ISSN: 2745-3673
http://www.Rcsj.ir

مجله علمی پژوهش در علوم رایانه (سال پنجم)
شماره ۱۹ / پاییز ۱۳۹۹ / ص ۷۵- ۵۵

ارائه‌ی یک مکانیسم تحمل‌پذیر خطا در پردازش ابری بلادرنگ با استفاده از اعتباردهی به ماشین‌های مجازی

مازیار گرامی[۱]، نگار ورشوی[۲]

[۱] گروه کامپیوتر، دانشکده تحصیلات تکمیلی۲ ، دانشگاه آزاد اسلامی، واحد کرمانشاه، کرمانشاه، ایران.

[۲] گروه کامپیوتر، دانشکده تحصیلات تکمیلی۲ ، دانشگاه آزاد اسلامی، واحد کرمانشاه، کرمانشاه، ایران.

نام نویسنده مسئول:
مازیار گرامی

تاریخ دریافت: ۱۳۹۹/۶/۲۷
تاریخ پذیرش: ۱۳۹۹/۹/۱

چکیده

رایانش ابری یک مفهوم محاسباتی مهم است که سرویس‌هایی را با هزینه‌های کم برای کاربران فراهم می‌کند. تحمل‌پذیری خطا و قابلیت اطمینان مفاهیم مهمی هستند که به معنای تولید پاسخ‌های مناسب در حضور مؤلفه‌های خطاکار هستند. به‌منظور دستیابی به قابلیت اطمینان در محاسبات بلادرنگ، درخواست‌ها برای تحمل‌پذیری خطا افزایش یافته است. در محاسبات، بیشتر کارهای بلادرنگی که در رایانش ابری ثبت می‌شود، پردازش‌ها به‌صورت از راه دور است؛ بنابراین به علت از دست رفت کنترل در گره‌های محاسباتی، امکان به وجود آمدن خطا افزایش می‌یابد. تکنیک‌های تحمل‌پذیری خطا برای مقابله با چنین مواردی قرار داده‌شده‌اند. قابلیت اطمینان ماشین‌های مجازی پس از گذشت دوره‌های زمانی مختلف تغییر خواهد کرد. در این مطالعه ما با خوشه‌بندی ماشین‌های مجازی بر اساس نحوه‌ی کارکرد آن‌ها و بروز خطا در آن‌ها، قابلیت اطمینان آن‌ها را مقایسه خواهیم کرد. پس از مشخص شدن قابلیت اطمینان و خوشه‌بندی آن‌ها در سه خوشه‌ی مختص به کارهای بلادرنگ سخت، بلادرنگ نرم و کارهای غیر بلادرنگ، علاوه بر اعمال تحمل‌پذیری خطا، اجرای کارها در زمان پیش از مهلت نهایی آن‌ها نیز برآورده خواهد شد. پس از شبیه‌سازی روش پیشنهادی در محیط کلود سیم مشخص شد که روش پیشنهادی در برآورده سازی مهلت‌های زمانی پیشرفت چشمگیری داشته است.

واژگان کلیدی: رایانش ابری، تحمل‌پذیری خطا، تخصیص منابع، ابر بلادرنگ.

ISSN: 2745-3673
http://www.Rcsj.ir

مقدمه

یکی از طعنه‌آمیزترین نکات فناوری اطلاعات این است که باوجوداینکه قدرت کامپیوترها بسیار بالا رفته اما به نظر می‌رسد کار با کامپیوترها کندتر شده و کامپیوترها سرعت سابق را ندارند! دلیل این موضوع این است که هرروز سیستم‌عامل‌ها و بسته‌های نرم‌افزاری پیچیده می‌شوند؛ اما امروزه با استفاده از فناوری رایانش ابری[1] دیگر نگران نخواهیم بود، زیرا با استفاده از ایده‌ی پردازش ابری آن را بر روی سروری در اینترنت نگهداری می‌کنیم و از هر جای دنیا با استفاده از یک کامپیوتر شخصی ساده، به پردازش و آن می‌پردازیم [1,2]. با نیاز روزافزون کاربران به منابع مختلف رایانش ابری به‌عنوان یکی از فنّاوری، امروزه سرویس‌های مبتنی بر آن به‌سرعت در حال پیشرفت هستند. فراهم آورندگان سرویس‌های رایانش ابری[2]، یک نرم‌افزار گران‌قیمت را که نیازمند توان محاسباتی بالا و حافظه بالایی است را از طریق ابر در اختیار متقاضیان قرار می‌دهند. اهمیت سرویس ابر این است که تمام منابع و سرویس‌های مختلف در اختیار فراهم‌کنندگان بوده و مشتریان نهایی نیازی به اطلاعات تخصصی و پرداخت هزینه نصب این‌گونه سرویس‌ها ندارند؛ بنابراین مشتریان تنها برای استفاده از سرویس هزینه پرداخت می‌کنند [3,4].

با توجه به این موضوع، بدیهی است که سرویس‌های مبتنی بر رایانش ابری به یکی از محبوب‌ترین سرویس‌ها در فضای مجازی بدل شده‌اند؛ اما این سرویس‌ها دارای چالش‌های مختلفی ازجمله امنیت و حریم خصوصی[3]، هزینه‌ها و نحوه‌ی تحویل سرویس‌ها[4]، قابلیت حمل[5]، قابلیت اطمینان و در دسترس بودن[6]، کارایی و هزینه‌ی پهنای باند هستند [5].

قابلیت اعتماد و در دسترس بودن منابع در رایانش ابری از طریق قرارداد سطح سرویس تضمین می‌گردد که به‌موجب آن، فراهم‌کننده‌ی سرویس تعهد می‌دهد که قابلیت اعتماد سیستم او بسیار بالا باشد [6]. عدم قطعیت در تخصیص منابع در رایانش ابری منجر به ایجاد پتانسیل بروز خطاهای[7] متعدد در هنگام اجرای وظایف در سرویس‌های ابری شده است. بسیاری از سیستم‌های موجود در زمره‌ی سیستم‌های بلادرنگ قرار می‌گیرند که در آن‌ها ممکن است پاسخگویی به‌موقع حیاتی باشد [7-10].

در سیستم‌های بلادرنگ وظیفه‌های مربوط به درخواست‌ها باید در کمتر از زمان مشخص‌شده‌ای اجرا شوند. ازجمله کاربردهای این نوع سیستم‌ها می‌توان به سیستم‌های حساس پزشکی، برخی سیستم‌های نظامی، کنترل سیستم‌های نیروگاه‌های هسته‌ای و ... اشاره کرد [11]. در این‌گونه سیستم‌ها باید پاسخ درخواست‌ها حتماً در زمان مشخصی ارسال گردد و در غیر این صورت سیستم دچار اختلال شده و حتی در کاربردهای حساس می‌تواند منجر به یک فاجعه گردد [12]. ازاین‌روست که نوع پیاده‌سازی، کنترل زمان پاسخ‌گویی، سربار و نحوه الگوریتم‌های پیاده‌سازی شده و همچنین بستر سیستم‌عامل و سخت‌افزار حائز اهمیت فراوان است [13].

در این مطالعه، ما بر آن هستیم که پس از مطالعه‌ی قابلیت اطمینان و در دسترس بودن در رایانش ابری، روشی را برای افزایش قابلیت اطمینان و افزایش تحمل‌پذیری این سرویس‌ها ارائه دهیم.

در واقع هدف اصلی ما در این مقاله، ارائه‌ی یک مکانیسم تخصیص وظایف برای پردازش به‌موقع وظایف در محیط ابر بلادرنگ است.

اگرچه رایانش ابری به‌صورت گسترده توسط صنایع مورداستفاده قرارگرفته است، اما هنوز هم چالش‌های بزرگی همچون تحمل‌پذیری خطا، زمان‌بندی کارها، امنیت و ... در آن وجود دارد. تحمل‌پذیری خطا یکی از بزرگ‌ترین مسائل در این زمینه است که به معنای بکار گیری تمامی تکنیک‌های موردنیاز برای آنکه یک سیستم تمامی خطاهای نرم‌افزاری به وجود آمده در آن را پس از توسعه پشتیبانی کند. هنگامی‌که یک خطا در سیستم اتفاق می‌افتد، این تکنیک‌ها مکانیسمی برای سیستم نرم‌افزاری

[1] Cloud Computing
[2] Service Providers
[3] Security and privacy
[4] Costs and delivery
[5] Portability
[6] Reliability and Avialibility
[7] Faults

ISSN: 2745-3673
http://www.Rcsj.ir

فراهم می‌کنند تا از وقوع سقوط[8] در سیستم ممانعت کنند [14,15]. مزیت اصلی پیاده-سازی تحمل‌پذیری خطا در رایانش ابری شامل بازگشت از شکست‌ها، هزینه‌ی پایین‌تر و بهبود کارایی است. در این زیر بخش سعی می‌کنیم که در مورد چالش‌های تحمل‌پذیری خطا و شناسایی ابزارهای و تکنیک‌های مختلف برای تحمل‌پذیری خطا بحث کنیم؛ اما پیش از آنکه این موارد موردبحث قرار گیرند، تفاوت واژه‌های اشتباه[9]، خطا[10] و شکست[11] موردبررسی قرار خواهند گرفت.

خطا: یک گام، فرآیند یا تعریف داده‌ی ناصحیح در یک برنامه‌ی کامپیوتری که باعث می‌شود کامپیوتر به یک حالت ناخواسته و پیش‌بینی‌نشده برود. این موضوع درواقع ضعف ذاتی طراحی یا پیاده‌سازی است. یک خطا ممکن است برای مدت‌ها در سیستم پنهان باشد. یک خطا ممکن است درنهایت منجر به یک شکست گردد [16].

- باگ‌های نرم‌افزاری
- از بین رفتن داده‌ها در هنگام انتقال

شکست: عدم توانایی یک سیستم یا مؤلفه در اجرای توابع موردنیاز آن برای عملکردی خاص [17].

اشتباه: یک اختلاف بین یک مقدار یا شرط محاسبه‌شده، مشاهده‌شده یا اندازه‌گیری شده و مقدار و شرط صحیحی که به‌صورت تئوریک مشخص‌شده است.

هنگامی‌که یک خطا اتفاق می‌افتد، باعث ایجاد اشتباه جانبی می‌گردد؛ هنگامی‌که این اشتباه بر روی سرویس‌ها تأثیر می‌گذارد، یک شکست رخ می‌دهد؛ بنابراین، اشتباه در سیستم و شکست در سرویس آشکار می‌شود[17].

در رایانش ابری خطاهای مختلفی وجود دارد. بر اساس سیاست‌های تحمل‌پذیری خطا، تکنیک‌های مختلفی را می‌بایست مورداستفاده قرارداد؛ بنابراین در ابتدا انواع خطاها را موردبررسی قرار خواهیم داد [18].

- تحمل‌پذیری خطای واکنشی

سیاست‌های واکنشی تحمل‌پذیری خطا، اثر شکست‌ها را در هنگامی‌که خطاها در زمان اجرای برنامه به وجود می‌آیند را کاهش می‌دهد. برای این دسته از سیاست‌ها، تکنیک‌های مختلفی ازجمله قرار دادن نقطه‌ی بازگشت، شروع مجدد، تلاش مجدد و ... وجود دارد [18].

- تحمل‌پذیری خطای فعال

این نوع سیاست‌ها به‌منظور اجتناب از بازگشت از خطاها، اشتباهات و شکست‌ها با پیش‌بینی آن‌ها و جایگزینی آن‌ها با مؤلفه‌هایی که به‌درستی کار می‌کنند برخی از تکنیک‌هایی که در این زمینه انجام‌گرفته است عبارت است از مهاجرت و جوان‌سازی[12]. [18].

چالش‌های پیاده‌سازی تحمل‌پذیری خطا در رایانش ابری

فراهم‌سازی تکنیک‌های تحمل‌پذیر خطا، نیازمند دقت و تحلیل‌های پیچیده است زیرا [19]:

- نیاز به پیاده‌سازی خودکار تکنیک‌های تحمل‌پذیر خطا برای چندین نمونه از یک برنامه‌ی در حال اجرا بر روی چندین ماشین مجازی دارد.
- برای ایجاد یک سیستم تحمل‌پذیر خطا، نیازمند تکنولوژی‌های مختلفی است که ممکن است مربوط به فراهم آورندگان سرویس‌های مختلفی باشد که با یکدیگر رقابت دارند.
- برای تضمین قابلیت اعتماد بالا و در دسترس بودن، چندین فراهم آورنده‌ی سرویس با پشته‌های نرم‌افزاری وابسته باید با یکدیگر در ارتباط باشند
- می‌بایست یک سیستم تحمل‌پذیر خودکار و یکپارچه در بین ابرهای مختلف به کار گرفته شود.

[8] Crash
[9] Error
[10] Fault
[11] Failure
[12] Rejuvenation

مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۱۹، پاییز ۱۳۹۹، ص ۷۵- ۵۵

ISSN: 2745-3673
http://www.Rcsj.ir

انواع تکنیک‌های ارائه‌شده برای رایانش ابری تحمل‌پذیر خطا

در جدول ۱، تکنیک‌های مختلف تحمل‌پذیری خطا آمده است [20].

جدول ۱ تکنیک‌های بکار گرفته‌شده برای تحمل‌پذیری خطا در رایانش ابری [20]

تکنیک بکار گرفته‌شده	سیاست	محیط	سیستم
بهبود خودکار، مهاجرت کار، تکثیر برداری	واکنشی/ فعال	ماشین مجازی	HAProxy
بازرسی[۱۳]	واکنشی	ماشین مجازی	SHelp
بازرسی، تلاش مجدد، بهبود خودکار	واکنشی/ فعال	ماشین مجازی	Assure
مهاجرت کار، تکثیر برداری	واکنشی/ فعال	محیط ابری	Hadoop
تکثیر برداری، ثبت مجدد وظایف	واکنشی/ فعال	محیط ابری	AmazonEC2

بر این اساس سیاست بکار گرفته‌شده توسط ما نیز از نو واکنشی/فعال بوده زیرا در هنگام اجرا به‌طور منظم وضعیت سیستم را چک کرده و از طرف دیگر در قبال خطاهای به وجود آمده واکنش نشان می‌دهد؛ بنابراین در ابتدا ماشین‌های مجازی را به دسته‌های مختلف تقسیم کرده تا از وقوع خطا جلوگیری کند و قابلیت اطمینان سیستم را با تخصیص کارهای بحرانی به دسته‌هایی که دارای قابلیت اطمینان بالا هستند افزایش دهد. در ادامه پس از وقوع هر نوع خطایی، با ثبت کردن مجدد آن کار، برای اجرای مجدد آن تلاش می‌کند.

مرور ادبیات پیشین

در این بخش به بررسی چند نمونه از معماری‌های ارائه‌شده که تحمل‌پذیری خطا را در رایانش ابری فراهم می‌آورند موردبررسی قرارداده‌ایم. باید گفته شود که این معماری‌ها را به‌صورت طبقه‌بندی‌شده و بر اساس سیاست بکار گرفته‌شده (فعال/واکنشی) تقسیم‌بندی کرده‌ایم. همچنین، باید گفته شود که در ساختار بعضی از معماری‌ها، بیش از یک تکنیک بکار گرفته‌شده است.

پیش از ادامه‌ی کار باید متذکر شده که دو معیار برای اندازه‌گیری قابلی تحمل‌پذیری خطا در رایانش ابری ارائه‌شده است که عبارت‌اند از PRO و RTO. PRO میزان داده‌هایی است که در زمان بروز خطا از بین رفته‌اند. RTO حداقل زمان موردنیاز برای ریکاوری است [21]. همان‌طور که پیش‌ازاین نیز گفته شد، در سیاست فعال، احتمال وقوع خطا تخمین زده می‌شود و سپس از بروز آن جلوگیری می‌شود.

معماری فعال

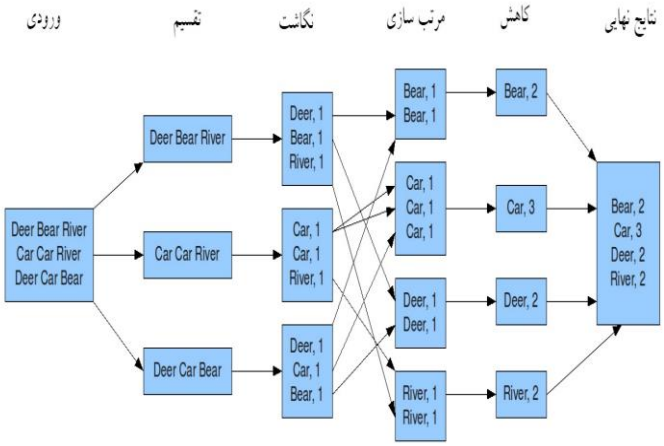
معماری نگاشت-کاهش[۱۴]

درزمینه‌ی معماری‌های فعال، معروف‌ترین این معماری‌ها نگاشت-کاهش است. نگاشت-کاهش کارها را به قطعات کوچک‌تری تقسیم می‌کند. این قطعات کوچک، کار را بر روی ماشین‌های مختلف به‌صورت موازی اجرا می‌کنند. پس از اتمام کار، نتایج نهایی با ترکیب خروجی‌ها با یکدیگر تولید می‌شود. واضح است که اگر اجرای بخش‌های کوچک در یک ماشین دچار مشکل شود، آن را به ماشین دیگری ارسال می‌کند تا خروجی را بدون اشکال اجرا کند. این معماری، اصولاً برای پردازش کلان‌داده‌ها[۱۵] در محیط ابر مورداستفاده قرار می‌گیرد. معماری گفته‌شده احتمال ذخیره‌سازی حجم بالایی از داده‌ها و پردازش موازی آن‌ها فراهم می‌کند[22,23].

---

[13] Checkpoint
[14] Map-Reduce
[15] Big Data

ISSN: 2745-3673
http://www.Rcsj.ir



شکل .Error! No text of specified style in document۱ تحمل‌پذیری خطا با استفاده از چارچوب نگاشت-کاهش

معماری FT-Cloud

کاربردهای رایانش ابری، معمولاً در مقیاس‌های بزرگی و با پیچیدگی‌های خاص خود فراهم‌شده‌اند؛ اما متأسفانه قابلیت اطمینان آن‌ها هنوز مناسب نیست. این چارچوب که FT-Cloud نام دارد، بر مبنای رتبه‌بندی است. این چارچوب شامل دو فاز رتبه‌بندی و تحمل‌پذیری خطا است. این معماری در صورت به وجود آمدن خطا، مکانیسم‌های مربوطه را بکار می‌گیرد. ساختار این معماری در شکل ۲ آمده است.

همان‌طور که در شکل ۲ نشان داده‌شده است، این معماری شامل ۲ بخش است: (۱) رتبه‌بندی و (۲) انتخاب تحمل‌پذیری خطا. رویه‌ی کار این معماری به‌صورت زیر است[24]:

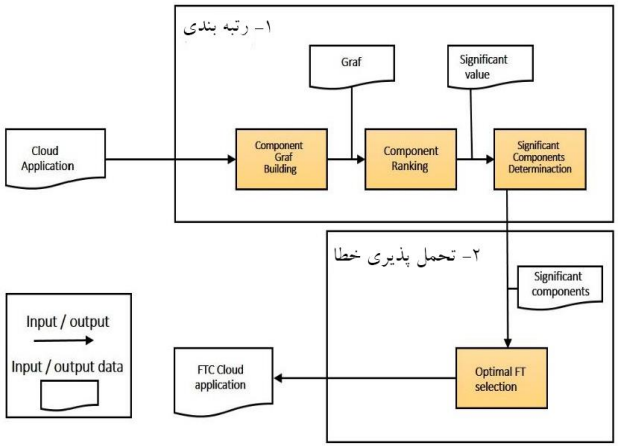
۱) طراحی اولیه‌ی برنامه‌ی کاربردی ابری، به‌صورت مجموعه‌ای از مؤلفه‌ها تحویل ابر داده می‌شود.

۲) یک الگوریتم رتبه‌بندی برای محاسبه‌ی مقدار مؤلفه‌های ابری بکار گرفته می‌شود. بر پایه‌ی این مقادیر، مؤلفه‌ها می‌توانند رتبه‌بندی شوند.

۳) مؤثرترین مؤلفه در برنامه‌ی کاربردی ابری مشخص می‌شود.

۴) کارایی استراتژی‌های مختلف تحمل‌پذیری خطا محاسبه‌شده و بهترین استراتژی بر رأی هر مؤلفه انتخاب می‌شود.

بر این اساس مؤلفه‌ها اجراشده و نتایج بازگردانده می‌شوند.



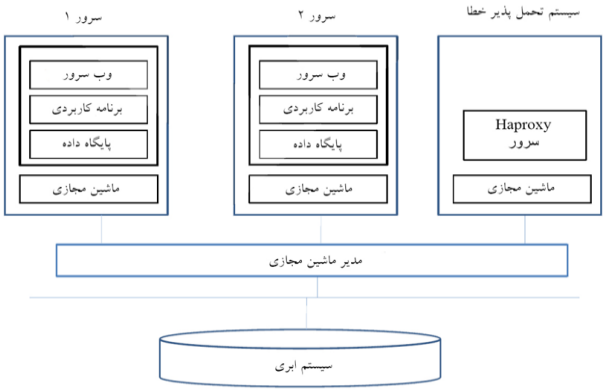
شکل .Error! No text of specified style in document۲ تحمل‌پذیری خطا با معماری FT-Cloud [24]

ISSN: 2745-3673
http://www.Rcsj.ir

معماری واکنشی

معماری Haproxy

معماری HAProxy، اولین معماری واکنشی است که تولیدشده است. در این معماری، مهاجرات کارها و تکنیک تکثیر برداری مورداستفاده قرارگرفته است. در این معماری، یکی از ماشین‌ها نقش افزونه برای سایر ماشین‌ها را دارد. به‌عبارت‌دیگر، اگر سرور اول از کار بیفتد، سرور دوم از تمامی عملیات سیستم بدون به وجود آمدن هرگونه وقفه‌ای، پشتیبان گیری خواهد کرد. اگر دومین سرور دچار مشکل شود، سیستم اول بدون هیچ‌گونه وقفه‌ای این عملیات را در دست خواهد گرفت. سرور Haproxy، مسئولیت مدیریت این وظایف را بر عهده دارد. همواره یکی از سرورها در سیستم فعال است، اما ممکن است و به‌صورت لحظه‌ای ممکن است در برخی مواقع هر دو سرور نیز فعال نباشند.



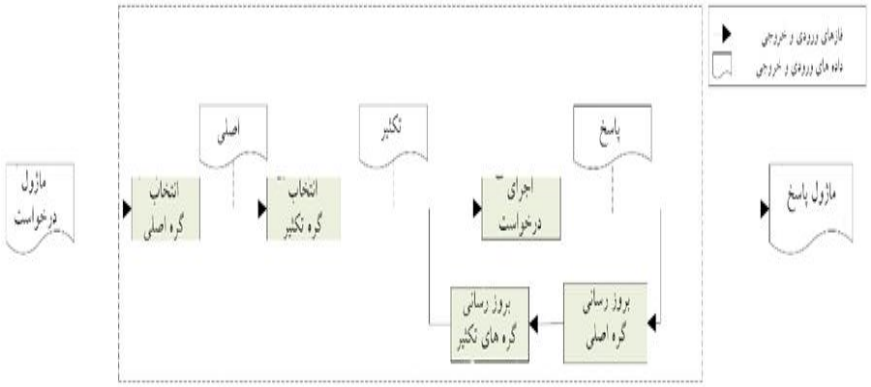
شکل .Error! No text of specified style in document۳ **تحمل‌پذیری خطا با معماری Haproxy [25]**

همچنین این امکان وجود دارد که هردو سرور نیز در سیستم فعال باشند، اما این امکان وجود ندارد که هر دو سرور به‌طور هم‌زمان غیرفعال باشند. در هنگام به وجود آمدن هر نوع خطا، عمل مهاجرت کار به سرور دیگر، مورداستفاده قرار خواهد گرفت [25]. در شکل ۳، این معماری نشان داده‌شده است.

معماری BFT-Cloud

معماری BFT-Cloud نیز از دیگر روش‌های واکنشی است که در [26] ارائه‌شده است. تکنیک تکثیر در این ساختار مورداستفاده قرارگرفته است. هنگامی‌که یک درخواست وارد سیستم رایانش ابری می‌شود، درخواست باید در گره‌های مختلف پیاده‌سازی شود. یکی از گره‌ها به‌عنوان گره‌ی اصلی شناخته‌شده و گره‌ی دیگر به‌عنوان گره‌های پشتیبان گیری شناخته می‌شوند. در هنگام اجرای درخواست، تمامی برنامه‌ها به‌صورت محلی بر روی ماشین اجرا می‌شوند. اگر نتایج اجرا بر ری گره‌های پشتیبان و گره‌ی اصلی مشابه باشد، نتیجه نیز صحیح بوده و به ماژول‌های درخواستی پاسخ می‌دهد. اگر یکی از پشتیبان‌ها یا گره‌ی اصلی دچار شکست گردد، پاسخ‌های مختلفی نسبت به یکدیگر خواهند داشت. در چنین شرایطی، آن گره به‌عنوان گره‌ی دارای خطا شناخته‌شده و عملیات ریکاوری باید بر روی آن انجام گیرد. اگر گره‌ای که در آن خطابه وجود آمده است یکی از گره‌های پشتیبان باشد، باید با یکی از گره‌های مناسب جایگزین گردد. ساختار اجرایی این معماری همانند شکل ۴ است.

ISSN: 2745-3673
http://www.Rcsj.ir

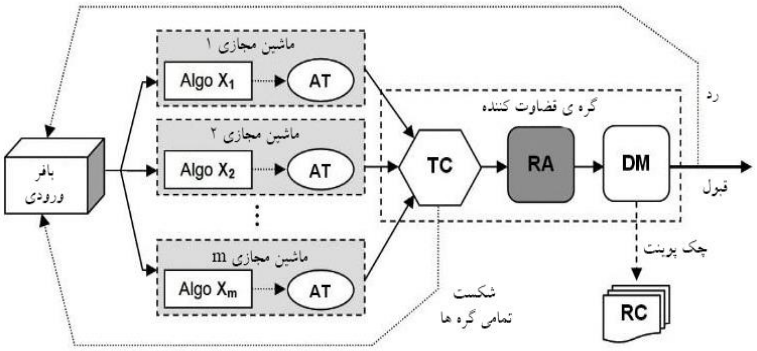


شکل ۴ تحمل‌پذیری خطا با معماری BFT-Cloud [26]

معماری AFTRC

باید اذعان داشت که بسیاری از کارهایی که در سیستم‌های ابری ثبت نمی‌شوند از نوع کارهای بلادرنگ هستند؛ بنابراین ازیک‌طرف این سیستم‌ها به‌شدت به زمان و پاسخ سیستم حساس هستند. به همین دلیل، اگر سیستم پس از زمان تعیین‌شده پاسخ دهد، نتایج آن ارزشی نخواهد داشت. از طرف دیگر تکنیک‌های تحمل‌پذیر خطا، باعث افزایش زمان پاسخ سیستم و تأخیر در تولید خروجی می‌شوند.

با ورود کارهای ورودی به بافر، تکثیرهای مختلف برای اجرا بر روی ماشین‌های مجازی تولید می‌شوند. گام بعدی با اعمال سیاست مهاجرت کار و انتقال به ماشین‌های مجازی دیگر در سیستم انجام می‌شود. کارها بر اساس الگوریتم‌های بلادرنگ مختلف در ماشین‌های مجازی مختلف اجرا می‌شوند. همان‌طور که در شکل ۵ نیز نشان داده‌شده است، ماژولی بانام AT وجود دارد: آزمایش مقبولیت[16] دقیقاً پس از خروجی هر الگوریتم در تمام ماشین‌های مجازی که وظیفه‌ی آن‌ها تحمل‌پذیری خطاست انجام می‌شود. به‌منظور چک کردن خروجی، به ماژول TC ارسال می‌شوند. در گام بعدی، ماژول TC به چک کردن زمان پاسخ کارها می‌پردازد... ماژول RA قابلیت اطمینان ماشین‌های مجازی بر اساس دو پارامتری که در مورد آن‌ها صحبت شد، ارزیابی می‌گردند. وظیفه‌ی دوم ماژول RA آن است که حدود حداقلی و حداکثری را برای قابلیت اطمینان تعیین می‌کند و اگر قابلیت اطمینان یک ماشین مجازی از حد آستانه‌ای کمتر بود، آن گره حذف می‌شود. در ادامه، ماژول DM که تصمیم‌گیرنده است، بر اساس قابلیت اطمینان چرخه‌ی محاسبات، خروجی نهایی را بر اساس این مکانیسم تولید می‌کند. ماژول RC نیز یک کش برای ذخیره‌سازی چک پوینت‌ها است [27].



شکل ۵Error! No text of specified style in document. تحمل‌پذیری خطا با معماری AFTRC [27]

[16] Acceptance test

ISSN: 2745-3673
http://www.Rcsj.ir

روش پیشنهادی

در این بخش به‌طورکلی به بررسی مواد و روش‌ها و روش پیشنهادی و به طور جزئی به بررسی جامعه‌ی مورد مطالعه، نمونه ی مورد مطالعه، نوع پژوهش، ابزارهای جمع‌آوری اطلاعات، متغیرهای موردبررسی، الگوریتم‌ها و فلوچارت‌های روش پیشنهادی و درنهایت روش پیشنهادی خواهیم پرداخت.

الگوریتم

رویه‌ی مدل پیشنهادی ما در غالب الگوریتم به‌صورت زیر است:

۱- شروع

۲- ماشین‌های مجازی را به تعداد n مورد تولید کنید.

۳- با استفاده از توزیع پواسون خطاهایی را برای برخی از ماشین‌های مجازی در نظر بگیر.

۴- وظایف را با استفاده از توزیع نرمال تولید کن

۵- به ازای اجرای هر m وظیفه کارهای ۶ تا ۱۱ را انجام بده

۶- ماشین‌هایی که دچار خطا شده‌اند را از لیست ماشین‌های تحت تعمیر قرار بده و ماشین‌هایی که به‌درستی کار می‌کنند را در لیست ماشین‌های مجازی سالم قرار بده

۷- بر اساس میزان کارهای در دسترس، نسبت هر یک از کارهای معمولی، بلادرنگ سخت و بلادرنگ نرم را حساب کن و خوشه‌بندی ماشین‌های مجازی را بر اساس اعتبار آن‌ها و میزان موردنیاز انجام بده.

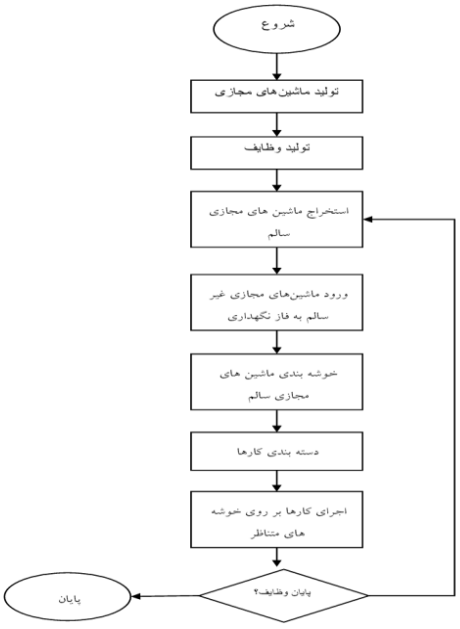
۸- کارهای مربوط به این دوره‌ی زمانی را بر اساس میزان تأخیر آن‌ها به سه دسته‌ی کارهای معمولی، کارهای بلادرنگ نرم و کارهای بلادرنگ سخت تقسیم کن

۹- کاره‌های مربوط به هر خوشه را بر روی ماشین‌های مجازی خوشه‌ی متناظر اجرا کن

۱۰- درصورتی‌که دوره‌ی تعمیر ماشین‌مجازی به اتمام رسیده است، آن را در لیست ماشین‌های مجازی سالم قرار بده.

۱۱- به ماشین‌های مجازی که در این دوره‌ی زمانی به‌درستی کارکرده‌اند اعتبار بده و از ماشین‌های مجازی که در این دوره دچار خطا شده‌اند اعتبار کاهش بده

فلوچارت روش پیشنهادی در شکل ۶ نشان داده‌شده است:



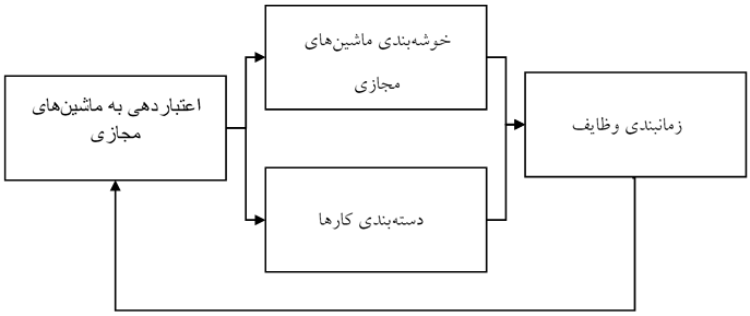
شکل* Error! No text of specified style in document. ۶ *فلوچارت روش پیشنهادی

مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۱۹، پاییز ۱۳۹۹، ص ۷۵- ۵۵

ISSN: 2745-3673
http://www.Rcsj.ir

ابزار تجزیه‌وتحلیل داده‌ها

همان‌طور که تا بدین جای کار بحث شده است، الگوریتم پیشنهادی ما با استفاده از یک مکانیسم اعتبار دهی، سعی بر پیاده‌سازی یک روش کارا برای افزایش تحمل‌پذیری خطا در محیط رایانش ابری داشته است و درعین‌حال با در نظر گرفتن لختی هر یک از کارهای رسیده، سعی در اجرای کارها در زمان از پیش تعیین‌شده دارد. بنابراین این روش به‌طور کارا می‌تواند در ابرهای بلادرنگ که کارهای رسیده از کاربران به‌صورت بلادرنگ نیز می‌باشند، مورداستفاده قرار گیرد. الگوریتم پیشنهادی ما به نحوی یک الگوریتم تکاملی است که پس از گذشت زمان با گرفتن بازخورد از دوره‌های زمانی پیشین، تکامل یافته و ماشین‌های مجازی بر اساس کارایی خود، دریکی از خوشه‌های با اطمینان بسیار بالا، با اطمینان بالا و با اطمینان کمتر تقسیم می‌شوند تا کارهای بلادرنگ سخت، بلادرنگ نرم و کارهای معمولی به ترتیب به آن‌ها اختصاص یابند. این الگوریتم دارای ماژول‌های مختلفی است که در شکل ۷ نشان داده‌شده است. در ادامه در مورد هر یک از این ماژول‌ها به‌طور مفصل بحث خواهیم نمود.



شکل .Error! No text of specified style in document۷ ماژول‌های مختلف و ترتیب و نحوه‌ی اجرای هر یک

مؤلفه‌ی موجود

مؤلفه اعتبار دهی به ماشین‌های مجازی

این مؤلفه وظیفه‌ی افزایش و کاهش اعتبار در ماشین‌های مجازی را بر عهده دارد. درواقع وظایف اصلی آن به‌صورت زیر است:

- ✓ دادن اعتبار به ماشین‌های مجازی بر اساس کارکرد آن‌ها در دوره‌ی زمانی پیشین. بر این اساس پس از انتهای هر دوره ی زمانی، به هر یک از ماشین‌های مجازی که بدون خطا کارکرده‌اند یک امتیاز مثبت داده می‌شود و از ماشین‌های مجازی که دچار خطا شده‌اند یک امتیاز کاسته می‌شود.
- ✓ مرتب‌سازی ماشین‌های مجازی به‌صورت نزولی بر اساس میزان اعتبار آن‌ها (اعتبار در ابتدا برای همه ماشین‌های مجازی ۱ در نظر گرفته می‌شود و با گذشت زمان و گرفتن بازخورد از محیط، به مقدار آن‌ها کم یا اضافه خواهد شد).

مؤلفه اختصاص ماشین‌های مجازی به خوشه‌ها

همان‌طور که پیشتر نیز گفته شد، تمامی ماشین‌های مجازی در سه خوشه‌ی مختلف بانام‌های اطمینان بسیار بالا، اطمینان بالا و اطمینان کمتر بر اساس اعتبارشان تقسیم می‌شوند. تعداد ماشین‌های مجازی موجود در هر خوشه باید به تعداد کارهای موجود در دسترس باشد، بنابراین باید مو لف‌های وجود داشته باشد که ماشین‌های مجازی را بر اساس اعتبارشان و با توجه به میزان کارهای در دسترس تقسیم‌بندی کند. بر این اساس، میزان زمان اجرای کارها نیز با کاهش اندازه‌ی صف ایجادشده برای هر خوشه و درنتیجه میزان زمان اجرای کل، کاهش خواهد یافت. بنابراین در این مؤلفه کارهای زیر انجام می‌شود:

- اندازه‌گیری تعداد دستورالعمل‌های موجود در صف از هر یک از انواع بلادرنگ سخت، بلادرنگ نرم و کارهای معمولی
- محاسبه‌ی درصد هر یک از انواع دستورالعمل‌های بلادرنگ سخت، بلادرنگ نرم و کارهای معمولی

ISSN: 2745-3673
http://www.Rcsj.ir

تخصیص ماشین‌های مجازی بر اساس تناسب محاسبه‌شده. به این‌گونه که تعداد ماشین‌های متناسب با هر دسته به آن دسته اختصاص داده‌شده و در اجرای این کار، اعتبار ماشین‌های مجازی نیز در نظر گرفته‌شده است. به این صورت که ماشین‌های مجازی که در ابتدای لیست مرتب‌شده قرار می‌گیرند و درنتیجه دارای اعتبار بیشتری هستند، به خوشه‌ی ماشین‌های مجازی با اطمینان بسیار بالا تخصیص داده خواهند شد.

مؤلفه دسته‌بندی کارها

این مؤلفه وظیفه‌ی تقسیم‌بندی کارها به سه نوع بلادرنگ سخت، بلادرنگ نرم و وظایف معمولی را دارا است. این مؤلفه این مهم را بر اساس میزان لختی هر یک از کارها انجام می‌دهد.

مؤلفه زمان‌بندی

این مؤلفه مسئولیت اجرای کارها را بر عهده دارد. بر این اساس هر یک از کارهای سه‌گانه (بلادرنگ سخت، بلادرنگ نرم و کارهای معمولی) به ماشین‌های مجازی خوشه‌ی متناظر خود (با اطمینان بسیار بالا، با اطمینان بالا و با اطمینان کم) تخصیص داده‌شده تا هر یک از کارها در زمان مقررشده‌ی خود با کمترین احتمال خطا اجرا شوند.

کلاس‌های نوشته‌شده برای این مطالعه

روش پیشنهادی با استفاده از شبیه‌ساز کلودسیم[17] و با استفاده از زبان برنامه‌نویسی جاوا شبیه‌سازی و مورد ارزیابی قرار گرفت. بدین منظور کلاس‌هایی نوشته‌شده و کلاس‌هایی دوباره‌نویسی شده است. در این بخش کلاس‌های مختلف نوشته‌شده به همراه قسمتی از کدهای آن‌ها آمده است.

در کلودسیم، برای موجودیت‌هایی که در داخل کلود وجود دارند (ازجمله وظایف (cloudlet)، دیتاسنتر، واسط[18]، ماشین‌های مجازی و موارد دیگر کلاس‌هایی تعریف شده است که به‌طور پیش‌فرض تمامی خصوصیات لازم برای اجرای کارها در رایانش ابری را دارا است. اما باید توجه داشت که همواره برای پیاده‌سازی روش موردنظر خود، نیازمند اعمال تغییراتی در آن‌ها هستیم. کلاس‌های تغییریافته و کلاس‌های جدید نوشته‌شده در این مطالعه در بخش زیر آمده است. شبه کد روش پیشنهادی در زیر نشان داده شده است:

1	**Input:**
2	Set of Virtual Machines (VMs)
3	Tasks or cloudlets (T)
4	**Output:**
5	deadline violation Number of
6	= number of virtual machines $NVMs$
7	//Set initial credit for each VM
8	$= \{1,1, \dots 1\}; C = \{C_i, C_{i+1}, \dots C_{NVMs}\}$
9	//At first, All Vms are healthy $HealthyVMs = VMs$
10	Define HRT as *Hard real-time Threshold*
11	Define H as *Set of hard real time tasks*
12	Define S as *Set of soft real time tasks*
13	
14	**While** (true)
15	// Recognizing tasks in the queue

[17] Cloudsim
[18] Broker

ISSN: 2745-3673
http://www.Rcsj.ir

NC = number of tasks in current period

$T_c = \{t_j, t_{j+1}, \dots t_{N_c}\}$

// Recognizing Healthy Virtual Machines

// classify tasks

For each T_c

//Calculate slack time for current task

// LFT (Last allowable Finish Time), PFT (Predicted Finish Time)

$Slack_i = LFT_i - PFT_i$

IF ($Si < HRT$) //HRT (Hard real-time Threshold)

$H = H + \{T_{c_i}\}$

Else IF ($HRT < Si < SRT$)

$S = S + \{T_{c_i}\}$

Else

// O = Other tasks $O = O + \{T_{c_i}\}$

End IF

End for

// Clustering Healthy Virtual Machines

//Sort VMs according to their Credits

SVM = Sorted VMs

NHVMs = Number of Healthy VMs

For each VM in SVM (C as Counter)

IF ($C < \left[\text{NHVMs} \times \frac{Size(H)}{N_c}\right]$)

} //HR = High Reliable VMs $HR = HR + \{VM_c$

(**Else IF**

$\left[\text{NHVMs} \times \frac{Size(H)}{NC}\right] < C <$

$\left[\text{NHVMs} \times \frac{Size(N)}{NC}\right]$

)

} //MR = Moderate Reliable VMs $MR = MR + \{VM_c$

Else

} // OV = Other VMs $OV = OV + \{VM_c$

End IF

End for

//Run tasks on corresponding clusters

Run H on HR

Run S on MR;

Run O on OV;

// Give credit to Virtual Machines

For each VM in $HealthyVMs$

IF (VM_i was healthy during this period)

$C_i + +;$

Else

$C_i - -;$

ISSN: 2745-3673
http://www.Rcsj.ir

	$WastedVMs = WastedVMs + \{VM_i\}$
	End IF
	End for
	Maintenance Phase // Enter Not Healthy Virtual Machines into
	$= HealthyVMs \_ WastedVMs; HealthyVMs$
	For each VM in $WastedVMs$
	IF (VM_i maintenance is finished)
	$WastedVMs = WastedVMs - \{VM_i\}$
	$= HealthyVMs + \{VM_i\} HealthyVMs$
	End IF
	End for
	End While

ارزیابی نتایج

در این بخش نتایج حاصل از پیاده‌سازی پوشش داده‌شده است. روش پیشنهادی با الگوریتم زمان‌بندی چرخشی[19] و همچنین AFTRC که به روش پیشنهادی شباهت دارد مقایسه خواهد شد. معیارهای مقایسه‌ی این الگوریتم‌ها عبارت است از:

- تعداد نقض مهلت زمانی برای هر یک از خوشه‌ها
- تعداد کل نقض‌ها در یک دوره زمانی
- زمان شبیه‌سازی (به‌عنوان یک معیار نادقیق برای اندازه‌گیری میزان سربار)

نتایج به‌دست‌آمده از شبیه‌سازی برای هر سه روش در جدول ۲ آمده است. آزمایشات با استفاده از کتابخانه‌ی کلودسیم در محیط Netbeans و به زبان جاوا شبیه‌سازی شده است. همچنین سخت افزار مورد استفاده عبارت است از: پردازنده ۲,۶ گیگاهرتزی Core i7 6500Hq و ۱۲ گیگابایت حافظه‌ی اصلی.

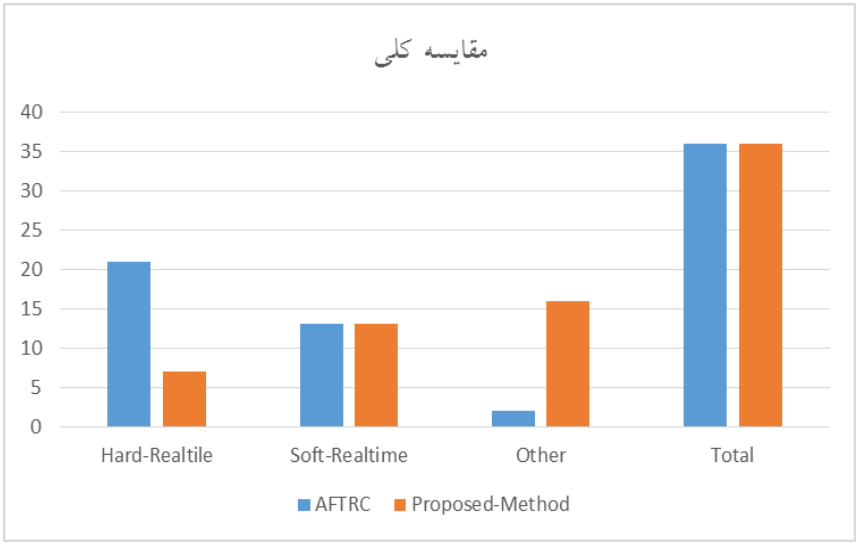
در ابتدای کار روش‌های نام‌برده ازنظر تعداد خطاهای به وجود آمده مقایسه می‌شوند. این مقایسه در جدول ۲ نشان داده‌شده است. اولین چیزی که کاملاً مشخص است، تفاوت فاحش روش پیشنهادی و روش AFTRC با روش زمان‌بندی چرخشی است. در این مطالعه با در نظر گرفتن مهلت‌های زمانی نسبتاً کوچک، روش زمان‌بندی چرخشی درمجموع ۹۹۶۲ خطا تولید کرده است. این مقدار برای هر دو روش AFTRC و روش پیشنهادی برابر با ۳۶ است. کاهش بیش از ۹۶۰۰ خطا به‌وسیله‌ی روش‌های تحمل‌پذیر خطا نشان‌دهنده‌ی کارایی این روش‌ها و ضرورت استفاده از آن‌ها در سیستم‌های محاسباتی بزرگ، ازجمله رایانش ابری است. ازآنجایی‌که تعداد خطاهای به وجود آمده درروش زمان‌بندی چرخشی با دو روش تحمل‌پذیر خطا قابل‌مقایسه نیست، در ادامه در اکثر موارد، این دو روش تحمل‌پذیر خطا با یکدیگر مقایسه خواهد شد. مقایسه‌ی کلی روش AFTRC و روش پیشنهادی در شکل ۸ نشان داده‌شده است. همان‌طور که از جدول ۳ می‌توان متوجه شد، مجموع تعداد خطاهای به وجود آمده در هر دو روش مساوی و برابر با ۳۶ بوده است. بر اساس آنچه در شکل ۸ دیده می‌شود، قدرت روش پیشنهادی در کاهش تعداد خطاها برای کارهای بلادرنگ سخت و انتقال تعداد خطاها بر روی کارهایی است که خطا بلادرنگ محسوب نمی‌شوند.

جدول ۲ مقایسه روش‌های مورد مقایسه ازنظر تعداد کل نقض‌های مهلت زمانی

	AFTRC	Proposed-Method	RoundRobin
Hard-Realtime	21	7	3294
Soft-Realtime	13	13	3301
Other	2	16	3367
Total	36	36	9962

[19] Round Robin

ISSN: 2745-3673
http://www.Rcsj.ir



شکل *Error! No text of specified style in document.*۸ مقایسه کلی روش AFTRC و روش پیشنهادی از نظر تعداد نقض‌های مهلت زمانی

بر اساس شکل ۵-۱ می‌توان نتیجه گرفت که هرچند تعداد کل خطاها در هر دو روش پیشنهادی با یکدیگر برابر بوده است، اما تعداد خطاهای به وجود آمده در کارهای بلادرنگ سخت به‌شدت کاهش‌یافته و به یک‌سوم رسیده است. هرچند تعداد خطاهای به وجود آمده از نوع بلادرنگ نرم ثابت بوده است، اما تعداد خطاهای موجود در کارهای غیر بلادرنگ، به‌صورت قابل‌توجهی افزایش‌یافته است. درواقع مابه تفاوت آن خطاهایی که از خوشه‌ی بلادرنگ سخت کاسته شده است، بر روی کارهای غیر بلادرنگ اضافه‌شده است.

بر اساس آنچه در شکل ۵-۲ نشان داده‌شده است، هرچند روش AFTRC توانسته است تعداد خطاها را به‌صورت قابل‌ملاحظه‌ای کاهش دهد اما خیلی در ایجاد توازن و کاهش تعداد خطاهای بلادرنگ سخت موفق نبوده است، به‌طوری‌که تقریباً در تمامی بازه‌های زمانی تعداد خطاهایی که در کارهای بلادرنگ سخت اتفاق افتاده است بیشتر از خطاهای به وجود آمده بر روی کارهای نوع بلادرنگ نرم و کارهای غیر بلادرنگ بوده است. دلیل این امر این است که روش AFTRC هیچ کلاس‌بندی بر روی نوع کارهای تولیدشده انجام نمی‌دهد و تنها بر اساس سطح دسترسی کاربران است که تقسیم به توزیع کارها بر روی ماشین‌های مجازی دارای امتیاز بهتر می‌گیرد. این در حالی است که همچنان این امکان وجود دارد که یک کاربر دارای سطح بالای دسترسی که ماشین‌های مجازی دارای قابلیت اطمینان بالا در دسترس دارد کارهای بلادرنگ نرم یا کارهای غیربلادرنگ زیادی تولید کند و برعکس، آن‌هایی که دارای سطح دسترسی پایین‌تر و به طبع ماشین‌های مجازی دارای قابلیت اطمینان پایین‌تری هستند، همچنان امکان تولید کارهای بلادرنگ سخت برای آن‌ها وجود دارد، هرچند این مقدار کم است. درنهایت می‌توان مشاهده کرد که تعداد کل خطاهای به وجود آمده بر روی کارهای بلادرنگ سخت ۲۱ مورد، بلادرنگ نرم ۱۳ مورد و کارهای غیر بلادرنگ ۲ مورد است.

جدول *Error! No text of specified style in document.*۳ تعداد نقض‌های مهلت زمانی ایجادشده درروش AFTRC به تفکیک بازه‌ی زمانی

AFTRC	p1	p2	p3	p4	p5		sum
Hard-Realtime	4	3	7	2	5		21
Soft-Realtime	2	2	3	4	2		13
Other	0	1	0	0	1		2
							36

ISSN: 2745-3673
http://www.Rcsj.ir

در جدول ۴ تعداد خطاهای ایجادشده درروش پیشنهادی به تفکیک بازه‌ی زمانی آمده است. همان‌طور که در این شکل مشخص است، روش پیشنهادی هرچند نتوانسته است تعداد کل خطاهای به وجود آمده را کم کند، اما در کاهش تعداد خطاهای به وجود آمده در کارهای بلادرنگ سخت. بلادرنگ نرم و انتقال آن‌ها به کارهای غیر بلادرنگ بسیار موفق بوده است. درنهایت می‌توان مشاهده کرد که تعداد کل خطاهای به وجود آمده بر روی کارهای بلادرنگ سخت ۷ مورد، بلادرنگ نرم ۱۳ مورد و سایر کارها ۱۶ مورد است. درنهایت می‌توان بهبود نسبی روش پیشنهادی را با مشاهده‌ی کاهش تعداد خطاهای به وجود آمده در کارهای بلادرنگ سخت را نتیجه گرفت.

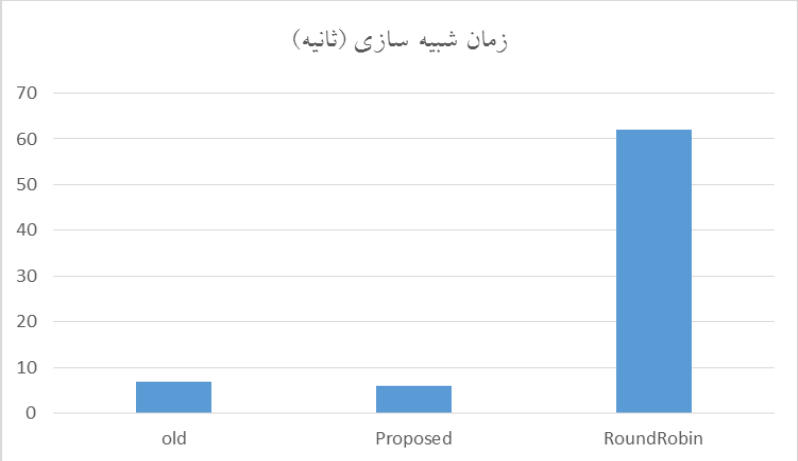
جدول *Error! No text of specified style in document.*۴ تعداد خطاهای ایجادشده درروش پیشنهادی به تفکیک بازه ی زمانی

Proposed-Method	p1	p2	p3	p4	p5	sum
Hard-Realtime	1	2	2	1	1	7
Soft-Realtime	2	2	4	0	5	13
Other	3	2	4	5	2	16
						36

همان‌طور که پیش‌ازاین گفته شد، زمان شبیه‌سازی به‌عنوان یک معیار نادقیق برای اندازه‌گیری سربار روش‌ها مورداستفاده قرار می‌گیرد. بر اساس آنچه در جدول ۵ و شکل ۹ به بررسی زمان شبیه‌سازی بر روی نرم‌افزار و سخت‌افزار یادشده پرداخته‌شده است. همان‌طور که مشخص است، روش AFTRC و روش پیشنهادی به همدیگر نزدیک بوده و با روش نوبت‌دهی چرخشی تفاوت زیادی داشته است. دلیل این امر سربار بالای روش نوبت‌دهی چرخشی در مواردی است که طول صفوف برای ماشین‌های مجازی بالا رفته که در این صورت هرچند روش نوبت‌دهی چرخشی تلاش در کاهش زمان پاسخ داشته است، اما به دلیل سربار ناشی از تعویض وظایف موجود در صف، زبان بازگشت و درنهایت زمان اتمام کل کارها بالا می‌رود.

جدول ۵ زمان شبیه‌سازی روش‌های مورد مقایسه

روش	زمان (ثانیه)
AFTRC	7
Proposed	6
RoundRobin	62



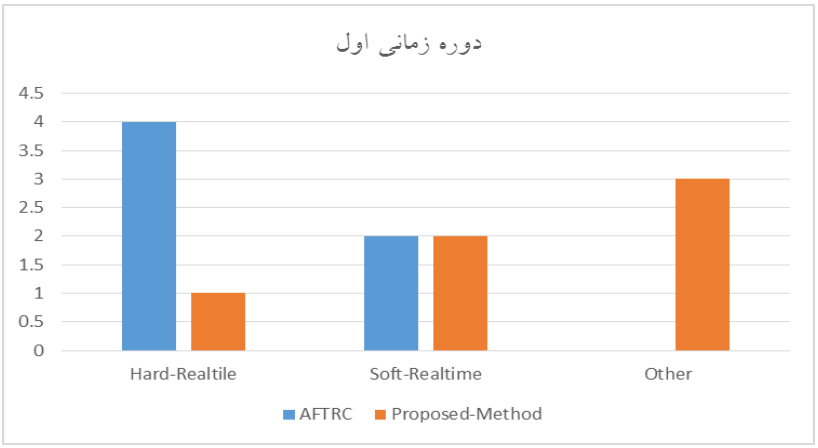
شکل ۹ زمان شبیه‌سازی روش‌های مورد مقایسه

ISSN: 2745-3673
http://www.Rcsj.ir

بر اساس شکل ۵-۲ می‌توان نتیجه گرفت که دو روش ارائه‌شده و روش AFTRC تقریباً زمانی برابر با یک‌دهم زمان شبیه‌سازی روش نوبت‌دهی چرخشی برای اجرای کارهای مشابه داشته‌اند.

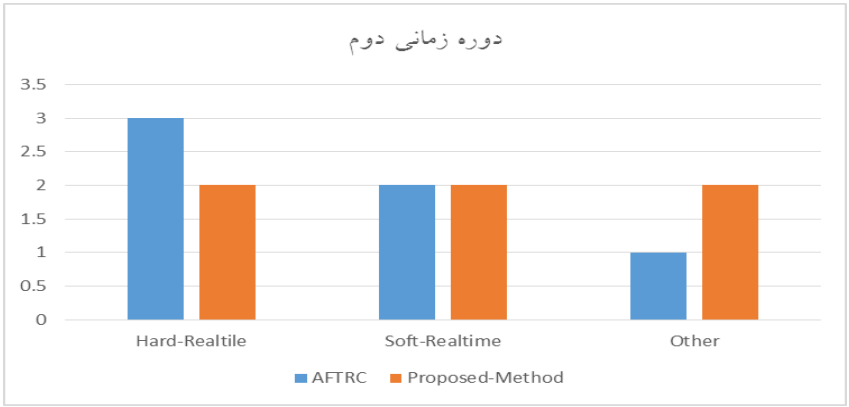
در ادامه‌ی کار، دو روش AFTRC و روش پیشنهادی به تفکیک نتایج هر بازه‌ی زمانی با یکدیگر مقایسه خواهند شد.

بر اساس شکل ۱۰، در بازه‌ی زمانی اول، روش پیشنهادی بسیار کارا عمل کرده، به‌نحوی‌که تعداد خطاهای موجود در کارهای از نوع بلادرنگ سخت را یک‌چهارم کرده و از ۴ به یک رسانیده است. هرچند تعداد خطاهای به وجود آمده در کارهای بلادرنگ نرم تغییری پیدا نکرده است، اما تعداد خطاهای به وجود آمده در کارهایی که مهلت زمانی ندارند از صفر به چهار افزایش پیداکرده است. با توجه به آنکه هدف اصلی ما کاهش تعداد خطاها در کارهای بلادرنگ سخت و بلادرنگ نرم است، روش پیشنهادی در بازه‌ی زمانی اول به‌خوبی عمل کرده است.



شکل ۱۰ مقایسه روش پیشنهادی و روش AFTRC ازنظر تعداد نقض مهلت زمانی در دوره زمانی اول

بر اساس شکل ۱۱، در بازه‌ی زمانی دوم، روش پیشنهادی به‌طور نسبی کارا عمل کرده، به‌نحوی‌که تعداد نقض‌های مهلت زمانی موجود در کارهای از نوع بلادرنگ سخت را به دوسوم مقدار روش AFTRC کاهش داده و از ۲ به ۳ رسانیده است. هرچند تعداد خطاهای به وجود آمده در کارهای بلادرنگ نرم تغییری پیدا نکرده است، اما تعداد نقض‌های مهلت زمانی به وجود آمده در کارهایی که مهلت زمانی ندارند از یک‌به‌دو افزایش پیداکرده است. با توجه به آنکه هدف اصلی ما کاهش تعداد نقض‌های مهلت زمانی در کارهای بلادرنگ سخت و بلادرنگ نرم است، روش پیشنهادی در بازه‌ی زمانی دوم نیز عملکرد مطلوب داشته است.

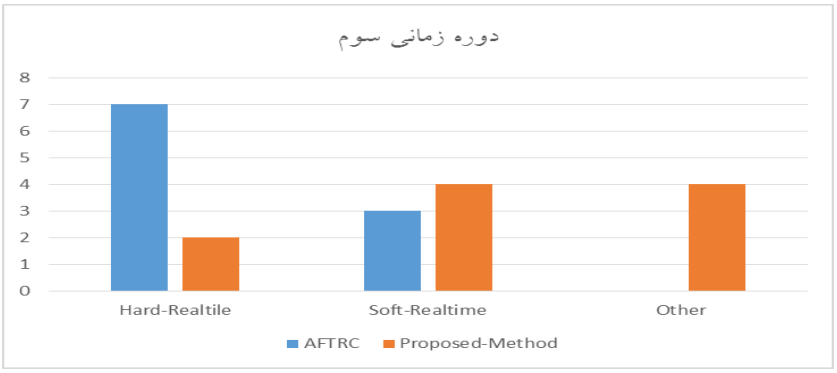


شکل ۱۱ مقایسه روش پیشنهادی و روش AFTRC ازنظر تعداد نقض مهلت زمانی در دوره زمانی دوم

بر اساس شکل ۱۱، در بازه‌ی زمانی سوم، روش پیشنهادی کارایی بسیار مناسبی داشته است، به‌نحوی‌که تعداد نقض‌های مهلت زمانی موجود در کارهای از نوع بلادرنگ سخت را به یک‌سوم مقدار روش AFTRC کاهش داده و از ۳ به ۲ رسانیده است. هرچند تعداد نقض‌های مهلت زمانی به وجود آمده در کارهای بلادرنگ سخت بسیار کمتر شده است، اما تعداد نقض‌های

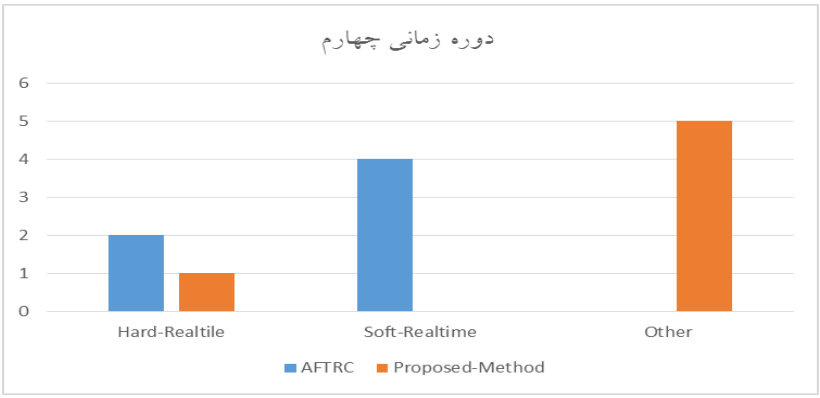
ISSN: 2745-3673
http://www.Rcsj.ir

مهلت زمانی به وجود آمده در کارهایی که از نوع بلادرنگ نرم هستند یا مهلت زمانی ندارند افزایش پیداکرده است. با توجه به آنکه هدف اصلی ما کاهش تعداد نقض‌های مهلت زمانی در کارهای بلادرنگ سخت و بلادرنگ نرم است، روش پیشنهادی در بازه ی زمانی سوم نیز عملکرد مطلوب داشته است.



شکل *Error! No text of specified style in document.*۱۱ مقایسه روش پیشنهادی و روش AFTRC ازنظر تعداد خطاها در دوره زمانی سوم

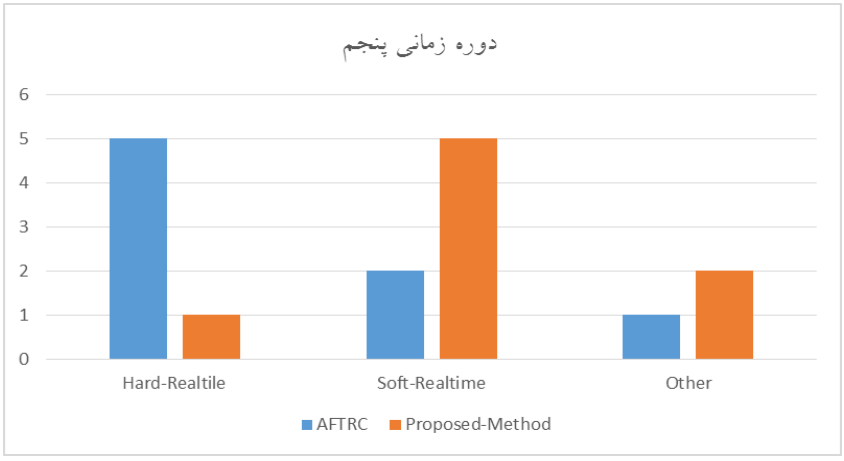
بر اساس شکل ۱۲، در بازه‌ی زمانی چهارم، روش پیشنهادی کارایی مناسبی داشته است، به‌نحوی‌که تعداد نقض‌های مهلت زمانی موجود در کارهای از نوع بلادرنگ سخت نسبت به روش AFTRC یک مورد کاهش داده و تعداد نقض‌های مهلت زمانی به وجود آمده بر روی کارهای بلادرنگ نرم را از ۴ به صفر رسانیده است. هرچند تعداد نقض‌های مهلت زمانی به وجود آمده در کارهای بلادرنگ سخت و نرم کاهش‌یافته است، اما تعداد نقض‌های مهلت زمانی به وجود آمده بر روی دسته‌ای از کارها که مهلت زمانی ندارند افزایش پیداکرده است. با توجه به آنکه هدف اصلی ما کاهش تعداد نقض‌های مهلت زمانی در کارهای بلادرنگ سخت و بلادرنگ نرم است، روش پیشنهادی در بازه‌ی زمانی چهارم نیز عملکرد مطلوب داشته است.



شکل ۱۲ مقایسه روش پیشنهادی و روش AFTRC ازنظر تعداد خطاها در دوره زمانی چهارم

و درنهایت بر اساس شکل ۱۳، در بازه‌ی زمانی پنجم، روش پیشنهادی همچنان کارایی مناسبی داشته است، به‌نحوی‌که تعداد نقض‌های مهلت زمانی موجود در کارهای از نوع بلادرنگ سخت نسبت به روش AFTRC چهار مورد و معادل ۸۰ درصد کاهش داده ولی تعداد نقض‌های مهلت زمانی به وجود آمده بر روی کارهای بلادرنگ نرم را از ۲ به ۵ رسانیده است. هرچند تعداد نقض‌های مهلت زمانی به وجود آمده در کارهای بلادرنگ سخت کاهش‌یافته است، اما تعداد نقض‌های مهلت زمانی به وجود آمده بر روی دسته‌ای از کارها که مهلت زمانی ندارند و یا از نوع بلادرنگ نرم هستند افزایش پیداکرده است. با توجه به آنکه هدف اصلی ما کاهش تعداد نقض‌های مهلت زمانی در کارهای بلادرنگ سخت و بلادرنگ نرم است، روش پیشنهادی در بازه ی زمانی پنجم نیز عملکرد مطلوب داشته است و درنهایت توانسته است در کل بازه‌های زمانی و درنهایت در کل زمان اجرا، تعداد نقض‌های مهلت زمانی موجود در کارهای بلادرنگ سخت را به‌طور قابل‌ملاحظه کاهش دهد.

ISSN: 2745-3673
http://www.Rcsj.ir



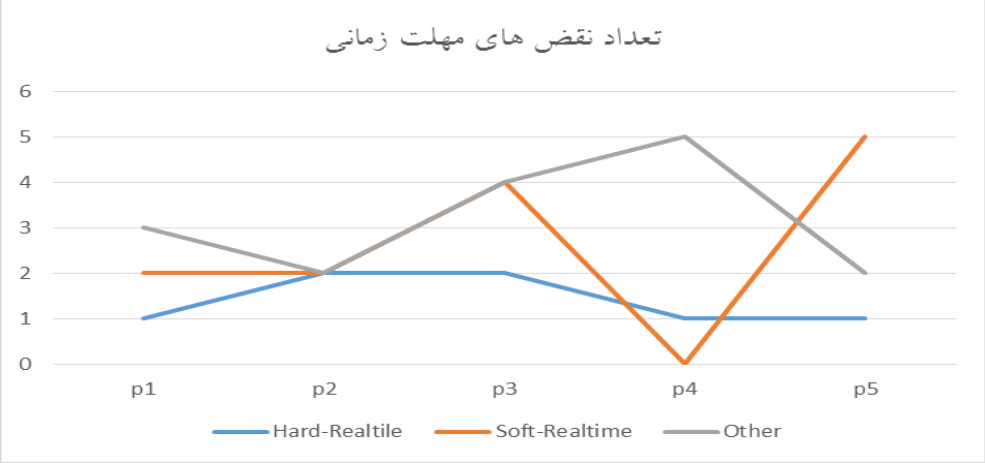
شکل ۱۳ مقایسه روش پیشنهادی و روش AFTRC ازنظر تعداد خطاها در دوره زمانی پنجم

همان‌طور که می‌دانیم، تمرکز اصلی این مطالعه بر روی کاهش تعداد نقض‌های مهلت زمانی در کارهای بلادرنگ، به‌خصوص کارهای بلادرنگ سخت بوده است. بر اساس آنچه دیده شد، روش پیشنهادی به‌طور کلی نتوانسته است که تعداد نقض‌های مهلت زمانی را کاهش دهد، اما به‌طور قابل‌ملاحظه‌ای تعداد نقض‌ها را برای کارهای بلادرنگ سخت کاهش داده است، به‌طوری‌که پس از گذشت زمان و پس از سپری شدن پنج دوره‌ی زمانی، هیچ‌گاه تعداد نقض‌های مهلت زمانی در کارهای بلادرنگ سخت، از کارهای بلادرنگ نرم و کارهای غیر بلادرنگ بیشتر نشده است و نمودار آن همواره (به‌جز یک مورد) در زیر نمودار تعداد نقض‌های به وجود آمده برای کارهای بلادرنگ نرم و کارهای غیر بلادرنگ قرار می‌گیرد (شکل ۱۴).

با مقایسه‌ی روش پیشنهادی با روش AFTRC می‌توان دریافت که روش پیشنهادی در کاهش تعداد نقض‌های زمانی موجود در کارهای بلادرنگ سخت بسیار بهتر عمل کرده است. هرچند تعداد کل نقض‌های مهلت زمانی هر دو روش با یکدیگر برابر بوده است، اما درروش AFTRC تعداد نقض‌های مهلت زمانی بر روی کارهای بلادرنگ سخت بیشتر بوده است. در این صورت هرچند که کاربران دارای دسترسی بالا احتمالاً تعداد کارهای بلادرنگ سخت تولیدشده توسط آن‌ها بیشتر است، اما همچنان امکان تولید کارهای بلادرنگ نرم و حتی کارهای غیر بلادرنگ برای وی وجود داشته است و در صورت تخصیص ماشین‌های مجازی دارای قابلیت اطمینان بالا بر اساس سطح اشخاص، مقدار زیادی از منابع به همین دلیل از بین خواهد رفت.

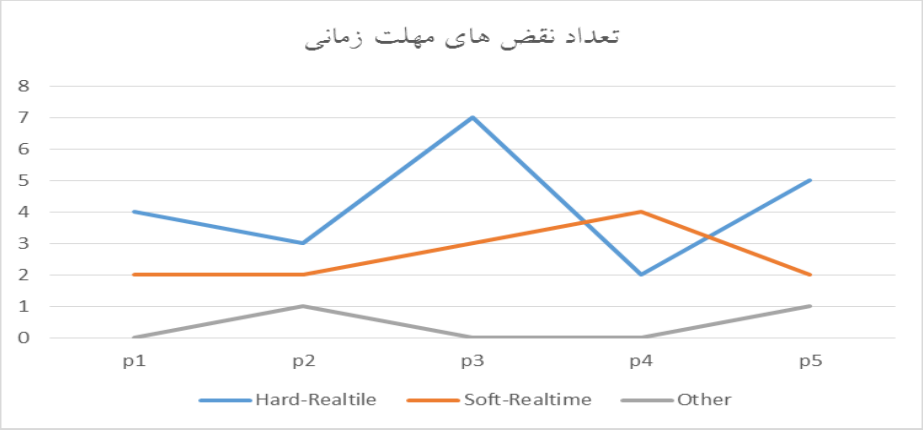
بنابراین در این مطالعه با ایجاد تغییر در تخصیص وظایف به ماشین‌های مجازی توانسته‌ایم کارایی بسیار مناسب‌تری را نسبت به روش AFRTC داشته باشیم.

نمودار روند نقض مهلت‌های زمانی درروش AFRTC در شکل ۱۴ نشان داده‌شده است.



شکل ۱۴ مقایسه‌ی تعداد نقض‌های مهلت زمانی در بازه‌های زمانی متوالی

ISSN: 2745-3673
http://www.Rcsj.ir



شکل ۱۵ مقایسه‌ی تعداد نقض‌های مهلت زمانی درروش AFTRC در بازه‌های زمانی متوالی

بر اساس آنچه در شکل ۱۵ نشان داده‌شده است، روش AFTRC تقریباً برعکس روش پیشنهادی عمل کرده است. این روش به دلیل عدم آگاهی نسبت به نوع و ماهیت مهلت‌های زمانی در کارهای تولیدشده و ارسال آن‌ها تنها بر مبنای هویت کاربر، هرچند که تعداد کل نقض‌های مهلت زمانی را نسبت به روش‌های غیر تحمل‌پذیر خطا همچون RoundRobin بسیار بهتر کرده است، اما انتظار می‌رفت که تعداد نقض‌های مهلت زمانی کارهای بلادرنگ سخت را نیز کاهش دهد. ازآنجایی‌که روش آشنایی با نوع کارها و نوع مهلت زمانی آن‌ها ندارد، تعداد نقض‌های مهلت زمانی کارها از نوع بلادرنگ سخت، بیشتر از تعداد نقض‌های مهلت‌های زمانی از نوع بلادرنگ نرم و کارهای غیر بلادرنگ بوده است، به‌طوری‌که نمودار تعداد نقض‌های مهلت‌های زمانی به وجود آمده در کارهای بلادرنگ سخت همواره (به‌جز یک مورد) بیشتر از تعداد نقض‌های مهلت‌های زمانی در کارهای بلادرنگ نرم و کارهای غیر بلادرنگ بوده است.

نتیجه‌گیری و پیشنهاد کارهای آینده

در این مطالعه یک چارچوب برای زمان‌بندی وظایف بلادرنگ در رایانش ابری با خصوصیت تحمل پذیری خطا ارائه گردید. روش پیشنهادی با در نظر گرفتن الگوریتم‌های مجزا بر روی هر یک از ماشین‌های مجازی، خطاهای مختلفی را برای هر یک از آن‌ها در فواصل زمانی مختلف با توزیع پواسون در نظر گرفت. نتایج شبیه‌سازی بطور کلی نشان داد که روش پیشنهادی هرچند بطور کلی نتوانسته است تعداد تمامی نقض‌های مهلت‌های زمانی را بر طرف کند، اما توانسته تعداد نقض‌های مربوط به وظایف بلادرنگ سخت را بطور قابل ملاحظه کاهش دهد. با در نظر گرفتن ۵ دوره‌ی زمانی، تعداد نقض‌های بوجود آمده در روش پیشنهادی نسبت به روش AFTRC در مجموع یکسان و برابر با ۳۶ بوده است. در روش AFTRC تعداد نقض‌ها به ترتیب برای کارهای بلادرنگ سخت، بلادرنگ نرم و سایر کارها برابر با ۲۱ ، ۱۳ و ۲ بوده است که نشان از تعداد نقض‌های بالا در کارهای بلادرنگ سخت است. این مقادیر برای الگوریتم پیشنهادی به ترتیب به ۷، ۱۳ و ۱۶ رسیده است که نشان از کاهش قابل ملاحظه‌ی نقض‌های مهلت زمانی بلادرنگ سخت شده است. این امر از طریق دادن تعداد ماشین‌های مجازی به تعداد مناسب به هر خوشه انجام شده است. از طرفی زمان اجرای شبیه سازی و به طبع آن سربار الگوریتم پیشنهادی از ۷ ثانیه به ۶ ثانیه رسیده است که نشان از کم بودن سربار الگوریتم دارد. تعداد نقض‌ها در الگوریتم نبت دهی چرخشی بسیار بالا بوده و همچنین سربار آن بطور قابل ملاحظه ای از روش پیشنهادی و AFTRC بیشتر بوده است

مزایای و معایب روش پیشنهادی

همان‌طور که در بخش ۴ نیز گفته شد، روش پیشنهادی به‌طور قابل‌توجهی تعداد نقض‌های مهلت زمانی به وجود آمده را کاهش داده است، اما در کاهش تعداد کل نقض‌های مهلت زمانی ناکام بوده است. تعداد کل نقض‌های مهلت زمانی نیز به‌خودی‌خود کوچک بوده که می‌توان از کاهش تعداد آن نیز صرفه نظر کرد. مزایای و معایب روش پیشنهادی عبارت است از:

ISSN: 2745-3673
http://www.Rcsj.ir

- ✓ کارایی بالا در کاهش تعداد کل نقض‌های مهلت زمانی در کارهای دارای مهلت زمانی کم (کارهای بلادرنگ سخت)
- ✓ کارایی بالا در کاهش تعداد کل نقض‌های مهلت زمانی در تمامی خوشه‌ها نسبت به روش‌های غیر تحمل‌پذیر خطا
- ✓ سربار کم

به عنوان کار پیشنهادی در آینده نیز، سعی داریم که روش پیشنهادی را بر روی یک سیستم واقعی محاسباتی با کارایی بالا[20] و یا بر روی سکوی هادوپ[21] به‌صورت واقعی پیاده‌سازی کنیم. انتظار می‌رود که روش پیشنهادی در یک پیاده‌سازی واقعی نیز کارایی مناسبی داشته باشد.

[20] High Performance Computing (HPC)
[21] Hadoop

ISSN: 2745-3673
http://www.Rcsj.ir

منابع و مراجع

[1] R. L. Grossman, “The case for cloud computing,” *IT professional*, vol. 11, no. 2, pp. 23-27, 2009.
[2] Dinh, Hoang T., et al. "A survey of mobile cloud computing: architecture, applications, and approaches." *Wireless communications and mobile computing* 13.18 (2013): 1587-1611.
[3] Y. Jadeja, and K. Modi, "Cloud computing-concepts, architecture and challenges." pp. 877-880.
[4] Avram, Maricela-Georgiana. "Advantages and challenges of adopting cloud computing from an enterprise perspective." *Procedia Technology* 12 (2014): 529-534.
[5] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *Journal of internet services and applications*, vol. 1, no. 1, pp. 7-18, 2010.
[6] Zhang, Congyingzi, Robert Green, and Mansoor Alam. "Reliability and Utilization Evaluation of a Cloud Computing System Allowing Partial Failures." *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. IEEE, 2014.
[7] S. Malik, and F. Huet, "Adaptive fault tolerance in real time cloud computing." pp. 280-287.
[8] Zhani, Mohamed Faten, and Raouf Boutaba. "Survivability and fault tolerance in the cloud." *Cloud Services, Networking, and Management*. John Wiley & Sons, Inc, 2015. 295-308.
[9] Patra, Prasenjit Kumar, Harshpreet Singh, and Gurpreet Singh. "Fault tolerance techniques and comparative implementation in cloud computing." *International Journal of Computer Applications* 64.14 (2013).
[10] Cheraghlou, Mehdi Nazari, Ahmad Khadem-Zadeh, and Majid Haghparast. "A survey of fault tolerance architecture in cloud computing." *Journal of Network and Computer Applications* 61 (2016): 81-92.
[11] Kumar, Karthik, et al. "Resource allocation for real-time tasks using cloud computing." *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*. IEEE, 2011.
[12] T. Tan, R. T. Ma, M. Winslett, Y. Yang, Y. Yu, and Z. Zhang, "Resa: realtime elastic streaming analytics in the cloud." pp. 1287-1288.
[13] Zhu, Xiaomin, et al. "Real-time tasks oriented energy-aware scheduling in virtualized clouds." *IEEE Transactions on Cloud Computing* 2.2 (2014): 168-180.
[14] W. Zhao, P. Melliar-Smith, and L. E. Moser, "Fault tolerance middleware for cloud computing." pp. 67-74.
[15] Jhawar, Ravi, Vincenzo Piuri, and Marco Santambrogio. "A comprehensive conceptual system-level approach to fault tolerance in cloud computing." *Systems Conference (SysCon), 2012 IEEE International*. IEEE, 2012.
[16] Johnson, Barry W. *Design & analysis of fault tolerant digital systems*. Addison-Wesley Longman Publishing Co., Inc., 1988.
[17] Laprie, Jean-Claude. "Dependable computing and fault-tolerance." *Digest of Papers FTCS-15* (1985): 2-11.
[18] Ganesh, Amal, M. Sandhya, and Sharmila Shankar. "A study on fault tolerance methods in cloud computing." *Advance Computing Conference (IACC), 2014 IEEE International*. IEEE, 2014.
[19] Rimal, Bhaskar Prasad, Eunmi Choi, and Ian Lumb. "A taxonomy and survey of cloud computing systems." *INC, IMS and IDC* (2009): 44-51.
[20] A. Bala, and I. Chana, “Fault tolerance-challenges, techniques and implementation in cloud computing,” *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 1, pp. 1694-0814, 2012.
[21] A. Ganesh, M. Sandhya, and S. Shankar, "A study on fault tolerance methods in Cloud Computing." pp. 844-849.

ISSN: 2745-3673
http://www.Rcsj.ir

[22] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, “The rise of “big data” on cloud computing: Review and open research issues,” *Information Systems*, vol. 47, pp. 98-115, 2015.
[23] V. N. Inukollu, S. Arsi, and S. R. Ravuri, “Security issues associated with big data in cloud computing,” *International Journal of Network Security & Its Applications*, vol. 6, no. 3, pp. 45, 2014.
[24] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, "FTCloud: A component ranking framework for fault-tolerant cloud applications." pp. 398-407.
[25] V. Kaushal, and A. Bala, “Autonomic fault tolerance using haproxy in cloud environment,” *Int. J. of Advanced Engineering Sciences and Technologies*, vol. 7, no. 2, pp. 54-59, 2011.
[26] Zhang, Yilei, Zibin Zheng, and Michael R. Lyu. "BFTCloud: A byzantine fault tolerance framework for voluntary-resource cloud computing." *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011.
[27] S. S. Lakshmi, “Fault Tolerance in Cloud Computing,” *IJESR International Journal Of Engineering Sciences Research*, vol. 4, 2013