ICWR2020

کنفرانس بین المللی وب پژوهی
ایران، تهران، ۲۶ خرداد ۱۳۹۹

Proposed Method for Reducing Bandwidth Utilization in Live Container Migration at Fog Layer

MohammadJavad Asadi [1], Azadeh Tabatabaei [2]

[1] Master Student of Software Engineering, University of Science and Culture, Tehran, Iran
mj.asadi72@gmail.com

[2] Assistant Professor, Computer Engineering Department, University of Science and Culture, Tehran, Iran
a.tabatabaei@usc.ac.ir

Abstract

One of the problems of fog computing comparing to cloud computing's layers is the lack of hardware resources. Hence, this limitation causes restriction on the number of services that can be implemented simultaneously in fog layers and this restriction affects the services in dynamic networks. To illustrate, in fog networks that provide services to IOV, there are several different machines with contrasting services that each due to the movement of the IOV machines in local fog network, must be able to provide services to all local networks. Since there is not enough resources, this is not possible in the fog layers.

The intention of this paper is providing a solution to optimize migration in fog computing. The solution for lack of the resource in fog layer that we present in this paper are: using a container-based virtualization solution called LXC/LXD, and using CRIU for real time migration of container. Furthermore, to minimize the amount of transmitted data that takes most of the migration time, this paper will include the using of a lightweight template for LXC/LXD, called alpine which will reduce the amount of data migrated in the migration process.

Keywords: Fog Computing, Internet of Things, Live Container Migration, Network, Cloud Computing.

JCWR2020

کنفرانس بین المللی وب پژوهی
ایران، تهران، ۲۶ خرداد ۱۳۹۹

روش پیشنهادی برای کاهش استفاده از پهنای باند در مهاجرت زنده کانتینر در لایه مه

محمدجواد اسدی[۱]، آزاده طباطبائی[۲]

[۱]دانشجو کارشناسی ارشد مهندسی نرم‌افزار، دانشگاه علم و فرهنگ، تهران، ایران
mj.asadi72@gmail.com

[۲] استادیار گروه مهندسی کامپیوتر، دانشگاه علم و فرهنگ، تهران، ایران
a.tabatabaei@usc.ac.ir

چکیده

یکی از مشکلاتی که در رایانش مه وجود دارد کمبود منابع سخت‌افزاری نسبت به لایه ابر است. ازاین‌رو این مسئله باعث به وجود آمدن محدودیت در تعداد سرویس‌هایی که می‌توان در لایه مه به‌صورت هم‌زمان پیاده‌سازی کرد می‌شود و این محدودیت یک مشکل اساسی در سرویس‌دهی در شبکه‌های پویا به وجود می‌آورد. به‌طور مثال در شبکه‌های مه ای که به Iov سرویس می‌دهند؛ تعداد ماشین‌های مختلف با سرویس‌های مختلفی وجود دارد که هریک به دلیل جابه‌جایی دستگاه‌های Iov در شبکه‌های محلی مه، باید سرویس‌هایشان در تمام شبکه‌های محلی موجود باشد. به دلیل محدودیت در منابع لایه مه این کار امکان‌پذیر نیست.

در این مقاله یک راهکار برای بهینه‌سازی مهاجرت سرویس‌ها در مه، ارائه می‌شود. راهکاری که در این پژوهش برای مقابله با مشکل کمبود منابع در لایه مه ارائه‌شده است، استفاده از یک راهکار مجازی سازی به بر پایه کانتینر به نام LXC/LXD و استفاده از CRIU برای مهاجرت زنده کانتینر می‌باشد. همچنین برای به حداقل رساندن حجم داده‌های منتقل شده که بیشترین زمان مهاجرت را به خود اختصاص می‌دهد، استفاده از یک قالب سبک برای LXC/LXD به نام آلپاین است که حجم داده‌های منتقل شونده در مهاجرت را کاهش دهد.

کلمات کلیدی

رایانش مه، اینترنت اشیاء، مهاجرت زنده کانتینر، شبکه، رایانش ابری.

۱- مقدمه

امروزه با افزایش تولید داده‌ها مواجه هستیم به‌طوری‌که این رقم در سال ۲۰۱۹ به عدد ۱۰ زتابایت[۱] می‌رسد. همچنین تعداد دستگاه‌های متصل به اینترنت به عدد ۵۰ میلیارد می‌رسد [1,2]. همین سرعت تولید داده ما را با کمبود پهنای باند در ارسال این داده‌ها به سمت ابر مواجه کرده است. همچنین به دلیل زمان پاسخ‌دهی بالای پردازش ابری، نیاز پردازشی برخی داده‌ها مانند داده‌های پزشکی و یا داده‌های تولیدشده توسط Iov[۲] را، رفع نمی‌کند [3,4].به همین منظور مدلی برای حل این مشکل ارائه‌شده است که پیشنهاد می‌کند که برخی از داده‌ها را در لبه شبکه محلی پردازش و یا پیش‌پردازش کند و سپس داده‌ها را به سمت ابر بفرستند. این لایه جدید را لایه مه[۳] می‌نامند که در شکل (۱) یک شمای کلی از سیستم را مشاهده می‌کنید.

به دلیل ماهیت لایه مه، این لایه ما را با چالش‌هایی روبه‌رو می‌کند. از این چالش‌ها می‌توان به محدودیت منابع نسبت به ابر نام برد. به دلیل

محدودیت منابع در گره‌های لایه مه نمی‌توان تمام سرویس‌های موجود در ابر را به مه منتقل کرد. همچنین لینک ارتباطی گره‌ها در لایه مه انواع مختلفی دارد و معمولاً به دلیل توزیع‌شدگی گره‌ها، به‌طور مثال در شهر هوشمند آن‌ها به‌صورت بی‌سیم به هم متصل هستند، باعث کاهش پهنای باند این گره‌ها می‌شود.

محدودیت دیگر لایه مه نامطمئن بودن گره‌ها است، به‌طوری‌که بعضی از گره‌ها با استفاده از باتری کار می‌کنند و یا محل نگه‌داری گره‌ها جای امنی از نظر مراقبت فیزیکی نیست و امکان از دست رفتن گره‌ها به دلیل حوادثی مانند قطعی برق، رطوبت بالا، گرمای زیاد و ... بالا است زیرا این گره‌ها معمولاً در محیط امنی مانند مراکز داده قرار نمی‌گیرند. به همین دلیل باید سازوکاری فراهم کرد که سرویس‌ها را بتوانیم در بین گره‌های مه و همچنین میان مه و ابر مهاجرت[۴] دهیم تا سرویس‌دهی مطلوبی را داشته باشیم.

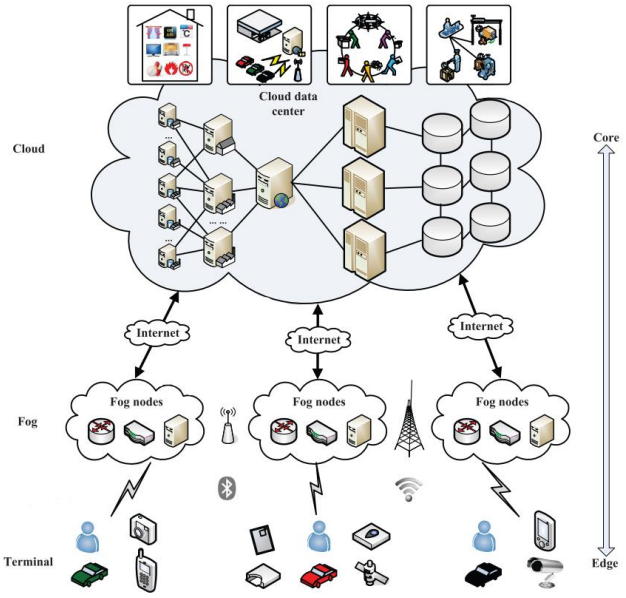
برای مهاجرت سرویس‌ها در میان گره‌ها می‌توان از ماشین مجازی و یا کانتینر[۵] استفاده کرد. بهترین راه استفاده از این روش استفاده از یک مخزن ذخیره اشتراکی[۶] است زیرا در فرایند مهاجرت، داده‌های بر روی دیسک جابه‌جا نمی‌شوند و فقط اطلاعات ماشین مجازی و وضعیت آن جابه‌جا می‌شود اما با توجه به این‌که لینک شبکه گره‌ها به‌صورت بی‌سیم است و پهنای باند کمی دارد، با استفاده از مخزن ذخیره اشتراکی پهنای باند ما درگیر داده‌های مخزن ذخیره اشتراکی می‌شود و از کارایی سرویس‌ها می‌کاهد. در مورد کانتینر[۷] نیز به همین صورت است و پهنای باند توسط مخزن ذخیره اشتراکی پر می‌شود و از کارایی گره می‌کاهد[6].

به همین دلیل برای جابه‌جایی سرویس‌ها در میان گره‌ها، عدم استفاده از مخزن ذخیره اشتراکی گزینه بهتری به نظر می‌رسد ولی عدم استفاده از این روش به معنی انتقال تمام داده‌های دیسک در زمان مهاجرت است که خود پهنای باند شبکه ما را درگیر می‌کند. همچنین به دلیل منابع محدود ذخیره‌سازی داده‌ها بر روی دیسک‌های محلی باید یک روش بهتر برای راه‌اندازی سرویس‌ها استفاده کرد.

برای کم کردن حجم فضای دیسک سرویس‌ها، می‌توان از کانتینر استفاده کرد. زیرا کانتینر منابع ذخیره‌سازی کمتری نسبت ماشین مجازی مصرف می‌کند و به همین دلیل حجم کمتری را از منابع ذخیره‌سازی محلی ما می‌گیرد و در زمان جابه‌جایی پهنای باند کمتری مصرف می‌کند.

کانتینر یک نوع مجازی سازی در سطح سیستم عامل است که در آن هسته سیستم عامل اجازه وجود چندین فضای کاربری[۸] ایزوله نسبت به هم را می‌دهد.

اصل کار این مجازی سازی بر اساس هسته لینوکس است که چندین محیط کاربری ایزوله نسبت به هم ایجاد می‌کند و تمام سیستم عامل‌ها در



شکل (۱): نمای کلی از معماری مه به‌صورت سلسله مراتبی [5]

حقیقت بر روی یک هسته اجرا می‌شوند. از نرم‌افزارهایی که از این قابلیت هسته لینوکس استفاده می‌کنند و سیستم عامل‌های مختلفی را می‌توان به‌عنوان کانتینر بر روی یک هسته مشترک بارگذاری کرد، می‌توان به داکر[۹]، اپن‌وی‌زد[۱۰] و LXC/LXD[۱۱] اشاره کرد.

تفاوت عمده‌ای که کانتینر به‌عنوان مجازی ساز با ماشین مجازی دارد نوع پیاده‌سازی آن است. در کانتینر به دلیل وجود هسته مشترک در سیستم عامل دیگر نیازی به فایل‌های راه‌اندازی[۱۲] و هسته نیست و به همین دلیل شروع به کار کانتینر سریع‌تر از ماشین مجازی است زیرا لازم نیست دوباره کرنل راه‌اندازی شود و سخت‌افزار شناسایی شود. از طرفی به دلیل اجراشدن بر روی هسته و عدم نیاز به یک ابرناظر[۱۳]، منابع کمتری برای اجرا لازم دارد چون هم نیازی به اشغال شدن دوباره فضای حافظه و فضای ذخیره‌سازی از کرنل نیست و هم ابرناظری بر روی سیستم اجرا نمی‌شود که منابع سیستم را درگیر کند. از طرفی به دلیل اجرای سیستم عامل بر روی هسته و نه مجازی سازی مانند ابرناظر، سرعت اجرا نیز نسبت به ماشین مجازی بالاتر است و افت کارایی کمتری به دلیل نگرفتن منابع مدیریتی به‌مانند ابرناظر، نسبت به ماشین مجازی دارد. منابع موردنیاز برای این نوع مجازی سازی بسیار کمتر از ماشین مجازی است و به همین دلیل گزینه ایده آلی برای پیاده‌سازی سرویس‌ها در لایه مه شده است. همچنین این نوع مجازی سازی در مقایسه با مجازی سازی با استفاده از ابرناظر یک برتری دیگر دارد و آن‌هم افت کمتر کارایی در هنگام افزایش تعداد سرویس‌ها می‌باشد. همان‌طور که از پژوهش [7] مشخص‌شده است با افزایش تعداد سرویس‌ها افت کارایی کمتری در کانتینر نسبت به ماشین مجازی وجود دارد.

همچنین در شکل (۲) نمونه‌ای از تفاوت مقدار زمان و حجم داده‌های منتقل شده توسط ماشین مجازی و کانتینر را می‌توان دید.

همان‌طور که مشاهده می‌شود مقدار داده منتقل شده توسط مهاجرت کانتینر برای یک سرویس به‌خصوص بسیار کمتر از داده منتقل شده توسط مهاجرت با استفاده از ماشین مجازی است به همین دلیل استفاده از کانتینر در شبکه مه به دلیل حجم کمتر داده‌های منتقل‌شده، پهنای باند کمتری از سیستم را مصرف می‌کند.

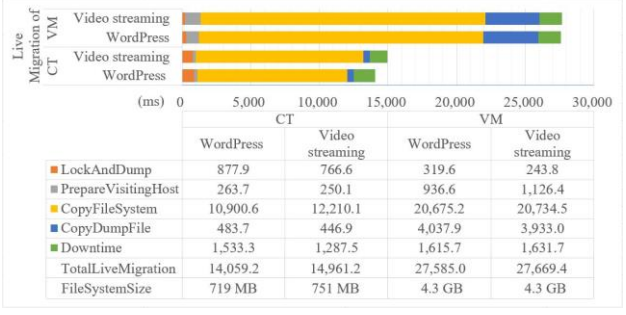
چون کانتینر از محیط کاربر[۱۴] برای راه‌اندازی کانتینرها استفاده می‌کند با استفاده از CRIU[۱۵] می‌توان مهاجرت کانتینرهای LXC/LXD را به‌صورت مهاجرت زنده میان گره‌ها انجام داد [9].

در این پژوهش با استفاده از LXC/LXD و CRIU و استفاده از یک توزیع سبک از لینوکس به نام آلپاین[۱۶] به‌عنوان قالب[۱۷] تلاش می‌کنیم که پهنای باند مصرفی و زمان مهاجرت سرویس را کاهش دهیم.

۲- کارهای مرتبط

در پژوهش [10] امکان سنجی پیاده سازی رایانش مه با استفاده از کانتینر بر روی رسپری‌پای[۱۸] بررسی شده است. این مقاله از دو تکنولوژی داکر و چهارچوب[۱۹] که توسط اپاچی برای درگاه[۲۰] دستگاه های Iot به نام کورا[۲۱] معرفی شده است ، استفاده میکند. که پلتفرمی که برای پیاده سازی این سیستم به کار برده است، استفاده از چیپ‌های رسپری‌پای است و در این مقاله به مشاهدات کیفی و کمی که این روش پیاده سازی داشته است، پرداخته می‌شود. به این صورت که به دلیل ضعیف بودن سخت افزار سیستم مه از کانتینر برای اضافه کردن اجزا، که نیاز حال رایانش مه است، استفاده می‌شود و به صورت پویا این اجزا تغییر می‌کنند بدون آن که کل نرم افزار در هنگام به روز رسانی از دسترس خارج شود و پیچیدگی اعمال تغییرات در شبکه مه کاهش یابد.

در پژوهش[7] یک روش زمان بندی وتخصیص منابع در رایانش مه با استفاده از کانتینر را بررسی شده است. یکی از راه های پیاده سازی این روش



	CT		VM	
	WordPress	Video streaming	WordPress	Video streaming
LockAndDump	877.9	766.6	319.6	243.8
PrepareVisitingHost	263.7	250.1	936.6	1,126.4
CopyFileSystem	10,900.6	12,210.1	20,675.2	20,734.5
CopyDumpFile	483.7	446.9	4,037.9	3,933.0
Downtime	1,533.3	1,287.5	1,615.7	1,631.7
TotalLiveMigration	14,059.2	14,961.2	27,585.0	27,669.4
FileSystemSize	719 MB	751 MB	4.3 GB	4.3 GB

شکل(۲): مقایسه زمان و میزان داده منتقل شده در مهاجرت میان ماشین‌مجازی و کانتینر [8]

استفاده از ماشین مجازی است ولی چند مشکل را برای این نوع طراحی معرفی کرده است. اولین مشکل زمان چند ثانیه ای برای به رسیدن به در حالت آماده به کار ماشین مجازی است و ایراد دیگر که در پژوهش ذکر شده این است که با افزایش تعداد ماشین های مجازی کارایی این سیستم کاهش پیدا میکند و یکی از دلایل دیگر پر هزینه بودن مانیتورینگ ماشین مجازی است. به همین منظور مقاله استفاده از راهکار دیگری در مجازی سازی که از نظر پیاده سازی سبک تر است به نام کانتینر معرفی کرده است. به گفته مقاله ماشین مجازی از ابرناظر استفاده میکند اما کانتینر از سی‌جی‌گروپ که یک قابلیت در کرنل لینوکس است استفاده میکند که توانایی ایزوله کردن منابع دارد. این قابلیت این امکان را به ما میدهد که منابع را در زمان اجرای کانتینر هم افزایش و یا کاهش دهیم. به طور کلی مزیت های کانتینر در این سیستم ها به شرح زیر است یک این که شروع به کار کانتینر سریع تر از ماشین مجازی است زیرا چیزی به نام ابرناظر وجود ندارد. دوم این که کانتینر از نظر کارایی بهتر از ماشین مجازی است. بنا به پژوهش ای‌بی‌ام[۲۲]، کانتینر در بخش پردازنده، حافظه و I/O بهتر عمل کرده است ولی با اختلاف خیلی کمی در بخش شبکه کارایی کمتری داشته است. در حجم بالای بارگذاری از نظر تعداد بر روی سرور کارایی ماشین مجازی به مراتب دچار کاهش میشود ولی مثلا بارگذاری صد کانتینر تاثیر زیادی بر روی کارایی آن ها نمیگذارد.

در پژوهش [11] به‌منظور رفع مشکل پهنای باند و افزایش سرعت انتقال داده‌ها در مهاجرت کانتینر LXC از MTCP[۲۳] استفاده‌شده است. کاری که در این پژوهش انجام‌شده ترکیب لینک‌های شبکه بر روی گره‌ها به‌منظور افزایش پهنای باند و درنتیجه افزایش سرعت انتقال داده در میان گره‌ها است. با استفاده از MTCP این امکان را فراهم کرده است که لینک‌های شبکه را باهم ترکیب کنند و داده‌ها را به‌طور هم‌زمان از دو لینک انتقال دهند. البته در این پژوهش به حجم داده‌های منتقل شده و قالب مورداستفاده برای کانتینر LXC اشاره‌ای نشده است. و درنهایت در بهترین حالت به ۲۰ درصد سرعت بیشتر و در بدترین حالت به حدود یک درصد افزایش سرعت منجر شده است. همچنین احتمال به مشکل خوردن مهاجرت سرویس را در اثر از بین رفتن یک لینک در هنگام جابه‌جایی کاهش داده است.

در پژوهش[8] یک روش برای جابه‌جایی میان گره‌های متحرک در شبکه WAN معرفی کرده است که توسط یک سری گره به نام گره‌های دسترسی این جابه‌جایی انجام می‌شد. همچنین با توجه به شکل (۲) زمان در دسترس نبودن[۲۴] در هنگام جابه‌جایی سرویس‌ها در کانتینر کمتر از ماشین مجازی است و از سمتی دیگر میزان استفاده از پهنای باند در کانتینر به‌مراتب بسیار کمتر از ماشین مجازی است و چیزی در حدود یک پنجم است.

همچنین عمده زمان مصرف شده در زمان مهاجرت سرویس، مربوط به جابه‌جایی فایل‌های سیستمی سرویس بوده است.

در ادامه پژوهش[8] ، در این مقاله به منظور کاهش زمان مهاجرت پردازنده و همچنین کاهش مصرف پهنای باند سعی شده است با استفاده از یک توزیع کمینه[۲۵] از لینوکس به نام آلپاین به‌عنوان قالب کانتینر و LXC/LXD به عنوان مدیریت کننده کانتینر، میزان استفاده از شبکه را با کاهش حجم کانتینر، کاهش دهیم.

۳- روش پیشنهادی

با توجه به نتایج به دست آمده در پژوهش [8] بیشترین دلیل افزایش زمان مهاجرت مربوط به جابه‌جایی فایل‌های سیستمی کانتینر می‌باشد. به همین منظور در این پژوهش راهکاری پیشنهاد شده است که میزان این فایل‌های سیستمی را کاهش و در مواردی در صورت عدم نیاز انتقال داده‌های سیستم آن‌ها را منتقل نکند. این راهکار استفاده از یک توزیع کمینه برای قالب کانتینر و همچنین استفاده از یک مدیریت کننده کانتینر با قابلیت کاهش میزان داده انتقالی در بستر شبکه است. برای این منظور، از مدیریت کننده کانتینر LXC/LXD استفاده می‌کنیم که از طریق API موجود بر روی نرم افزار دسترسی به مهاجرت کانتینر را برای ما میسر می‌کند و قابلیت این را دارد که کانتینر را به دو صورت، بنا به موجودیت فایل قالب در گره ای که مهاجرت صورت می‌گیرد، انجام دهد. در حالت اول ابتدا از کانتینر با استفاده از CRIU یک تصویر لحظه‌ای[۲۶] کامل[۲۷] در هنگام کار از کانتینر گرفته و آن را فشرده کرده و به گره هدف بر روی بستر شبکه منتقل می‌نماید. در این انتقال به دلیل فشرده سازی کانتینر قبل از انتقال بر بستر شبکه، مقداری از زمان مهاجرت به فشرده سازی کانتینر مصرف می‌شود و اگر گره های پردازشی قویتری در لایه مه داشته باشیم باعث بهبود سرعت مهاجرت می‌شود، گرچه حتی بر روی گره های ضعیف تر نیز سرعت این مهاجرت در مقایسه با سرعت مهاجرت با روشی که در پژوهش [8] تفاوت چندانی نمی‌کند و حتی امکان بهبود سرعت را دارد، زیرا در شبکه مه میزان پهنای باند به دلیل بی سیم بودن ارتباطات میان گره ها، محدود است و فشرده سازی در کانتینر باعث کاهش چشمگیر در حجم داده انتقالی می‌شود. در حالت دوم به دلیل وجود قالب در گره هدف عملکرد مهاجرت تغییر می‌کند و میزان جابه‌جایی داده را به‌صورت بسیار زیادی کاهش می‌دهد. ابتدا با استفاده از CRIU یک تصویر لحظه‌ای بی تابعیت[۲۸] از کانتینر می‌گیرد. سپس بر روی گره هدف و با استفاده از قالب موجود از کانتینر در حال مهاجرت، یک کانتینر ایجاد می‌کند. در مرحله بعد تصویر لحظه‌ای گرفته شده را از طریق شبکه به گره هدف منتقل کرده و بر روی کانتینر جدید ایجادشده بازنشانی می‌کند. بهترین عملکرد در مهاجرت کانتینر مربوط به این روش است زیرا در مهاجرت فقط یک تصویرلحظه ای بی تابعیت از کانتینر منتقل می‌شود که بنا به سرویس مورد نظر و میزان تغییرات نسبت به کانتینر اولیه، در بیشتر مواقع از حجم کلی کانتینر کمتر است. همچنین با مقایسه قالب‌های مختلف برای کانتینرهای LXD به یک قالب به نام آلپاین رسیده‌ایم که یک توزیع لینوکس کامل را بر پایه بیزی‌باکس[۲۹] و ماسل لیب‌سی[۳۰] شکل داده است و برخلاف توزیع‌های گنو/لینوکس[۳۱] مانند اوبونتو[۳۲] از حجم پایین‌تری برخوردار است. استفاده از این توزیع به‌عنوان قالب برای کانتینرهایمان، حجم فایل‌های سیستمی مربوط به سیستم عامل را به‌شدت پایین می‌آوریم به‌طوری‌که حجم اشغالی از دیسک این توزیع به‌عنوان کانتینر برابر ۱۱ مگابایت است و حتی استفاده از آن به‌عنوان سیستم عامل مستقل حدود ۱۳۰ مگابایت حجم را اشغال می‌کند و از میزان حجم کانتینر اوبونتو که بدون هیچگونه نرم افزار دیگری، برابر ۴۳۹ مگابایت است، کمتر است. به همین منظور این توزیع به دلیل کاهش حجم اولیه کانتینر، در حدود ۴۲۵ مگابایت، می‌تواند سرعت مهاجرت زنده را به دلیل کم کردن حجم داده‌های منتقل شونده کاهش دهد. به‌منظور کم کردن پهنای باند مصرفی در زمان مهاجرت زنده کانتینر از LXC/LXD استفاده شد که دو نوع حالت برای مهاجرت زنده در اختیار ما قرار می‌دهد. زمانی که در گره هدف قالب کانتینر وجود داشته نداشته باشد و حالتی قالب کانتینر در گره مقصد وجود داشته باشد.

محیط آزمایش بر روی یک سرور با پردازنده Intel(R) Xeon(R) CPU E5-2698 v4 و همچنین 128GB حافظه[۳۳] پیاده‌سازی شده است و سرور از مجازی سازی ESXi 6.7 استفاده می‌کند. بر روی سرور دو ماشین مجازی با مشخصات دو پردازنده مجازی[۳۴] و همچنین 2GB حافظه قرار داده شده است و بر روی هرکدام از آن‌ها اوبونتو نسخه هسته ۱۸.۰۴ نصب شده است. بر روی هرکدام از گره‌ها نسخه ۳.۰.۳ از LXC/LXD [12] به‌منظور مدیریت کانتینرها و همچنین نسخه ۳.۶ از CRIU به‌منظور فراهم‌سازی مهاجرت زنده کانتینر نصب شده است.

با استفاده از LXC/LXD یک کانتینر از توزیع آلپاین ساخته و برای شباهت آزمایش نسبت به پژوهش قبلی یک سرویس وردپرس[۳۵] با استفاده از ابزارهای موردنیاز آن (انجین‌اکس[۳۶]، ماریادی‌بی[۳۷] و پی‌اچ‌پی[۳۸]) پیاده‌سازی کرده‌ایم و مقادیر حجم داده‌های جابه‌جاشده و همچنین زمان این عمل برای نتیجه‌گیری جمع‌آوری‌شده است. به منظور به دست آوردن مدت زمان انتقال کانتینر از دستور time در گره مبدا استفاده شده است. همچنین از wireshark برای اندازه گیری میزان داده منتقل شده بین دو گره استفاده شده است.

٤- نتیجه‌گیری

حجم داده کانتینر نسبت به پژوهش قبلی از 719MB به 438MB تقلیل یافت و همچنین حجم قالب به 68MB رسید. مهاجرت در حالت اول به

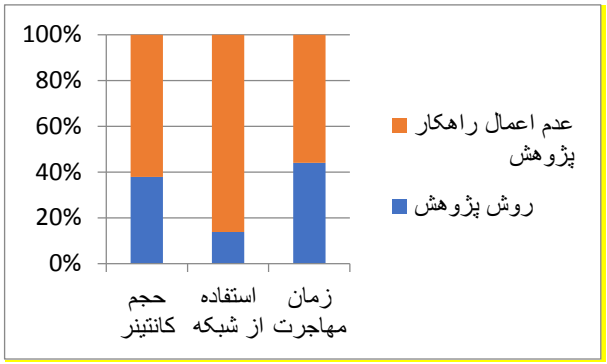
دلیل فشرده‌سازی قبل از جابه‌جایی اطلاعات کانتینر، میزان انتقال داده را در شبکه به عدد ۷۲ مگابایت می‌رساند و در مقایسه با حجم اولیه ۴۳۸ مگابایتی کانتینر، حدود ۸۵٪ کاهش انتقال داده صورت می‌گیرد. مهاجرت در روش دوم انتقال به دلیل عدم انتقال کامل قالب و حجم بسیار پایین تصویر لحظه‌ای، میزان جابه‌جایی داده بسیار کاهش پیدا می‌کند. به‌طور مثال در روش اول حدود ۷۲ مگابایت داده بر بستر شبکه منتقل می‌شود ولی در این حالت میزان داده منتقل شده در حدود ۳ مگابایت است و این روش در مقایسه به حالت اول حدود ۹۵٪ کاهش جابه‌جایی داده بر بستر شبکه را می‌دهد.

در جدول (۱) میزان زمان انجام مهاجرت در حالت‌های مختلف جابه‌جایی کانتینر قرار داده شده است که نسبت به پژوهش قبلی که در حدود ۱۴ ثانیه زمان مهاجرت طول می‌کشد، با در نظر گرفتن حالت اول این عدد به ۱۲ ثانیه، حدود ۱۵٪ کاهش زمان مهاجرت و در حالت دوم ۱۰.۵ ثانیه، حدود ۲۵٪ کاهش زمان مهاجرت صورت گرفته است.

همانطور که در شکل (۳) آمده است، حجم کلی کانتینر به عدد 438MB رسید که در مقایسه با پژوهش قبلی که این عدد 719MB بوده، کاهش حدود ۴۰درصدی در حجم کانتینر انجام‌شده است. همچنین در حالت اول زمان مهاجرت زنده از ۱۴ به ۱۲ ثانیه تقلیل یافته است و در حالت دوم، این عدد برابر ۱۰ ثانیه بوده است. همچنین به دلیل انتقال حداقلی داده‌ها در بستر شبکه، استفاده از شبکه در حالت اول، نسب به پژوهش قبلی که کل داده‌های کانتینر منتقل می‌شد، با حساب حجم کانتینری که با استفاده از یک توزیع کمینه به نام آلپاین حجم به مقدار 438MB رسید، چیزی حدود 70MB جابه‌جایی داشته‌ایم. یعنی در حدود ۸۵ درصد کاهش مصرف پهنای باند نسبت به روش قبلی به دست آمده است.

جدول (۱): زمان مهاجرت در حالت اول و دوم

زمان(S)	حجم داده منتقل‌شده(MB)	حالت
۱۰.۵۸	۴	دوم
۱۲.۴	۷۲	اول
۱۰.۸۳	۲.۸	دوم
۱۲.۲۹	۶۸	اول
۱۱.۵	۲	دوم
۱۳.۶۳	۷۰	اول
۱۰.۴۳	۳	دوم



شکل(۳): مقایسه درصدی بهبود های حاصل شده در مقایسه با عدم اعمال راهکار پژوهش

به‌طورکلی با استفاده از یک توزیع کمینه از لینوکس به نام آلپاین و همچنین استفاده از LXC/LXD به همراه CRIU برای مهاجرت زنده کانتینر مصرف پهنای باند ما به اندازه یک هشتم کاهش یافته است همچنین مقدار زمان مهاجرت زنده نیز به‌اندازه ۱۵ درصد و در بهترین حالت حدود ۳۰ درصد بهبود داشته است.

به دلیل فشرده‌سازی و همچنین جابه‌جایی حداقلی داده هنگام مهاجرت زنده این روش مناسب‌تری برای مهاجرت زنده سرویس‌ها در لایه مه می‌باشد.

۵- پیشنهاد

در شبکه‌های مه، به دلیل ارتباطات بی‌سیم میان گره‌ها، امکان از بین رفتن و یا اختلال در ارتباط در زمان مهاجرت زنده وجود دارد و ازآنجایی که مکانیسم موجود در LXC/LXD برای این نوع اختلال گاهی با اشکال مواجه می‌شود می‌توان با استفاده از روش‌هایی در لایه‌های پایین‌تر شبکه بدون تغییر در ساختار LXC/LXD و مهاجرت زنده آن، این مشکلات را کم کرد. به‌طور مثال بررسی استفاده از MTCP[۳۹] در این روش و میزان تحمل‌پذیری در زمان خطای شبکه، بررسی شود. کار دیگری که می‌توان در ادامه این پژوهش انجام داد استفاده از یک سیستم مسیریابی و یا تغییر آی‌پی و تغییر اطلاعات دامنه سرویس است. در شبکه مه به دلیل تفاوت در دامنه آی‌پی بین منطقه‌های مختلف، باید راهکاری تعبیه شود که پس از مهاجرت زنده سرویس در گره‌هایی با دامنه آی‌پی مختلف، سرویس همچنان در دسترس باشد. زیرا در مهاجرت زنده آی‌پی سرویس به خودی‌خود تغییر نمی‌کند.

مراجع

[1] D. Evans, “The Internet of Things How the Next Evolution of the Internet Is Changing Everything,” no. April, 2011.
[2] “Cisco Global Cloud Index: Forecast and Methodology, 2014–2019,” 2015.
[3] A. Lbath, M. Nahri, A. Boulmakoul, L. Karim, and A. Lbath, “IoV distributed architecture for real-time traffic data analytics IoV distributed architecture for real-time traffic data analytics,” *Procedia Comput. Sci.*, vol. 130, pp. 480–487, 2018, doi: 10.1016/j.procs.2018.04.055.
[4] A. M. Rahmani *et al.*, “Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog

computing approach," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 641–658, Jan. 2018, doi: 10.1016/J.FUTURE.2017.02.014.

[5] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *J. Netw. Comput. Appl.*, vol. 98, pp. 27–42, Nov. 2017, doi: 10.1016/J.JNCA.2017.09.002.

[6] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. K. R. Choo, and M. Dlodlo, "From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework," *IEEE Access*, vol. 5, pp. 8284–8300, 2017, doi: 10.1109/ACCESS.2017.2692960.

[7] L. Yin, J. Luo, and H. Luo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," *IEEE Trans. Ind. Informatics*, vol. 14, no. 10, pp. 4712–4721, Oct. 2018, doi: 10.1109/TII.2018.2851241.

[8] J. Ha, J. Park, S. Han, and M. Kim, "Live migration of virtual machines and containers over wide area networks with distributed mobility management," in *ACM International Conference Proceeding Series*, 2018, pp. 264–273, doi: 10.1145/3286978.3286996.

[9] S. Pickartz, N. Eiling, S. Lankes, L. Razik, and A. Monti, "Migrating LinuX containers using CRIU," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9945 LNCS, pp. 674–684, doi: 10.1007/978-3-319-46079-6_47.

[10] P. Bellavista and A. Zanni, "Feasibility of Fog Computing Deployment based on Docker Containerization over RaspberryPi," in *Proceedings of the 18th International Conference on Distributed Computing and Networking - ICDCN '17*, 2017, pp. 1–10, doi: 10.1145/3007748.3007777.

[11] Y. Qiu, C. H. Lung, S. Ajila, and P. Srivastava, "LXC Container Migration in Cloudlets under Multipath TCP," in *Proceedings - International Computer Software and Applications Conference*, 2017, vol. 2, pp. 31–36, doi: 10.1109/COMPSAC.2017.163.

[12] "Linux Containers." [Online]. Available: https://linuxcontainers.org/. [Accessed: 21-Jul-2019].

زیرنویس

[1] Zetta bytes
[2] Internet of vehicle
[3] Fog
[4] Migration
[5] Container
[6] Share storage
[7] Container
[8] User space
[9] Docker
[10] OpenVZ
[11] Linux Contaner
[12] Boot
[13] Hypervisor
[14] User space
[15] Check point/Restore in users pace
[16] Alpine
[17] Template
[18] RaspberryPi
[19] Framework
[20] Gateway
[21] Kura
[22] IBM
[23] Multipath TCP
[24] Downtime
[25] Minimal
[26] Snapshot
[27] Stateful
[28] Stateless
[29] BusyBox
[30] Musl libc
[31] Gnu/linux
[32] Ubuntu
[33] Ram
[34] VCPU(Virtual CPU)
[35] Wordpress
[36] Enginx
[37] Mariadb
[38] PHP
[39] Multipath Transmission Control Protocol