Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 10, Number 2, Summer 2021, pp. 100-115
Received: 03/13/2020, Revised: 03/29/2021, Accepted: 06/14/2021

Reactive Fault-tolerant Scheduling in a Fog-based application

Ahmad Sharif[1], Mohsen Nickray[2*] and Ali Shahidinejad[3]
1- Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran.
2*- Department of Computer Engineering and Information Technology, University of Qom, Qom, Iran.
3- Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran.
[1] a.sharif@qom-iau.ac.ir, [2*] m.nickray@qom.ac.ir, and [3] a.shahidinejad@qom-iau.ac.ir

Corresponding author's address: Mohsen Nickray, Department of Computer Engineering and Information Technology, University of Qom, Qom, Iran.

Abstract- Fog environment is growing as a vital platform for IoT. With the growing scale of IoT, network failures become inevitable. Communication reliability should be considered to achieve high performance. Fault tolerance becomes a vital issue for improving the reliability of fog. Most studies in fault tolerance have been carried out only on the cloud system. To address this issue, we propose a novel fault-tolerant scheduling algorithm for hybrid modules in fog. One of the main contributions of this approach is proposing a CRBC model, which composes the profits of Checkpoint-Restart and Primary Back up model with Classification. Besides, a novel classification method for different modules is another originality of this paper. We evaluate the performance of the proposed method by comparing it with three methods in terms of delay, energy consumption, execution cost, network usage, and total executed modules. Analysis and simulation results show the reliability and effectiveness of CRBC.

Keywords- Fog Computing, Checkpoint-Restart, Primary-Backup, Fault-tolerant scheduling.

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۱۰، شماره ۲، تابستان ۱۴۰۰، صفحه ۱۰۰-۱۱۵
دریافت: ۱۳۹۸/۱۲/۲۳، بازنگری: ۱۴۰۰/۰۱/۰۹، پذیرش: ۱۴۰۰/۰۳/۲۴

مدل زمانبندی مقاوم در برابر اشکال در کاربرد مبتنی بر مه

احمد شریف[۱]، محسن نیک رای*[۲]، علی شهیدی نژاد[۳]
۱- دانشکده فنی مهندسی، گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی‌واحد قم، قم، ایران.
*۲- دانشکده مهندسی کامپیوتر و فن آوری اطلاعات، دانشگاه قم، قم، ایران.
۳- دانشکده فنی مهندسی، گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی‌واحد قم، قم، ایران.
[1] a.sharif@qom-iau.ac.ir, [2*] m.nickray@qom.ac.ir, [3] a.shahidinejad@qom-iau.ac.ir

* نشانی نویسنده مسئول: محسن نیک رای، قم، بلوار غدیر، دانشگاه قم، دانشکده فنی و مهندسی.

چکیده- محیط مه به عنوان یک بستر مهم برای IoT در حال رشد است. با افزایش مقیاس IoT، خرابی شبکه اجتناب ناپذیر می شود. برای دستیابی به کارایی بالا باید به قابلیت اطمینان در ارتباطات توجه نمود. تحمل پذیری اشکال به یک مسئله مهم برای بهبود قابلیت اطمینان در محیط مه تبدیل شده است. بیشتر مطالعاتی که در مورد تحمل اشکال بوده، فقط در سیستم ابر صورت گرفته است. برای پرداختن به این موضوع در محیط مه، ما الگوریتم زمانبندی تحمل اشکال برای ماژول‌های ترکیبی در مه را پیشنهاد می‌کنیم. یکی از برجستگیهای این رویکرد، ارائه مدل CRBC[1] در کنار روش طبقه بندی است که تلفیقی از مزایای Checkpoint-Restart و Primary-Backup است. علاوه بر این، ارائه یک روش طبقه بندی برای ماژول های مختلف، نوآوری دیگر این مقاله است. در این مقاله عملکرد روش پیشنهادی را با مقایسه آن با سه روش دیگر از نظر تأخیر، مصرف انرژی، هزینه اجرا، میزان استفاده از شبکه و تعداد کل ماژولهای اجرا شده ارزیابی می‌کنیم. نتایج تجزیه و تحلیل و شبیه سازی، قابلیت اطمینان و اثربخشی CRBC را نشان می‌دهد.

واژه‌های کلیدی: محاسبات مبتنی بر مه، زمانبندی مقاوم در برابر اشکال، Checkpoint-Restart، Primary-Backup.

۱- مقدمه

محاسبات مه[۲]، محیط محاسباتی توزیع شده ای است که باعث گسترش محاسبات ابری تا لبه است و همینطور باعث تقویت IoT است [۱]. محاسبات مه، مدل محاسبات جدیدی است که در آن باعث می‌شود گره‌ها را به جایی که داده تولید می‌شود نزدیکتر کند تا محاسبات و پردازشها آنجا صورت گیرد. می‌توان بیان داشت مهمترین مزیت رایانش مه‌ای در مقایسه با رایانش ابری، نزدیکی آن به دستگاه های نهایی مانند سنسورها، محرک ها، دوربین های هوشمند، تلفن های هوشمند و دستگاه های IoT است [۲]. مدیریت منابع موجود در محاسبات مه، پیچیده است. تعداد زیادی از گره‌های مه با در نظر گرفتن محدودیت منابع برای پاسخگویی به تقاضای محاسباتی سیستم با قابلیت IoT به صورت توزیع شده رفتار می‌کنند [۳]. در کنار تمامی‌مزایایی که محاسبات مه‌ای دارد، با اختلالاتی نیز روبرو است؛ همچون اشکال و خرابی، که در ذات سیستمهای توزیع شده است. بنابراین استفاده از تکنیک های مقاوم در برابر اشکال[۳] ضروری است. علاوه بر این باید توجه داشت که خطا و اشکال در سیستمهای توزیع شده اجتناب ناپذیر است، به گونه‌ای که ۸٪ از ماشینهای مجازی در زمان اجرا با خطا روبرو می‌شوند [۴].

انواع مختلف اشکالات در سیستمهای مبتنی بر مه ممکن است رخ دهد، که باید با استراتژی زمانبندی برطرف گردد [۵]. استراتژی های زمانبندی وظیفه و منابع به سه دسته ایستا، پویا و ترکیبی طبقه بندی می‌شوند. در زمانبندی ایستا، وظایف به طور هم زمان به گره های مه می‌روند و روش های زمانبندی قبل از ارسال

وظایف انجام می‌شود. در حالی که در زمانبندی پویا، زمان ورود کارها نامشخص است و تنها هنگام ورود به سیستم وظایف زمانبندی می‌شوند. علاوه بر این، روش ترکیبی از زمانبندی ایستا و پویا استفاده می‌کند. زمانبندی وظیفه ترکیبی نیاز به تحمل پذیری اشکال بالایی دارد [۶].

جهت پیاده سازی الگوریتم های زمانبندی و مدیریت منابع در اینترنت اشیاء، رایانش مه بسیار موثرتر از رایانش ابری عمل می کند. مواردی همچون انرژی مصرفی، تاخیر زمانی، مصرف منابع شبکه، تاخیر زمانی در رایانش مه بسیار بهتر و راحت تر از رایانش ابری –ولو ابر کوچک- صورت می‌گیرد. پردازش های سنگین و زیاد به جای انتقال به مراکز داده ابری در ابزارهای مه نزدیک به محل کاربران، انجام می شود و در نتیجه آن مصرف منابع شبکه نیز بهینه خواهد شد. در تحقیقات قبلی، موضوع زمانبندی تحمل اشکال در محاسبات مه به طور واضح مورد توجه قرار نگرفته است. در این مقاله، ما مدل خلاقانه‌ای را برای زمانبندی تحمل اشکال برای وظایف گوناگون در محیط مه ارائه کرده‌ایم. اهداف این مقاله عبارتند از: ۱) معرفی یک معماری جدید در یک محیط سلسله مراتبی که مقاوم در برابر اشکال است. ۲) طراحی مدل CRBC که ترکیبی است از مزایای مدل checkpointing و primary backup در کنار دسته‌بندی گره‌های مه و ماژول‌ها. ۳) علاوه بر این، پیشنهاد یک روش طبقه بندی جدید برای ماژول‌های مختلف در زمان اجرا و گره‌های مه، که می‌تواند ماژول‌ها و گره‌های مه را در گروه‌های مختلف طبقه بندی کند. این روش تضمین می‌کند که می‌توان انواع مختلفی از ماژول‌ها را به دستگاه‌های مناسب اختصاص دهد. نتایج تجزیه و تحلیل و شبیه سازی، قابلیت اطمینان و اثربخشی CRBC را اثبات می‌کند. از این رو می‌توان به این نتیجه رسید که این مدل پیشنهادی، تحمل اشکال در محیط مبتنی بر مه و استفاده بهینه از منابع و کاهش زمان پاسخ را تضمین می‌کند. ما عملکرد روش پیشنهادی را با بررسی آن و مقایسه با سه روش، بر پایه تأخیر، مصرف انرژی، هزینه اجرا، میزان استفاده از شبکه و تعداد ماژولهای اجرا شده ارزیابی می-کنیم.

مقاله در ادامه به شرح زیر سازماندهی شده است: مفاهیم پایه‌ای از رویکردهای تحمل اشکال در محاسبات مه، در بخش ۲ آورده شده است. بخش ۳ مروری بر کارهای مرتبط است. بخش ۴ مدل پیشنهادی CRBC، مدل کاربردی و مزایای آن مورد بحث قرار می‌گیرد. همچنین الگوریتم پیشنهادی در این بخش شرح داده شده است. بخش ۵ ارزیابی عملکرد CRBC را تشریح می‌کند. درآخر، بخش ۶ نتیجه گیری را ارائه می‌دهد و به کارهای آینده می‌پردازد.

۲- مفاهیم

خطاها یا اشکالات به نمای کلی[۴] و نمای پردازنده[۵] طبقه بندی می شوند. خطاهای نمای کلی خود سه نوع است: گذرا[۶]، متناوب[۷] و دائمی[۸]. خطاهای گذرا تنها به اجرای وظیفه‌ی در حال اجرا آسیب می‌رساند، و پس از شروع دوباره سیستم یا اجرای مجدد وظیفه، آن خطا مرتفع می‌گردد. خطاهای متناوب در فواصل زمانی اتفاق می‌افتد و بالاخره خطاهای دائمی‌خطاهایی هستند که اثرات آنها قابل جبران نیست. از چشم‌انداز پردازنده، خطاها به سه دسته تقسیم می‌شوند: خرابی کامل[۹]، خرابی-توقف[۱۰]، و بیزانس[۱۱]. عمدتاً این سه دسته برای خرابی منابع یا دستگاهها مورد استفاده قرار می‌گیرند. در مدل خرابی کامل، پردازنده به طور غیر منتظره‌ای در یک نقطه خاص اجرایش متوقف می‌شود. در مدل خرابی-توقف، محتویات داده ها هنگام رخداد خطا از دست می‌رود و بازیابی نمی‌شود. خطاهای بیزانس هم به دلیل شکست غیرمنتظره مانند پیری یا اشکال در زیرساخت ها ظاهر می‌شوند. این دست خطاها را می‌توان در هر پردازنده یا پیام مشاهده کرد.

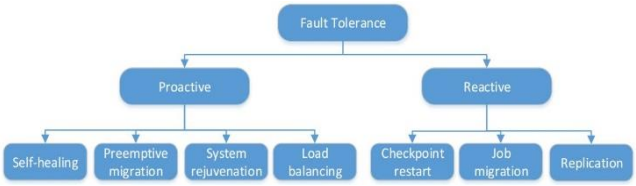
خطا در یک محیط توزیع شده می‌تواند در موارد مختلف رخ دهد: سخت افزار، سیستم عامل، واسط کار، وظیفه، جریان کاری و کاربر. نمونه‌هایی از ایرادات قابل توجهی که در سیستم رخ می‌دهد مربوط است به خرابی شبکه، خرابی دستگاه، کمبود حافظه[۱۲]، مشکلات تصدیق، خطاهای مرحله بندی پرونده[۱۳]، استثنائات غیرقابل برداشت[۱۴]، مشکلات مربوط به حرکت داده‌ها و استثنائات تعریف شده توسط کاربر [۶].

رویکردهای تحمل خطا به دو بخش کنش گرا[۱۵] و واکنشی[۱۶] تقسیم می‌شوند که در شکل ۱ نشان داده شده است. بر اساس این رویکردها، روش‌های مختلفی برای دستیابی به تحمل پذیری اشکال اتخاذ شده است.

رویکردهای واکنشی تحمل اشکال، خطاها را پس از وقوع آنها کنترل می‌کند. کارکرد رویکردهای واکنشی پاسخ محور است نه بر اساس پیش بینی. چک‌پوینتینگ و تکثیر[۱۷] دو رویکرد مهم تحمل خطا در زمانبندی توزیع شده است [۷]. رویکردهای واکنشی معمولاً محافظه کارانه است و نیازی به بررسی رفتار سیستم در آن نیست. از این رو، هیچ گونه سربار اضافی به وجود نمی‌آورد.

کنش گرا به این شکل تعریف می‌شود که توانایی سیستم در وضعیت آماده یا کنترل شده برای دستیابی به وقفه های احتمالی

(اشکال، خطا، خرابی) پیش از آنکه رخ دهد، می باشد. حالت سیستم در رویکردهای کنش گرایانه به طور مداوم مورد بررسی قرار می‌گیرد و وقوع خطا با استفاده از تکنیک های هوش مصنوعی تخمین زده می‌شود. سپس اقدامات لازم جهت جلوگیری از بروز خطا انجام می‌شود.



شکل ۱: طبقه بندی رویکردهای تحمل اشکال در سیستم های توزیع

این رویکرد مبتنی بر تجربه و انتظار عمل می‌کند. سیستم های کنش گرایانه تحمل پذیر خطا تا زمانی که انتظارات با تجربه مطابقت داشته باشد بدون وقفه باقی می‌ماند [۶]. رویکردهای تحمل خطا در سیستم مبتنی بر مه نیز توسعه می‌یابد همانگونه که در انواع سیستمهای توزیع شده استفاده می‌گردد. موارد در جدول ۱ ذکر شده است.

جدول ۱: رویکرد و مکانیزم های تحمل پذیری اشکال

ردیف	تکنیک	رویکرد	توصیف
1	Self-Healing	کنش گرا	توانایی یک سیستم برای بازیابی مستقل خطاها با استفاده از اعمال متناوب روالهای خاص رفع نقص متشکل از وظایف نظارت.
2	Pre-emptive migration	کنش گرا	توانایی یک سیستم برای جابجایی محاسبات از گره های محاسبات مشکوک به روش کنشگرایانه.
3	System rejuvenation	کنش گرا	فرآیند گرفتن پشتیبان دوره‌ای از سیستم. پس از هر نسخه پشتیبان، سیستم پاک می‌شود و سپس نسخه پشتیبان مجدداً برقرار می‌شود، و یک حالت تازه سازی سیستم حاصل می‌شود.
4	Load Balancing	کنش گرا	برای تعادل بار حافظه و CPU هنگامی‌که بیش از حد باشد، استفاده می‌شود. بار بیش از حد CPU به برخی دیگر از CPU ها منتقل می‌شود تا از حداکثر حد خود فراتر نروند.
5	Checkpoint-restart	واکنشی	حالتهای اجرای وظیفه بطور دوره‌ای در این روش ذخیره می‌شوند. در صورت بروز هرگونه خرابی، کار به جای شروع مجدد از ابتدا ، از آخرین وضعیت ذخیره شده دوباره شروع می‌شود.
6	Job migration	واکنشی	در صورت خراب شدن منبعی، وظیفه به منبع مشابه دیگری منتقل می‌شود.
7	Replication	واکنشی	برای ایجاد چندین نسخه از کارها و ذخیره آن در مکانهای مختلف استفاده می‌شود.

۳- کارهای مرتبط

رایانش ابری حجم گسترده ای از سرورهای متصل، مراکز داده و تغییر مداوم برای ارائه خدمات به درخواست کاربر از طریق رابط front-end است. با رشد سریع استقرار IoT، پرداختن به موضوعات تحمل پذیری اشکال به موضوعی اساسی و حیاتی تبدیل می‌شود. اخیراً تعداد قابل توجهی از کارهای دیگران در مورد زمانبندی تحمل اشکال و مدیریت منابع در سیستم های توزیع شده منتشر شده‌است [۸، ۹، ۱۰، ۱۱]. خرابی یک وظیفه‌ی در حال اجرا اتفاقی نیست که یکبار رخ دهد بلکه در سیستم‌های توزیع شده در مقیاس بزرگ رخ دادی تکرار شونده است. عبدالحمید و همکاران طرح نویی برای زمانبندی تحمل پذیر در برابر اشکال در سیستم های ابری طراحی کرده‌اند. آنها از چک پوینتینگ و مهاجرت وظایف در برخورد با خرابی های مستقل استفاده کرده‌اند. در تحقیق خود ثابت کردند که طرح ارائه شده بهبود چشمگیر و عملکرد بهتری نسبت به کاهش خرابی وظایف دارد [۱۲]. هان و همکاران مدل چک پوینتینگ و نسخه اصلی– پشتیبان را با مکانیسم طبقه بندی در ابر ارائه کرده‌اند و سه مکانیسم از جمله مکانیزم هدایت وظیفه به جلو، دگرگونی و تبدیل زمان را ابداع نموده‌اند. اهداف آنها تأمین منابع بیشتر برای کارهای اصلی، بهبود استفاده از منابع و تضمین تحمل اشکال بوده‌است [۴].

تام و همکاران تحمل پذیری اشکال کنش‌گرا و واکنشی را ترکیب کرده و تحمل پذیری اشکال انطباقی را ایجاد کرده‌اند که منابع ابر را بر اساس بار ابر انتخاب می‌کند. هدف آنها ساختن معماریی بود که بتواند اشکالات را بعد و قبل از وقوع آنها تحمل کند، درست مثل آنچه چین و همکاران در مورد خطاهای بیزانس انجام دادند [۱۳]. همچنین راه حلی برای حفظ ترافیک داده و در دسترس بودن سیستم ارائه دادند [۱۴]. هایدر و همکاران همان کارها را برای شبکه های محاسباتی انجام دادند که با کمک تحمل پذیری اشکال به روش کنش گرایانه و با استفاده از موقعیت مکانی و در دسترس بودن، منابع معتبر را تشخیص می‌دهد [۱۵]. چندین زمانبندی تحمل خطا برای جریان کاری طراحی شده‌است. هدف آنها افزایش بهره وری زمانبندی و منابع ضمن تحمل خرابی هاست [۱۶، ۱۷، ۱۸، ۱۹، ۲۰]. فن و همکاران مدل سازی و تجزیه و تحلیل استراتژی تحمل خطا را برای زمانبندی دقیق وظایف مهلت دار در محاسبات ابری در نظر گرفته اند[۲۱]. مه، محاسبات نزدیک به دستگاه های نهایی را ارائه می‌دهد، و یکی از گزینه های گوناگونی است که در کارخانه ها می‌توان در نظر گرفت. بنابراین، زمانبندی در کارخانه های هوشمند و محیط مه، زمینه جدیدی است که در بسیاری از تحقیقات مانند [۲۲، ۲۳، ۲۴] مورد توجه قرار گرفته‌است. یین و همکاران روشی جدید برای تخصیص منابع و زمانبندی وظایف در تولید هوشمند ارائه دادند. مدل آنها با هدف بهبود استفاده از منابع و کاهش تأخیر وظایف انجام شده است [۲۵]. برخی کارهای دیگر با هدف تمرکز بر مشکلات پیچیده مصرف انرژی در کارخانه های هوشمند [۲۶]، و تخصیص منابع در زیرساخت های سلسله مراتبی ارائه شده است

[۲۷]. ZNG و همکاران، سیستم رایانش مه‌ای تعبیه شده توسط نرم افزار را مورد بررسی قرار داده است. نسخه وظیفه در سرور ذخیره سازی قرار می‌گیرد، در حالی که محاسبات بر روی یک دستگاه توکار یا یک سرور محاسباتی مدیریت می‌شود [۲۸].

یانگ و همکاران الگوریتم دیگری برای کاهش مصرف انرژی کل با به حداقل رساندن تأخیر سرویس در گره های مه، ابداع کردند [۲۹]. و بالاخره تحقیق [۳۰] روشی چند هدفی را برای زمانبندی بارهای قابل تقسیم در یک شبکه درخت چند سطحی ارائه داده است.

آنچه در این بررسی ها می‌توان یافت این است که اکثر مطالعات در زمینه زمانبندی تحمل خطا روی سیستم های ابری متمرکز شده اند. هیچ تحقیق واحدی وجود ندارد که به طور مجزا زمانبندی تحمل خطا در سیستم های مه را پوشش دهد. البته نویسندگان مقاله حاضر در مقاله ای دیگر به موضوع موازنه بار بر اساس روش یادگیری ماشین برای یافتن راه حل بهتر برای مسئله زمانبندی در محیط رایانش مه توجه کرده اند[۳۱].

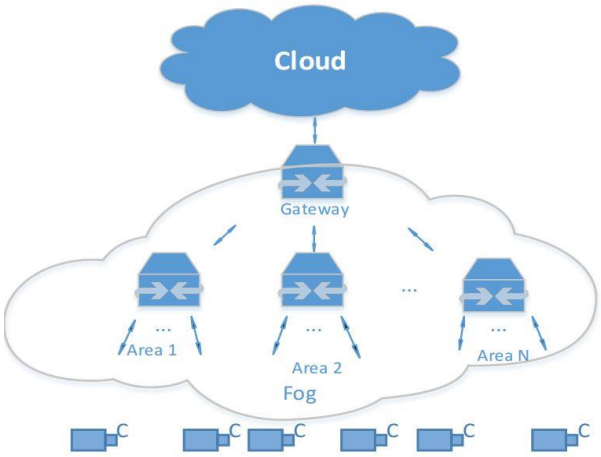
۴- مدل کاربردی و و مدل سیستمی

این بخش شامل دو مدل است از جمله مدل کاربرد و مدل سیستم. مدل سیستم ما بر روی یک مدل کاربردی اجرا می‌شود. بنابراین ابتدا یک مدل کاربردی را معرفی می‌کنیم و سپس مدل پیشنهادی را توصیف می‌کنیم.

۴-۱- مدل کاربردی

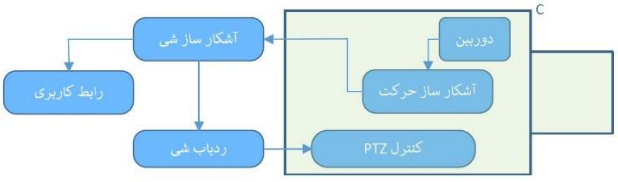
برای بیان اینکه چگونه زمانبندی تحمل خطا در رایانش مه ای می‌تواند باعث کاهش خرابی در سیستم های کاربردی گردد، مدل کاربردی نظارت تصویری / ردیابی شی[۱۸] (VSOT) را مد نظر قرار می‌دهیم. سرویس نظارت تصویری، که از طریق دوربین ها تصاویر زنده را دریافت کرده و آنرا برای کاربران نهایی ارسال می‌کند، به یکی از عمومی‌ترین خدمات تبدیل شده است [۳۲].

همانطور که در شکل ۲ نشان داده شده‌است، این‌کاربرد مدل به مجموعه ای از دوربین های توزیع شده وابسته است که می‌توانند حرکت یک شی را دنبال کنند. به منظور بررسی عملکرد مدل سیستم تحت معیارهایی مانند مصرف انرژی، هزینه اجرا و تأخیر، مقادیر دامنه پارامترها را تغییر می‌دهیم. یکی از این پارامترها تعداد نواحی و دوربینها است. ما چهار مجموعه طراحی و مدل پیشنهادی را بر روی هر مجموعه بطور جداگانه اجرا می‌کنیم. مجموعه ۱ دارای دو منطقه و شش دوربین در هر منطقه و مجموعه ۲ دارای سه منطقه و هفت دوربین است. مجموعه ۳ نیز دارای چهار منطقه و هشت دوربین و درنهایت مجموعه ۴ دارای پنج منطقه و ۹ دوربین است.



شکل۲ : مدل کاربرد سیستم های نظارتی

همانطور که در شکل ۲ نشان داده شده‌است، این‌کاربرد مدل به مجموعه ای از دوربین های توزیع شده وابسته است که می‌توانند حرکت یک شی را دنبال کنند. به منظور بررسی عملکرد مدل سیستم تحت معیارهایی مانند مصرف انرژی، هزینه اجرا و تأخیر، مقادیر دامنه پارامترها را تغییر می‌دهیم. یکی از این پارامترها تعداد نواحی و دوربینها است. ما چهار مجموعه طراحی و مدل پیشنهادی را بر روی هر مجموعه بطور جداگانه اجرا می‌کنیم. مجموعه ۱ دارای دو منطقه و شش دوربین در هر منطقه و مجموعه ۲ دارای سه منطقه و هفت دوربین است. مجموعه ۳ نیز دارای چهار منطقه و هشت دوربین و درنهایت مجموعه ۴ دارای پنج منطقه و ۹ دوربین است.



شکل۳ : ماژولهای مدل کاربرد

این کاربرد مدل دارای شش ماژول است: دوربین، آشکارساز حرکت، ردیاب شی، آشکارساز شی، رابط کاربری و کنترل PTZ[۱۹] همانطور که در شکل ۳ نشان داده شده است. این نوع دوربین قادر به کنترل از راه دور و کنترل زوم است. ویدیو توسط دوربین به ماژول ردیاب حرکت منتقل می‌شود. این ماژول جریان ورودی را فیلتر می‌کند و فیلم شناسایی شده حرکت را به ماژول آشکارساز شی منتقل می‌کند. در مرحله بعد، ماژول ردیاب شیء اشیاء

متحرک را تشخیص می‌دهد، سپس اطلاعات موقعیت و شناسایی شی را به ردیاب شی ارسال می‌کند، که PTZ برتر را محاسبه می‌کند و دستور را به ماژول کنترل PTZ می‌فرستد. نیاز به ذکر است که ردیاب حرکت و ماژول‌های کنترل PTZ همگی درون دوربین قرار می‌گیرند، در حالی که رابط کاربری همواره در ابر قرار دارد [۳۲].

۴-۲- مدل سیستم برای زمانبندی مقاوم در برابر اشکال

ما مدل خلاقانه‌ای برای زمانبندی مقاوم در برابر اشکال در محیط مه را پیشنهاد می‌کنیم. نوآوری‌های اصلی مدل پیشنهادی به شرح زیر است: ارائه یک روش طبقه بندی جدید برای گره های مه‌ای و ماژول های متنوع زمان اجرا، که با این روش می‌توان ماژول ها و دستگاه‌های مه‌ای را به مجموعه‌های مختلف طبقه بندی کند. این شکل طبقه بندی این اطمینان را می‌دهد که انواع مختلفی از ماژول‌ها به دستگاه‌های مناسب اختصاص داده می‌شود. علاوه بر این، مدلی را طراحی می‌کنیم که مزایای استفاده از مدل چک پوینت-راه اندازی مجدد و مدل نسخه اصلی-پشتیبان را در کنار ابزار طبقه بندی ارائه می‌دهد. به عبارت دیگر، مانند کار مرتبط [۴] که مدل چک پوینت و مدل پشتیبان اولیه سنتی را با مکانیسم طبقه بندی در محیط ابری ترکیب نمود، ما هم به شکل نو آورانه ای با وجود محدودیت ها از این مکانیسم ها در محیط مه استفاده می‌کنیم. شکل ۴ نمودار جریان مدل پیشنهادی را نشان می‌دهد.

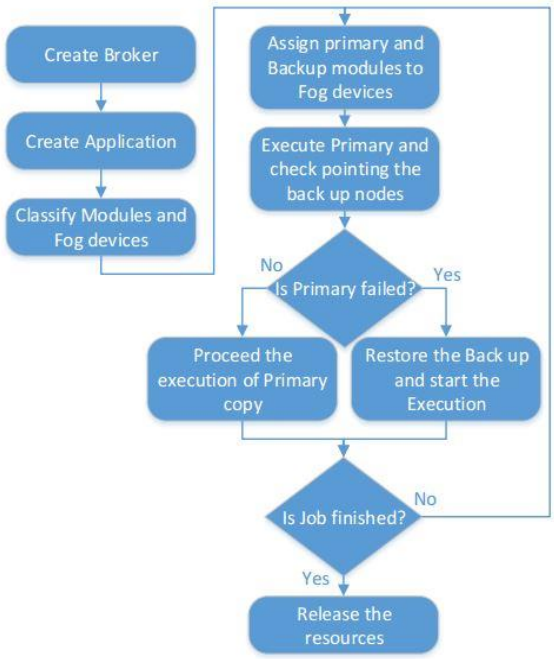
در این کار، ابتدا روشهای طبقه بندی گره های مه و وظایف را پیشنهاد می‌کنیم تا کارآیی زمانبندی وظایف را بهبود ببخشد. ماژول‌های وظیفه و گره های مه به گروه های مختلف طبقه بندی می‌شوند:

الف) دسته بندی ماژول ها: طبقه بند ماژول، ماژول ها را به انواع زیر از هم جدا می‌کند، ماژول های محاسبات فشرده (CIM)[۲۰]، ماژول های داده فشرده (DIM)[۲۱] و ماژول های متوازن (BM)[۲۲].

طبقه بندی ماژول‌ها به زمان انتقال داده و زمان پردازش ماژول وظایف بستگی دارد. زمان انتقال داده، مدت زمان صرف شده برای بارگیری داده های مورد نیاز برای اجرای در ماژول‌ها است.

$$DTT = \frac{Module_{Size}}{Module_{BW}} + \frac{ParentModule_{Size}}{ParentModule_{BW}} \qquad (۱)$$

که $Module_{Size}$ داده‌های مورد نیاز برای یک ماژول است، $Module_{BW}$ سرعت انتقال داده است. $ParentModule_{Size}$ نتیجه ماژول والدین است و $ParentModule_{BW}$ سرعت انتقال داده ها بین دو گره مه است، همانطور که در معادله (۱) نشان داده شده است.



شکل۴ : نمودار جریان مدل پیشنهادی

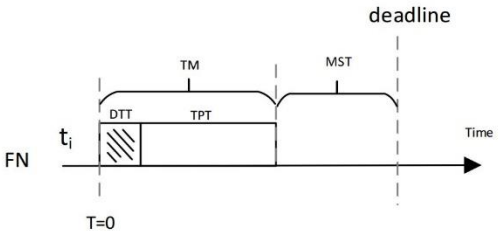
زمان پردازش ماژول وظیفه (TPT)[۲۳] توسط معادله (۲) محاسبه می‌شود که همان زمان لازم برای اجرای یک ماژول است.

$$TPT = \frac{Module_{MIPS}}{Device_{MIPS}} \qquad (۲)$$

که در آن $Module_{MIPS}$ اندازه محاسبه یک ماژول است که با MIPS اندازه گیری می‌شود. علاوه بر این، $Device_{MIPS}$ قابلیت CPU گره مه‌ای است که آن نیز با MIPS اندازه گیری می‌شود.

برای توضیح کاملتر می توان اینطور بیان کرد که زمان انتقال اطلاعات یا همان DTT، زمان صرف شده برای بارگیری داده هایی است که برای اجرا در ماژول ها مورد نیاز است. این به مقدار داده و سرعت انتقال بستگی دارد.

برای طبقه بندی ماژولها به DTT و MPT نیاز داریم زیرا طبقه بندی ماژولها بر پایه زمان انتقال داده و زمان پردازش کار استوار است. اگر DTT بسیار بزرگتر از MPT باشد ، به عنوان ماژول های فشرده داده (DIM) طبقه بندی می شود و اگر DTT بسیار کوچکتر از MPT باشد ، به عنوان ماژول های محاسبات فشرده (CIM) طبقه بندی می شود و در نهایت، چنانچه مقدار DTT نزدیک به MPT باشد، به عنوان ماژول های متعادل (BM) طبقه بندی می‌شود.



شکل۵ : اجرای ماژول وظیفه بر روی گره مه

ب) دسته بندی گره‌های مه: به طور کلی یک گره دارای پارامترهای مختلفی است که باعث ایجاد تنوع در آن می گردد. برای آنکه بتوان گره ها را دسته بندی کرد به دو پارامتر اصلی آن توجه گردید: پهنای باندی که گره از آن استفاده می کند و توان عملیاتی پردازنده آن. برخی گره ها توان عملیاتی بالایی دارند وبرخی پهنای باند بالاتری دارند و برخی نیز بین این دو حالت اند. لذا برای دسته بندی گره ها این دو پارامتر مورد توجه قرار گرفت و بر این اساس گره ها به سه دسته تقسیم شدند: دسته ۱، گره هایی که سرعت انتقال اطلاعات آنها بالاست و توانایی عملیاتی پردازنده آن کمتر است. دسته ۲، گره های مه ای که سرعت انتقال داده پایین است در عوض توانایی عملیاتی پردازنده آن بالاست. و دسته ۳ شامل گره های مه ای است که سرعت انتقال داده ها و توانایی عملیاتی پردازنده در آن متوازن است.

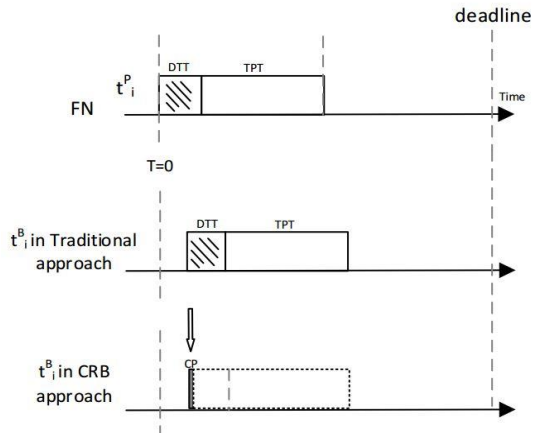
با استفاده از روش طبقه بندی، ماژول‌ها و گره‌های مه به دسته-های مختلفی تقسیم می‌شوند سپس ماژول مناسب در گره مناسبی نگاشت می‌شود. این کار باعث افزایش استفاده از منابع مه و کاهش زمان پاسخ می‌شود.

مدلی طراحی کردیم که فواید مدل Checkpoint-Restart و Primary Backup را برای ماژول های مختلف با هم ترکیب می-کند. این فرض را در نظر گرفتیم که حداکثر یک گره در یک بازه زمانی ممکن است خراب شود. همانطور که در شکل ۵ نشان داده شده‌است، وظیفه باید قبل از رسیدن به زمان پایانی اش به طور کامل اجرا شود. TM زمان اجراست که از معادله (۳) بدست می‌آید.

$$TM = DTT + TPT \qquad (۳)$$

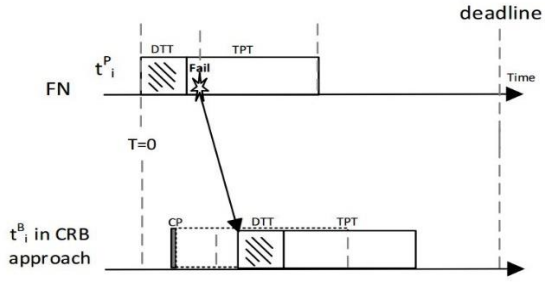
زمان فراخی ماژول (MST) زمان بین زمان پایان ماژول و مهلت آن است.

همانطور که شکل ۶ نشان می‌دهد، در یک مدلی که فقط PB یا همان مدل نسخه اصلی-پشتیبان است، یکی از گره‌های مه به طور کامل نسخه اولیه t_i^P را اجرا می‌کند، و گره مه دیگر نسخه پشتیبان t_i^B را به طور کامل اجرا می‌کند. t_i^P به معنای ماژول i ام مربوط به نسخه اولیه یا همان نسخه اصلی و t_i^B به معنای ماژول i ام مربوط به نسخه پشتیبان است.



شکل۶ : PB سنتی در مقایسه با PB پیشنهادی
PB: Primary Backup

در این حالت، حتی اگر اجرای ماژول وظیفه تضمین شده باشد، تعداد قابل توجهی از منابع توسط نسخه های پشتیبان اشغال می‌شوند. یعنی در حالی که نسخه اصلی در گرهی بدرستی در حال اجراست نسخه پشتیبان نیز همزمان در یک گره دیگری به شکل کامل اجرا می‌شود. به تبع این کار، مصرف انرژی گره‌های پشتیبان افزایش می‌یابد. در حالیکه چنانچه بر اساس مدل پیشنهادی‌مان نسخه پشتیبان به روش چک پوینتینگ ذخیره شده باشد، و به شکل کامل اجرا نگردد، باعث صرفه جویی در مصرف انرژی می‌شود. و تنها زمانی نسخه پشتیبان کامل اجرا می-شود که نسخه اولیه خراب گردد در این صورت اجرای نسخه پشتیبان از سر گرفته می‌شود و اجرای خود را کامل می‌کند.



شکل۷ : رویکردCRB

همانطور که در شکل ۷ نشان داده شده است، t_i^P به شکل کامل اجرا می‌شود مگر اینکه خرابی در آن رخ دهد. در این حالت نسخه پشتیبان t_i^B اجرا را از سر می‌گیرد تا نقش خود را برای دستیابی به هدف تحمل پذیری خطا ایفا نماید.

الگوریتم ۱ شبه کد مدل CRBC را نشان می‌دهد. در ابتدا

Broker سپس Application ایجاد می‌شود. کارگزار به عنوان یک عامل بین سنسورها و سطح بالای دستگاه است.کاربرد به عنوان ترکیبی از ماژول ها ایجاد می‌شود که عناصر پردازش داده ها را ترکیب می‌کند [۳۳]. این کاربرد برای استقرار در مه ساخته شده است و بر اساس مدل توزیع داده جریان داده شده است. سپس دستگاه‌های مه ای در سطح معماری تعیین می‌شوند. پس از آن، روش پیشنهادی همه ماژول‌ها و دستگاه‌های مه را طبقه بندی می‌کند تا اطمینان حاصل کند که ماژول‌ها به طور دقیق با دستگاه‌های مه هماهنگ و تطبیق می‌یابد.

ما فرض می‌کنیم که نسخه اصلی t_k^P و نسخه پشتیبان t_k^B از ماژول t_k برای دستیابی به قابلیت مقاومت در برابر اشکال، به همان گره مه نمی‌روند. چرا که، اگر نسخه اصلی و پشتیبان ماژول t_k در یک گره مه قرار گیرند و آن گره خراب شود، هر دو به طور همزمان خراب می‌شوند.

Algorithm 1: CRBC Model

```
1  Create Broker.
2  Create Application.
3  Create Fogdevices.
4  for (i=1 to last FD) do
5    for (j=1 to last Module) do
6      if(IsModuleCompatibleWithDevice(Module_j,FD_i))
       as Algorithm 2 then
7            BackupModule(Module_j, FD_i) as Algorithm 3
8      end if
9    end for
10 end for
11 Create Controller(FDs,Sensors,Actuators).
12 Submit Application to FDs.
   Scheduling:
13 for (k=1 to last Modules) do
14     Parent_k=FindParentModule(Module_k ).
15     Calculate DTT_k and TPT_k by Eqs.(1)and (2).
16     if(!IsPrimaryModuleFailed(Module_k) then
17       Calculate EST_k as Eq(4).
18       Calculate EFT_k as Eq(5).
19     else
20       Calculate EST_k as Eq(6).
21       Calculate EFT_k as Eq(5).
22     end if
23     VMList=getVMList().
24     for (i=1 to last VM) do
25       if (Module_k not in VM_i ) then
26           Allocate VM_i to Module_k.
27       end if
28     end for
29 end for
```

شبه کد مربوط به دسته بندی ماژول‌ها و دستگاه‌های مه‌ای در الگوریتم ۲ آمده‌است. ماژول‌ها به سه دسته بر طبق معادله ۱و ۲ تقسیم می‌شوند (رجوع شود به الگوریتم ۲، خط ۱۰ - ۲۲). روش دسته بندی دستگاه‌های مه‌ای مانند ماژول‌ها است. گره‌های مه به سه دسته تقسیم می‌شوند (رجوع شود به الگوریتم ۲، خط ۱-۸). تمامی‌ماژول‌ها و دستگاه‌های مه‌ای را به این منظور طبقه بندی می‌کنیم تا بتوان بعد از آن با اجرای ادامه الگوریتم اطمینان حاصل کرد که هر ماژول می‌تواند دقیقاً به دستگاه مه مناسب خود منتسب شود. در نهایت هنگامی‌که پرچم Tag مقدار درست را برگرداند، به این معنی است که ماژول و دستگاه مه سازگار هستند. در ادامه‌ی الگوریتم ۱ چنانچه ماژول اصلی دچار خرابی نشود، سریعترین زمان شروع ماژول (EST) بر مبنای معادله ۴ محاسبه می‌شود.

$$EST_k = EFT_{Parent_K} + DTT_k + SD \qquad (۴)$$

این بدان معنی است که سریعترین زمان شروع ماژول اولیه t_k به عملکرد گره مه‌ای که قرار است ماژول t_k در آن اجرا شود و نیز به

Algorithm 2: Classification

```
   Input: Module and FD
   Output: True or False
    Classify Fog Devices:
1  Parent = GetModuleParent(module).
2  Tag = false.
3  if  (device.MIPS > device.BW) then
4       deviceType = 1.
5  else if  (device.MIPS < device.BW) then
6          deviceType = 2.
7  else
8          deviceType = 3.
   Classify Modules:
9  Calculate DTT and TPT as Eqs.(1) and (2).
10 if (DTT > TPT) then
11     if  (deviceType=2) then
12         Tag = true.
13     end if
14  else if (DTT < TPT) then
15     if (deviceType =1) then
16         Tag = true;
17     end if
18  else if (DTT - TPT < 0.05 ) then
19     if (deviceType=3) then
20       Tag = true.
21     end if
22 end if
23 Return  Tag.
```

پارامترهای ماژول والدین t_k بستگی دارد. EFT_{Parent_K} سریعترین زمان پایان ماژول والدین t_k است و SD تأخیر سیستم گره مه است. سریعترین زمان پایان ماژول t_k بر اساس معادله ۵ تعریف شده است.

$$EFT_k = EST_k + TPT_k \qquad (۵)$$

که EST_k سریعترین زمان شروع کار t_k است و TPT_k زمان پردازش ماژول وظیفه t_k است. اما در صورت خرابی ماژول اولیه، ماژول پشتیبان اجرای مجدد را از سر می‌گیرد و کار خود را برای دستیابی به هدف تحمل پذیری اشکال انجام می‌دهد. EST ماژول پشتیبان بر اساس معادله ۶ تعریف می‌شود.

$$EST_k = TFT_k + DTT_k + SD \qquad (۶)$$

سریعترین زمان شروع ماژول به زمان خرابی نسخه اصلی t_k^P (TFT) و زمان انتقال داده از ماژول k و تأخیر سیستم گره مه بستگی دارد. در آخر، ماژول برای اجرا به VM اختصاص داده می‌شود.

Algorithm 3: Backup Copy Processing

```
Input: Module and FD
for (i=1 to FDlast) do
    x=random * FDlist.size %  FDlist.size.
    if (! FDlist(x) !=Fogdevice) then
        Add Module to Fogdevice.
        Add Module to ModulePrimaryList.
        ModuleBackupList.put(Module,Fogdevice(x)).
    end if
    Break.
end for
```

سیستم چک پوینتینگ بر اساس الگوریتم ۳ اجرا می‌شود. ماژول پشتیبان به دستگاه مه مناسب اضافه می‌شود. دستگاه مه ماژول پشتیبان را ذخیره کرده و آنرا اجرا نمی‌کند، مگر اینکه ماژول اصلی از بین برود. در این حالت، دستگاه مه، ماژول پشتیبان را بازیابی و آنرا به طور کامل اجرا می‌کند.

در لبه ابر ، برخی از دستگاه های متصل ممکن است خراب شوند و در نتیجه، درخواست های اختصاص داده شده به این دستگاه ها تحت تأخیر قرار می گیرند. به طور کلی، شکست ها به طور تصادفی در محیط های مه اتفاق می افتند. مدل سازی خرابی ها به فاصله زمانی صرف شده بین دو شکست پی در پی بستگی دارد که می تواند به عنوان یک متغیر تصادفی از توزیع احتمال پوآسون نشان داده شود، که نشان می دهد تعداد خرابی دستگاه ها برای هر دو دوره زمانی مستقل یکسان نخواهد بود[۳۵]. البته به منظور ایجاد خرابی در سیستم باید آن را شبیه سازی نمود و به صورت تصادفی برخی دستگاه های مه به حالت توقف در آیند. برای طراحی خرابی هنگام اجرای ماژول ها در رایانش مه، الگوریتم ۴ را در نظر می گیریم. از احتمال شکست برای تنظیم نرخ خرابی استفاده می شود. زمان خرابی قبلی توسط آخرین ساعت خرابی توسط کلودسیم محاسبه می شود. از این الگوریتم برای شبیه سازی خرابی و غیرفعال کردن موقت دستگاه مه استفاده می‌کنیم.

Algorithm 4: Fail and Up Fog Device

```
1  if ((math.random < Fail.Probability)
     && FailCounter ==0)) then
2    FailCounter = 1;
3    Fail.FD.Id = getFD.Id;
4    FailTime = cloudsim.clock;
5  end if
6  if ((Fail.FD.Id ==FD.Failed) &&
     (cloudsim.clock - FailTime >= FailTimeDuration )) then
7    FailCounter = 0;
8    Fail.FD.Id = -1;
9  end if
```

به منظور بررسی عملکرد روش پیشنهادی، تمام عملیات بر روی یک پلتفرم شبیه سازی انجام می‌شود. عملکرد روش پیشنهادی را با مقایسه آن با سه روش دیگر ارزیابی می‌کنیم.

CRBC را با PBC (نسخه اصلی-پشتیبان به همراه طبقه بندی و بدون چک پوینتینگ) و CRB (چک پوینتینگ و نسخه اصلی-پشتیبان و بدون طبقه بندی) و First-Fit کلاسیک مقایسه می‌کنیم.

تفاوت های بین CRBC و روش های دیگر به شرح زیر است:

- PBC: از مدل PB سنتی در کنار طبقه بندی استفاده می‌کند اما از روش چک پوینت بهره نمی‌گیرد [۴]. با مقایسه CRBC با PBC می‌توان به اثربخشی روش CRBC اطمینان حاصل کرد.
- CRB: از تکنیکهای چک پوینت و نسخه اصلی-پشتیبان استفاده می‌کند، اما از طبقه بندی بی بهره است [۴]. مقایسه CRB با CRBC می‌تواند برتری روش طبقه بندی CRBC را تأیید کند.
- FF یا الگوریتم کلاسیک First-Fit: این روش دارای سریعترین جستجو است به این شکل که اولین دستگاه مه و ماژول را برای اختصاص یک فرآیند جستجو می کند. این کار باعث بهبود تاخیر و استفاده بهینه از منابع

می شود اما تحمل خطا را در زمانبندی ماژول ها ارائه نمی دهد که در iFogsim ارائه شده است[۳۳]. با مقایسه CRBC و FF می‌توان به این نتیجه رسید که FF فاقد مکانیزم تحمل پذیری اشکال است لذا برتری از آن CRBC است.

۵- ارزیابی عملکرد

در این بخش نتایج آزمایشی روش پیشنهادی ارائه شده است. بخش اول و دوم این بخش به تشریح راه اندازی آزمایشی و معیارهای ارزیابی عملکرد می‌پردازد. سپس بر اساس معیارها، پارامترهای تأثیر بر عملکرد روشها در بقیه این بخش آورده شده است.

۵-۱- راه اندازی آزمایشی

در این تحقیق از iFogsim به عنوان بستر شبیه سازی استفاده می‌کنیم که می‌توان با آن سناریوها را اجرا کرد و درستی الگوریتم ها را بررسی نمود[۳]. جدول ۲ نوع دستگاه‌های مه و تنظیمات آنها را نشان می‌دهد. هر دستگاه مه دارای تعدادی پارامترها و مقادیر مرتبط است.

این پارامترها شامل تعداد (میلیون) دستوراتی که در یک ثانیه اجرا می‌شود MIPS، میزان حافظه دستگاه بر حسب کیلو بایت RAM، پهنای باند ارتباطی با سطح بالایی بر حسب کیلو بایت بر ثانیه UpBW، پهنای باند ارتباطی با سطح پایینی برحسب بایت بر ثانیه DownBW، سطح گره در توپولوژی سلسله مراتبی Level، توان مصرفی Busy Power و توان بیکاری Idle Power بر حسب میلی وات است.

جدول ۳ به پیکربندی میزبان اشاره دارد. از خصوصیات میزبان می‌توان به ذخیره سازی، پهنای باند، معماری، سیستم عامل، مدل VM، منطقه زمانی، هزینه، هزینه حافظه و هزینه ذخیره سازی اشاره کرد[۳۳].

۵-۲- سنجه های ارزیابی

پنج سنجه[۲۴] زیر برای تأیید عملکرد روشهای ذکر شده در بالا که همان CRBC، PBC، CRB و FF است، استفاده می‌شود:

۱) انرژی مصرفی: مقدار انرژی یا توان مصرفی است. مصرف انرژی با معادله ۷ محاسبه می‌شود.

$$Energy = CEC + (CT - LUUT) * NP \qquad (۷)$$

میزان مصرف انرژی دستگاه مه با اندازه گیری توان گره‌ها در یک قاب زمانی خاص از زمان محاسبه می‌شود. در این رابطه CEC مصرف انرژی فعلی است، CT زمان فعلی است، LUUT آخرین زمان بهره وری به روز شده و NP قدرت گره است.

۲) هزینه اجرای کل: هزینه اجرای ماژول ها در دستگاه های مه است.

$$Cost = \sum_{i=1}^{N}[EC + (CC - LUUT) * RPM * LU * TM] \quad (۸)$$

برای محاسبه هزینه اجرا، از مجموع MIPS گره ها استفاده می‌کنیم که بر اساس قاب زمانی محاسبه می‌شود. قاب زمانی اختلاف بین زمان فعلی شبیه سازی و آخرین زمان استفاده است. در معادله (۸)، N تعداد گره های مه، EC هزینه اجرای است، CC کلاک CloudSim یا زمان فعلی شبیه سازی است.

LUUT آخرین زمان بهره وری به روز شده است، RPM نرخ در هر MIPS است که برای هر لبه بین ماژول ها متفاوت است. و LU آخرین استفاده ای می‌باشد که بر اساس LU=Min(1,TMA/TM) محاسبه شده‌است، که در آن TMA کل MIPS های اختصاص داده شده برای گره مه است و TM کل MIPS گره مه است.

جدول ۲: پیکر بندی دستگاه مه ای

Name	RAM	MIPS	UpBw	DownBw	Level	Busy power	Idle power
Cloud	40,000	44,800	100	10,000	0	1648	1.332
Proxy-server	4000	2800	10,000	10,000	1	107.339	83.433
Area	4000	2800	10,000	10,000	1	107.339	83.433
Camera	1000	500	10,000	10,000	3	87.53	82.44

جدول ۳: پیکر بندی میزبان

محیط	هزینه حافظه	هزینه	منطقه زمانی	مدلVM	سیستم عامل	معماری	پهنای باند	ذخیره سازی
0.01	0.05	3	10	Xen	Linux	X86	10,000 B/S	10,000,000 B

۳) تأخیر: مشخص می‌کند که داده ها چه اندازه طول می‌کشد که در بین گره ها بگذرد. تأخیر حلقه مدل کاربردی توسط ساعت Cloudsim و زمان پایان tuple محاسبه می‌شود. با کامل شدن تاپل، زمان پایان تاپل بوسیله معادله ۹ محاسبه می‌شود. TS زمان شروع تاپل است، TA متوسط زمان CPU یک تاپل است، CC کلاک Cloudsim است. (CC - TS) زمان اجرای tuple و C1 تعداد تاپل های اجرا شده است. پرچم در این معادله شرط است، بنابراین اگر TA محاسبه شود، پرچم مقدار درست می‌گیرد، در غیر این صورت مقدار پرچم نادرست خواهد بود.

$$TupleEndTime = \begin{cases} CC - TS & if \quad Flag = False \\ \frac{TA*C1+(CC-TS)}{C1+1} & otherwise \end{cases} \qquad (۹)$$

معادله (۱۰) تأخیر حلقه مدل کاربردی را محاسبه می‌کند، به این صورت که CC ساعت Cloudim است و ET زمان انتشار یک تاپل است. ET زمان ارسال تاپل از یک ماژول به ماژول دیگر است.

$$Delay = CC - ET \qquad (۱۰)$$

معادله (۱۱) زمان دریافت تاپل را محاسبه می‌کند، که TS و Delay در معادله ۹ استفاده می‌شوند. C2 تعداد انواع تاپل های دریافتی است [۳۴].

$$TupleReceiptTime = \frac{TS*C2+Delay}{C1+1} \qquad (۱۱)$$

۴) میزان استفاده از شبکه: مقدار داده ارسال و دریافت شده توسط شبکه را نشان می‌دهد.

۵) ماژول های اجرا شده در دستگاه مه: تعداد ماژول هایی است که به طور کامل در دستگاه های مه اجرا می‌شود. این سنجه معیار مفیدی برای اندازه گیری بار در دستگاه‌های مه است.

جدول ۴: پارامترهای متغیر آزمایشی

No	Parameter	Value (Fixed)	Values
1	Area and camera's set	set4(5 areas, 9 cameras)	set1(2,6), set2(3,7), set3(4,8), set4(5,9)
2	Failure rate	0.01	0.01, 0.03, 0.07, 0.1
3	Module size	10,000	1000, 2000, 5000, 10000
4	Fail time's duration	200	50, 100, 200, 500

پارامترهای مربوط به متغیرهای آزمایشی زیر را تعریف می‌کنیم تا بتوان بین عملکرد روشهای ذکر شده در بالا که همان CRBC، PBC، CRB و FF است بر اساس سنجه هایی که در بالا بدان اشاره شد، مقایسه انجام نمود.

۱) نواحی و دوربینها: که در ۴ مجموعه بر اساس شکل ۲ در نظر گرفته شده است. مجموعه ۱ دارای (۲ ناحیه و ۶ دوربین در هر منطقه) است، مجموعه ۲ دارای (۳ ناحیه، ۷ دوربین)، مجموعه ۳ است (۴ منطقه، ۸ دوربین) و مجموعه ۴ دارای (۵ ناحیه، ۹ دوربین) است. یعنی ما در هر مورد ۴ حالت جهت بررسی و مقایسه روشها با یکدیگر داریم. عملکرد چهار روش را با تغییر مجموعه ها (ناحیه و تعداد دوربین) آزمایش می‌کنیم از مجموعه ۱ تا مجموعه ۴. نتایج در بخش ۵.۳ ذکر شده است.

۲) میزان خرابی: بسامد خرابی های سیستم است که ۰.۰۱٪، ۰.۰۳٪، ۰.۰۷٪ و ۰.۱٪ است. به این شکل که پارامتر میزان خرابی سیستم را در هر بار اجرای کامل، تغییر می‌دهیم. درمرحله اول میزان خرابی ۰.۰۱٪ در نظر می‌گیریم و متدهای مختلف را با این پیش فرض اجرا می‌کنیم و نتایج را ثبت می‌کنیم سپس مقدار پارامتر را در مراحل بعدی تغییر می‌دهیم. یعنی میزان خرابی به تدریج از ۰.۰۱٪ به ۰.۱٪ افزایش می‌یابد. تاثیر پارامتر میزان خرابی را بر عملکرد هر روش را جداگانه مورد ارزیابی قرار می‌دهیم. نتایج آزمایش در زیر بخش ۵.۴ آورده شده است.

۳) اندازه ماژول: طول ماژول ها برحسب بایت است که اندازه ها ۱۰۰۰، ۲۰۰۰، ۵۰۰۰ و ۱۰۰۰۰ بایت است. تأثیر پارامتر اندازه ماژول را بر عملکرد هر روش آزمایش می‌کنیم. اندازه ماژول ها را از ۱۰۰۰ به ۱۰۰۰۰ بایت افزایش می‌دهیم و نتایج تغییر اندازه ماژول بر عملکرد هر متد را جداگانه ثبت می‌کنیم. نتایج به دست آمده از آزمایش در بخش ۵.۵ ارائه شده است.

۴) مدت زمان خرابی: مدت زمان است که برای خرابی یک گره مه در نظر گرفته می‌شود و مقادیر آن ۵۰، ۱۰۰، ۲۰۰ و ۵۰۰ ms است. تأثیر مدت زمان خرابی را بر عملکرد هر روش تحلیل و ارزیابی می‌کنیم. به عبارت دیگر، همه روشها را با یکدیگر بر اساس مقادیر مختلف این پارامتر (که چهار مقدار مختلف دارد) به شکل جداگانه که از ۵۰ به ۵۰۰ ms افزایش می‌یابد، مورد ارزیابی قرار می‌دهیم.

در زیر بخش ۵.۶ نتایج آزمایش را مورد بحث قرار داده‌ایم. برای بررسی صحت عملکرد روشها تحت پارامتر های مختلف ذکر شده، هر بار که اثرات یکی از پارامترها را بر روی روشها می‌خواهیم بررسی کنیم مقادیر آن پارامتر مورد نظر متغیر در نظر گرفته می‌شود و در هر بار اجرا مقدار پارامتر بر حسب جدول تغییر می‌یابد و سه پارامتر دیگر مقدار ثابتی به خود می‌گیرند. جدول ۴ پارامترها و مقادیر را نشان می‌دهد.

۵-۳- تاثیر نواحی و تعداد دوربین بر عملکرد

در این زیر بخش، عملکرد چهار روش را با تغییر ناحیه و شماره دوربین از مجموعه ۱ به مجموعه ۴ نشان می‌دهیم . به عبارتی، ۴

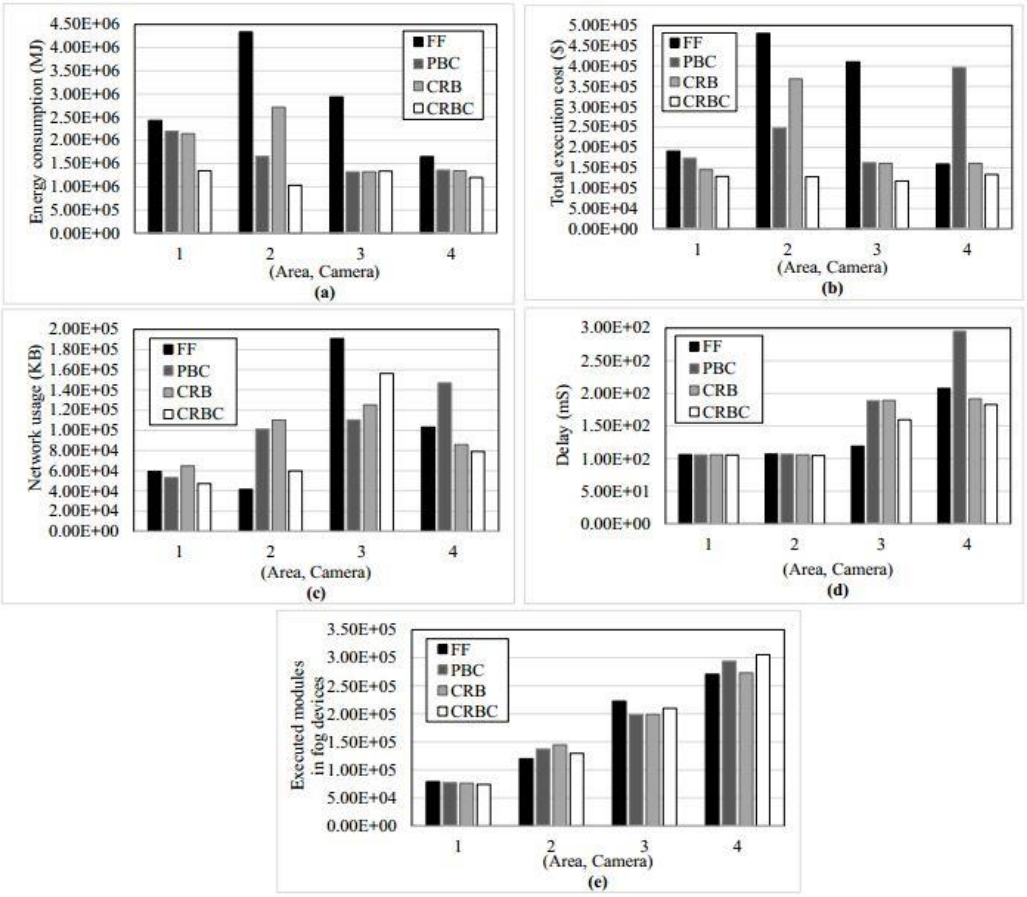
حالت برای آزمایش داریم و در هر حالت همه روشها را با هم مقایسه می‌کنیم. شکل ۸ نتایج آزمایش را نشان می‌دهد.

همانطور که در شکل ۸ (a) نشان داده شده است، با افزایش تعداد نواحی و در نتیجه دوربین ها، روش CRBC مصرف انرژی کمتری را نسبت به سه روش دیگر نشان می‌دهد. مصرف انرژی روش First-fit بالاتر از روشهای مقاوم در برابر اشکال است.

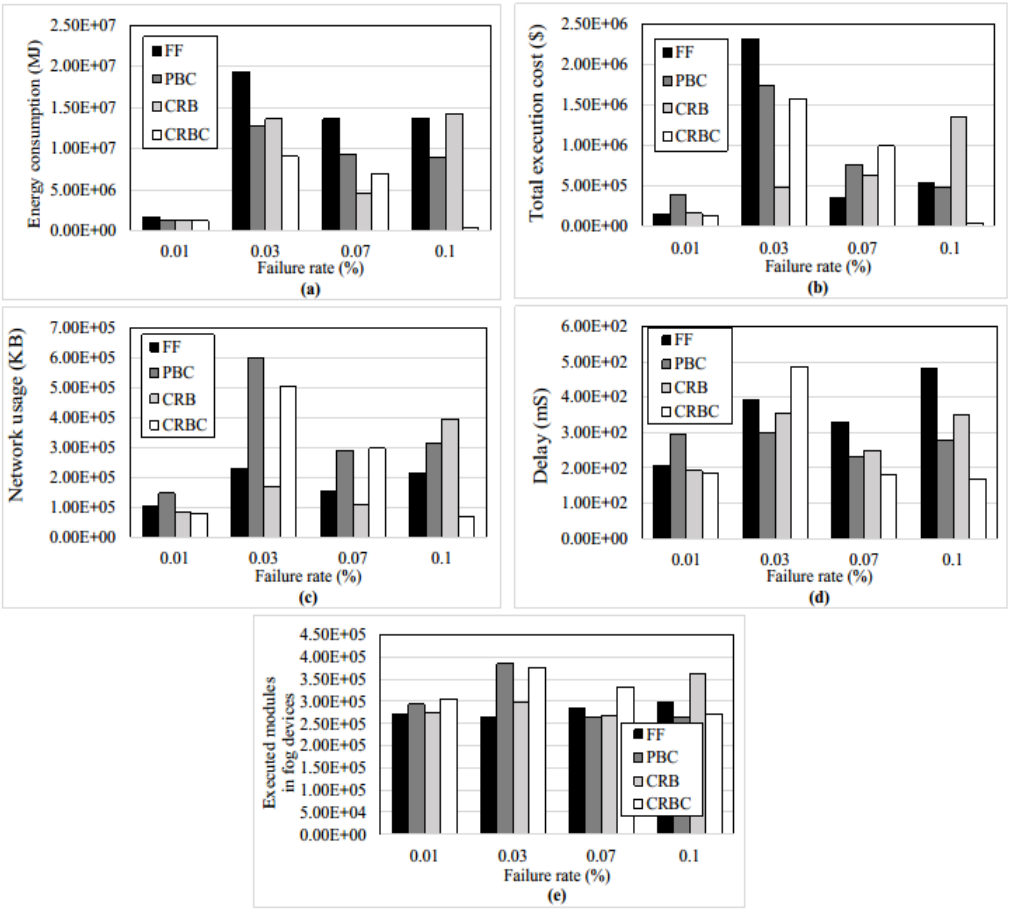
زیرا وقتی دستگاه و به تبع آن، ماژول از کار می‌افتد، پس از راه اندازی مجدد دستگاه ماژول باید از ابتدا اجرا شود؛ بنابراین، مصرف انرژی سیستم افزایش می‌یابد. شکل ۸ (b) نشان می‌دهد که روش ما بدلیل استفاده از چک پوینتیگ و رویکرد نسخه اصلی-پشتیبان هزینه اجرای کمتری نسبت به سایر روش ها دارد. روند سه روش دیگر نامنظم است. مصرف انرژی در آنها ابتدا افزایش و سپس کاهش می‌یابد. همانطور که در شکل ۸ (a) و (b) نشان داده شده است روند همه روشها تقریباً مشابه است.

در شکل ۸ (c) مشاهده می‌شود که روند همه روش ها برای سنجه میزان استفاده از شبکه نامنظم است. همانطور که روشن است میزان استفاده از شبکه در روش CRBC تنها در مجموعه ۱ و مجموعه ۴ نتیجه بهتری دارد. از شکل ۸ (d) می‌توان دریافت که میزن تأخیر در هر روش با افزایش تعداد نواحی و دوربینها افزایش می‌یابد. در واقع، هنگامی‌که تعداد نواحی و دوربینها دوربین کم است، تفاوت تأخیر ناچیز است.

در شکل ۸ (e) می‌توانیم ببینیم که با افزایش تعداد نواحی و تعداد دوربین، تعداد ماژولهای اجرا شده در دستگاههای مه برای هر روش افزایش می‌یابد. و درنهایت، در روش CRBC، ماژولهای اجرا شده در دستگاههای مه ای رشد بیشتری نسبت به بقیه روشها دارند. در این زیر بخش، ما تاثیر نرخ خرابی را بر عملکرد هر روش آزمایش می‌کنیم. نرخ خرابی به تدریج از ۰.۰۱ به ۰.۱٪ افزایش می‌یابد. بنابراین، ما ۴ حالت برای آزمایش داریم. نتایج به دست آمده از آزمایش در شکل ۹ ارائه شده است.



شکل ۸: تأثیر نواحی و تعداد دوربین بر عملکرد



شکل ۹: تأثیر نرخ خرابی بر عملکرد

۵-۴- تاثیر نرخ خرابی بر عملکرد

در شکل ۹ (a) با افزایش نرخ خرابی، روش CRBC نتایج بهتری را به دلیل انجام طبقه بندی ارائه می‌دهد یعنی مصرف انرژی کمتری نسبت به سه روش دیگر دارد. مصرف انرژی روش First-fit بالاتر از سایر روشها است زیرا وقتی خرابی رخ می‌دهد، این روش توانایی مقاومت در برابر اشکال را ندارد و باید ماژول از نو به شکل کامل در دستگاه مه اجرا شود. بنابراین، مصرف انرژی افزایش قابل توجهی می‌یابد.

شکل ۹ (b) نشان می‌دهد که روند همه روشها نامنظم است، ابتدا افزایش می‌یابند و سپس در ادامه کاهش می‌یابند. روش پیشنهادی ما وقتی نرخ خرابی به ۰.۱ برسد نتیجه بهتری را نشان می‌دهد. همانطور که در شکل ۹ (a) و (b) نشان داده شده است روند همه روشها تقریباً مشابه است.در شکل ۹ (c) مشاهده می‌شود که روند همه روش ها در بهره گیری از سنجه میزان استفاده از شبکه، نامنظم است. میزان استفاده از شبکه در روش CRBC هنگامی‌که میزان خرابی ۰.۰۱ و ۰.۱ باشد نتیجه بهتری می‌دهد و روش CRB زمانی که میزان خرابی ۰.۰۳ و ۰.۰۷ باشد نتیجه بهتری خواهد داشت. در شکل ۹ (d) مشاهده می‌شود که روند افزایش همه روشها با افزایش نرخ خرابی نامنظم است. اما بطور کلی، روش پیشنهادی ما در نتیجه‌ی استفاده از طبقه بندی ماژول و دستگاه نتیجه بهتری دارد. همانطور که در شکل ۹ (e) نشان داده شده است، هنگامی‌که میزان خرابی کمتر از ۰.۰۷ باشد، ماژول های اجرا شده در دستگاههای مه برای روش CRBC نتیجه بهتری دارند. شکل ۹ نشان می‌دهد که اگر میزان خرابی کل سیستم ۰.۰۱ باشد، نتایج کلیه سنجه ها نتایج بهتری دارند.

۵-۵- تاثیر اندازه ماژول بر عملکرد

در این زیر بخش تأثیر اندازه ماژول بر عملکرد هر روش را تحلیل می‌کنیم. ۴ حالت برای بررسی داریم که در هر حالت تمامی۴ روش را با یکدیگر مقایسه می‌کنیم. اندازه ماژول از ۱۰۰۰ به ۱۰۰۰۰ بایت افزایش می‌یابد، و با هر بار افزایش آزمایشات را ثبت و بررسی می‌کنیم. نتایج آزمایش در شکل ۱۰ ارائه شده است.

در شکل ۱۰ (الف) با افزایش اندازه ماژول، ملاحظه می‌شود که

همه روش‌ها رفتار متفاوتی نشان می‌دهند. البته وقتی اندازه ماژول ۱۰۰۰۰ B باشد نتیجه بهتری را می‌توان مشاهده کرد. همانطور که در شکل ۱۰ (b) نشان داده شده است، هنگامی‌که اندازه ماژول افزایش می‌یابد، هزینه اجرای کل برای روش های First-fit و CRBC کاهش می‌یابد، در حالیکه روند در روش های PBC و CRB نامنظم است. بهترین نتیجه زمانی حاصل می‌شود که اندازه ماژول ۱۰۰۰۰ B باشد.

شکل ۱۰ (c) نشان می‌دهد که میزان استفاده از شبکه در الگوریتم First-fit با افزایش اندازه ماژول کاهش می‌یابد و روش CRBC نتیجه بهتری نسبت به سایر روش ها دارد. شکل ۱۰ (d) نشان می‌دهد که هنگامی‌که اندازه ماژول ۱۰۰۰۰ B است، بهترین نتایج را شاهد هستیم. به طور کلی، روش پیشنهادی ما به دلیل استفاده از چک پوینتینگ نتیجه بهتری دارد.

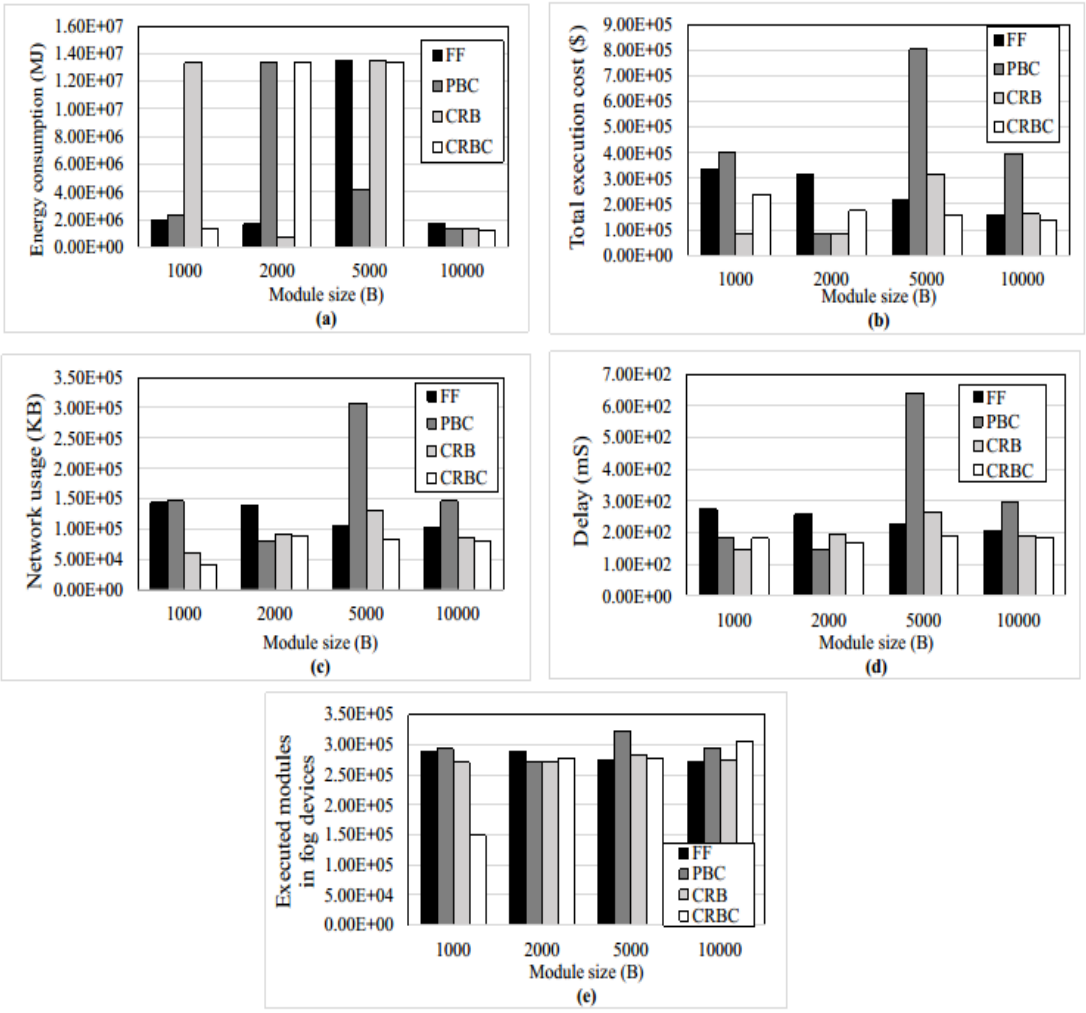
همانطور که در شکل ۱۰ (e) نشان داده شده است، هنگامی‌که اندازه ماژول ۱۰۰۰ B است، ماژول های اجرا شده در دستگاههای مه برای روش CRBC بدترین نتیجه را دارند. و هنگامی‌که اندازه ماژول افزایش می‌یابد نتیجه بهتر می‌شود.

در مجموع شکل ۱۰ نشان می‌دهد وقتی اندازه ماژول به ۱۰۰۰۰ B می‌رسد، نتایج حاصل از روش پیشنهادی ما خروجی بهتری دارند.

۵-۶- تاثیر مدت زمان خرابی بر عملکرد

تأثیر مدت زمان خرابی را بر عملکرد هر روش را در این زیر بخش آزمایش می‌کنیم. این پارامتر مدت زمانی را شامل می‌شود که گره در حالت خراب قرار دارد. مدت زمان خرابی را از ۵۰ به ۵۰۰ ms افزایش می‌دهیم که نتایج این آزمایش را می توان در شکل ۱۱ مشاهده نمود.

در شکل ۱۱ (a) واضح است که برای تمامی روشها نتیجه بهتر



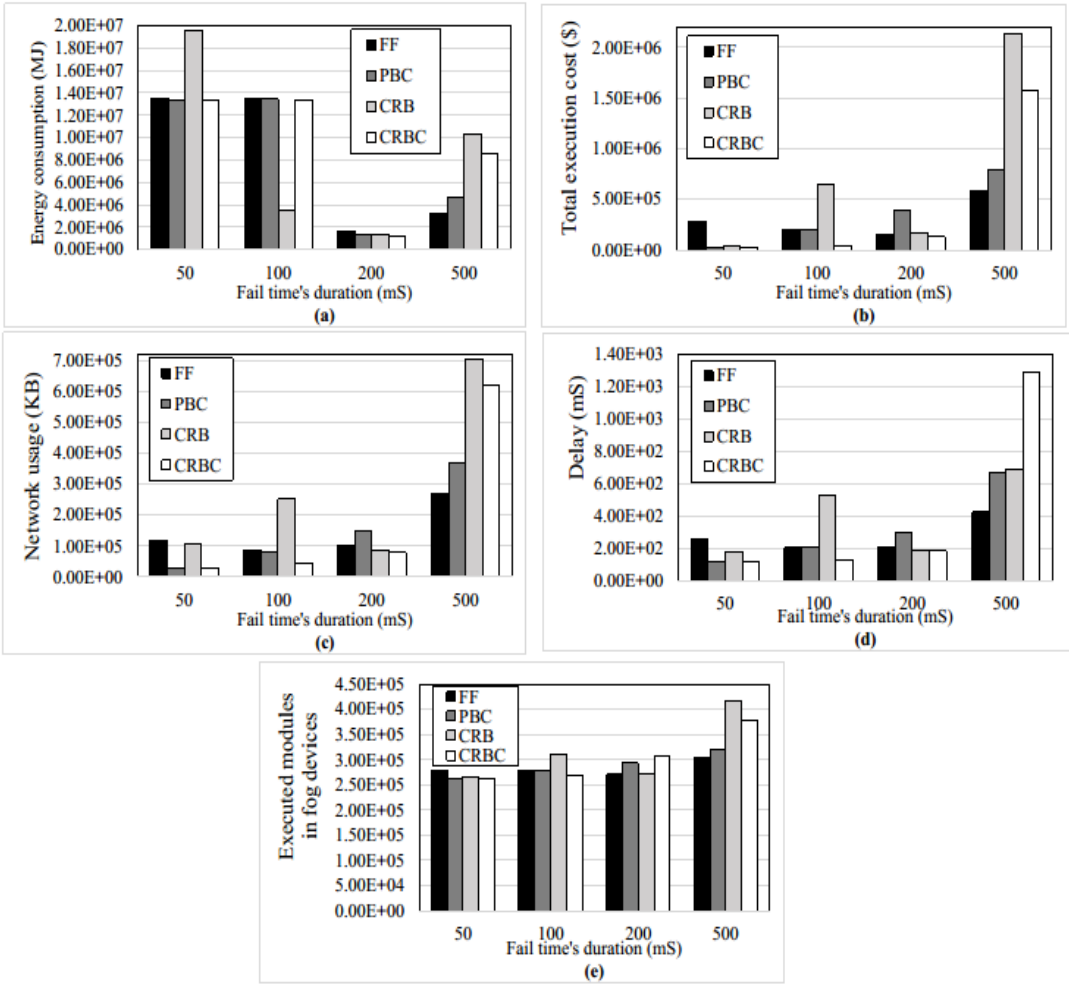
شکل ۱۰: تأثیر اندازه ماژول بر عملکرد

زمانی است که مدت زمان خرابی سیستم ۲۰۰ میلی ثانیه باشد که این بدان معنی است که مصرف انرژی در آن حداقل است. همانطور که در شکل ۱۱ (b) نشان داده شده است، وقتی که مدت زمان خرابی کمتر از ۲۰۰ ms باشد، زمان اجرا در روش CRBC کمینه است، و هنگامی‌که مدت زمان خرابی به ۵۰۰ ms افزایش یابد، هزینه اجرای روش های مبتنی بر تحمل اشکال افزایش می‌یابد و بدتر از First-fit می‌گردد که این بدان معنی است که روش First-fit نتیجه بهتری دارد. شکل ۱۱ (c) و (d) نشان می‌دهد زمانی که مدت زمان خرابی کمتر از ۲۰۰ ms باشد، میزان استفاده از شبکه و تاخیر در روش CRBC نتیجه بهتری نسبت به روش های دیگر دارد و به نظر می‌رسد چنانچه مدت زمان خرابی برابر ۵۰۰ میلی ثانیه باشد CRBC به درستی عمل نمی‌کند و نتایج مورد قبولی نخواهد داشت. همانطور که در شکل ۱۱ (e) نشان داده شده است، وقتی مدت زمان خرابی ۲۰۰ میلی ثانیه باشد، ماژول های اجرا شده در دستگاههای مه برای روش CRBC بهترین نتیجه را دارند.

با توجه به بحث فوق در شکل ۱۱ (الف) - (ه) به نظر می‌رسد که اگر مدت زمان خرابی ۲۰۰ میلی ثانیه باشد، نتایج حاصل از روش پیشنهادی ما، خروجی قابل اطمینان تری دارد. از این رو، به منظور طراحی سیستم در محیط مه، پیشنهاد می‌شود که سیستمی‌طراحی گردد که مدت زمان خرابی در آن کمتر از ۲۰۰ ms باشد.

۶- نتیجه گیری و کارهای آینده

زمانبندی مقاوم در برابر اشکال یک مسئله اساسی در محاسبات مه است. مقالات قبلی این موضوع را به طور دقیق مورد مطالعه قرار نداده‌اند. بنابراین، ما یک الگوریتم زمانبندی مقاوم در برابر اشکال برای ماژول‌های مختلف در مه را پیشنهاد می‌کنیم. در مدل ارائه شده، ماژول‌ها و دستگاه‌های مه به گروه‌های مختلف طبقه بندی می‌شوند و سپس با ترکیب آن با روش‌های چک پوینتینگ و نسخه اصلی-پشتیبان، مدل کارآمدی ایجاد می‌گردد. نتیجه شبیه سازی نشان می‌دهد که روش پیشنهادی باعث کاهش قابل توجه



شکل ۱۱: تأثیر مدت زمان خرابی بر عملکرد

خرابی در ماژول‌های وظیفه می‌گردد.

این روش تضمین می‌کند که انواع مختلف ماژول‌ها را می‌توان به دستگاه‌های مناسب اختصاص داد. با در نظر گرفتن نتایج شبیه سازی، معلوم می‌شود که روش پیشنهادی نتایج زمانبندی برتری نسبت به PBC و سایر روش ها ارائه می‌دهد.

جدول ۵: درصد بهبود CRBC نسبت به سایر روشهای مقایسه شده

سنجه	FF	PBC	CRB
انرژی مصرفی	۲۷	۱۲	۱۰
هزینه اجرای کل	۱۶	۶۶	۱۷
میزان استفاده از شبکه	۲۳	۴۶	۷.۸
تاخیر	۱۲	۳۸	۴.۵
ماژول‌های اجرا شده	۱۳	۴	۱۲

به منظور ارزیابی به روش شبیه سازی، مدل پیشنهادی با سه روش دیگر از نظر مصرف انرژی، تأخیر، هزینه اجرا، میزان استفاده از شبکه و تعداد ماژولهای اجرا شده، اجرا و مقایسه می‌شود. هنگامی‌که پارامترهای (دستگاه‌های مه، اندازه ماژول، احتمال خرابی، مدت زمان خرابی) به ترتیب (۵۲، ۱۰۰۰۰، ۰.۰۱، ۲۰۰) بودند، نتیجه نشان می‌داد که کل هزینه اجرا، انرژی مصرفی، تاخیر در حلقه کاربرد، میزان استفاده از شبکه و تعداد ماژولهای اجرا شده در دستگاه‌های مه در روش CRBC نسبت به الگوریتم‌های FF، PBC و CRB بهبود یافته تر است.

در مصرف انرژی، الگوریتم پیشنهادی ما نسبت بهFF، PBC، CRB به ترتیب ۲۷٪، ۱۲٪ و ۱۰٪ بهتر است، همچنین در مجموع هزینه اجرا، ۱۶٪، ۶۶٪ و ۱۷٪ در تأخیر حلقه کاربرد، ۱۲٪، ۳۸٪ و ۴.۵٪ همچنین در میزان استفاده از شبکه، ۲۳٪، ۴۶٪ و ۷.۸٪ و در آخر، تعداد ماژولهای اجرا شده در دستگاه‌های مه، ۱۳٪، ۴٪ و ۱۲٪ برتری دارد.

علاوه بر این، به نظر می‌رسد که اگر مدت زمان شکست ۲۰۰ میلی ثانیه باشد، نتایج حاصل از روش پیشنهادی نتایج بسیار معتبرتری به دست می‌آورد. به عبارت دیگر، در برنامه ما بهترین نتیجه را با مدت زمان شکست برابر با ۲۰۰ ms کسب کردیم.

به عنوان کار بعدی، بر روی روشهای موازنه بار، فرااکتشافی و آماری در زمانبندی مقاوم در برابر اشکال در محیط مبتنی بر مه توجه خواهیم نمود. و در ادامه می‌خواهیم برای بررسی قابلیت اطمینان سیستم، پیاده سازی واقعی این مسئله را در سیستم محاسبات مبتنی بر مه پیشنهاد کنیم.

مراجع

[1] A. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," Computer, vol. 49, pp. 112–116, Aug 2016.

[2] Bellendorf, J. Mann, Z. A. Classification of optimization problems in fog computing, Future Generation Computer Systems, January 2020.

[3] M. Mahmud and R. Buyya, Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit. 04 2018.

[4] H. Han, W. Bao, X. Zhu, X. Feng, and W. Zhou, "Fault-tolerant scheduling for hybrid real-time tasks based on cpb model in cloud," IEEE Access, vol. 6, pp. 18616– 18629, 2018.

[5] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for internet of things services," IEEE Internet Computing, vol. 21, pp. 16–24, Mar 2017.

[6] M. Hasan and M. S. Goraya, "Fault tolerance in cloud computing environment: A systematic survey," Computers in Industry, vol. 99, pp. 156 – 172, 2018. M.B.A. Haghighat, "Biometrics for Cybersecurity and Unconstrained Environments", Ph.D. Thesis, University of Miami, USA, 2016.

[7] D. Poola, M. A. Salehi, K. Ramamohanarao, and R. Buyya, "Chapter 15 – a taxonomy and survey of fault-tolerant workflow management systems in cloud and distributed computing environments," in Software Architecture for Big Data and the Cloud (I. Mistrik, R. Bahsoon, N. Ali, M. Heisel, and B. Maxim, eds.), pp. 285– 320, Boston: Morgan Kaufmann, 2017.

[8] J. Liu, P. Wang, J. Zhou, and K. Li, "Mctar: A multi-trigger checkpointing tactic for fast task recovery in mapreduce," IEEE Transactions on Services Computing, pp. 1–1, 2019.

[9] Y. Sharma, W. Si, D. Sun, and B. Javadi, "Failure-aware energy-efficient vm consolidation in cloud computing systems," Future Generation Computer Systems, vol. 94, pp. 620 – 633, 2019.

[10] G. Levitin, L. Xing, and L. Luo, "Joint optimal checkpointing and rejuvenation policy for real-time computing tasks," Reliability Engineering and System Safety, vol. 182, pp. 63 – 72, 2019.

[11] H. Yan, X. Zhu, H. Chen, H. Guo, W. Zhou, and W. Bao, "Deft: Dynamic faulttolerant elastic scheduling for tasks with uncertain runtime in cloud," Information Sciences, vol. 477, pp. 30 – 46, 2019.

[12] S. M. Abdulhamid and M. S. A. Latiff, "A checkpointed league championship algorithm-based cloud scheduling scheme with secure fault tolerance responsiveness," Applied Soft Computing, vol. 61, pp. 670 – 680, 2017.

[13] S. Chinnathambi, A. Santhanam, J. Rajarathinam, and M. Senthilkumar, "Scheduling and checkpointing optimization algorithm for byzantine fault tolerance in cloud clusters," Cluster Computing, Mar 2018.

[14] T. Tamilvizhi and B. Parvathavarthini, "A novel method for adaptive fault tolerance during load balancing in cloud computing," Cluster Computing, Jul 2017.

[15] S. Haider and B. Nazir, "Dynamic and adaptive fault tolerant scheduling with qos consideration in computational grid," IEEE Access, vol. 5, pp. 7853–7873, 2017.

[16] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang, and L. Liu, "Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds," IEEE Transactions on Parallel and Distributed Systems, vol. 27, pp. 3501–3517, Dec 2016.

[17] G. Yao, Y. Ding, and K. Hao, "Using imbalance characteristic for fault-tolerant workflow scheduling in cloud systems," IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 3671–3683, Dec 2017.

[18] Y. Ding, G. Yao, and K. Hao, "Fault-tolerant elastic scheduling algorithm for workflow in cloud systems," Information Sciences, vol. 393, pp. 47 – 65, 2017.

[19] V. K., S. M. D. Kumar, R. S., and V. K. R., "Cost and fault-tolerant aware resource management for scientific workflows using hybrid instances on clouds," Multimedia Tools and Applications, vol. 77, pp. 10171–10193, Apr 2018.

[20] H.-L. Li, J. Wu, Z. Jiang, X. Li, and X.-H. Wei, "A task allocation method for stream processing with recovery latency constraint," Journal of Computer Science and Technology, vol. 33, pp. 1125–1139, Nov 2018.

[21] G. Fan, L. Chen, H. Yu, and D. Liu, "Modeling and analyzing dynamic faulttolerant strategy for deadline constrained task scheduling in cloud computing," IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp. 1–15, 2018.

[22] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Yl¨a-J¨a¨aski, "Folo: Latency and quality optimized task allocation in vehicular fog computing," IEEE Internet of Things Journal, pp. 1–1, 2019.

[23] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "Dats: Dispersive stable task scheduling in heterogeneous fog networks," IEEE Internet of Things Journal, pp. 1–1, 2019.

[24] G. Zhang, F. Shen, N. Chen, P. Zhu, X. Dai, and Y. Yang, "Dots: Delay-optimal task scheduling among voluntary nodes in fog networks," IEEE Internet of Things Journal, pp. 1–1, 2019.

[25] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," IEEE Transactions on Industrial Informatics, vol. 14, pp. 4712–4721, Oct 2018.

[26] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energyaware load balancing and scheduling in smart factory," IEEE Transactions on Industrial Informatics, vol. 14, pp. 4548–4556, Oct 2018.

[27] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," IEEE Cloud Computing, vol. 4, pp. 26–35, March 2017.

[28] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," IEEE Transactions on Computers, vol. 65, pp. 3702–3712, Dec 2016.

[29] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "Debts: Delay energy balanced task scheduling in homogeneous fog networks," IEEE Internet of Things Journal, vol. 5, pp. 2094–2106, June 2018.

[30] SH. Ghanbari, M. Othman, M. Abu Bakar, "Multi-objective method for divisible load scheduling in multi-level tree network," Future Generation Computer Systems, vol.54,pp.132-143,2016.

[31] A. Sharif, M. Nickray, A, Shahidinejad, "Fault-tolerant with load balancing scheduling in a fog-based IoT application," IET Communications, vol. 14, pp. 2646-2657, Oct 2020.

[32] Y.-H. Gao, H.-D. Ma, and W. Liu, "Minimizing resource cost for camera stream scheduling in video data center," Journal of Computer Science and Technology, vol. 32, pp. 555–570, May 2017.

[33] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," Software: Practice and Experience, vol. 47, no. 9, pp. 1275–1296, 2017.

[34] D. Rahbari and M. Nickray, "Low-latency and energy-efficient scheduling in fogbased iot applications," Turkish Journal of Electrical Engineering and Computer Sciences, vol. 27, pp. 1406–1427, Feb 2019.

[35] Rehani, N., Garg, R. Meta-heuristic based reliable and green workflow scheduling in cloud computing. Int. J. Syst. Assur. Eng. Manag. 1–10, 2018.

پاورقی‌ها:

[1] Checkpoint-Restart and Primary Back up model with Classification

[2] Fog Computing

[3] Fault tolerance

[4] Generic view

[5] Processor view

[6] Transient

[7] Intermittent

[8] Permanent

[9] Crash

[10] Fail-stop

[11] Byzantine

[12] Out-of-memory

[13] File staging errors

[14] Uncaught exceptions

[15] Proactive

[16] Reactive

[17] Duplication

[18] Video surveillance/object tracking

[19] Pan, tilt, and zoom

[20] Computing-intensive modules

[21] Data-intensive modules

[22] Balanced modules

[23] Task Module Processing Time

[24] Metric