Journal of Soft Computing and Information Technology (JSCIT)

Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 8, Number 1, Spring 2019, pp. 65-79
Received: 01/06/2018, Revised: 12/24/2018, Accepted: 02/03/2019

An algorithm for minimizing energy consumption with reliability goal on multiple processors in a cloud computing environment

Monireh Safari Habib Abadi[1], Reihaneh Khorsand Motlagh Esfahani[2*], Mohammadreza Ramezanpour[3]
1- Department of Computer Engineering, Dolatabad Branch, Islamic Azad University, Isfahan, Iran
2*- Department of Computer Engineering, Dolatabad Branch, Islamic Azad University, Isfahan, Iran
3- Department of computer engineering, Mobarakeh branch, Islamic Azad University, Mobarakeh, Isfahan, Iran
[1]Msafari_1992@yahoo.com, [2*]R.khorsand@iauda.ac.ir, and [3]Ramezanpour@mau.ac.ir

Corresponding author address: Reihaneh Khorsand, Department of Computer Engineering, Dolatabad Branch, Islamic Azad University, Isfahan, Iran, Post Code: 83418-75185.

Abstract- The energy and time constrained task scheduling on multi-processing environments with different frequency levels is considered as an important optimization issue in a cloud computing. In addition, a processor executed with a reduced frequency will dynamically increase transient faults, which will weaken the reliability of the applications. Reliability is an important figure of merit of the system and it must be satisfied in safety-critical applications. In this paper, the parallel task scheduling problem in multi-processors has been explored to reduce energy and to increase reliability in the scheduling. Multi-processing and Dynamic Voltage and Frequency Scaling (DVFS) techniques can decrease the processor's frequency level as much as possible; therefore, the voltage consumption of the processor will be reduced. Here, a two-phase algorithm is proposed to minimize energy consumption with reliability goal on multiple processors. The first phase is for initial assignment and the second phase is for either satisfying the reliability goal or improving energy efficiency. Specifically, when the application's reliability goal cannot be achieved via initial assignment, based on our defined current reliability ratio, an enhanced algorithm is designed to satisfy application's reliability goal. The proposed algorithm compared with existing algorithms. Experimental results demonstrate that the proposed algorithm consume less energy while satisfying the application's reliability goal.

Keywords- Energy consumption reduction; Energy-aware scheduling; Reliability; Workflow tasks; DVFS

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۸، شماره ۱، بهار ۱۳۹۸ ، صفحه ۷۹-۶۵
دریافت: ۹۶/۱۰/۱۶، بازنگری: ۹۷/۱۰/۰۳، پذیرش: ۹۷/۱۱/۱۴

ارائه یک الگوریتم کاهش مصرف انرژی با هدف قابلیت اطمینان در پردازنده‌های چندگانه در محیط محاسبات ابری

منیره صفری حبیب آبادی [۱]، ریحانه خورسند مطلق اصفهانی [۲*]، محمدرضا رمضان پور [۳]
۱- گروه مهندسی کامپیوتر، واحد دولت آباد، دانشگاه آزاد اسلامی، اصفهان، ایران
۲*- گروه مهندسی کامپیوتر، واحد دولت آباد، دانشگاه آزاد اسلامی، اصفهان، ایران
۳- گروه مهندسی کامپیوتر، واحد مبارکه، دانشگاه آزاد اسلامی، مبارکه، اصفهان، ایران
[1]msafari_1992@yahoo.com, [2*]R.khorsand@iauda.ac.ir, and [3]Ramezanpour@mau.ac.ir

* نشانی نویسنده مسئول: ریحانه خورسند، اصفهان، دولت‌آباد، دانشگاه آزاد اسلامی واحد دولت‌آباد، دانشکده مهندسی کامپیوتر، کد پستی: ۷۵۱۸۵-۸۳۴۱۸.

چکیده- زمان‌بندی وظایف با محدودیت زمان و انرژی در محیط‌های چند پردازنده‌ای با سطوح مختلف فرکانس به عنوان یک مسئله بهینه‌سازی مهم در رایانش ابری مطرح است. به‌علاوه یک پردازنده که با فرکانس کاهش‌یافته کار می‌کند، به‌طور پویا باعث افزایش خرابی‌هایی می‌شود که این امر باعث ضعیف شدن قابلیت اطمینان برنامه‌های کاربردی می‌شود. قابلیت اطمینان یک عنصر مهم از شایستگی سیستم است و باید در برنامه‌های کاربردی امنیتی مورد توجه قرار گیرد. در این مقاله مسئله زمان‌بندی وظایف در چندین پردازنده با هدف کاهش مصرف انرژی و افزایش قابلیت اطمینان مورد توجه قرار گرفته است. با استفاده از چند پردازنده‌ای و تکنولوژی مقیاس‌بندی ولتاژ و فرکانس پویا (DVFS) می‌توان سطح فرکانس پردازنده را تا حد امکان پایین آورد و در نتیجه ولتاژ مصرفی آن پردازنده را کاهش داد. در اینجا یک الگوریتم دو مرحله‌ای برای به حداقل رساندن مصرف انرژی با هدف قابلیت اطمینان در پردازنده-های چندگانه ارائه می‌شود. مرحله اول برای تخصیص اولیه و مرحله دوم برای ارضاء هدف قابلیت اطمینان و بهبود بهره‌وری انرژی است. به‌طور خاص، هنگامی که هدف قابلیت اطمینان نرم افزار از طریق تخصیص اولیه نمی‌تواند به‌دست آید، بر اساس ضریب اطمینان فعلی تعریف شده‌ی ما، یک الگوریتم بهبودیافته طراحی می‌شود که می‌تواند هدف قابلیت اطمینان را ارضاء کند. الگوریتم پیشنهادی با الگوریتم‌های موجود مقایسه شد. نتایج آزمایش‌ها نشان می‌دهد که الگوریتم پیشنهادی مصرف انرژی کمتری دارد درحالی‌که هدف قابلیت اطمینان برنامه‌های کاربردی را نیز برآورده می‌کند.

واژه‌های کلیدی: کاهش مصرف انرژی، زمانبندی انرژی آگاه، جریان کار، قابلیت اطمینان، DVFS.

۱- مقدمه

رشد سریع سیستم‌های کامپیوتری و ارتباطی منجر به پدیده محاسبات ابری شد که با استفاده از اینترنت، خدمات مختلفی را به کاربران ارائه می‌دهد. کاربران به‌منظور استفاده از سرویس‌های مختلف ابری نیاز دارند به مرکز داده متصل شوند و بر طبق تقاضا سرویس‌های خود را تحویل بگیرند. در طول سال‌های اخیر با رشد نیازهای محاسباتی کاربران، مصرف انرژی مراکز داده افزایش چشمگیری یافته و به یک مسئله بحرانی تبدیل شده است [۱]. یکی از تحقیقات تخمین می‌زند که یک مرکز داده با ۵۰۰۰۰ گره محاسبه‌گر ممکن است بیش از یک صد میلیون کیلووات ساعت، انرژی مصرف کند[۲]. این مصرف بالای انرژی با ایجاد زیر ساخت ابر باعث افزایش هزینه‌ها و کاهش میزان سود شده است. همچنین باعث تولید گازهای گلخانه‌ای مانند دی‌اکسید کربن که عامل مخربی برای محیط زیست به شمار می‌آید، شده است. این موضوع منجر به پیدایش مبحثی به نام محاسبات ابری سبز شد.

محاسبات ابری سبز قصد دارد تا با طراحی الگوریتم‌ها و روش‌هایی برای کاهش مصرف انرژی از منابع به شکلی موثر و مقرون به صرفه، استفاده کند. برای دستیابی به این امر نیاز است که منابع مراکز داده با تکنیک‌های بهره‌وری انرژی مدیریت شوند به‌طوری که انرژی مصرفی را کاهش دهند. یکی از جنبه‌های مهم محاسبات ابری زمان‌بندی تعداد زیادی درخواست‌های وظیفه گردش‌کار است که به وسیله کاربران ارسال می‌شوند و منابع در مراکزداده ابر نه تنها باید این وظیفه‌های گردش‌کار را اجرا کنند بلکه باید با تکنیک‌های بهره‌وری انرژی مدیریت شوند. هدف زمان‌بندی پیدا کردن یک منبع از مجموع منابع پردازشی است که یک وظیفه برای پردازش به آن نیاز دارد به‌طوری که بتوان وظایف بیشتری را در زمان تعیین شده توسط کاربر پردازش کرد. زمان‌بندی مناسب از یک طرف موجب کاهش میزان مصرف انرژی مرکز داده و کاهش نقض توافقات سطح سرویس (SLA[۱]) می‌شود و از طرف دیگر افزایش بهره‌وری منابع را فراهم می‌کند.

تاکنون تکنیک‌های بهره‌وری انرژی زیادی وجود دارد که باعث کاهش مصرف انرژی مراکز داده می‌شود. به عنوان مثال DPM [۲] با خاموش و روشن کردن اجزا به صورت پویا کار می‌کند[۳]، DVFS[۳] با تغییر ولتاژ و فرکانس اجزا به صورت پویا کار می‌کند [۴-۵]، DCD [۴] با غیر فعال کردن اجزا کار می‌کند [۶] و DPS [۵] با تنظیم کارایی متناسب با مصرف انرژی به صورت پویا کار می‌کند که یکی از کاراترین این روش‌ها مورد دوم یعنی DVFS می‌باشد.

روش DVFS یک روش ذخیره انرژی معمول برای سیستم‌های کامپیوتری عظیم می‌باشد که قادر است مصرف انرژی پردازنده‌های سیستم‌های کامپیوتری را با کاهش فرکانس عامل پایین آورد. بر اساس این‌که مصرف انرژی ارتباط مستقیمی با مربع ولتاژ عرضه و فرکانس دارد [۷-۸] کاهش مصرف برق زیادی را می‌توان با تغییر بین ولتاژ / فرکانس پردازنده در حین اجرای وظیفه به‌دست آورد، در حالی‌که کارایی تضمین شود. در حال حاضر، بسیاری از پردازنده‌های جریان اصلی مانند Intel، AMD و ARM وجود دارد که تکنیک‌های DVFS را برای بهبود بهره‌وری انرژی مورد استفاده قرار می‌دهند ولی با کاهش میزان فرکانس اجرا، احتمال خطاهای گذرا در پردازنده به وضوح وجود دارد که قابلیت اطمینان برنامه‌ها را تضعیف می‌کند [۹-۱۰]. قابلیت اطمینان یک برنامه به صورت احتمال اینکه برنامه به درستی اجرا شود تعریف می‌شود [۹]. برای بسیاری از برنامه های جاسازی شده، قابلیت اطمینان سیستم یک معیار کارایی مهم است. از این رو، ارضاء هدف قابلیت اطمینان برنامه برای سیستم‌های امنیتی بسیار مهم است [۱۰]. به طورکلی، بهبود قابلیت اطمینان سیستم ممکن است همراه با مصرف انرژی بیشتری باشد. بنابراین، باید میان قابلیت اطمینان و مصرف انرژی تعادل برقرار شود.

تاکنون تعدادی از الگوریتم ها برای بهبود بهره‌وری انرژی با تضمین قابلیت اطمینان برنامه های کاربردی ارائه شده است [۱۰-۱۲] ولی این مقالات عمدتا براساس سیستم های تک پردازنده کار می‌کنند.

تاکنون تحقیقات زیادی در رابطه با زمانبندی و استفاده از تکنیک DVFS ارائه شده است[۱۳-۱۵] ولی این الگوریتم‌ها قابلیت اطمینان نرم افزار را درنظر نمی‌گیرند. برخی از الگوریتم‌ها برای برنامه های موازی در سیستم های ناهمگن پیشنهاد شده‌اند، مانند الگوریتم MaxRe [۱۶]، الگوریتم MRCRG [۱۷] و الگوریتم ESRG [۱۸] ولی MaxRe و الگوریتم MRCRG روش DVFS را برای بهبود بهره وری انرژی استفاده نکردند. الگوریتم ESRG از تکنیک DVFS برای بهبود بهره‌وری انرژی استفاده می‌کند اما همیشه نمی‌تواند قابلیت اطمینان برنامه را برآورده سازد. علاوه بر این، تمام این الگوریتم‌ها ابتدا هدف قابلیت اطمینان برنامه را به هدف قابل اطمینان هر وظیفه انتقال می‌دهند و سپس از روش اکتشافی برای به حداقل رساندن مصرف منابع (انرژی) استفاده می‌کنند. در این مقاله روشی جدید برای کاهش مصرف انرژی با در نظر گرفتن هدف قابلیت اطمینان بر روی سیستم‌های

چندپردازنده پیشنهاد شده است. بهره‌وری انرژی روش پیشنهادی بسیار بالاتر از روش های ذکر شده در بالا است.

در روش پیشنهادی، ابتدا وظایف گردش‌کار زمان‌بندی اولیه می-شوند تا قابلیت اطمینان اولیه و مصرف انرژی اولیه بدست آید، سپس اگر قابلیت اطمینان بدست آمده کمتر از هدف بود آن‌را افزایش می‌دهد در غیر اینصورت مصرف انرژی را با استفاده از روش DVFS کاهش می‌دهد.

ادامه مقاله به شرح زیر می‌باشد: بخش ۲ یک مرور کلی از کارهای مرتبط فراهم می‌کند. بخش ۳ مدل‌های سیستم، وظیفه، انرژی و قابلیت اطمینان را توضیح می‌دهد. بخش ۴ الگوریتم کاهش مصرف انرژی با هدف قابلیت اطمینان در پردازنده‌های چندگانه را بیان می‌کند. بخش ۵ ارزیابی روش پیشنهادی را بیان می‌کند و در نهایت در بخش ۶ نتیجه گیری و کارهای آینده بحث می‌شود.

۲- کارهای مرتبط

پیدایش رایانش ابری یک راه حل برای این است که اجازه دسترسی به داده‌های مشترک را برای اجرای محاسبات به‌طور موثر در منابع ابر فراهم کند[91]. در محیط ابر، زمان‌بندی وظیفه برای تخصیص وظایف به منابع مناسب برای به حداکثر رساندن استفاده از منابع، افزایش عملکرد و کیفیت سرویس استفاده می-شود. با توجه به ماهیت NP-Complete بودن زمان‌بندی وظایف در حالت کلی تحقیقات زیادی در این زمینه انجام شده است. برای این منظور، روش‌های اکتشافی و فراشناختی برای حل مساله پیچیدگی چند جمله‌ای تلاش می‌کنند [۲۰-۲۵]. گروه دیگری از مطالعات این مسئله NP-complete را با استفاده از رابطه‌های ریاضی مورد بررسی قرار دادند [۲۶-۲۷]. یکی از محبوب‌ترین روش‌ها برای زمان‌بندی وظایف گردش‌کار، زمان‌بندی لیست می-باشد. الگوریتم‌های زمان‌بندی لیست موجود عبارتند از: مسیر بحرانی پویا (DCP) [۲۸]، زمان‌بندی سطح پویایی (DLS) [۲۹]، مسیر بحرانی در پردازنده)CPOP(، اولین دوره پایان ناهمگن (HEFT) [۳۰] و غیره. به‌عنوان مثال، الگوریتم HEFT که در مرجع [۳۰] معرفی شده، یکی از معروف‌ترین الگوریتم‌های زمان-بندی لیست ایستا بر روی پردازنده‌های ناهمگن است. در این روش وظیفه با بالاترین رتبه را انتخاب می‌کند و آن را به یک منبع پردازشی اختصاص می‌دهد که منجر به کاهش زمان اجرای وظیفه می‌شود. نویسندگان در مرجع [۲۰] پیشنهاد دو روش پیش‌بینی شده LOSS و GAIN (براساس HEFT) را برای گردش‌های کاری در محیط شبکه‌ای دادند که هر دو برای برآورد بودجه مشخص شده کاربر سعی در بهینه‌سازی زمان یا هزینه دارند. بنابراین در یک زمان، تنها یکی از اهداف، یعنی زمان یا هزینه، بهینه می‌شود.

DVFS ابزار دستیابی به صرفه جویی در انرژی در تجهیزات سخت‌افزاری است[۲۱]. DVFS براساس نیازهای متفاوت ظرفیت محاسبات برنامه کاربردی، برای دستیابی به صرفه‌جویی در انرژی، به صورت پویا فرکانس و ولتاژ در حال اجرای تراشه را تنظیم می-کند. فرکانس پایین می‌تواند مصرف انرژی را کاهش دهد. با این حال، صرفه‌جویی در انرژی را نمی‌توان با صرفه‌جویی در فرکانس ذخیره کرد، زیرا برای یک کار خاص، مصرف انرژی تنها زمانی می‌تواند کاهش یابد که ولتاژ و فرکانس به طور همزمان کاهش یابند. الگوریتم‌های بسیاری از روش DVFS استفاده کرده‌اند. در واقع پیاده‌سازی DVFS بستگی به پیش‌بینی موفقیت‌آمیز تعداد و زمان اجرای وظایف پردازشی در میزبان دارد. بر مبنای تکنولوژی DVFS، نویسندگان در مرجع [۲۲] یک الگوریتم برای کاهش مصرف انرژی وظایف موازی پیشنهاد کردند. با فرض این‌که زمان اجرای کلی افزایش نیابد، الگوریتم به طور مناسب زمان اجرای کارهای غیررسمی را افزایش می‌دهد درحالی‌که ولتاژ و فرکانس پردازنده ها را کاهش می‌دهد تا هزینه‌های انرژی کاهش یابد. شکی نیست که این الگوریتم یک سیاست برنامه‌ریزی انرژی با استفاده از محیط چند هسته ای است. با این وجود، عملیات موازی برای تقاضای منابع دیگر علاوه بر CPU را در نظر نمی‌گیرد. در مرجع[۲۳] نویسندگان با تمرکز بر مشکلات مصرف بالای انرژی و انتشار کربن در مراکز داده ابر، با استفاده از تکنیک DVFS به طراحی استراتژی زمانبندی برای صرفه‌جویی در انرژی برای حل این مشکل پرداختند. در مرجع [۲۴] نویسندگان یک روش زمان-بندی برای کاهش مصرف انرژی وظایف موازی که دارای محدودیت تقدم می‌باشند را در خوشه همگن با تکنولوژی DVFS ارائه دادند. این مدل برای برنامه هایی که دارای زمان بیکاری برای کارهای غیر حساس هستند با پایین آوردن ولتاژ منبع برای کاهش مصرف انرژی و گسترش زمان اجرای کار ارائه شد. این مدل گره-های همگن را به عنوان عناصر پردازشی با سرعت پردازش یکسان در نظر می‌گیرد و از توافق سطح سرویس سبز به منظور پذیرش قابلیت تحمل عملکرد از دست داده استفاده می‌کند. مطالعه زمان بیکاری برای کارهای غیر حساس و گسترش زمان اجرای آنها در مرجع [۲۵] انجام شده است. نویسندگان در مرجع [۲۵] دو الگوریتم PATC و PALS را ارائه کردند. روش پیشنهادی آنها به طورکلی به کاهش مصرف انرژی بدون افزایش زمان اجرای وظیفه پرداخت ولی پیچیدگی‌های چالش‌برانگیزی برای پیاده‌سازی در محیط واقعی داشت. در تلاشی دیگر یک روش کاهش مصرف انرژی با استفاده از روش مقیاس‌گذاری فرکانس/ولتاژ پویا در مرجع

[۲۶] ارائه شد که در آن سرویس‌دهنده توسط کاهش فرکانس عامل با سطوح مختلف ولتاژ کار می‌کند که در تعداد وظایف پایین بهبود مصرف انرژی کمتری نسبت به سایر حالات دارد. پس از آن یک مدل زمان‌بندی انرژی آگاه که داده های بزرگ را در محیط ابر تقسیم و زمان‌بندی می‌کند در [۲۷] ارائه شد. هدف اصلی از آن مدل افزایش بهره‌وری نرم‌افزار و کاهش مصرف انرژی روی منابع اساسی است. این الگوریتم ظرفیت محاسبات موجود و ظرفیت حافظه و مصرف انرژی توسط هر قطعه‌بندی وظایف را در نظر می‌گیرد و زمان‌بندی می‌کند. الگوریتم مدیریت قدرت بر خط برای به حداقل رساندن مصرف انرژی برای زمان‌بندی تک پردازنده‌های زمان واقعی تحت شرایط قابلیت اطمینان در مرجع [۲۸] می‌باشد. این روش به صورت پویا فرکانس وظایف را توسط بهینه‌سازی زمان اجرا که توسط بازگرداندن بلوک‌های اضافی افزایش می‌یابد دستکاری می‌کند. در مرجع [۲۹] نویسندگان زمان‌بندی را برای تحمل‌پذیری خطا برای یک برنامه موازنه قابل اطمینان بر روی سیستم‌های توزیع شده ناهمگن زمان‌بندی کردند و یک الگوریتم زمان‌بندی تحمل‌پذیری خطا با صرفه جویی انرژی با هدف قابلیت اطمینان ارائه دادند. هدف کاهش مصرف انرژی با درنظرگیری قابلیت اطمینان بر اساس یک برنامه تکرار فعال بود. ژو و همکاران [۱۴] قابلیت اطمینان یک سیستم زمان واقعی را به عنوان احتمال کامل اجرای تمام وظایف بدون رخداد خطا بیان کردند. آنها همچنین یک مدل خطی و نمایشی برای ضبط اثرات DVS [۶] بر میزان خطای گذرا ارائه دادند و نشان دادند که مدیریت انرژی از طریق DVS می‌تواند قابلیت اطمینان سیستم را کاهش دهد. بر اساس این مدل، آن‌ها یک طرح بهبودی را برای برنامه-ریزی در زمان واقعی انجام دادند که می‌تواند مصرف انرژی را بدون کاهش اعتبار کاهش دهد. زی و همکاران [۱۸] الگوریتم MRCRG را برای برنامه‌های موازی بر روی سیستم های جاسازی شده ناهمگن پیشنهاد کردند. الگوریتم آن‌ها ابتدا قابلیت اطمینان هر وظیفه را محاسبه می‌کند و سپس کار را به پردازنده با کمترین مصرف منابع اختصاص می‌دهد. بر اساس سیستم های جاسازی شده ناهمگن، زای و همکاران در مرجع [۱۰] الگوریتم ESRG را برای به حداقل رساندن مصرف انرژی با هدف قابلیت اطمینان یک برنامه موازی پیشنهاد دادند. ژانگ و همکاران [۱۳] همچنین الگوریتم ژنتیک دو هدفه برای بهینه سازی کارایی و قابلیت اطمینان انرژی برای برنامه موازی بر روی سیستم‌های محاسباتی ناهمگن را پیشنهاد دادند. ژائو و همکاران [۱۶] یک الگوریتم MaxRe را که با استفاده از تکثیر پویای وظایف مختلف، قابلیت اطمینان برنامه کاربردی را برآورده می‌کند پیشنهاد کردند. در [۱۲] چندین الگوریتم زمان‌بندی با صرفه جویی انرژی با تضمین قابلیت اطمینان برای مجموعه ای از وظایف زمان واقعی در یک سیستم تک پردازنده ارائه شد. این تکنیک بر اساس سیستم های تک پردازنده است و برای بهبود بهره‌وری انرژی از DVFS استفاده می‌شود.

۳- مدل‌ها

در این بخش، مدل‌های سیستم، وظیفه، انرژی و قابلیت اطمینان استفاده شده در این مقاله ارائه می‌شوند:

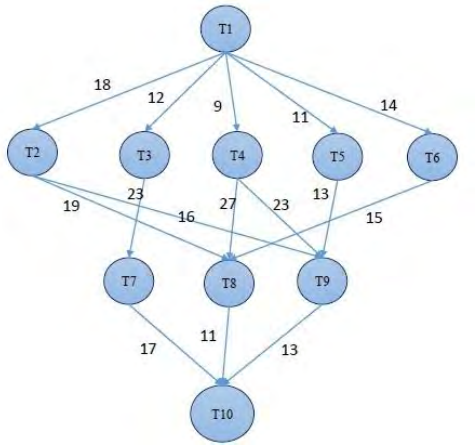
۳-۱- مدل سیستم

در این مقاله سرویس‌دهنده‌هایی در نظر گرفته شد که در یک مرکز داده با تکنولوژی DVFS فعال هستند که قابلیت اجرای هر برنامه کاربردی را دارا می‌باشند. $R = \{r_i\}$ مجموعه منابع همگن می‌باشد که در یک پیکربندی کامل به همدیگر مرتبط هستند. هر منبع در مجموعه R دارای تکنولوژی DVFS می‌باشد که این بدین معناست که می‌توانند در سطوح ولتاژ و فرکانس مختلف کار کنند. ما مجموعه ولتاژ $v = \{v_i\}$ و مجموعه فرکانس ساعت$f = \{f_i\}$ را تعریف می‌کنیم. زمانی‌که منبع در سطح ولتاژ v_iکار می‌کند، سطح فرکانس ساعت برابر مقدارf_i خواهد بود و در زمانی که منبع بیکار است مقدار سطح ولتاژ آن به مقدار v_{min} تغییر می‌کند که کمترین مقدار ولتاژ برای منبع می‌باشد و از مقدار صفر بیشتر است. لازم به ذکر است این کار برای ذخیره بیشتر انرژی انجام می‌شود. در این کار محاسبات و ارتباطات با همدیگر همپوشانی دارند یعنی زمانی که پردازنده در حال انجام کاری روی وظیفه است می‌تواند از پردازنده دیگری داده دریافت کند.

۳-۲- مدل وظیفه

برنامه‌های موازی متشکل از مجموعه‌ای از وظایف با محدودیت تقدم می‌باشند که اصطلاحاً به آنها گردش‌کار گفته می‌شود که با گراف جهت دار مستقیم(DAG[۷]) نشان داده می‌شود. $G = \{T, E\}$ نشان دهنده گراف گردش‌کار است که T مجموعه گره‌ها در گراف G است که نشان دهنده n وظیفه مختلف $t_i \in T(1 \le i \le n)$ است که در هر منبع می‌تواند اجرا شود. E مجموعه یال‌های گراف G می‌باشد که $e_{i,j} = (t_i, t_j) \in E(1 \le i \le n\,, 1 \le j \le n, i \ne$ j) نشان‌دهنده وابستگی بین وظایف است که t_i وظیفه والد و t_j وظیفه فرزند می‌باشد. وظیفه t_j شروع نمی‌شود مگر این‌که کار وظیفه t_i به اتمام رسیده باشد. وظیفه بدون یال ورودی را وظیفه ورودی n_{entry} می‌نامیم و وظیفه بدون یال خروجی را وظیفه خروجی n_{exit} می‌نامیم. شکل ۱ گراف گردش‌کار نمونه را نشان

می‌دهد. از آنجا که زمان اجرای وظایف یکسان بر روی پردازنده-های مختلف متفاوت است، جدول ۱ نشان‌دهنده زمان اجرای وظایف متفاوت گردش‌کار نمونه در پردازنده‌های متفاوت است.



شکل۱: گراف نمونه گردش کار[۷]

جدول۱: زمان اجرای وظایف در پردازنده های متفاوت از گراف نمونه گردش کار

task	r1	r2	r3	Rank
t1	14	16	9	108
t2	13	19	18	77
t3	11	13	19	80
t4	13	8	17	80
t5	12	13	10	69
t6	13	16	9	63.3
t7	7	15	11	42.7
t8	5	11	14	35.7
t9	18	12	20	44.3
t10	21	7	16	14.7

وزن $w(t_i)$ اختصاص داده شده به هر وظیفه t_i نشان‌دهنده تعداد میلیون دستورالعمل (MI) برای اجرای وظیفه است و $w(e_{i,j})$ اختصاص داده شده به هر یال $e_{i,j}$ نشان‌دهنده مقدار داده منتقل شده از وظیفه t_i به وظیفه t_j است که در دو منبع متفاوت هستند. هزینه ارتباط درون پردازنده‌ها صفر در نظر گرفته شده است. زمان اجرا $ET(t_i,r_j)$ بر طبق رابطه ۱ نشان‌دهنده زمان اجرای تخمینی وظیفه t_i است.

$$ET(t_i,r_j)=\frac{w(t_i)\times CPI}{f_{j,k}} \qquad (۱)$$

وظیفه t_i روی منبع r_j در فرکانس $f_{j,k}$ اجرا می‌شود، k سطح فرکانس منبع می‌باشد که منبع می‌تواند در سطوح مختلف فرکانس کار کند. CPI تعداد سیکل دستورالعمل منبع r_j است که توسط ساختار کامپیوتر و دستورالعمل تعیین می‌شود.

زمان/هزینه مورد نیاز برای انتقال واحد داده از وظیفه والد t_p (واقع در منبع r_s) به وظیفه جاری t_i (واقع در منبع r_j) توسط رابطه ۲ محاسبه می‌شود.

$$C_{p,i}^{s,j}=w(e_{p,i}).B_{s,j} \qquad (۲)$$

که $B_{s,j}$ زمان مورد نیاز برای انتقال واحد داده در بین دو منبع است. در این مقاله هر وظیفه مقدار مهلت زمانی مشخصی دارد که در آن مهلت زمانی باید اجرا شود. در واقع وظایف بر اساس مهلت زمانی خود در واحد سطح مرتب می‌شوند. برای هر وظیفه مقدار EST (ti) [۸] نشان‌دهنده اولین زمانی است که وظیفه t_i می‌تواند اجرای خود را در منبع مشخص انجام دهد و مقدار LFT [۹] نشان دهنده آخرین زمانی است که وظیفه می‌تواند اجرا شود. این مقادیر توسط رابطه‌های ۳ و ۴ محاسبه می‌شود.

$$EST(t_i,r_j)=\begin{cases}0 & if\ t_i=Entry\ task\\ max_{t_p\in pred(t_i)}\{EST(t_p)+ET(t_p)+C_{p,i}\} & otherwise\end{cases} \qquad (۳)$$

$$LFT(t_i,r_j)=\begin{cases}D & if\ t_i=Exit\ task\\ min_{t_j\in succ(t_i)}\{LFT(t_j)-ET(t_j)-C_{j,i}\} & otherwise\end{cases} \qquad (۴)$$

بدین معنی که برای آخرین وظیفه، آخرین زمان پایان مجاز برابر با مهلت زمانی می‌باشد و برای حالات غیر از گره آخر، کمترین آخرین زمان پایان مجاز وظیفه بعدی منهای زمان اجرای آن منهای زمان ارتباطات آن، زمان پایان مجاز برای این وظیفه فعلی می‌شود.

۳-۳- مدل انرژی و هزینه

مصرف انرژی منابع برای اجرای کار به دو بخش مصرف انرژی پویا ($E_{dynamic}$) و مصرف انرژی ایستا (E_{static}) تقسیم می‌شود[۳۰]. روش‌های ایستا عملیات بهینه‌سازی را در زمان طراحی انجام می-دهند و تکنیک‌های پویا شامل روش‌هایی است که در زمان اجرا رفتار سیستم را با نیازمندی‌های فعلی منابع و مشخصات پویای سیستم تطبیق می‌دهند. از آنجا که گران‌ترین و زمان‌برترین قسمت، مصرف انرژی پویا است[۳۰]، مصرف انرژی ایستا در این مقاله در نظر گرفته نشده است. اتلاف توان پویا به صورت رابطه ۵ محاسبه می‌شود:

$$P_{dynamic}=K\times v_{j,s}^2\times f_s \qquad (۵)$$

که K یک پارامتر ثابت مربوط به توان پویا بسته به ظرفیت دستگاه است. $v_{j,s}$ ولتاژ عرضه شده در سطح S روی منبع r_j و f_s نشان-دهنده فرکانس مطابق با $v_{j,s}$ است. بر این اساس، کل مصرف انرژی در هنگام کار ماشین آلات را می‌توان به صورت رابطه ۶ محاسبه کرد:

$$E_{busy}(t_i)=\sum_{i=1}^{n}K\times v_{i,p_{j,s}}^2\times f_{r_{j,s}}\times t_{i,j} \qquad (۶)$$

که $t_{i,j}$ زمان اجرای وظیفه n_i روی منبع r_j و $v_{i,r_{j,s}}$ نشان‌دهنده این است که وظیفه n_i بر روی منبع r_j تحت ولتاژ v_s زمان‌بندی شده است. علاوه بر این، $f_{r_{j,s}}$ نشان‌دهنده فرکانس منبع r_j با سطح ولتاژ s است. ولتاژ عرضه و فرکانس را نمی‌توان در طول دوره آماده به کار منبع با مقدار صفر تنظیم کرد، ولتاژ در کمترین حالت v_{lowest} باعث صرفه جویی در انرژی می‌شود، مصرف انرژی در دوره بیکاری برای تمام منابع موجود را می‌توان به صورت زیر تعریف کرد:

$$E_{idle} = \sum_{j=1}^{p} K \times v_{i_{lowest}}^2 \times f_{j_{lowest}} \times t_{j_{idle}} \qquad (۷)$$

که در آن $v_{i_{lowest}}$ و $f_{j_{lowest}}$ ولتاژ و فرکانس منبع r_j تحت کمترین ولتاژ به ترتیب هستند و $t_{j_{idle}}$ نشان دهنده زمان بیکاری منبع r_j است. بر اساس رابطه ۷، کل انرژی مصرفی یک برنامه DAG را می‌توان به صورت زیر محاسبه کرد:

$$E_{total} = E_{busy} + E_{idle} \qquad (۸)$$

نقض توافق سطح سرویس زمانی اتفاق می‌افتد که فراهم‌کننده سرویس نتواند شرایط و قوانین توافق سطح سرویس را تامین کند. در این مقاله ما تفاوت بین مهلت زمانی و زمان پایان واقعی را محاسبه می‌کنیم و مشخص می‌کنیم که نقض توافق سطح سرویس اتفاق افتاده یا خیر که مقادیر آن توسط رابطه ۹ محاسبه می‌شود[۳۱].

$$TD_{t_i} = FT_{t_i} - DL_{t_i} \qquad (۹)$$

که TD_{t_i} تاخیر زمانی برای وظیفه t_i می‌باشد و FT_{t_i} زمان پایان و DL_{t_i} مهلت زمانی می‌باشد. اگر تاخیر زمانی برای یک وظیفه گردش کار بزرگتر از صفر باشد، نقض توافق سطح سرویس اتفاق افتاده است.

۳-۴- مدل قابلیت اطمینان

خطا ها ممکن است در طی اجرای یک برنامه رخ دهند که می-تواند به خطا های گذرا و خطا های دائمی تقسیم شوند. از آنجا که خطا های گذرا بیشتر از خطاهای دائمی رخ می‌دهند [۸]، تنها خطاهای گذرا در این مقاله در نظر گرفته می‌شوند. فرض کنید که خطاهای گذرا برای یک کار در یک برنامه مبتنی بر DAG براساس توزیع پواسون باشند [۳۲]. هنگامی که وظیفه t_i بر روی پردازنده rj با حداکثر فرکانس اجرا می‌شود، قابلیت اطمینان از طریق رابطه ۱۰ محاسبه می شود:

$$R(t_i, rj, f_{j,max}) = e^{-\lambda_{j,max} \times w_{i,j}} \qquad (۱۰)$$

که $\lambda_{j,max}$ حداکثر مقدار خطای گذرا در واحد زمان روی اجرای پردازنده rj با حداکثر فرکانس را نشان می‌دهد.

با توجه به [۳]، در یک سیستم DVFS فعال، نرخ خطاهای گذرا $\lambda_{j,k}$ از پردازنده rj با فرکانس $f_{j,k}$ توسط رابطه ۱۱ محاسبه می‌شود:

$$\lambda_{j,k} = \lambda_{j,max} \times 10^{\frac{d_j \times (f_{j,max} - f_{j,k})}{f_{j,max} - f_{j,min}}} \qquad (۱۱)$$

که d_j بزرگتر از ۰ نشان دهنده میزان حساسیت خطا به مقیاس ولتاژ / فرکانس پردازنده rj است. بر اساس رابطه (۱۰) و (۱۱)، هنگامی که وظیفه t_i بر پردازنده rj با فرکانس $k_{f,j}$ اجرا می‌شود، قابلیت اطمینان وظیفه t_i را می‌توان با استفاده از رابطه۱۲ محاسبه کرد:

$$R(t_i) = R\left(t_i, rj, f_{j,k}\right) e^{-\lambda_{j,max} \times 10^{\frac{d_j \times (f_{j,max} - f_{j,k})}{f_{j,max} - f_{j,min}} \times \frac{w_{i,j} \times f_{j,max}}{f_{j,k}}}} \qquad (۱۲)$$

هنگامی که تمام وظایف اختصاص داده شد، می‌توان قابلیت اطمینان برنامه را با استفاده از رابطه ۱۳ محاسبه کرد:

$$R(G) = \prod_{i=1}^{N} R(t_i, r_{ap(t_i)}, f_{ap(t_i),af(t_i)}) \qquad (۱۳)$$

که N تعداد وظایف می‌باشد $r_{ap(t_i)}$ و $f_{ap(t_i),af(t_i)}$ منبع و فرکانس اختصاص داده شده به وظیفه t_i است. بر اساس معادله (۱۲)، حداکثر قابلیت اطمینان وظیفه t_i را می‌توان محاسبه کرد:

$$R_{max}(t_i) = max_{r_j \in r}\{R\left(t_i, rj, f_{j,max}\right)\} \qquad (۱۴)$$

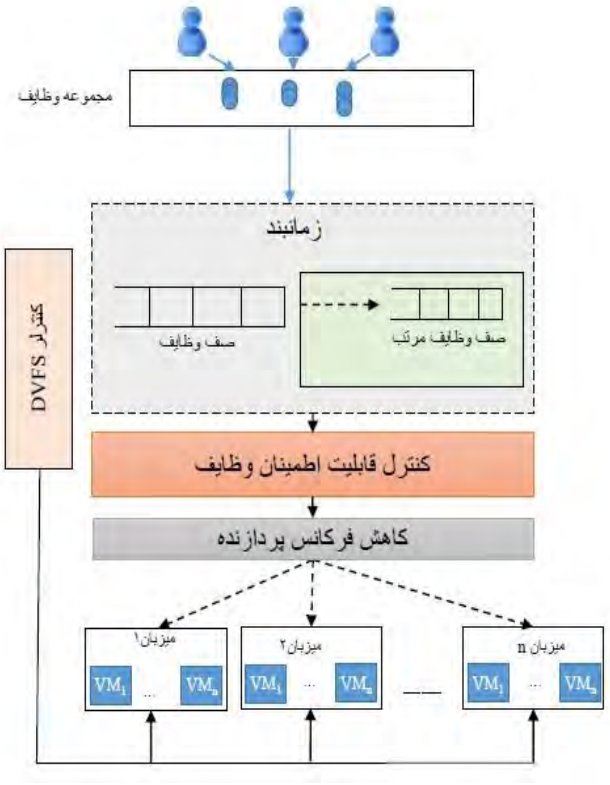
اگر به تمام وظایف، حداکثر قابلیت اطمینان اختصاص داده شود، قابلیت اطمینان نرم افزار حداکثر می‌شود که می‌توان آن را محاسبه کرد:

$$R_{max}(G) = \prod_{i=1}^{N} R_{max}(t_i) \qquad (۱۵)$$

۴- روش پیشنهادی

در محاسبات ابری، تکنولوژی مجازی‌سازی، ناهمگونی منابع را پنهان می‌کند. منابع دیگر ماهیت فیزیکی ندارند، اما به جای آن یک استخر بزرگ از منابع وجود دارد که شامل ماشین‌های مجازی (VMs) فراوان است. زمان‌بندی وظیفه یک روش پایه‌ای در محاسبات ابری است که توزیع وظایف محاسباتی در میان استخر

منابع مجازی را به عهده دارد. در این میان محاسبات ابری سبز قصد دارد تا با طراحی الگوریتم‌ها و روش‌هایی برای کاهش مصرف انرژی، از منابع به شکلی موثر و مقرون به صرفه، استفاده کند. برای دستیابی به این امر نیاز است که منابع مراکز داده با تکنیک‌های بهره‌وری انرژی مدیریت شوند به‌طوری که انرژی مصرفی را کاهش دهند و همچنین قابلیت اطمینان برنامه نیز مورد توجه قرار گیرد. چارچوب الگوریتم پیشنهادی حداقل مصرف انرژی با هدف قابلیت اطمینان در شکل ۲ نشان داده شده است. الگوریتم پیشنهادی به دو مرحله تقسیم می‌شود. مرحله اول برای تخصیص اولیه است و مرحله دوم برای ارضاء هدف قابلیت اطمینان و بهبود بهره‌وری انرژی است. تخصیص اولیه برای پیدا کردن پردازنده با حداقل مصرف انرژی برای اجرای کار بدون استفاده از تکنیک DVFS است. در مرحله دوم، زمانی که قابلیت اطمینان نرم افزار به‌دست آمده در مرحله اول پایین‌تر از قابلیت اطمینان هدف باشد، الگوریتم باید قابلیت اطمینان وظایف خاص را تا زمانی که هدف قابل اطمینان برنامه برآورده شود افزایش دهد. در غیر این صورت، روش DVFS می‌تواند برای بهبود بهره-وری انرژی استفاده شود. برای اطمینان از نظم وظیفه در هنگام اجرای برنامه لازم است اولویت وظایف معرفی شود.



شکل ۲: الگوریتم کاهش مصرف انرژی برای برآوردن هدف قابلیت اطمینان در پردازنده‌های چندگانه در محیط محاسبات ابری

۴-۱- طرح ریزی

۴-۱-۱- مرتب سازی وظایف و انتخاب منبع

به منظور اطمینان از ترتیب وظایف در هنگام اجرای برنامه به-طوری که اولویت تقدم وظایف برآورده شود، همانند [۳۳]، مقدار رتبه برای ایجاد پیکربندی وظایف استفاده می‌شود. مقدار رتبه به صورت رابطه ۱۶ است:

$$Rank(t_i) = \overline{w_i} + max_{t_j \in succ(t_i)}\{C_{p,i}^{s,j} + Rank(t_i)\} \quad (۱۶)$$

وظیفه با مقدار رتبه بیشتر دارای اولویت بالاتری است. بنابراین، قبل از اجرای برنامه، وظایف را بدون افزایش رتبه‌ها مرتب می‌کنیم که می‌تواند اجرا را در جهت درست انجام دهد. مرحله اول تخصیص وظایف با حداقل مصرف انرژی می‌باشد.

الگوریتم ۱ مرتب سازی و انتخاب منبع را نشان می دهد. در این الگوریتم هر وظیفه به پردازنده با حداقل مصرف انرژی پویا اختصاص داده می‌شود که به‌طور کلی شامل حلقه های تودرتو است. حلقه بیرونی تمام وظایف را انتقال می‌دهد (خطوط ۳-۱۱) و حلقه درونی هر وظیفه را به پردازنده با حداقل مصرف انرژی پویا (خطوط ۵-۱۰) اختصاص می‌دهد. پیچیدگی زمانی الگوریتم ۱ به شرح زیر است: فرایند تخصیص تمام وظایف به پردازنده باید تمام وظایف را انجام دهد که در زمان $O(N)$ انجام می‌شود. محاسبه حداقل مصرف انرژی هر کار را می‌توان در زمان $O(N \times M)$ انجام داد. از این‌رو الگوریتم دارای پیچیدگی زمان $O(N^2 \times M)$ است که برابر با الگوریتم HEFT [۳۰] است.

الگوریتم۱: مرتب سازی و انتخاب منبع

Input: List of sorted task
P: a set of DVFS enabled Processor
Output: Schedule Tasks to Resource,R(G),E_{total}
1.sort the tasks to queue queue by non-increasing order of Rank value
2. r_list: put all available resources in the resource list
3. **For each** task ti according to task ordering
4. e=∞
5. **For each** resource rj in r_list
6. calculate E_{busy} (t_i)by EQ(6)
7. **If** (E_{busy} (t_i)<e)
8. e= E_{busy} (t_i)
9.
10. **End for**
11. **End for**
12. **For each** resource rj in each time slot
13. **If** rj is idle or communication **then**
14. Place the resource in low power state, at lowest voltage frequency level.
15. **End for**
16.Calculate R(G) by EQ(13)
17.Calculate E_{total} by EQ(8)

۴-۱-۲- افزایش قابلیت اطمینان

در این بخش، یک الگوریتم برای بهبود قابلیت اطمینان وظایف پیشنهاد شده است. برای انتخاب وظیفه مناسب برای بهبود قابلیت اطمینان، نسبت قابلیت اطمینان فعلی به عنوان تعریف ۱ مشخص می‌شود.

تعریف ۱: (نسبت قابلیت اطمینان فعلی). نسبت قابلیت اطمینان فعلی CRR[۱۰]، نسبت قابلیت اطمینان فعلی وظیفه به حداکثر قابلیت اطمینان روی وظیفه است:

$$CRR(t_i) = \frac{R_{cur}(t_i)}{R_{max}(t_i)} \qquad (۱۷)$$

در معادله (۱۷)، $\mathbf{R_{cur}(t_i)}$ نشان دهنده قابلیت اطمینان فعلی وظیفه ti است. بدیهی است، $\mathbf{CRR(t_i)}$ کمتر از ۱ یا برابر با ۱ است. $\mathbf{CRR(t_i) = 1}$ نشان می‌دهد که وظیفه $\mathbf{t_i}$ با حداکثر قابلیت اطمینان تعیین شده است. اگر مقادیر CRR تمام وظایف برابر با ۱باشد، قابلیت اطمینان برنامه به $\mathbf{R_{max}(G)}$ می‌رسد؛ در-غیر این صورت، قابلیت اطمینان برنامه کمتر از $\mathbf{R_{max}(G)}$ است. هنگامی‌که الگوریتم ۱ تکمیل می‌شود، ارزش اولیه CRR تمام وظایف را می‌توان به دست آورد. به‌عنوان مثال، مقدار اولیه CRR تمام وظایف در برنامه می‌تواند مانند جدول ۳ پس از تکمیل الگوریتم ۱ محاسبه شود. ما فرض می‌کنیم که پارامترهای قدرت و قابلیت اطمینان برای تمام پردازنده ها شناخته شده‌اند و در جدول ۲ نشان داده شده است، در حالی‌که حداکثر فرکانس f_j برای هر پردازنده ۱,۰ است. حداکثر قابلیت اطمینان نرم افزار در مثال ۰,۹۷۶۲۸۵۷ است که توسط رابطه ۱۵ بدست آمده است.

جدول ۲: مقدار اولیه پارامترهای قابلیت اطمینان و انرژی

r_j	$p_{dynamic}$	$\lambda_{j,max}$	d_j
r_1	0.003	0.0002	2.2
r_2	0.005	0.0004	2.0
r_3	0.007	0.0005	2.5

جدول ۳: مقدار اولیه CRR بعد از اجرای الگوریتم ۱

ti	proj	R(ti)	Rmax(ti)	CRR
t1	3	0.9955101	0.9972039	0.9983014
t3	3	0.9905450	0.9978024	0.9927266
t4	2	0.9968051	0.9974034	0.9994002
t2	3	0.9910404	0.9974034	0.9936204
t5	3	0.9950125	0.9976029	0.9974034
t6	3	0.9955101	0.9974034	0.9981018
t9	2	0.9952115	0.9964065	0.9988007
t7	3	0.9945151	0.9986010	0.9959084
t8	3	0.9930244	0.9990005	0.9940180
t10	3	0.9920319	0.9972039	0.9948135

همانطور که در جدول ۳ نشان داده شده است، ارزش CRR تمام وظایف کمتر از ۱ است که نشان می‌دهد قابلیت اطمینان این وظایف را می‌توان افزایش داد. در این مطالعه، الگوریتم IRT کار را با کوچکترین CRR برای بهبود قابلیت اطمینان انتخاب می‌کند. هنگامی که قابلیت اطمینان یک کار بهبود می‌یابد، قابلیت اطمینان برنامه افزایش می‌یابد. فرض بر این است که قابلیت اطمینان وظیفه $\boldsymbol{t_i}$ از $\boldsymbol{R_{cur}(t_i)}$ به $\boldsymbol{R_{new}(t_i)}$ افزایش یافته است، افزایش قابلیت اطمینان روی برنامه را می‌توان با استفاده از رابطه زیر به‌دست آورد:

$$R(G) = \frac{R_{cur}(G)}{R_{cur}(t_i)} \times R_{new}(t_i) \qquad (۱۸)$$

که $\mathbf{R_{cur}(G)}$ نشان دهنده قابلیت اطمینان فعلی برنامه قبل از افزایش است. بر اساس تجزیه و تحلیل فوق، الگوریتم IRT در الگوریتم ۲ با جزئیات گفته شده است. ایده‌های اصلی الگوریتم IRT به شرح زیر است. هنگامی که قابلیت اطمینان نرم افزار پایین تر از هدف قابلیت اطمینان است، IRT چندین مرتبه وظیفه را با کمترین مقدار CRR برای افزایش قابلیت اطمینان انتخاب می‌کند تا زمانی که هدف قابلیت اطمینان برنامه برآورده شود. گاهی اوقات هدف قابلیت اطمینان روی برنامه ممکن است کمتر از قابلیت اطمینان تولید شده توسط الگوریتم۱ باشد. در این حالت می‌توان فرکانس پردازنده را برای بهبود بهره وری انرژی کاهش داد.

الگوریتم۲: بهبود قابلیت اطمینان (IRT)

Input: R_{cur} , R(G), resource_list
Output: R(G),E_{total}
1. **For each** task t_i
2. calculate CRR(t_i) by EQ(17)
3. **End for**
4. **While** R(G)<R_{goal}(G)
5. **For each** $t_i \in T$
6. input task into t_s
7. **End for**
8.e= ∞
9. **For each** $r_j \in$ resource
10. calculate $R_{new}(t_s)$ by EQ(12)
11. calculate $E_{total}(t_s)$ by EQ(6)
12. **If** $R_{new}(t_s) > R_{cur}(t_s)$ and $E_{total}(t_s)$<e
13. **e**= $E_{total}(t_s)$
14. **End If**
15. **End for**
16.Update R(G) by EQ(18)
17.**End while**
18.Calculate E_{total}(G) by EQ(8)
19.**If** R(G)>R_{goal}(G)
20. **For each** $r_j \in$ resource_list
21.**For each** $t_i \in$ tasks
22.reduce $f_{j,k}$ of r_j from $f_{j,max}$ to $f_{j,low}$
23. calculate $R_{new}(t_i)$ by EQ(12)
24. calculate R(G) by EQ(18)
25. **If** R(G)<R_{goal}(G)
26.return j,k,i
27. **End If**
28.**End for**
29.**End for**
30.**End If**

در ادامه جزئیات ایده اصلی توضیح داده شده است. در خطوط ۱-۳ الگوریتم IRT مقدار CRR تمام وظایف را محاسبه می‌کند. ساختار الگوریتم خطوط ۴-۱۸ یک حلقه تو در تو است که در آن IRT تعیین می‌کند که آیا هدف قابلیت اطمینان از برنامه برآورده شده است یا خیر. هنگامی که هدف قابلیت اطمینان برآورده نمی‌شود، IRT یک کار را با کمترین مقدار CRR برای افزایش قابلیت اطمینان انتخاب می‌کند. در حلقه داخلی (خطوط ۵-۱۶)، IRT اولا وظیفه را با کوچکترین CRR برای افزایش قابلیت اطمینان (خطوط ۵-۷) انتخاب می‌کند و سپس IRT یک پردازنده را با قابلیت اطمینان بالاتر و حداقل مصرف انرژی برای اجرای وظیفه انتخاب می‌کند (خطوط ۹-۱۶). در خط ۱۹ الگوریتم IRT، $\mathbf{E}(\mathbf{G})$ را محاسبه می‌کند. خطوط ۱۹-۳۰ یک حلقه تو در تو است، در ادامه فرکانس اجرای پردازنده را به ترتیب پردازنده یک-به‌یک کاهش می‌دهد. در حلقه داخلی (خطوط ۲۱-۲۸)، فرکانس اجرای پردازنده یک سطح در هر بار کاهش می‌یابد. برای وظیفه t_i که به پردازنده r_j اختصاص داده شده مقادیر $\mathbf{R_{NEW}}()$ و $\mathbf{R}(\mathbf{G})$ محاسبه می‌شود. اگر $\mathbf{R}(\mathbf{G})$ بزرگتر از $\mathbf{R_{goal}}(\mathbf{G})$ باشد، وظیفه بعدی در پردازنده r_j بررسی می‌شود. در غیر این صورت، فرآیند پایان خواهد یافت. در الگوریتم IRT، حلقه while برای یک تکرار اجرا می شود. از آنجا که هر تکرار قابلیت اطمینان یک کار را افزایش می‌دهد، بنابراین قابلیت اطمینان برنامه می‌تواند به طور مداوم تا $\mathbf{R}(\mathbf{G}) > \mathbf{R_{goal}}(\mathbf{G})$ با استفاده از الگوریتم IRT تکراری افزایش یابد. پیچیدگی زمان الگوریتم IRT به شرح زیر است:

در طی هر تکرار، علامت‌گذاری وظایف با کوچکترین مقدار CRR دارای پیچیدگی زمان$O(N)$ (خطوط ۵-۷) است. در خطوط ۹-۱۶، الگوریتم IRT نیاز به عبور از تمام پردازنده‌ها برای پیدا کردن پردازنده مناسب با قابلیت اطمینان بالاتر و مصرف حداقل انرژی است، که پیچیدگی زمان $O(M \times PT_s)$.

در اینجا PTs نشان دهنده حداکثر تعداد وظایف سابق وظیفه TS است. برای یک نمودار متراکم زمانی که PT با N متناسب است، بدترین حالت پیچیدگی زمانی هر تکرار$O(M \times N)$ است. از آنجا که N وظیفه و M پردازنده وجود دارد، حداکثر تعداد تکرارها $M \times N \times F_s$ است ، که Fs حداکثر تعداد فرکانس های گسسته موجود پردازنده را نشان می‌دهد. برای حالت کاهش فرکانس نیز ابتدا وظایف را برای هر ترکیبی از پردازنده و فرکانس پیمایش می‌کند که دارای پیچیدگی زمان $O(M \times mdf \times N)$ است، در حالی‌که MDF نشان‌دهنده حداکثر تعداد فرکانس‌های گسسته پردازنده‌ها است. بنابراین بدترین حالت پیچیدگی زمان الگوریتم IRT، $O(M^2 \times N^2 \times F_s \times mdf)$ است.

در این قسمت برای شرح بهتر الگوریتم IRT یک مثال ارائه می‌شود: در این مثال، تمام پارامترهای جدول ۲ را دنبال می‌کنیم، هدف قابلیت اطمینان برنامه ۰,۹۶است که بیشتر از الگوریتم۱ (۰,۹۴۰۸۲۳۲۴) است. جدول ۴ نتایج نمونه با استفاده از IRT را پس از تکمیل الگوریتم ۱ نشان می‌دهد. "✓" پشت شماره سریال پردازنده نشان می‌دهد که وظیفه به پردازنده مربوطه مجددا اعطا می‌شود و شماره سریال پردازنده با "زیر خط" نشان می‌دهد که قابلیت اطمینان این کار به حداکثر رسانده شده است. همان‌طور که در جدول ۴ نشان داده شده است، قابلیت اطمینان وظایف t_3، t_2، t_8 و t_{10} افزایش می‌یابد و اطمینان از وظایف t_2، t_8 و t_{10} به حداکثر مربوطه رسیده است. در نهایت، قابلیت اطمینان واقعی برنامه $R(G) = 0.96165454$ است و کل مقدار مصرف انرژی $Etotal(G) = 162.327$ است که بیشتر از مقدار تولید شده توسط الگوریتم۱ است. فاصله زمانی بین زمان اجرای واقعی وظیفه و زمانی که به عنوان مهلت زمانی به آن وظیفه داده شده در واقع زمانی است که منبع بیکار است که به آن زمان اسلک گفته می‌شود و از طریق رابطه (۱۹) محاسبه می‌شود.

جدول ۴: نتایج نمونه با استفاده از IRT پس از تکمیل الگوریتم ۱

ti	R(ti)	proj	E_{total}
t1	0.9955101	3	7.26
t3	0.9948135	2✓	19.07
t4	0.9968051	2	12.54
t2	0.9974034	1✓	24.64
t5	0.9950125	3	8.07
t6	0.9955101	3	7.26
t9	0.9952115	2	26.56
t7	0.9945151	3	8.88
t8	0.9990005	1✓	27.02
t10	0.9972039	2✓	21.04

جدول ۵: زوج های متفاوت ولتاژ و سرعت مرتبط [۳۱]

level	Pair1		Pair2		Pair3	
	Supply voltage (V)	Relative speed (%)	Supply voltage (V)	Relative speed (%)	Supply voltage (V)	Relative speed (%)
0	1.5	100	1.50	100	1.48	100
1	1.4	90	1.35	85	1.32	80
2	1.3	80	1.2	70	1.15	60
3	1.2	70	1.05	55	0.95	40
4	1.1	60	0.9	40		
5	1.0	50				
6	0.9	40				

$$slack(t_i) = LFT(t_i) - EST(t_i) - ET(t_i) \quad (۱۹)$$

اگر زمان اسلک یا بیکاری برای وظیفه وجود داشت، زمان اجرای جدید آن وظیفه توسط رابطه (۲۰) متناسب با فرکانس عامل انتخاب شده از مقادیر جدول فرکانس تغییر می‌کند.

$$\acute{ET}(t_i) = \frac{ET(t_i) \times f_{j_{lowest}}}{f_{j,k}} \quad (۲۰)$$

پس از اینکه مشخص شد وظیفه‌ای دارای زمان بیکاری است زمان اجرای آن وظیفه برای کاهش فرکانس عامل توسط تکنیک DVFS به مقدار $\acute{E}T(t_i)$ تغییر پیدا می‌کند و وظیفه روی منبع با فرکانس پایین‌تری اجرا می‌شود و زمان بیکاری بلا استفاده توسط اجرای وظیفه پوشانده می‌شود. مزیت استفاده از این روش این است که زمان‌هایی که منبع بیکار است به طور بهینه استفاده می‌شود و همچنین زمانی که منبع در حالت ارتباطات می‌باشد می‌توان از این روش برای کاهش انرژی مصرفی استفاده کرد.

۵- ارزیابی

محاسبات ابری یک راه حل با توانایی بازدهی انرژی بیشتری نسبت به مدل‌های محاسبات قبلی است. فروشندگان مختلف فناوری اطلاعات و ارتباطات محاسبات ابری را به عنوان فن‌آوری ارجح‌تری برای غلبه بر بحران انرژی پیش‌بینی کرده‌اند[۳۴]. در این بخش تنظیمات آزمایش و معیارهای کمی ارزیابی شده ارائه می‌شوند و روش پیشنهادی در طول چند آزمایش مورد مطالعه قرار می‌گیرد.

۵-۱- تنظیمات آزمایش

آزمایشاتی که در این بخش ارائه شده، با استفاده از ابزار کاربردی شبیه ساز ابر [۱۱] توسعه داده شده است که ابزاری برای شبیه سازی و مدل کردن ساختارهای بر پایه ابر می‌باشد [۳۵]. مدل‌های گراف گردش‌کار به دو دسته علمی[۱۲] و تجاری[۱۳] تقسیم می‌شوند که مدل های گردش‌کار علمی اغلب برای تجسم محاسبات علمی پیچیده غیر متمرکز استفاده می‌شود و مدل گردش‌کار تجاری شامل مستند سازی فرایندهایی است که در طول فرایند معمول کسب و کار یک سازمان اتفاق می‌افتد[۳۶]. همچنین تعداد متفاوتی از گراف وظایف با تعداد متفاوتی گره در نظر گرفته شده است. برای تکمیل شبیه‌سازی مجموعه پردازنده همگن در نظر گرفته شده که هر پردازنده دارای فن‌آوری DVFS می‌باشد که می‌تواند در سطوح مختلف ولتاژ و فرکانس کار کند. به ازای هر پردازنده یک مجموعه v_j از سطوح ولتاژ در نظر گرفته می‌شود که بین سه سطح نشان داده شده در جدول ۲ با الهام از مرجع [۳۷]، توزیع می‌شود.

برای اجتناب از نتایج یکسان چندین مجموعه ورودی در نظر گرفته شده و هر مجموعه در واقع DAG های تولید شده با شاخص های مختلف می‌باشد. نتایج آزمایش میانگین مقادیر بدست آمده از ۱۰ بار اجرای آزمایش است. بعد از مشخص شدن سطح ولتاژ توسط پردازنده، سرعت/فرکانس مرتبط با آن نیز تعیین می‌شود. سرعت/فرکانس را در بازه $[f_{min}, f_{max}]$ در نظر گرفتیم که f_{min} برابر با ۴۰٪ از f_{max} است همان‌طور که در جدول ۲ آمده است. مصرف انرژی پردازنده در سطوح ولتاژ متفاوت نیز از رابطه (۱) محاسبه می‌شود. در این مطالعه مصرف انرژی در حالت ارتباطات در نظر گرفته نشده است[۳۱].

۵-۲- معیارهای ارزیابی روش پیشنهادی

معیارهای زیر برای مقایسه روش پیشنهادی ارائه شده با دیگر روش‌ها معرفی می‌شوند:

قابلیت اطمینان: الگوریتم پیشنهادی فرض می‌کند که وظایف اختصاص داده نشده حداکثر قابلیت اطمینان را دارند و هدف قابلیت اطمینان وظیفه t_i توسط فرمول زیر تعیین می‌شود [۱۸]:

$$R_{goal}(t_i) = \frac{R_{goal}(G)}{\prod_{x=1}^{i-1} R(t_x) \times \prod_{x=i+1}^{N} R_{max}(x_y)} \quad (۲۱)$$

که با حاصل تقسیم قابلیت اطمینان کل بر حاصل‌ضرب حداکثر قابلیت اطمینان و قابلیت اطمینان وظایف قبلی، قابلیت اطمینان وظیفه t_i محاسبه می‌شود.

سودمندی: سودمندی در واقع مقدار نرخ MIPS اختصاص داده شده به پردازنده به مقدار MIPS که توسط پردازنده ارائه شده در بازه زمانی Δt است.

$$U_g(\Delta t) = \sum_{i=1}^{Resource_{NUM}} Resource\ i\ Total\ MIPS * Resource\ i\ Relative\ Speed * TimeInterval \quad (۱۲)$$

$$U_i(\Delta t) = \frac{\sum_{i=1}^{Num_{level}} \sum_{j=1}^{Request_{num_{level_i}}} Requested\ J\ MIPS}{U_g(\Delta t)} \quad (۱۳)$$

میانگین زمان اجرا: میانگین زمان شروع تا زمان اتمام زمانبندی می‌باشد.

میانگین مصرف انرژی: میانگین کل انرژی مصرفی منابع در دوره Δt تحت ولتاژ و فرکانس تعیین شده.

$$E_{total} = \frac{(totalEnergy - energySavingRatio)}{\text{totalEnergy}} \quad (۱۴)$$

نقض SLA: نقض SLA در واقع تفاوت بین مقدار زمان پاسخ مشخص شده توسط کاربر SLA (به‌عنوان مثال: مهلت زمانی) و زمان پاسخ واقعی (زمان پایان) برای درخواست c در دوره زمانی Δt است.

$$\text{SLAViolation}(\Delta t) = {}^{1}/_{c} \sum_{c=1}^{C} \text{SLAViolation}(\text{request}_c) \quad (۱۵)$$

$$\text{SLAViolation}(\text{request}_c) = \text{FT}(\text{request}_c) - \text{DL}(\text{request}_c) \quad (۱۶)$$

۵-۳- ارزیابی آزمایش‌ها

برای ارزیابی عملکرد الگوریتم پیشنهادی، ۴ معیار مهم برای ارزیابی در نظر گرفته شده است که عبارتند از: مصرف انرژی، میانگین زمان اجرا ، میانگین نقض SLA و قابلیت اطمینان وظایف. علاوه بر این، الگوریتم پیشنهادی در آزمایشات با الگوریتم‌های MRCRG [۱۷] و MaxRe [۱۶] مقایسه و ارزیابی می‌شود.

الگوریتم MaxRe ، تجزیه و تحلیل قابلیت اطمینان را به طرح تکرار فعال اضافه می‌کند و از یک تعداد پویا از تکثیر برای وظایف مختلف استفاده می‌کند.

الگوریتم MRCRG با هدف قابلیت اطمینان و به حداقل رساندن هزینه مصرف منابع، هدف قابلیت اطمینان روی برنامه را به هر وظیفه انتقال می‌دهد و هر وظیفه با پردازنده‌ای با حداقل هزینه مصرف منابع اجرا می‌شود.

با توجه به اینکه حذف گاوسی و تبدیل فوریه به طور گسترده‌ای به عنوان برنامه‌های کاربردی موازی مبتنی بر DAG برای ارزیابی الگوریتم استفاده می‌شوند [۱۴-۱۸] و همچنین حذف گاوسی و تبدیل فوریه به ترتیب دو برنامه کاربردی موازی با فاکتور موازی سازی پایین و بالا است. برای بررسی اثربخشی و اعتبار الگوریتم پیشنهاد شده، این دو نوع از برنامه‌های کاربردی موازی واقعی برای مقایسه نتایج الگوریتم‌ها استفاده شد. مقدمه‌ای کوتاه برای حذف گاوسی و تبدیل فوریه به شرح زیر است.

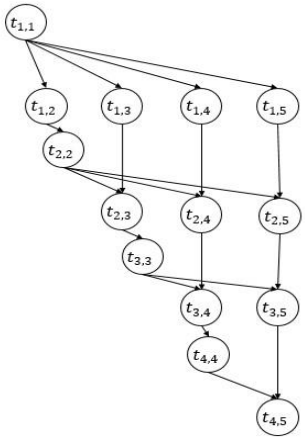
حذف گاوس: پارامتر s برای توصیف اندازه برنامه حذف گاوس استفاده می‌شود، تعداد کل وظایف در برنامه $N = \frac{s^2+s-2}{2}$ است. شکل ۳ یک برنامه حذف گاوسی را با $s = ۵$ نشان می‌دهد.

تبدیل فوریه: پارامتر s برای توصیف اندازه برنامه کاربردی فوریه استفاده می‌شود، تعداد کل وظایف در برنامه $N = (2 \times s - 1) + s \times \log_2 s$ است با $s = 2^y$ که y یک عدد صحیح مثبت است. شکل۴ یک برنامه تبدیل فوریه با $s = 4$ را نشان می‌دهد.

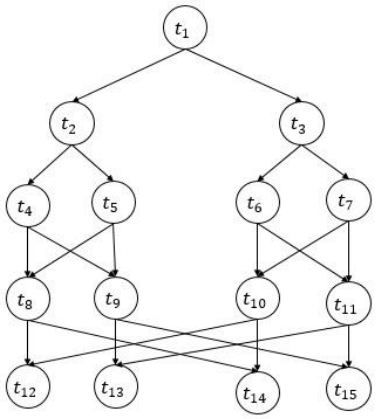
۵-۳-۱- مجموعه آزمایش۱ (ارزیابی در تعداد متفاوتی از وظایف، بررسی پارامترهای مصرف انرژی و قابلیت اطمینان)

در این آزمایش مصرف انرژی و قابلیت اطمینان واقعی برنامه‌های حذف گاوسی برای تعداد کل وظایف مختلف مقایسه شد. تعداد ۳۲ پردازنده با الهام از مرجع [۳۸] در این آزمایش استفاده شده و هدف قابلیت اطمینان نرم افزار با الهام از مرجع [۱۸] ۰,۹۵ است. مقدار S از ۱۶به ۴۰ با توان ۴ افزایش می‌یابد (به‌عنوان مثال، تعداد کل وظایف ۱۳۵، ۲۰۹، ۲۹۹، ۴۰۵، ۵۲۷، ۶۶۵، و ۸۱۹).

مصرف انرژی و قابلیت اطمینان واقعی با الگوریتم‌های مختلف در شکل ۵ نشان داده شده است.



شکل ۳: نمودار حذف گاوسی با مقدار s=5



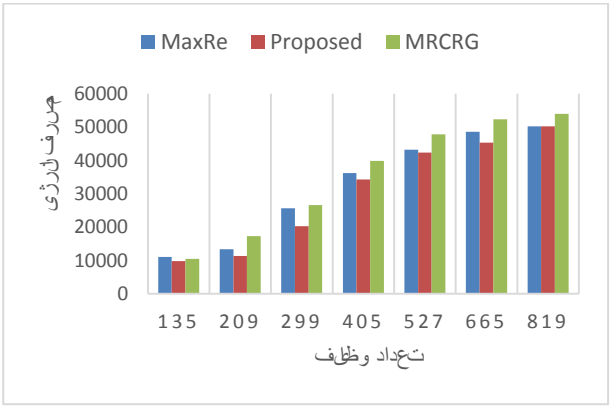
شکل ۴: نمودار تبدیل فوریه با مقدار s=4

شکل ۵ (الف) مصرف انرژی الگوریتم‌های مورد مقایسه را نشان می‌دهد. با افزایش تعداد کل وظایف، مصرف انرژی از سه الگوریتم افزایش می‌یابد. به طور کلی، MRCRG انرژی بیشتری مصرف می‌کند و سپس MaxRe، ولی الگوریتم پیشنهادی کمترین مصرف انرژی را دارد.

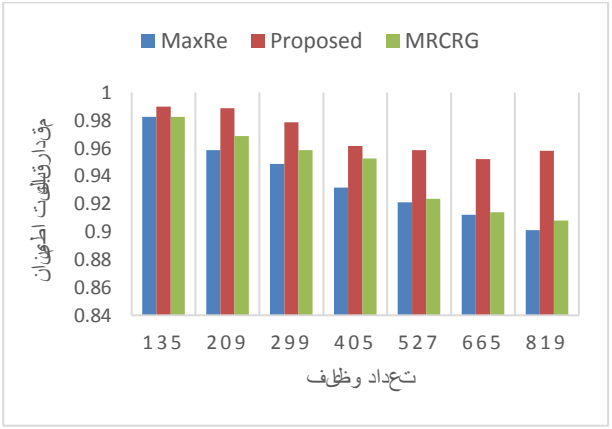
شکل ۵ (ب) قابلیت اطمینان برنامه را با الگوریتم‌های مورد مقایسه نشان می‌دهد. قابلیت اطمینان به دست آمده از MRCRG و MaxRe تقریبا یکسان است ولی قابلیت اطمینان الگوریتم پیشنهادی بالاتر از قابل اطمینان هدف است اما با توجه به زمانی- که تعداد کل وظایف بیشتر از ۸۱۹ یا برابر است، MRCRG و MaxRe نمی‌توانند قابلیت اطمینان برنامه را به ۰,۹۵ برسانند. دلایل اصلی نتایج فوق می‌تواند به شرح زیر توضیح داده شود.

در الگوریتم پیشنهادی، وظیفه به یک پردازنده با کمترین مصرف انرژی در مرحله اول با استفاده از الگوریتم ۱ اختصاص می‌یابد. هنگامی که هدف قابلیت اطمینان از برنامه بالاتر از قابلیت اطمینان تولید شده توسط الگوریتم ۱ است، الگوریتم پیشنهادی

فرکانس اجرای هر کار را کاهش نمی‌دهد. از آنجا که الگوریتم ۱ وظایف را با کمترین مقدار CRR برای افزایش قابلیت اطمینان انتخاب می‌کند، بنابراین، وظایف به پردازنده با بهره‌وری انرژی بیشتر به راحتی ترتیب داده می‌شود. هنگامی که هدف قابلیت اطمینان برنامه پایین تر از قابلیت اطمینان تولید شده توسط الگوریتم ۱ باشد، الگوریتم IRT فرکانس اجرای بسیاری از وظایف را کاهش می‌دهد، وظایف دیگر نیز به پردازنده‌های با مصرف انرژی کم اختصاص می‌یابد. بنابراین الگوریتم پیشنهادی انرژی کمتر از سایر الگوریتم‌ها تولید می‌کند.



الف



ب

شکل ۵: مجموعه آزمایش ۱ نتایج مقایسه الگوریتم پیشنهادی در تعداد متفاوتی از وظایف(برنامه های کاربردی تبدیل گاوسی)

وقتی روش DVFS در الگوریتم‌ها استفاده می‌شود، به منظور بهبود کارایی انرژی، فرکانس اجرای کار تا حد ممکن کاهش می‌یابد. در نتیجه، قابلیت اطمینان واقعی کارهای تعیین شده، کمی بالاتر از قابل اطمینان هدف خواهد بود. در مقابل، هنگامی که الگوریتم بدون استفاده از روش DVFS باشد، قابلیت اطمینان واقعی از وظایف اختصاص داده شده می‌تواند بسیار بالاتر از قابلیت اطمینان هدف باشد. با این حال، قابلیت اطمینان واقعی الگوریتم پیشنهادی کمی بالاتر از قابلیت اطمینان هدف است.

۵-۳-۲- مجموعه آزمایش۲ (ارزیابی در تعداد متفاوتی از پردازنده، بررسی پارامترهای مصرف انرژی ، میانگین سودمندی منابع و میانگین زمان اجرا)

این آزمایش مصرف انرژی و قابلیت اطمینان واقعی برنامه‌های کاربردی تبدیل فوریه را برای تعداد کل کارهای مختلف مقایسه می‌کند. ۶۴ پردازنده در این آزمایش مورد استفاده قرار می‌گیرد، قابلیت اطمینان هدف برنامه ۰,۹۵ است، مقادیر S برابر با ۸، ۱۶، ۳۲، ۶۴ و ۱۲۸ است (یعنی تعداد کل کارها ۳۹، ۹۵، ۲۲۳، ۵۱۱ و ۱۱۵۱ است)، مصرف انرژی و قابلیت اطمینان واقعی الگوریتم‌ها در شکل ۶ نشان داده شده است. همان‌طور که در شکل۶ (الف) نشان داده شده است، وقتی که تعداد کل وظایف افزایش می‌یابد، مصرف انرژی سه الگوریتم نیز افزایش می‌یابد.

همان‌طور که در شکل ۶ (ب) نشان داده شده است، در تعداد ۳۹ وظیفه اطمینان حاصل شده توسط MRCRG ، MaxRe و الگوریتم پیشنهادی کمی بالاتر از قابلیت اطمینان هدف است. زمانی‌که تعداد وظایف کل بیش از ۱۱۵۱ یا برابر است، با استفاده از MRCRG، MaxRe قابلیت اطمینان برنامه را نمی‌توان به ۰,۹۵ رساند. نتیجه آزمایش ۲ همانند آزمایش ۱ است، الگوریتم پیشنهادی همیشه می‌تواند برنامه را به هدف قابلیت اطمینان برساند و حداقل مصرف انرژی را تولید کند. شکل ۶(ج) متوسط زمان اجرای سه الگوریتم را نشان می‌دهد. میانگین زمان اجرا از تقسیم زمان کل اجرا بر تعداد وظایف محاسبه می‌شود.

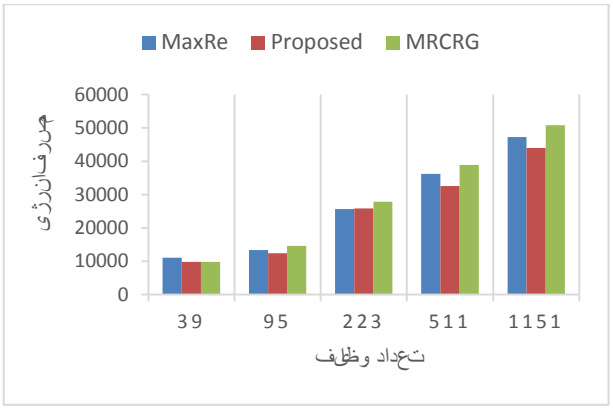
با افزایش تعداد وظایف، متوسط زمان اجرای سه الگوریتم افزایش می‌یابد. این به این دلیل است که با افزایش تعداد وظایف، مقیاس الگوی تصادفی DAG نیز افزایش می‌یابد. همان‌طور که در شکل۶(ج) مشخص است الگوریتم پیشنهادی زمان اجرای بیشتری نسبت به الگوریتم‌های دیگر دارد که این به این دلیل است که در روش پیشنهادی تمرکز بیشتر بر روی مصرف انرژی و قابلیت اطمینان است و زمان اجرا از اولویت پایین‌تری برخوردار است.

۵-۳-۳- مجموعه آزمایش ۳ (ارزیابی در مقادیر متفاوتی از پردازنده ، بررسی پارامتر زمان اجرا و مصرف انرژی)

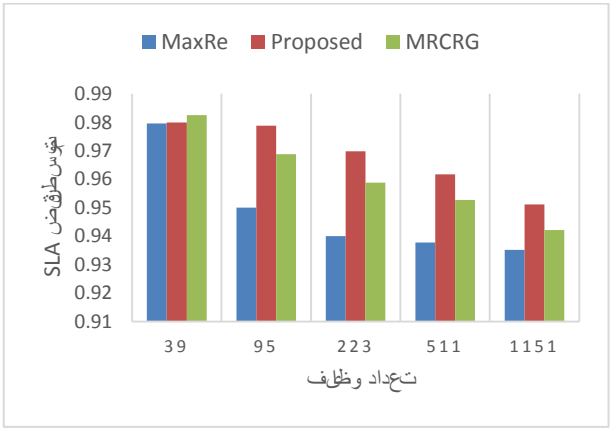
در سومین سناریو از آزمایشات الگوریتم‌های مورد مقایسه، الگوریتم‌ها در تعداد متفاوتی از پردازنده مقایسه می‌شوند.

شکل ۷ مقایسه عملکرد سه الگوریتم در شرایط مختلف پردازنده ها را نشان می‌دهد. در حالی‌که گراف تصادفی با ۲۰۰ وظیفه است و تعداد پردازنده‌ها با مقادیر ۲، ۴، ۸، ۱۶، ۳۲ و ۶۴ با الهام از مرجع [۳۸] متغیر می‌باشد. مشابه آزمایشات پیشین روش

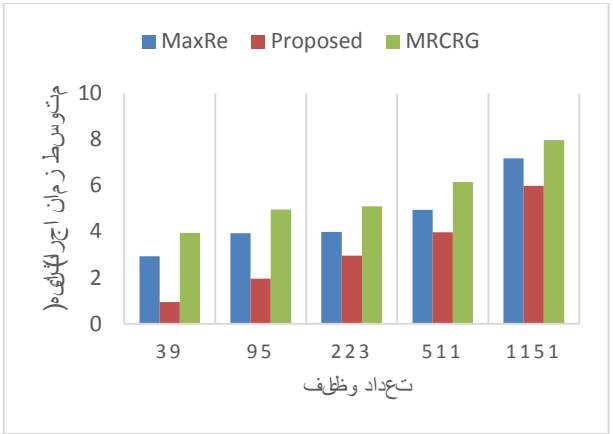
پیشنهادی با افزایش تعداد پردازنده‌ها نیز بهتر از دو الگوریتم دیگر انجام می‌شود.



الف



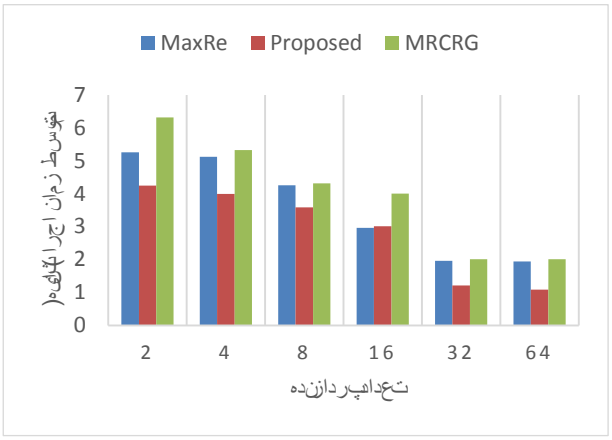
ب



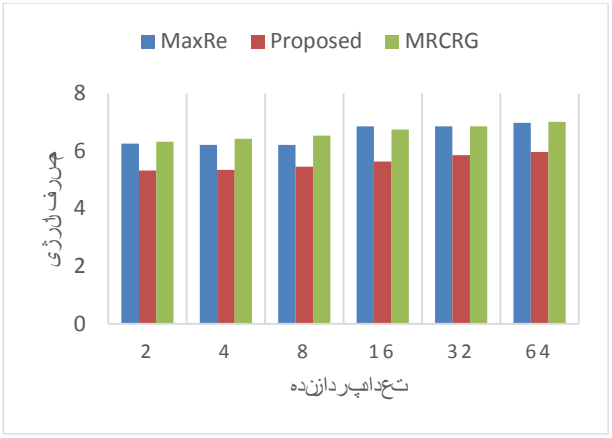
ج

شکل ۶: مجموعه آزمایش ۲ نتایج مقایسه الگوریتم پیشنهادی در تعداد متفاوتی از وظایف(برنامه های کاربردی تبدیل فوریه)

در شکل ۷ (الف) تفاوت میانگین زمان اجرا در میان سه الگوریتم، به تدریج کاهش می‌یابد. توجه داشته باشید که تعداد ۸ پردازنده در نمودار ارائه شده مهم است. هنگامی که تعداد پردازنده ها بزرگتر از ۸ است، عملکرد متوسط زمان اجرا از افزایش تعداد پردازنده‌ها به خوبی بهره می‌برد. به دلیل این واقعیت است که مقیاس گراف به اندازه کافی بزرگ می‌باشد. در شکل ۷ (ب) در مقادیر متفاوت پردازنده، مصرف انرژی تغییر مشخصی ندارد که این به‌دلیل مدیریت انرژی در هر سه الگوریتم است.



الف



ب

شکل۷: مجموعه آزمایش ۳ نتایج مقایسه الگوریتم پیشنهادی در تعداد متفاوت پردازنده

۶- نتیجه‌گیری و کارهای آینده

در سال‌های اخیر، بهره‌وری انرژی به‌عنوان یکی از مهم‌ترین الزامات طراحی برای سیستم‌های محاسباتی مدرن، اعم از تک سرویس‌دهنده تا مراکز داده و ابرها پدید آمده که مقدار زیادی از انرژی برق را مصرف می‌کند. به غیر از هزینه‌های عملیاتی بالای منابع محاسباتی، منجر به تولید گازهای گلخانه‌ای قابل توجهی نیز در محیط زیست می‌شود. با این حال، با کاهش مصرف انرژی احتمال خطاهای گذرا در پردازنده که قابلیت اطمینان برنامه‌ها را تضعیف می‌کند وجود دارد. رویکردهای زیادی برای کاهش مصرف انرژی و کاهش نرخ خطا معرفی شده است تا مراکز داده را به یک طرح اقتصادی‌تر و سبزتر هدایت کند. مقیاس گذاری پویای وظایف (DVFS) یکی از روش‌های موثر در کاهش مصرف انرژی می‌باشد که به کمک کاهش

فرکانس عامل منابع به کاهش انرژی دست می‌یابد. ترکیب این روش با یک روش زمان‌بندی و کنترل قابلیت اطمینان می‌تواند تاثیر زیادی در کاهش انرژی مصرفی و افزایش قابلیت اطمینان داشته باشد. در این مقاله زمان‌بندی وظایف گردش‌کار را در محیط ابر نا همگن با امید به کاهش مصرف انرژی و کاهش نرخ خطا انجام می‌شود در حالی‌که محدودیت مهلت زمانی مشخص شده توسط SLA در نظر گرفته شود. برای دستیابی به این هدف ما الگوریتم کاهش مصرف انرژی با هدف قابلیت اطمینان در پردازنده‌های چندگانه در محیط محاسبات ابری را ارائه دادیم که وظایف را بین پردازنده‌های با مصرف انرژی پایین‌تر زمان‌بندی می‌کند به طوری‌که قابلیت اطمینان وظایف و در نتیجه قابلیت اطمینان برنامه به قابلیت اطمینان هدف نزدیک باشد. برای ارزیابی آزمایشات انجام شده از گراف تصادفی وظایف استفاده کردیم که وظایف با توزیع گوسی و انتقال فوریه تولید می‌شوند. نتایج آزمایش‌ها نشان می‌دهد که الگوریتم پیشنهادی قابلیت اطمینان برنامه را حفظ می‌کند و انرژی کمتری مصرف می‌کند و همچنین زمان اجرای روش پیشنهادی نیز نسبت به دو الگوریتم مورد مقایسه کمتر است.

مراجع

[1] J.Koomey, "Growth in data center electricity use 2005 to 2010", A report by Analytical Press, completed at the request of The New York Times, 2011.

[2] A.Greenberg, J.Hamilton, DA.Maltz, P.Patel, "The cost of a cloud: research problems in data center networks", ACM SIGCOMM computer communication review, vol. 39, no. 1, pp. 68-73, 2008.

[3] M. Lin, Y.Pan, LT.Yang, M. Guo, N.Zheng, "Scheduling co-design for reliability and energy in cyber-physical systems", IEEE Transactions on Emerging Topics in Computing, vol. 1, no. 2, pp. 353-65, 2013.

[4] L Benini, A.Bogliolo, G. De Micheli, "A survey of design techniques for system-level dynamic power management", IEEE transactions on very large scale integration (VLSI) systems, vol. 8, no. 3, pp. 299-316, 2000.

[5] EJ. Hogbin. "ACPI: Advanced Configuration and Power Interface", Phoenix Usa, pp. 1–24, 2004.

[6] V. Venkatachalam, M. Franz, "Power reduction techniques for microprocessor systems", ACM Computing Surveys (CSUR), vol.37, no. 3, pp. 195-237, 2005.

[7] G. Xie, R. Li, K. Li, "Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems", Journal of Parallel and Distributed Computing, vol. 83, pp. 1-12, 2015.

[8] D. Zhu, H. Aydin, "Reliability-aware energy management for periodic real-time tasks", IEEE Transactions on Computers, vol. 58, no. 10, pp. 1382-1397, 2009.

[9] D. Zhu, "Reliability-aware dynamic energy management in dependable embedded real-time systems", In: 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), pp. 397-407, 2006.

[10] G. Xie, Y. Chen, X. Xiao, C. Xu, R. Li, K. Li, "Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems", IEEE Transactions on Sustainable Computing, vo-181, 2018.

[11] G. Xie, J. Jiang, Y. Liu, R. Li, K. Li, "Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems", IEEE Transactions on Industrial Informatics, vol. 13, no. 3, pp. 1068-1078, 2017.

[12] H. Xu, R. Li, L. Zeng, K. Li, C. Pan, "Energy-efficient scheduling with reliability guarantee in embedded real-time systems", Sustainable Computing: Informatics and Systems, vol. 18, pp. 137-148, 2018.

[13] L. Zhang, K. Li, C. Li, K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems", Information Sciences, vol. 379, pp. 241-256, 2017.

[14] D. Zhu, R. Melhem, D. Mossé, "The effects of energy management on reliability in real-time embedded systems", In Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design, pp. 35-40, 2004.

[15] L. Zhang, K. Li, K. Li, Y. Xu, "Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems", International Journal of Electrical Power & Energy Systems, vol. 78, pp. 499-512, 2016.

[16] L. Zhao, Y. Ren, Y. Xiang, K. Sakurai, "Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems", In IEEE 12th International Conference on High Performance Computing and Communications (HPCC), pp. 434-441, 2010.

[17] H. Topcuoglu, S. Hariri, MY. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", IEEE transactions on parallel and distributed systems, vol. 13, no. 3, pp. 260-274, 2002.

[18] G. Xie, Y. Chen, Y. Liu, Y. Wei, R. Li, K. Li, "Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems", IEEE Transactions on Industrial Informatics, vol. 13, no. 4, pp. 1628-1640, 2017.

[19] G. Xie, Y. Chen, X. Xiao, C. Xu, R. Li, K. Li, "Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems", IEEE Transactions on Sustainable Computing, vol. 3, no. 3, pp. 167-181, 2018.

[20] M. Fan, Q. Han, X. Yang, "Energy minimization for on-line real-time scheduling with reliability awareness", Journal of Systems and Software,vol. 127, pp. 168-176, 2017.

[21] L. Ismail, A. Fardoun, "Eats: Energy-aware tasks scheduling in cloud computing systems", Procedia Computer Science, vol. 83, pp. 870-877, 2016.

[22] MH. Kumar, SK. Peddoju, "Energy efficient task scheduling for parallel workflows in cloud environment", In 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 1298-1303, 2014.

[23] YK. Kwok, I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors", ACM Computing Surveys (CSUR), vol. 31, no. 4, pp. 406-471, 1999.

[24] GC. Sih, EA. Lee, "A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures", IEEE transactions on Parallel and Distributed systems, vol. 4, no. 2, pp. 175-187, 1993.

[25] R. Sakellariou, H. Zhao, E. Tsiakkouri, MD Dikaiakos, "Scheduling workflows with budget constraints", In Integrated research in GRID computing, pp. 189-202, 2007.

[26] L. Wang, SU. Khan, D. Chen, J. KołOdziej, R. Ranjan, CZ. Xu, A. Zomaya, "Energy-aware parallel task scheduling in a cluster", Future Generation Computer Systems, vol. 29, no. 7, pp. 1661-1670, 2013.

[27] L. Wang, G. Von Laszewski, J. Dayal, F. Wang, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS", In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 368-377, 2010.

[28] SK. Garg, CS. Yeo, A. Anandasivam, R. Buyya, "Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers", Journal of Parallel and Distributed Computing, vol. 71, no. 6, pp. 732-749, 2011.

[29] 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, Clouds and e-Science, p. 1, 2009.

[30] R. Garg, AK. Singh, "Energy-aware workflow scheduling in grid under QoS constraints", Arabian Journal for Science and Engineering, vol. 41, no. 2, pp. 495-511, 2016.

[31] R. Khorsand, M. Ghobaei-Arani, M. Ramezanpour, "FAHP approach for autonomic resource provisioning of multitier applications in cloud computing environments", Software: Practice and Experience, vol. 48, no. 12, pp. 2147-2173, 2018.

[32] R. Khorsand, F. Safi-Esfahani, N. Nematbakhsh, M. Mohsenzade, "ATSDS: adaptive two-stage deadline-constrained workflow scheduling considering run-time circumstances in cloud computing environments", The Journal of Supercomputing, vol. 73, no. 6, pp. 2430-2455, 2017.

[33] R. Khorsand, F. Safi-Esfahani, N. Nematbakhsh, M. Mohsenzade, "Taxonomy of workflow partitioning problems and methods in distributed environments", Journal of Systems and Software, vol. 132, pp. 253-271, 2017.

[34] T. Kaur, I. Chana, "Energy efficiency techniques in cloud computing: A survey and taxonomy", ACM computing surveys (CSUR), vol. 48, no. 2, pp. 22-46, 2015.

[35] YH. Wang, IC. Wu, "Achieving high and consistent rendering performance of Java AWT/Swing on multiple platforms", Software: Practice and Experience, vol. 39, no. 7, pp. 701-736, 2009.

[36] M. Ghobaei-Arani, S. Jabbehdari, MA. Pourmina, "An autonomic approach for resource provisioning of cloud services", Cluster Computing, vol. 19, no. 3, pp. 1017-1036, 2016.

[37] V. Venkatachalam, M. Franz, "Power reduction techniques for microprocessor systems", ACM Computing Surveys (CSUR), vol. 37, no. 3, pp. 195-237, 2005.

[38] Z. Tang, L. Qi, Z. Cheng, K. Li, SU. Khan, K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment", Journal of Grid Computing, vol. 14, no. 1, pp. 55-74, 2016.

زیرنویس‌ها:

[1] Service Level Agreement
[2] Dynamic Power Management
[3] Dynamic Voltage and Frequency Scaling
[4] Dynamic Component Deactivation
[5] Dynamic Performance Scaling
[6] Dynamic Voltage Scaling
[7] Directed acyclic graph
[8] Earliest Start Time
[9] Latest Finish Time
[10] Current Reliability Ratio
[11] cloudsim
[12] Scientific
[13] Business