طراحی و پیاده سازی ماشین مجازی گسترده برای کامپیوترهای شبکه ای

چکیده

این مقاله ، انگیزه ، معماری و عملکرد ماشین مجازی گسترده (DVM) برای کامپیوترهای شبکه ای را توصیف میکند . DVM برای تامین نیازهای اداره پذیری ، امنیت و یکنواختی خوشه های ناهمگن بزرگ کامپیوترهای شبکه ای، برپایه معماری سرویس گسترده ، استوار هستند. در DVM ، سرویس های سیستم ، از قبیل تایید، تقویت امنیت ، کامپایل و بهینه سازی ، خارج از این بخش بندی کاربرد پذیری سیستم ، نیازهای منبع به مشتریان شبکه را کاهش داده ، امنیت سایت را از طریق جداسازی فیزیکی بهبود بخشیده و اداره پذیری شبکه بزرگ و ناهمگن را بدون قربانی کردن عملکرد ، افزایش میدهد . DVM ما ، ماشین مجازی Java را پیاده کرده ، در پردازشگرهای X86 و DEC Alpha اجرا کرده و مشتریان توانمند شده با Java موجود را پشتیبانی میکند.

1- مقدمه

2- ماشین های مجازی (VM ها) ، دارای این پتانسیل هستند که در محیط های محاسبه شبکه ای فردا، نقش مهمی را ایفاء نمایند. روند های جاری نشان میدهند که شبکه های آینده ، احتمالاً با کد متحرک [Thorn 97] ، تعداد زیادی میزبان های شبکه ای در هر حوزه [ISC 99] و تعداد زیادی دستگاه برای هر کاربر که معماریهای سخت افزار متفاوت و سیستم های عامل تحت پوشش قرار میدهند، شناخته میشوند . نوع جدید ماشین های مجازی که به وسیله سیستم هایی از قبیل Inferno , Java الگوسازی میشوند، اخیراً معلوم شده که نیازهای یک چنین محیطی را تامین میکنند. این ماشینهای مجازی مدرن بسیار جالب هستند چون یک فرمت دودویی مستقل از پلاتفرم خاص فراهم میکنند، تضمین ایمنی قوی که اجرای ایمن برنامه غیرقابل اطمینان و مجموعه گسترده ای از واسط های برنامه سازی را که شامل واسطه های سیستم عامل چند منظوره هستند، سهولت میبخشد.

3- قابلیت بارگذاری پویا و اجرای ایمن برنامه غیرقابل اطمینان ، ماشین مجازی Java را در سیستم های گسترش پذیر از سارقین شبکه و سرورها تا موتورهای پایگاه داده ای و کاربردهای اداری ، جزء فراگیر می سازد. عدم وابستگی ماشین های مجازی مدرن به پلاتفرم ، اجرای برنامه های کاربردی مشابه را در دامنه وسیعی از دستگاههای محاسبه ازجمله سیستم های تعبیه شده ، سامانگرهای دستی، پلاتفرم های رومیزی قراردادی و سرویس دهندههای موسسه با محصولات گرانقیمت ، امکانپذیر میکند. علاوه بر این ، پلاتفرم اجرای منفرد پنانسیل سرویس های مدیریت یکپارچه را فراهم نموده و بنابراین برخی از مدیران سیستم را برای اداره کردن موثر هزاران یا حتی صدها هزار دستگاه قادر می سازد.

4- در حالیکه ماشین های مجازی مدرن، آیندۀ امیدوارکننده ای را ارائه میکنند، حال حاضر تا حدودی مبهم است. مثلاً ماشین مجازی Java ، علیرغم موفقیت تجاری و فراگیربودن خود ، معایب مهمی دارد.اولاً ، ولو اینکه ماشین مجازی Java برای

دستگاههای دستی و سیستم های تعبیه شده طراحی شد، به واسطهٔ پردازش بیش از حد و شرایط حافظه ، هنوز در این زمینه چندان جا نیفتاده است.

5- ثانیاً ، یافتن یک ماشین مجازی Java ، بجای اینکه یک اصل باشد ، استثناء است.

ثالثاً ، به جای ساده سازی ادارهٔ سیستم ، ماشین های مجازی مدرن ، مانند Java ، مسأله مدیریتی مهمی ایجادکرده که این امر موجب شده اکثر سازمانها به راحتی همهٔ ماشین های مجازی را غیر مجاز پندارند.

ما تایید میکنیم که این علائم ، نتیجه یک مساله بزرگ است که ذاتاً در طرح ماشنی های مجازی مدرن وجود دارد. خصوصاً ماشین های مجازی بسیار پیشرفته ، بر پایه معماری یکپارچه اسلاف خود استوار هستند کل مولفه های سرویس در VM یک پارچه از قبیل تایید ، مدیریت امنیت ، کامپایل و بهینه سازی ، بطور موضعی در میزبانی قرار دارند که برای اجرای برنامه های کاربردیVM در نظر گرفته شده است .

یک چنین معماری سرویس یکپارچه ، چهار عیب دارد:

1. اداره پذیری: چون هر ماشین مجازی مدرن یک ماهیت کاملاً مستقل است، نقطه کنترل مرکزی در سازمان وجود ندارد. روش های شفاف و جامع برای توزیع امنیت سطح بالا، گرفتن مسیرهای حسابرسی و خلاصه کردن شبکه برنامه های کاربردی نامطلوب به سختی اجرا میشوند.

2. عملکرد: سرویس های ماشین مجازی مدرن ، از قبیل معتبر سازی ، کامپایل به موقع و تایید ، دارای پردازش اصلی و ملزومات حافظه ، میباشند.

نتیجتاً ، سیستم های یکپارچه برای میزبانهایی از قبیل دستگاههای تعبیه شده مناسب نیستند چون برای پشتیبانی ماشین مجازی کامل ، فاقد منابع هستند.

3. امنیت: مبناء محاسبه معتبر (TCB) VM های مدرن ، کوچک ، خوش – تعریف یا از لحاظ فیزیکی جدا از برنامه کاربردی نیست TCB بزرگ با حدود بد – تعریف شده ، ساخت و تایید سیستم های ایمن را دشوار می سازد فقدان تمایز بین مولفه های

ماشین مجازی به این معناست که نقص در هر مولفه از ماشین مجازی میتواند کل ماشین را در معرض خطر قرار دهد.

علاوه بر این ، همگذاری سرویس های VM ، به سیستم های غیر مدولی منجر شده که میتوانند بر همکنش های بین مولفه ای پیچیده ای نشان دهند همانطوریکه در مورد سیستم های عامل یکپارچه ، مشاهده شد.

4.مقیاس پذیری: ماشین های مجازی یکپارچه در معماریهای متنوع و پلاتفرم های موجود در شبکه خاص به سختی انتقال می یابند . گذشته از این ، آنها در مقیاس بندی ملزومات کاربرد متفاوت در سازمانها ، مشکل داشته اند.

هدف تحقیق ما، عبارت از توسعه سیستم ماشین مجازی است که اداره پذیری، عملکرد، امنیت و ملزومات مقیاس پذیری محاسبه شبکه ای را معرفی میکند علاوه بر این ، چنین سیستمی باید سازگاری با مبناء وسیع ماشین های مجازی یکپارچه موجود را حفظ کنند تا استقرار را سهولت بخشند به این منظور، تکنیک های اجرا را مورد

بحث قرار میدهیم که واسط های بیرونی و API های پلاتفرم ماشنی های مجازی موجود را حفظ میکنند.

مسائل مربوط به ماشین های مجازی یکنواخت را با معماری ماشین مجازی گسترده نوین براساس فاکتورگیری و توزیع ، معرفی میکنیم. معماری سرویس گسترده ، سرویس های ماشین مجازی را درمولفه های منطقی فاکتورگیری کرد.

این سرویس های را از مشتری انتقال داده و آنها را در سرتاسر شبکه توزیع میکند. یک ماشین مجازی گسترده را برای Java براساس این معماری، طراحی و اجرا نمودیم DVM ما شامل زمان اجرای Java ، تایید کننده ، سامانگر ، سرویس کنترل عملکرد، و مدیر امنیت میباشد. آن از این لحاظ با سیستم های موجود متفاوت است که این سرویس ها در مولفه های خوش – تعریف فاکتورگیری شده و به هنگام ضرورت ، متمرکز میشوند.

بقیه مقاله بصورت زیر تدوین شده است بخش بعدی، معماری ما را توصیف کرده و سیستم ما را بطور کلی بررسی میکند. بخش 3 ، پیاده سازی سرویس های ماشین

مجازی قراردادی را تحت معماری ما ، توصیف میکند. بخش 4 ، ارزیابی این معماری را ارائه کرده و بخش 5 نشان میدهد که چطور سرویس بهینه سازی جدید را میتوان تحت این معماری تطبیق داد. بخش 6 ، کار مربوطه را بحث نموده و بخش 7 ، نتیجه گیری میکند.

2. بررسی کلی معماری

شناخت مهم در ارتباط با تحقیق ما اینست که سرویس های متمرکز ، مدیریت سرویس را از طریق کاهش تعداد و توزیع جغرافیایی واسط هایی ساده میکنند که باید به منظور اداره کردن سرویس ها، قابل دستیابی باشند . همانطوریکه با آرایش گستردهٔfirewall ها در دههٔ گذشته نشان داده شد، اداره کردن یک میزبان جا افتاده در شبکه نسبت به اداره کردن هر مشتری آسانتر است. بطور قابل مقایسه ، ماشین های مجازی یکپارچه را به مولفه های سرویس منطقی آنها تفکیک کرده و این مولفه ها را به خاطر مشتریان در سرویس دهنده های شبکه فاکتورگیری میکنیم.

معماری سرویس ماشین مجازی ، تعیین میکند که سرویس ها کجا، کی و چطور، اجرا میشوند . محل (یعنی کجا) ، زمان درخواست (یعنی کی) ، و اجرا (یعنی چطور) سرویس ها ، بوسیله اداره پذیری ، یکپارچگی و ملزومات عملکرد کل سیستم محدود شده و مستلزم ارزیابی های مهندسی میباشد . ماشین های مجازی یک پارچه ، نقطه طراحی خاصی را نشان میدهند که آنها همهٔ سرویس ها برای مشتریان قرار داشته و اغلب کاربردپذیری سرویس از جمله کامپایل fly و بازبینی امنیت، طی زمان اجرای برنامه های کاربردی ، اجرا میشود. در حالیکه این مقاله مزایایی تعیین محل سرویس ها را در داخل شبکه نشان میدهد ،تغییر محل سرویس ها بدون توجه به اجرای آنها ، میتواند بطور قابل توجهی عملکرد را نیز کاهش دهد ، بعنوان مثال ، یک روش ساده برای توزیع سرویس ، جائیکه سرویس ها در امتداد واسط های موجود تفکیک شده و بطور سالم به میزبانهای دورحرکت میکنند، احتمالاً بواسطهٔ هزینه ارتباط از راه دور از طریق پیوندهای بطور پتانسیل کند و تناوب بر همکنش های بین مولفه ای درماشین های مجازی یکپارچه، خیلی گزاف است. یک طرح جایگزین را توصیف میکنیم که

درآن کارآمدی سرویس از مشریان از طریق بخش بندی سرویس ها به مولفه های ایستا و پویا فاکتورگیری شده و یک استراتژی اجرایی را ارائه میکنیم که درمقایسه با ماشین های مجازی یکپارچه به عملکرد دست می یابد.

در ماشین مجازی گستردهٔ ما سرویس ها در سرویس دهنده های متمرکز قرار داشته و بخش عمدهٔ عملیات خود را قبل از اجرای برنامه کاربردی ، بطور ایستا اجرا میکنند مولفه های سرویس ایستا ، از قبیل تایید کننده ، کامپایلر، حسابرس ، پروفیلر، و بهینه سازی ، بخش آموزشی برنامه های کاربردی را قبل از اجرا بررسی میکنند.

تا اطمینان ایجاد کنند که برنامه کاربردی ، خواص سرویس مطلوبی را نشان میکنند بعنوان مثال، یک تایید کننده ممکن استکد تایپ – ایمنی را بازبینی کند. سرویس ایمنی شاید آرگومان های قابل تعیین بطور ایستاتیک در فراخوانی های سیستم بررسی کند. و یک بهینه ساز شاید ساختار کد را برای عملکرد خوب در امتداد یک مسیر خاص بازبینی کند.

مولفه های سرویس پویا، کاربردپذیری سرویس را طی اجرای برنامه های کاربردی فراهم میکنند. آنها مولفه های سرویس ایستاتیک را از طریق فراهم سازی سرویس هایی تکمیل میکنند که باید در زمان اجرای برنامه کاربردی در بافت مشتری خاص، اجرا شوند. مثلاً یک سرویس ایمنی شاید آرگومان های ارائه شده توسط کاربر را برای فراخوانی های سیستم بازبینی کند، یک پروفیلر شاید استاتیک زمان اجرا را جمع آوری نموده و سرویس حسابرس ممکن است براساس اجرای برنامه کاربردی ، رویدادهای حسابرسی را تولید نماید.

چسبی که مولفه های سرویس ایستا و پویا را به یکدیگر پیوند میدهد، بازنویسی دودویی است. وقتی مولفه های سرویس ایستا با عملیات وابسته به داده ها مواجه میشوند که بطور ایستا قابل اجرا نیستند ، فراخوانی‌ها را در مولفه های سرویس پویایی متناظر درج میکنند. مثلاً ، سرویس تایید ایستای ما، مطابقت برنامه های کاربردی را با شمخصات Java VM، کنترل می کند. جائیکه بازبینی ایستا نمی تواند امنیت برنامه را بطور کامل تأیید کنندهٔ ایستا، برنامه کاربردی حاصل، نتیجتاً خود تأیید

کننده است چون بازبینی های تعبیه شده بوسیلۀ مألفه سرویس ایستا، جزء لاینفک برنامه کاربردی هستند.

شکل 1، معماری ماشین مجازی گستردۀ ما را نشان می دهد. مؤلفه های سرویس ایستا، برنامه کاربردی خود – سرویس دهنده تولید می کنند که مستلزم کاربرد پذیری حداقل در مشتری ها هستند. مؤلفه های سرویس پویا، به هنگام لزوم، کاربردپذیری سرویس را طی زمان اجرا برای مشتریان، فراهم می کنند. سرویس ایستا در معماری ما در خطر لوله مجازی مرتب می شوند که در برنامه کاربردی عمل می کند همانطوریکه در شکل 2، ملاحظه می شود.

معماری سرویس گسترده، اجازه می دهد تا انبوه کاربرد پذیری سرویس VM در جایی قرار گیرد که خیلی مناسب است. استراتژی جاگذاری سرویس طبیعی عبارت از ساختاری کردن مؤلفه های سرویس به صورت پروکسی شبکه شفاف است که در میزبان ایمن از لحاظ فیزیکی، اجرا می کند. یک چنین پروکس که مانند Fire wall

در حد اطمینان شبکه قراتر دارد، می تواند تبدیل های برنامه را در کل برنامه بطور شفاف اجرا کند که در یک سازمان وارد می شود.

در برخی محیط ها، یکپارچگی برنامه های کاربردی تغییر شکل یافته را نمی توان بین سرویس دهنده و مشتری تضمین کرد. یا اینکه کاربرها ممکن است برنامه را در شبکه ای وارد کنند که بوسیلهٔ سرویس های ایستا، فرآوری نشده است. در چنین محیط های، امضاهای دیجیتالی الصاق شده با مؤلفههای سرویس ایستا می توانند اطمینان ایجاد کنند که بازبینی ها از برنامه های کاربردی جدا نشدنی بوده و مشتریان می توانند برای جهت دهی مجدد برنامه نادرست امضاء شده یا امضاء نشده، در سرویس های متمرکز، آموزش داده شوند.

DVM، مقدار متوسطی از کاربرد پذیری جدید را در مبناء محاسبه قابل اطمینان موجود یک سازمان، وارد می کند. مشتری DVM باید مطمئن شود که مؤلفه های سرویس ایستا و پویایی که برای ایمنی متکی بر آنها می باشد از جمله پروکس و بازنویس دودویی، بطور صحیح تحقیق می یابند.

علاوه بر این، هر طرح تأیید سرویس بکار رفته در این مشتریان، که ممکن است شاهد بازبین امضاء دیجیتال و مدیر اصلی باشد، بخشی از پایه محاسبه قابل اطمینان را طبق طرح ما، تشکیل می دهد. به هر حال، معتقدیم که اثر واقعی این افزایشات در TCBجزئی است. مشتریان یکپارچه تقریباً به همه مؤلفه های سرویس اطمنان دارند که VM معمولی را شکل داده و اغلب از قبل تدارکاتی برای پروتکل های سرویس اطمینان دارند که بازبینی امضاء دیجیتالی جهت پشتیبانی کاربردهای هدف خود، دارا می باشند. بطور کلی، افزایش متعادل در TCB، مشتریان DVM را قادر می سازد تا مؤلفه های قابل اطمینان را به میزبان های از لحاظ فیزیکی ایمن و از لحاظ حرفه ای مدیریت و اداره شده، انتقال دهند که برای نشان دادن مسائل عملیاتی که VM های یکپارچه را مختل کرده اند، خیلی مهم است.

معماری سرویس ما از چندین نظر، بی نظیر است. اولاً سرویس های متمرکز برای همهٔ مشتریان در یک سازمان الزامی هستند. مثلاً، بازبینی های ایمنی درج شده در رمز ورودی، در زمان اجرای آنها، از برنامه های کاربردی الینفک هستند و بنابر این

درسرتاسر شبکه پیوند می دهند. ثانیاً، برای همهٔ ماشین های مجازی در یک سازمان، فقط یک نقطه کنترل منطقی وجود دارد. در مورد سرویس ایمنی، خط مشی ها از یک محل مشخص شده و کنترل می شوند؛ نتیجتاً، تغییرات خط مشی مستلزم همکاری کاربرهای عادی نیست. ثالثاً، استفاده از بازنویسی دودویی به عنوان یک مکانیسم پیاده سازی سرویس، سازگاری با ماشین های مجازی یکپارچه موجود را حفظ می کند. ماشین مجازی یکپارچه شاید برنامه بازنویسی شده در بازبینی ها یا سرویس های اضافی را شامل شود، اما می تواند از کاربرد پذیری افزون بدون تغییر، استفاده کند.

در حالیکه معماری سرویس گسترده، مسائل موجود در ماشین های مجازی یکپارچه را معرفی میکند، شاید چالش های جدیدی را نیز مطرح کند. متمرکز سازی می تواند مخل عملکرد شده یا به یک نقطه خرابی در شبکه منجر گردد.

این مسائل را می توان از طریق پیاده سازی سرویس دهنده تکراری و بازسازی پذیر نشان داد. بخش بعدی نشان می دهد که چطور جداسازی بین مؤلفه های ایستا و پویا را می توان برای انتخاب کاربردپذیری مستلزم تحقیق مخلی را برای شبکه های متوسط

حتی در بدترین مورد، ایجاد نمی کند و برای تطبیق با تعداد زیادی از میزبان ها به سهولت تکرار پذیر است.

3- سرویس ها

معماری توصیف شده در بخش قبلی را برای پشتیبانی شبکه ماشین های مجازی Java (JVM ها)، محقق کرده ایم. در این بخش، تحقیق سرویس های ماشین مجازی قراردادی را تحت معماری خود توصیف کرده و نشان می دهیم که تحقیق گسترده این سرویس ها، معایب VM های یکپارچه توصیف شده در بخش اول را، نشان می دهد. سرویس های ما از ویژگی Java VM مشتق می شوند که عمدتاً محیط اجرای مقصد گرای نوع ایمن، مشتق می شوند. تحیقق های خاص از یک تأیید کننده تشکیل می شوند که برنامه مقصد را برای نوع ایمنی بازبینی می کند و یک مفسر و مجموعه ای از کتابخانه های زمان اجرا. در برخی تحقیق ها، به مفسر یک کامپایلر درست به موقع برای بهبود عملکرد، اضافه می شود. بخش های بعدی، طرح و تحقیق سرویس هایی

را که ایجاد کرده ایم، توصیف میکنند تا جای آنهایی که در ماشین های مجازی Java معمولی یافت میشوند، بگیرند.

همهٔ سرویس های ما بر پایه زیر ساخت های پروکس معمولی استوار هستند که مؤلفه های سرویس ایستا را جای می دهد. این پروکس بطور شفاف درخواست های برنامه را از مشتریان قطع کرده، کدهای بایتی JVM را تجزیه نموده و در فرمت دودویی مناسب، برنامه اندازه گیری شده، تولید می کند . یک API پالایش داخلی اجازه می دهد تا سرویس های از لحاظ منطقی جداگانه توصیف شده در این بخش، در میزبان پروکس ترکیب شوند. تجزیه و تولید برنامه برای هر سرویس ایستا فقط یک بار اجرا می شوند، در حالیکه ساختاری کردن سرویس ها به صورت صافی های تبدیل کد مستقل آنها را قادر می سازد تا طبق شرایط خاص سایت پشته ای شوند. این پروکس برای خوداری از بازنویسی کد مشترک بین مشریان، از یک حافظه نهانگاهی استفاده کرده و برای کنسول اداری راه دور، یک مسیر حسابرسی تولید می کند. کد مولفه های

سرویس پویا در پروکسی مرکزی قرار داشته و به مشتریان مورد تقاضا، توزیع می شود.

در حالیکه جزئیات تحقیق سرویس های ماشین مجازی ما عمدتاً متفاوت هستند، در میان همهٔ آنها، سه موضوع مشترک وجود دارد:

- مکان: فاکتور گیری سرویس های VM برای مشتریان و تعیین محل آنها در سرویس دهنده ها، مدیریت پذیری را از طریق کاهش حالت تکراری بهبود داده، از طریق جداسازی سرویس ها از کد بطور بالقوه نامطلوب، به یکپارچگی کمک کرده و توسعه و تدارک سرویس را ساده می کند.
- ساختار سرویس: بخش بنید سرویس ها به مؤلفه های ایستا و پویا می تواند عملکرد را برای استهلاک قطعات پر هزینه سرویس در همهٔ میزبان های شبکه محلی، تقویت کند.
- تکنیک تحقیق: بازنویسی دودویی بری تحقیق شفاف سرویس ها، بکار می رود. سرویس های بازنویسی دودویی را می توان برای ایجاد سربار عملکرد

نسبتاً کوچک، طراحی کرد در حالیکه سازگاری رو به عقب با مشتریان موجود، حفظ می شود.

3-1 تأیید

مجموعه کامل محدودیت های ایمنی به ماشین مجازی اجازه می دهد تا کد بطور بالقوه نامطلوب را در سیستم پایه ممتاز، تلفیق کند. در واقع، درخواست Java برای محاسبه شبکه اساساً از تضمین ایمنی قوی آن ناشی می شود که با تأیید کنندهٔ Java تقویت می شود.

تکلیف تأیید بایت کد Java، چالشی برای ماشین های مجازی یکپارچه بوده است. اولاً، چون ویژگی Java در توصیف آن راجع به اصول ایمنی رسمی نیست، تفاوتهایی بین تحقیق های تأیید کننده وجود دارد.

تأیید کننده ها از فروشندگان متفاوت در مورد موضوعات کمتر مشخص شده از قبیل محدودیت های استفاده از مقصدهای آغاز نشده، فراخوانی های زیر روال و معتبر سازی متقابل داده های اضافه در فایل های طبقه بندی شده، متفاوت هستند. ثانیاً،

تحقیق های یکپارچه، تأیید کننده را به بقیه VM پیوند داده و به این طریق از کاربرد تأیید کننده های قویتر به هنگام لزوم توسط کاربران، جلوگیری می کند. گذشته از این، تأیید کننده های یکپارچه، انتشار پچ های ایمنی را به همهٔ مشتریان در زمان معین، دشوار می سازد.

به عنوان مثال، 15٪ کل دستیابی به سایت شبکه ما از طریق سارقین اطلاعات منسوخ با حفره های ایمنی معروف ناشی می شود که اکثر پچ ها برای آنها منتشر شده اند. نهایتاً، حافظه و شرایط پردازش تأیید، VM های یکپارچه را برای مشتریان محدود منبع، نامناسب ارائه می کند از فبیل کارت های هوشمند و میزبان های تعبیه شده. برخی ماشین های مجازی یکپارچه برای سیستم های تعبیه شده و محدود منبع، تأیید را بطور یکجا برای مدل گسترش محدود براساس اطمینان، ترک کرده اند.

این معایب را از طریق تجزیه تأیید از بقیه VM، انتقال کاربردپذیری آن از مشتریان در سرویس تأیید شبکه و متمرکز سازی اداره این سرویس، معرفی می کنیم.

انتقال تأیید از مشتریان، چالش هایی را موجب می شود، چون بخش هایی از فرایند تأیید مستلزم دستیابی به فضای نام مشتری بوده و معمولاً مستلزم تزویج با JVM مشتری بوده خصوصاً تأیید Java شامل چهار مرحله جداگانه است. سه مرحله اول در یک فایل رده بندی ذاتاً سازگار بوده، اینکه کد فایل رده بندی از یکپارچگی آموزش پیروی کرده و اینکه کد نوع ایمن است. مرحله چهارم، واسط هایی را بازبینی می کند که یک رده در برابر امضاء های نوع صادر شده در فضای نام آن، وارد کرده و اطمینان حاصل می کند که فرض هایی که این رده در مورد رده های دیگر تقبل کرده، طی پیوند صدق می کنند.

در تحقق ما، سه مرحله اول تأیید در سرویس دهندهٔ شبکه که بطور ایستا اجرا شده در حالیکه بازبینی های زمان استباط بوسیلهٔ یک مؤلفه پویای کوچک در مشتری، اجرا می شوند. این بخش بندی کاربردپذیری، ارتباط غیر ضروری را برطرف کرده و تحقق سرویس را سهل می کند طی فرآوری سه مرحله اول، سرویس تأیید همهٔ فرض هایی را که یک رده در مورد محیط خود تقبل کرده، جمع آوری نموده و حوزهٔ این فرض

ها را محاسبه می کند. مثلاً، فرض های اساسی، از قبیل روابط توارث، بر اعتبار ردهٔ کامل تأثیر می گذارد، در حالیکه رفرنس فیلد فقط بر دستورالعمل هایی تأثیر می گذارد که بر پایه رفرنس استوار هستند. با تعیین این فرض ها و حوزهٔ آنها، سرویس تأیید، کد را اصلاح میکند تا بازبینی های متناظر را در زمان اجرا از طریق ایجاد یک مؤلفه سرویس ساده، انجام دهد. چون اکثر اصول ایمنی تا بحال بازبینی شده اند، کاربردپذیری در مؤلفه پویا به مراجعه توصیفگر و مقایسه رشته، محدود می شود. این طرح کند برای معوق کردن بازبینی های فاز پیوند اطمینان ایجاد می کند که رده هایی که یک برنامه کاربردی تشکیل می دهند، از سرویس دهندهٔ دور دست و بطور بالقوه کند واکنش نمی شوند مگر اینکه برای اجرا لازم باشند.

سرویس تأیید گسترده، خطاها را در مشتری از طریق ارسال به جلوی ردهٔ جایگزین منتشر می کند که طی آغازیدن آن، یک استثناء تأیید ایجاد می کند. بنابر این، خطاهای تأیید از طریق مکانیسم های منظم استثناء Java در مشتریان، منعکس می شوند. چون ویژگی Java VM، زمان و روش تأیید را عمدتاً تعریف نشده بر جای می گذارد به

جز اینکه بازبینی ها باید قبل از اینکه هر برنامه تحت تأثیر اجرا شود، انجام یابندکه روش ما با آن ویژگی، مطابقت دارد.

در حالیکه این روش برای تأیید، کار اصلی تأیید را آسانتر نمی کند، مسائل عملیاتی را معرفی میکند که مخل مشتریان یکپارچه شده اند. اولاً، آن به شبکه VM ها اجازه می دهد تا تأیید کنندۀ مشترکی داشته باشند، بنابراین اطمینان ایجاد می کند که کد پذیرفته شده برای اجرا در آن حوزۀ اداری، حداقل به اندازۀ محدودیت های اعمال شده توسط تأیید کنندۀ مرکزی، قوی هستند. بازنویسی دودویی شفاف اطمینان ایجاد می کند که حتی VM های یکپارچه موجود از تضمین فراهم شده توسط تأیید کنندۀ مرکزی بهره مند می شوند، اگرچه این کد را در تأیید محلی اضافی، مشمول می سازند. ثانیاً، سرویس تأیید مجزا با واسط های مشخص با استفاده از تکنیک های تست اتوماتیک، در مورد صحت، به سهولت بازبینی میشود. ثالثاً، مشتریان ماشین مجازی گسترده را می توان کوچکتر و ارزانتر ساخت چون مجبور نیستند تأیید کننده را پشتیبانی کنند. نهایتاً، در پاسخ به نقایص ایمنی، فقط یک مولفه نرم افزاری باید در هر حوزۀ اداری به هنگام

سازی می شود برخلاف وصله کردن هر مشتری شبکه منفرد، که اغلب مستلزم مساعدت کاربران، می باشد.

2-3 ایمنی

هدف کلی هر سرویس ایمنی عبارت از اجرای خط مشی ایمنی خاص در میزبان های موجود در یک حوزهٔ اداری است. یک چنین سرویسی باید سه هدف زیر را تأمین کند تا کامل، به سهولت قابل مدیریت و تطبیق پذیر باشد. اولاً، آن باید بطور یکنواخت خط مشی ایمنی سازمان را در همهٔ گره ها اجرا کند. ثانیاً، باید یک نقطه کنترل منفرد برای مشخص کردن خط مشی های سازمان، فراهم کند. و ثالثاً، باید به مدیر اجازه دهد تا بازبینی هایی را در هر کد مهم برای ایمنی، فراهم کند.

معماری مااین اهداف را از طریق فاکتورگیری بخش عمدهٔ کاربرد پذیری تامین میکند که برای ایمنی گرههای منفرد در سرویس ایمنی شبکه متمرکز ، مهم است سرویس ایمنی، کاربردها را برای تطبیق با خط مشی ایمنی سازمان از طریق درج بازبینی های دستیابی مناسب در بازنویسی دودویی ، سوق میدهد برنامه های کاربردی ایمن سپس

در میزبان های منفرد اجرا میکنند جائیکه مدیر اجرای کوچک ، بازبینی های دستیابی و درج شده را مطابق با خط مشی متمرکز، اجرا میکند.

مدل ایمنی ما از DTOS مشتق میشود جائیکه شناسه های ایمنی که حوزه های حفاظت را نشان میدهند، با رشته ها و مقاصد مهم ایمنی، همراه هستند. نوشته میشود، ماتریس دستیابی را مشخص میکندکه شناسه های ایمنی را به مجوزها مربوط میسازد که تعیین میکند چه کسی کدام عملیات را انجام دهد. این خط مشی همچنین نگاشت بین منابع نامگذاری شده و شناسه های ایمنی را مشخص میکند که محدودیت ها را درمنابع نامگذاری از قبیل فایلها و نگاشت بین عملیات ایمنی و کد کاربرد را تعیین میکند که محل درج بازبینی های دستیابی را برنامه های کاربردی تعیین میکند. سرویس ایمنی، این خط مشی را تفکیک کرده و برنامه های کاربردی وارده را بازنویسی کرده ، فراخوانی ها را در مدیری اجرایی متد و حدود ایجاد کننده را درج میکند بطوریکه دستیابی های منبع از بازبینی های دستیابی مناسب، تبعیت میکنند طی اجرای برنامه کاربردی بازنویسی شده ، مدیر اجرایی ، بازبینی های دستیابی درج شده

را اجرا نموده، سرویس ایمنی را براساس شناسه‌های ایمنی و مجوزهایی که نگهداری میکند، پرس و جو می نماید، در نتیجه ، سرویس ایمنی، انبود کاربردپذیری را برای فرآوری خط مشی اجرا کرده و بازبینی مشتری را به کنترل ساده و موثر کاهش میدهد تغییرات خط مشی که مستلزم اندازه گیری مسیرهای کد جدید هستند که نامتناوب به نظر می آیند، مستلزم این هستند که برنامه های کاربردی با استفاده از سرویس اداری از راه دور، دوباره شروع شوند.

طرح ما ، دو عیب اصلی سرویس های ایمنی را در JVM های یکپارچه، نشان میدهد. اولاً ، در این سیستم ها، خط مشی ایمنی مشخص شده و بطور جداگانه در هر گره منفرد ، اجرا میشود . نتیجتاً ، مدیر سایت باید حالت ایمنی را برای هر گره به طور مجزا حفظ کند که متناسب با تعداد گرهها، اقدام میکند.

ثانیاً ، قابلیت و طرحهای کنترل موجودی مورد استفاده در JVM های یکپارچه، خط مشی ایمنی را به تحقق این سیستم پیوند داد. و بنابراین مساعدت از طرف توسعه دهندگان سیستم اصلی را برای کار، ضروری میکند. اساساً ، توسعه دهندگان باید

شرایط ایمنی سیستم را پیش بینی کرده و بازبینی های ایمنی لازم را در مکانهای مناسب کتابخانه های سیستم تعبیه کنند در مورد Java ، توسعه دهندگان JVM های معروف ، بازبینی های ایمنی را پیش بینی و در فایل، شبکه و دستیابی های سیستم، ارائه کرده اند. به هر حال ، ابزارهای صوتی و رویدادهای ایجاد پنجره، بطور کافی پشتیبانی نشده اندکه انکار حملات سرویس را موجب میشوند. بطور کلی، مکانیسم های کنترل دستیابی که به پیش بینی ارجح و سنجش دستی بستگی دارند، احتمالاً کمتر انعطاف پذیر بوده و نسبت به سیستم هایی که جداسازی واضحی بین خط مشی ایمنی و تحقق فراهم میکنند، امنیت کمتری دارند.

3-3- کنترل از راه دور

اداره سستم خصوصاً برای شبکه های بزرگ سیستم های ناهمگن، چالش برانگیز است. تکالیفی از قبیل حسابداری منبع و تحلیل الگو کاربرد اگر چه در سیستم های مشترک زمانی گذاشته به سهولت اجرا میشوند، جهت اجرا در سیستم های گستردهٔ امروزی، دشوار هستند. سرویس کنترل از راه دور ما به مدیران اجازه میدهد تا کاربرد

منبع را در سرتاسر شبکه پیگیری کنند. سرویس کنترل از راه دور که پس از سرویس ایمنی الگوسازی میشود ، برنامه های کاربردی را تغییر شکل میدهد تا در مکانهای مناسب ، سرویس های حسابرس ایجاد کند از قبیل ورود به فراخوانی های متد و سازنده و خروج از آنها. وقتی هر برنامه کاربردی بالا می آید، با کنسول کنترل از راه دور در تماس بوده و یک پروتکل توافق نهایی ، اعتبارنامه های کاربر را برقرار کرده و برای این دوره کار یک شناسه تخصیص میدهد. این ارتباط بعداً برای ارسال اطلاعات به میزبان ادارهٔ مرکزی بکار می رود که پیکربندیهای سخت افزار مشتری ، کاربرها، موارد JVM نسخه های کد و رویدادهای مهم مشتری را کنترل میکند. چون لگاریتم های حسابرسی در یک میزبان بیرونی ذخیره میشوند که درمعرض برنامه های کاربردی غیرقابل اطمینان نیست، نقض ایمنی شاید ایجاد رویدادهای حسابرسی جدید را متوقف کرده اما نتواند لگاریتم های حسابداری موجود را فشرده سازی کند.

علاوه بر سرویس های کنترل از راه دور سطح متد، برای کنترل عملکرد برنامه کاربردی یک سرویس پروفیل کننده و ردیاب سطح دستورالعمل ارائه میکنیم. پروفیلر،

کدی را برای تولیدنمودار فراخوانی پویا از برنامه های کاربرد که در شبکه اجرا میشوند و همچنین آمار در کاربرد کد مشتری ، ایجاد میکند. از این سرویس ها برای اشکال زدایی سیستم خود و برای بهینه سازی ، بطور گسترده استفاده کرده ایم. خصوصاً ، از سرویس ردیابی برای بدست آوردن مسیرهای رفتار همگام سازی برنامه های کاربردی Java استفاده نموده و از داده های در طراحی سرویس بهینهسازی شفاف ، بهره جسته ایم.

3-4- کامپایل

کامپایل درماشین های مجازی یکپارچه نوعاً بوسیله کامپایلر درست به موقع (JIT) اجرا میشود که بایت کدها را به فرمت اصلی در صورت نیاز ، ترجمه میکند.

چون کامپایل در مشتریان برحسب تقاضا امجام میشود، فشارهای زمان و منبع قابل توجهی وجود دارد. نتیجتاً ، کامپایلرهای JIT نوعاً بهینه سازیهای اگر سیر انجام نمی دهند اگر چه تکنیک های کامپایل جدیدتر، تبادل زمانی بین کامپایل ، تفسیر و اجرا را از طریق تمرکز بر نقاط داغ در یک برنامه نشان میدهند، مشکلات اساسی اعمال شده

بوسیلهٔ محدودیت های زمانی و فقدان توان پردازش در میزبان های تعبیه شده ، هنوز به قوت خود باقی هستند.

در فرایند اجرای کامپایلر شبکه متمرکز هستیم که مسئولیت کامپایل را از مشتریان برطرف می سازد. همانطوریکه در بخش قبل توصیف شد، مشتریان معمولاً توافق نهایی را با سرویس اداره از راه دور انجام میدهندکه در آن، فرمت اصلی آنها را توصیف میکنند. یک کامپایلر در شبکه میتواند ترجه را برای آن پلاتفرم به موقع انجام دهد و بنابراین هزینه های راه اندازی را درمقادیر قابل توجه برنامه، استهلاک کند. سرمایه گذاریهای منبع در کامپایلر به نفع همهٔ مشتریان یک سازمان ، میباشد.

3-5- خلاصه

بحث قبلی، تحقق سرویس های ماشین مجازی قراردادی را در معماری ما توصیف کرده و مزایای تعین محل سرویس های NM را در مکانهای دقیقاً انتخاب شده در شبکه نشان داد. متمرکزسازی به مدیریت و اداره کمک کرده ، واسط یکنواختی را برای مشتریان ناهمگن فراهم نموده و یکپارچگی سرویس ها را از طریق تعیین محل آنها

درمیزبانهای قابل اطمینان از لحاظ فیزیکی، افزایش میدهد. بخش بندی سرویس های NM قراردادی به مولفه های ایستا و پویا، اجرای بخش عمدهٔ کاربرد پذیری سرویس را با هزنه ثابت به هنگام اولین بارگذاری یک برنامه کاربردی ، امکانپذیر میسازد.

7. نتایج کلی

معماری سیستم جدیدی را برای محاسبه شبکه براساس ماشین های مجازی گسترده طراحی و اجرا کرده ایم. سیستم ما سرویس های ماشین مجازی را از مشتریان فاکتورگیری کرده و آنها را در سرویس دهنده های شبکه سازمانی ، قرار میدهد. این سرویس ها با محدود کردن برنامه کاربردی و اصلاح آن در fly برای ارائه کاربردپذیری سرویس ، عمل میکنند. این مقاله نشان میدهدکه ماشین های مجازی گسترده میتوانند ملزومات منبع مشتری را کاهش داده، مدیریت را آسان کرده و سرویس های مهم ایمنی را از کد غیرقابل اطمینان و بطور بالقوه نامطلوب، جداسازند. استراتژی تحقق سرویس خاص ما برپ=ایه فاکتورگیری سرویس های NM از

مشتریان ، بخش بندی آنها به مولفه های ایستا و پویا و اجرای آنها در بازنویسی دودویی ، استوار است.

این روش، سرویس های NM متنوع را با عملکرد قابل مقایسه با ماشین های مجازی یکپارچه ، پشتیبانی میکند.