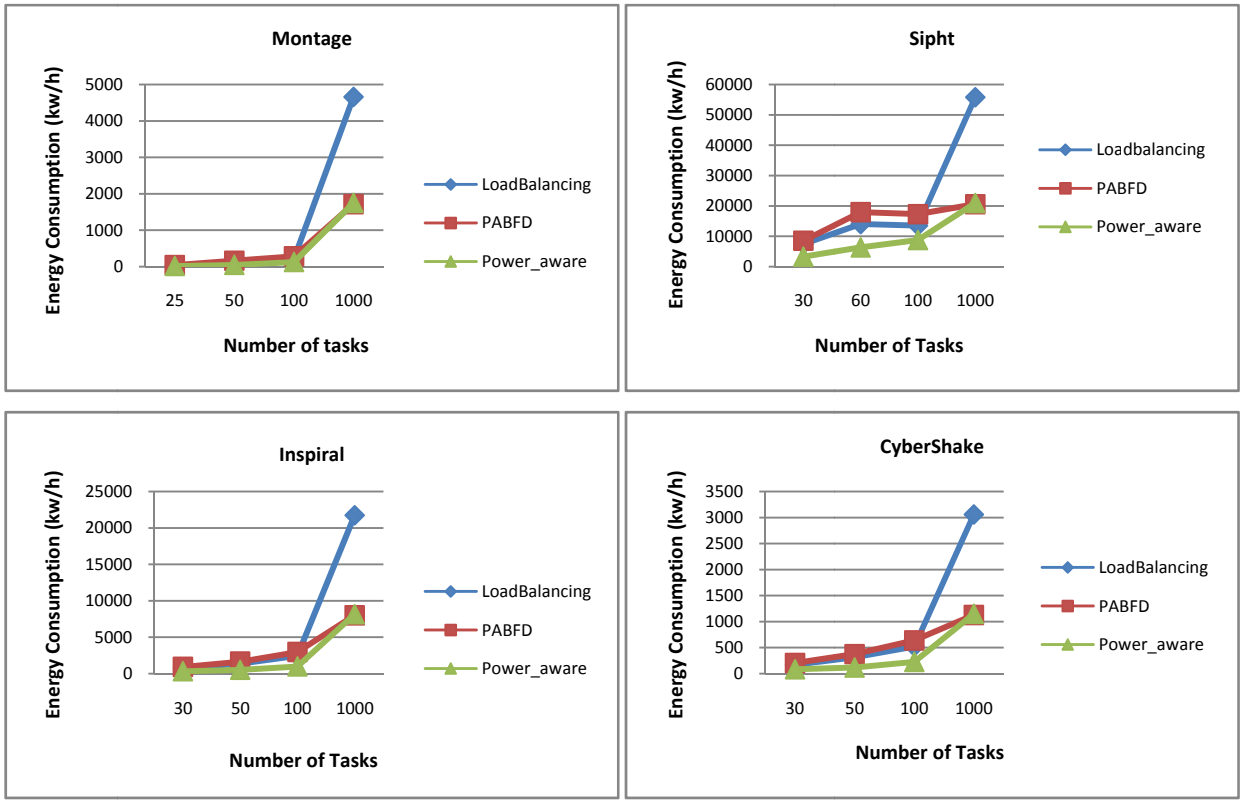
نمودارهای شکل ۸ میزان انرژی مصرفی مرکز داده را برای ۴ نوع جریان کاری مختلف با سه الگوریتم تخصیص منبع تعدیل بار، تخصیص منبع PABFD و تخصیص منبع آگاه به توان نشان می‌دهد. با توجه به نتایج حاصل از شبیه‌سازی که در شکل ۸ مشاهده می‌شود، می‌توان دریافت که با افزایش تعداد وظایف یک گراف، تعداد ماشین‌های مجازی مورد نیاز برای اجرای وظایف افزایش یافته و این امر موجب افزایش تعداد میزبان‌های مشغول به کار در مرکز داده شده است که در نتیجه‌ی آن میزان انرژی مصرفی مرکز داده افزایش یافته است.

همانطور که مشاهده می‌شود به‌طور میانگین این افزایش انرژی در الگوریتم آگاه به توان کمتر می‌باشد. چرا که با افزایش تعداد ماشین‌های مجازی مورد نیاز، تعداد میزبان‌های مشغول افزایش چشمگیری پیدا نمی‌کنند. بدلیل اینکه ما سعی کرده‌ایم که آنها را بر روی تعداد کمتری از میزبان‌ها جاسازی کنیم.

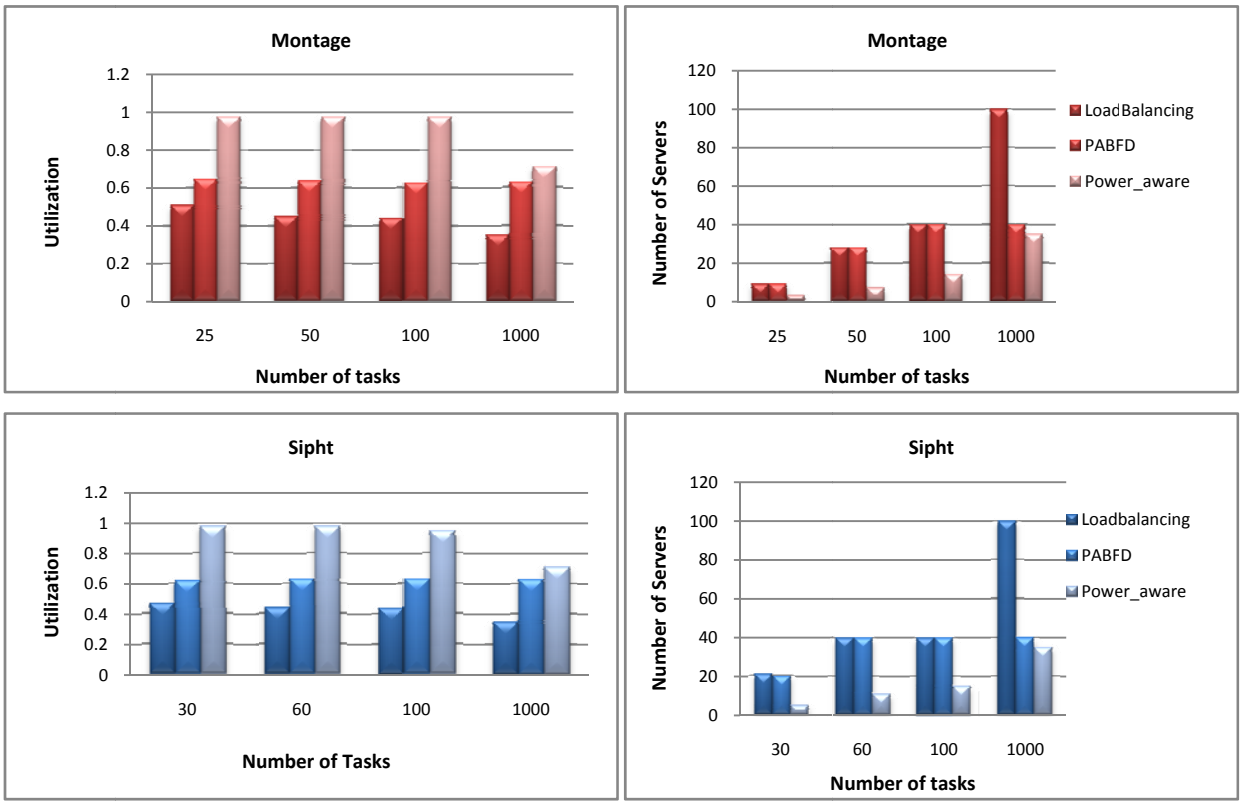
نمودارهای شکل ۹ میانگین میزان بهره‌وری میزبان‌های فعال (سمت چپ) و تعداد منابع فیزیکی مورد استفاده برای اجرای جریان‌های کاری مختلف (سمت راست) را در سه الگوریتم تخصیص منبع تعدیل بار، تخصیص منبع PABFD و تخصیص منبع آگاه به توان، نشان می‌دهد. براساس این نتایج، میانگین بهره‌وری منابع فیزیکی در الگوریتم پیشنهادی آگاه به توان نسبت به دو الگوریتم دیگر افزایش چشمگیری داشته است و این مقدار در الگوریتم تعدیل بار از همه کمتر است. علاوه بر این در الگوریتم آگاه به توان تعداد منابع فیزیکی مورد استفاده برای میزبانی ماشین‌های مجازی به منظور رسیدن به حداقل زمان اجرای ممکن، بسیار کمتر شده است و همین امر موجب شده تا بهره‌وری منابع فیزیکی افزایش یابد. در نتیجه با بالارفتن بهره‌وری منابع فیزیکی و کاهش تعداد منابع فیزیکی مورد استفاده برای اجرای برنامه‌های کاربردی، میزان توان مصرفی مرکز داده کاهش چشمگیری پیدا کرده است.



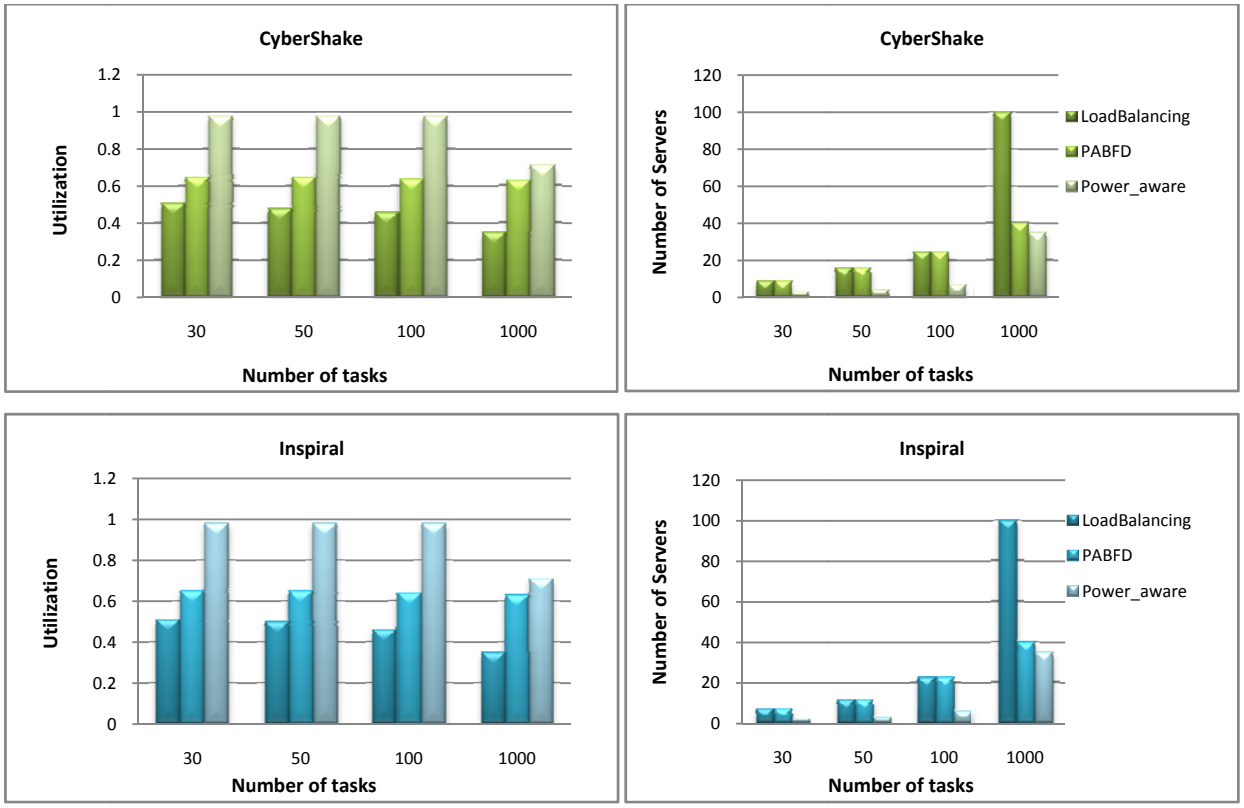
شکل ۷- نمودار زمان تکمیل وظایف (ثانیه) در مقابل توان مصرفی مرکز داده (وات) در جریان‌های کاری مختلف در الگوریتم تخصیص منبع تعدیل بار (۱) و الگوریتم تخصیص منبع پیشنهادی آگاه به توان (۲)



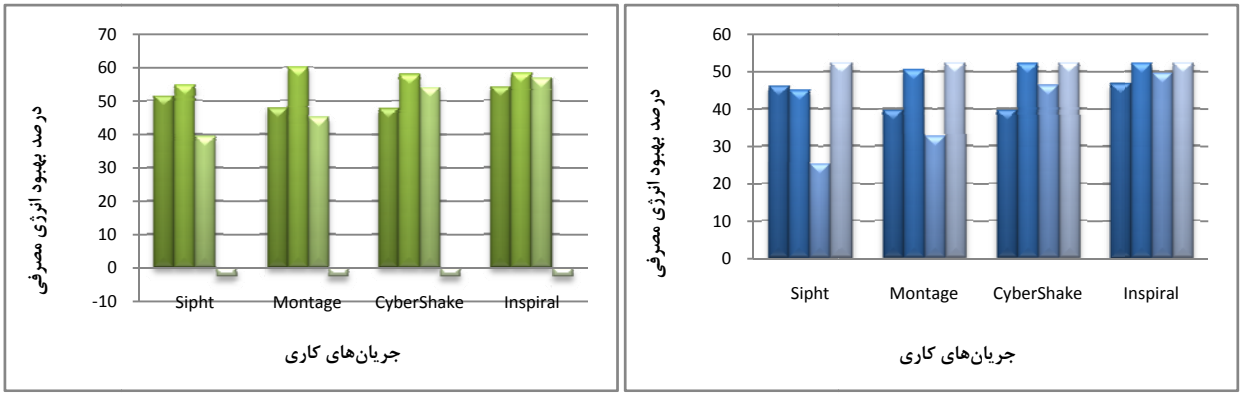
شکل ۸- نمودار انرژی مصرفی مرکز داده (کیلو وات بر ساعت) با افزایش تعداد وظایف جریان کاری در سه الگوریتم تخصیص متفاوت



شکل ۹- میانگین بهره‌وری میزبان‌ها (سمت چپ) و تعداد میزبان‌های فعال (سمت راست) با افزایش تعداد وظایف جریان‌های کاری



ادامه شکل ۹- میانگین بهره‌وری میزبان‌ها (سمت چپ) و تعداد میزبان‌های فعال (سمت راست) با افزایش تعداد وظایف جریان‌های کاری



شکل ۱۰- درصد بهبود انرژی مصرفی الگوریتم پیشنهادی در مقایسه با الگوریتم تعدیل بار (سمت راست) و الگوریتم PABFD (سمت چپ)

۹- نتیجه‌گیری

با توجه به نتایج حاصل از شبیه‌سازی می‌توان گفت که الگوریتم زمانبندی ایستای HEFT در مقایسه با دیگر الگوریتم‌های زمانبندی معرفی شده بهترین نتیجه را داشته است و براساس شکل ۶ مشاهده می‌شود که با افزایش تعداد وظایف گراف‌های ورودی، این الگوریتم نسبت به دیگر الگوریتم‌ها بهتر عمل می‌کند. همچنین با توجه به نمودارهای شکل ۷ که نشان‌دهنده‌ی زمان تکمیل وظایف در مقابل توان مصرفی مرکز داده است، می‌توان دریافت که نوع گراف ورودی و تعداد وظایف آن در میزان توان مصرفی مرکز داده تاثیرگذار است.

براساس نمودار شکل ۸ که نمایان‌گر درصد بهبود انرژی مصرفی الگوریتم پیشنهادی در مقایسه با الگوریتم تعدیل بار و PABFD است، می‌توان دریافت که که نوع گراف ورودی در میزان انرژی مصرفی مرکز داده تاثیرگذار بوده و الگوریتم پیشنهادی ما در مقایسه با الگوریتم تعدیل بار، حداقل برابر با ۲۵.۳۱٪ و حداکثر برابر با ۵۲.۴۵٪ کاهش انرژی را داشته و در مقایسه با الگوریتم PABFD

حداقل ۲.۸۵٪ افزایش انرژی و حداکثر ۶۰.۲٪ کاهش انـرژی را بـه همـراه داشـته است.

بنابراین به‌طور میانگین الگوریتم پیشنهادی ما در مقایسه با الگـوریتم تعـدیل بار، ۴۶.۰۲٪ و در مقایسه با الگوریتم PABFD، ۳۶.۳۳٪ کـاهش انـرژی را داشـته است.

مراجع

[1] M. Abu Sharkh, M, Jammal, A. Shami, and A. Ouda, "Resource Allocation in a Network-Based Cloud Computing Environment: Design Challenges," in *Proceeding IEEE Communications Magazine*, pp. 46–52, November 2013.

[2] H. Topcuoglu, S. Hariri, and Min-You Wu. "Performance-effective and low-complexity task scheduling for heterogeneous computing. Parallel and Distributed Systems," *IEEE Transactions on*, 13(3): 260–274, march 2002.

[3] P. Akilandeswari, and H. Srimathi, "Survey and Analysis on Task Scheduling in Cloud Environment," *Indian Journal of Science and Technology*, vol. 9(37), Issue 37, October 2016.

[4] S. Shruthi, and V. Nagaveni, "A Survey on Various Parallel Power Aware Task Scheduling Algorithms for Reducing Power Consumption," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, Issue: 4, April 2014.

[5] L. Ma, Y. Lu, F. Zhang, and S. Sun, "Dynamic Task Scheduling in Cloud Computing Based on Greedy Strategy," *Trustworthy Computing and Services Communication in Computer and Information Science*, vol. 320, pp. 156-162, 2013.

[۶] ه. رضایی و ق. حاجین، "مدیریت منابع مجازی‌سازی شده در محیط‌های رایانش ابر،" پایان‌نامه جهت اخذ درجه‌ی کارشناسی‌ارشد، دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، بهمن ۱۳۹۰.

[7] A. Beloglazov, and R. Buyya, "Energy Efficient Allocation of Virtual Machines in Cloud Data Centers," *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 577-578, 2010.

[۸] م. نجف‌زاده، "مقیاس‌پذیری خودمختار محیط‌های توزیعی مجازی براساس توافق‌های سطح سرویس،" دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، پایان‌نامه کارشناسی‌ارشد، ۱۳۸۹.

[9] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *Proceedings of the 34th annual international symposium on Computer architecture*, vol. 35, Issue 2, pp. 13-23, May 2007.

[10] S. Baskiyar, and K. Kumar Palli, "Low Power Scheduling of DAGs to Minimize Finish Times," *13th International Conference, Bangalore*, India, vol. 4297, pp. 353-362, December 2006.

[11] J. Durillo, H. Mohammadi Fard, and R. Prodan, "MOHEFT: A Multi-Objective List-based Method for Workflow Scheduling," *2012 IEEE 4th International Conference on Cloud Computing Technology and Science*, 2012.

[12] W. Chen, and E. Deelman, "WorkflowSim: Atoolkit for simulating scientific workflows in distributed environments," IEEE Xplore: 11 January 2013.

[13] P. S. Rawat, P. Dimri, and G. P. Saroha, "Tasks Scheduling in Cloud Computing Environment using Workflowsim," *Research Journal of Information Technology, 8: 98-104*, October 2016.

[14]"WorkflowGenerator,"https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator, September 2015.

[15] F. Soleimanian, M. Ahadi, R. Habibpour, and A. Kamalinia, "Analysis of Scheduling Algorithms in Grid Computing Environment," nternational Journal of Innovation and Applied Studies ISSN 2028-9324 vol. 4, no. 3, Nov. 2013, pp. 560-567.

[16] A. Beloglazov, and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," Journal Concurrency and Computation: Practice and Experience vol. 24, pp. 1397-1420, September 2012.

پیوست

الگوریتم PABFD

این الگوریتم، تمامی ماشین‌های مجازی را براساس میزان بهره‌وری پردازنده‌ی آنها به صورت نزولی مرتب می‌کنیم و هر ماشین‌های مجازی را به میزبانی تخصیص می‌دهیم که کمترین افزایش توان مصرفی را به ازای آن تخصیص داشته باشیم. شبه کد زیر الگوریتم PABFD را نشان می‌دهد.

```
Algorithm Power Aware Best Fit Decreasing (PABFD)
1.Input: hostlist, vmlist   Output:allocation of VMs
2.vmlist.SortDecreasingUtilization
3.foreach vm in vmlist do
4.   minpower ← MAX
5.   allocatedhost ← NULL
6.   foreach host in hostlist do
7.      if host has enough resources for vm then
8.         power ← estimatepower (host,vm)
9.         if power < minpower then
10.           allocatedhost ← host
11.           minpower ← power
12.  if allocatedhost ≠ NULL then
13.     allocation.add(vm , allocatedhost)
14. return allocation
```

جدول ۴- میزان توان مصرفی سرورها بر حسب وات در سطوح متفاوت بهره‌وری

سرور/سطوح متفاوت بهره‌وری	٪۰	٪۱۰	٪۲۰	٪۳۰	٪۴۰	٪۵۰	٪۶۰	٪۷۰	٪۸۰	٪۹۰	٪۱۰۰
HP ProLiant ML110 G3	۱۰۵	۱۱۲	۱۱۸	۱۲۵	۱۳۱	۱۳۷	۱۴۷	۱۵۳	۱۵۷	۱۶۴	۱۶۹
HP ProLiant ML110 G4	۸۶	۸۹.۴	۹۲.۶	۹۶	۹۹.۵	۱۰۲	۱۰۶	۱۰۸	۱۱۲	۱۱۴	۱۱۷
HP ProLiant ML110 G5	۹۳.۷	۹۷	۱۰۱	۱۰۵	۱۱۰	۱۱۶	۱۲۱	۱۲۵	۱۲۹	۱۳۳	۱۳۵
IbmX3250XeonX3470	۴۱.۶	۴۶.۷	۵۲.۳	۵۷.۹	۶۵.۴	۷۳	۸۰.۷	۸۹.۵	۹۹.۶	۱۰۵	۱۱۳
IbmX3250XeonX3480	۴۲.۳	۴۶.۷	۴۹.۷	۵۵.۴	۶۱.۸	۶۹.۳	۷۶.۱	۸۷	۹۶.۱	۱۰۶	۱۱۳

عاطفه یکتا اول متولد سال ۱۳۶۹، مدرک کارشناسی خود را در رشته‌ی مهندسی کامپیوتر گرایش نرم‌افزار در سال ۱۳۹۱ از دانشگاه علم و صنعت ایران گرفته است و سپس در سال ۱۳۹۳ مدرک کارشناسی ارشد را در رشته مهندسی فناوری اطلاعات گرایش شبکه‌های کامپیوتری از دانشگاه علم و صنعت ایران دریافت نمود. زمینه‌های تحقیقاتی مورد علاقه ایشان شبکه‌های کامپیوتری، سیستم‌های توزیع شده، رایانش ابری و امنیت شبکه می‌باشد.
آدرس پست‌الکترونیکی ایشان عبارت است از:
atefe.yekta@gmail.com

محمود فتحی مدرک کارشناسی خود را در رشته‌ی مهندسی برق- الکترونیک در سال ۱۳۶۳ از دانشگاه علم و صنعت ایران گرفته است، و سپس در سال ۱۳۶۶ مدرک کارشناسی‌ارشد را در گرایش معماری سیستم‌های کامپیوتری از دانشگاه برادفورد انگلستان، برادفورد غربی و دکترا را از دانشگاه یومیست منچستر انگلستان - موسسه‌ی علوم و فن‌آوری در سال ۱۳۷۰ در حوزه‌ی معماری کامپیوتر سیستم‌های پردازش تصویر دریافت کرده است. از سال ۱۳۷۰ تا سال ۱۳۹۶ وی به عنوان هیئت علمی در دانشکده کامپیوتر دانشگاه علم و صنعت ایران مشغول به آموزش و پژوهش بوده و در حال حاضر بازنشسته دانشکده‌ی مهندسی کامپیوتر می‌باشد. علاقه‌مندی‌های تحقیقاتی وی شامل: مدلسازی سخت‌افزار، شبکه‌های کامپیوتری (کیفیت سرویس، شبکه‌های خودرویی، همپوشان)، نظارت سلامت (محاسبات زیستی و زیست انفورماتیک)، سیستم‌های حمل و نقل هوشمند، پردازش توزیع شده‌ی ویدیو و تصویر، رایانش موازی، توزیع شده و ابری، فناوری اطلاعات سبز است. وی دارای تالیفات متعددی در زمینه کتاب (۷ کتاب)، مقالات مجلات (۱۳۲ مقاله)، مقالات کنفرانس‌های بین‌المللی (۱۷۶ مقاله) و مقالات کنفرانس‌های بین‌المللی و ایرانی (۱۸۴ مقاله) می‌باشد.
آدرس پست‌الکترونیکی ایشان عبارت است از:
mahfathy@iust.ac.ir

اطلاعات بررسی مقاله:

تاریخ ارسال: ۱۳۹۵/۰۸/۰۱
تاریخ اصلاح: ۱۳۹۶/۰۱/۲۶
تاریخ قبول شدن: ۱۳۹۷/۰۲/۰۵
نویسنده مرتبط: عاطفه یکتا اول، دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران.

[1]Data Center
[2]Scientific Workflow
[3]Application
[4]Utilization
[5]Workload
[6]Directed Acyclic Graph
[7]Nodes
[8]Edges
[9]Static Task Scheduling
[10]Dynamic Task Scheduling
[11]Non-Preemptive
[12]Task Duplication Heuristic
[13]Guided Random Search Base
[14]Dynamic Greedy Scheduling
[15]Greedy
[16]Dynamic Voltage Frequency Scaling (DVFS)
[17]Multi-Threading
[18]Migration
[19]Liza Wanga
[20]Earliest Task First
[21]Control Flow
[22]Data Flow
[23]Parent
[24]Haluk Topcuoglu