توزیع کد برنامه دینامیک در ابرهای فراساختار به شکل یک خدمات

چکیده

کاربردهای محاسبه ابر با مقیاس بندی الاستیک در چشم انداز IT امروزی بیشتر و بیشتر شایع شده است. یک مسئله ایجاد چنین کاربردهایی در یک ابر فراساختار به شکل یک خدمات توزیع زمانی کد برنامه ، فایلهای پیکربندی و سایر منابع است. درحالیکه امکان پذیر است که کلیه کد برنامه لازم را در تصاویر پایه IaaSبکاررفته بگنجانیم این امر به شدت پویایی قابل دسترسی را در زمان اجرا محدود می سازد. در این مقاله ما یک چارچوب برای توزیع کد برنامه دینامیک ارائه می دهیم. رهیافت ما با توزیع کد به طور کامل و شفاف در لایه نرم افزاری سروکار دارد. ما راه حل خود را طبق لایه نرم افزاری موجود به نام CloudScale بنا نهاده ایم. این مقاله درباره طراحی و اجرای رهیافت توزیع کدمان در راس CloudScale بحث می کند و از لحاظ عددی قابلیت کاربرد و اجرای این رهیافت را بر اساس مطالعه موردی توضیحی ارزیابی می کند.

1-مقدمه

در سالهای اخیر، پیشرفت محاسبه کامپیوتری به شیوه ابری کل صنعت IT را تغییر شکل داده است و فرصتها و توانایی های جدیدی را برای پدیدآورندگان و کاربران فراهم کرده است. وانگهی، محاسبه کامپیوتری به شیوه ابری اجرای ایده های نواورانه را برای شرکتهای کوچک یا افراد ساده سازی و هزینه های تولید و نگهداری برای کاربردهای صنعتی را پایین نگه داشته است. به دلیل محاسبه کامپیوتری به شیوه ابری، انعطاف پذیری منابع یک خصوصیت کاربردها به جای یک شرح مراکز داده ها شده است. ایده انعطاف پذیری برای کاربردهای به شیوه ابری به قابلیت انعطاف پذیری هزینه، منابع و کیفیت تقسیم بندی شده و ابعاد و توانایی های دیگری برای پدیدآورندگان نرم ا فزار جهت بهینه سازی کاربرد و خدمات بدست می دهد.

محاسبه کامپیوتری به شیوه ابری یک انتخاب تکنولوژیکی نویدبخش برای پروژه های تدوین کاربرد نوین است ولی اگر یک کاربرد بالواقع وجود داشته باشد، هزینه های جابجایی یا مهاجرت باید قبل از اینکه بتوان مزیت های ابر را افزایش داد، درنظر گرفته شود. جابجایی یا مهاجرت ابر را یک مسئله چالش اور می دانند. اغلب جابجایی یا مهاجرت مسائل معماری را اشکار می کند و نیاز به فاکتورگیری مجدد و طراحی مجدد کاربردها دارد. اما حتی زمانی که معماری کاربرد بالواقع با پارادیگم محاسبه کامپیوتری به شیوه ابری متناسب باشد، اندکی کار بیشتر برای اینکه کاربرد کاملا از ابر سود برد، لازم است.

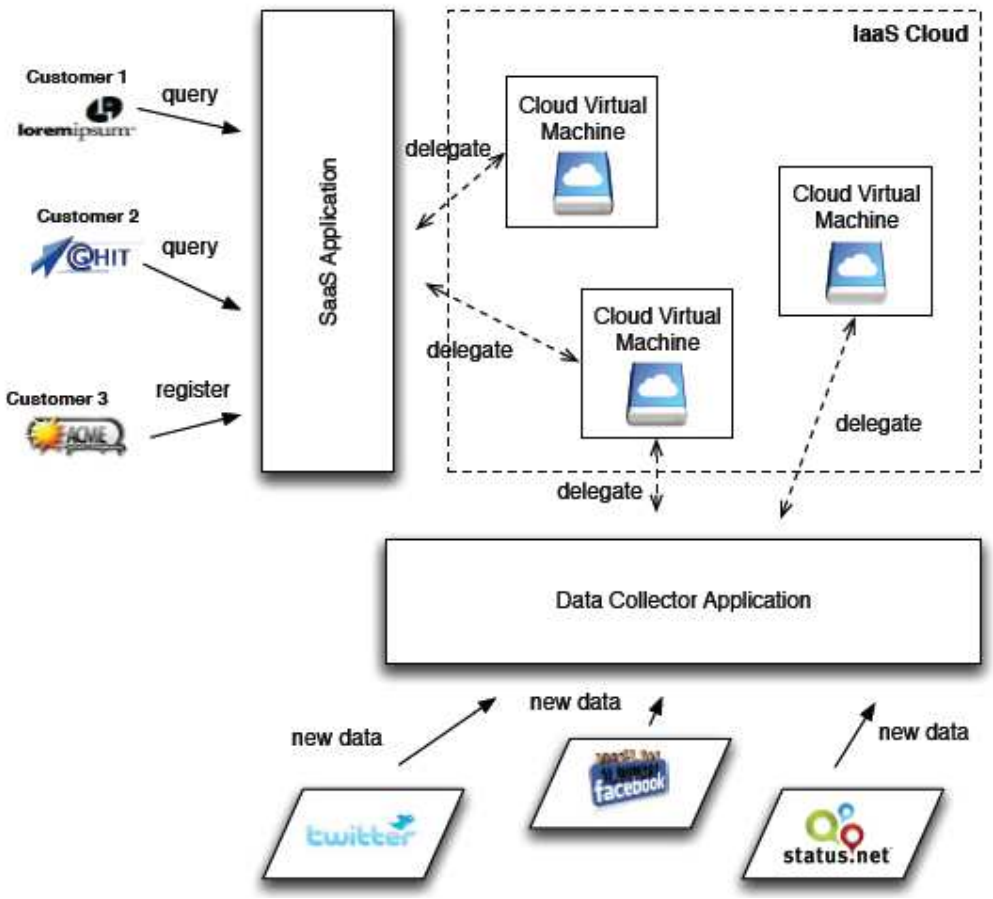
اساسی ترین مدل خدمات ابری رهیافت فراساختار به شکل یک خدمات یا IaaS می باشد. در این سطح، فراهم کنندگان خدمات ابری ماشین های مجازی دارای پیکربندی درخواستی و سیستم عملیاتی درخواستی را (معمولا به شکل تصویر درایو هارد) برای براورده سازی الزامات محاسبه کامپیوتری آن کاربرد ارائه می دهند. این لایه برای جابجایی یا مهاجرت ابر مرجح است به طوری که به تلاش جابجایی یا مهاجرت کمتری نسبت به پلتفورم به شکل یک خدمات یا PaaS نیاز دارد (و بهتراستانداردسازی می شود.) هنگامی که یک کاربرد IaaS نیاز به مقیاس بندی داشته باشد (یعنی از ماشین های مجازی بیشتری نسبت به قبل استفاده کند) یک مسئله این است که چگونه قابلیت دسترسی نسخه کنونی کد کاربردی، فایلهای پیکربندی و سایر منابع را می توان روی میزبان جدید تضمین نمود. در ذیل، ما از اصطلاح «کد برنامه» به شکل کوتاه نویسی شده برای کد کاربرد و کلیه فایلهای وابسته استفاده می کنیم. رهیافت سه تایی گنجاندن این کد برنامه در تصویر پایه ماشین مجازی است ولی این رهیافت تنها در موقعیتهایی منطقی است که در آن کاملا ایستایی داشته و طی عمر آن کاربرد تغییر شکل نیابد. در عوض، کد برنامه اغلب طی زمان تکامل و تغییر یافته و نسخ مختلف متعددی از یک کاربرد باید به موازات آن قابل اجرا باشد. در چنین سناریوهایی، کدنویسی سخت افزاری برای کد برنامه و سایر فایلها به داخل تصاویر ماشین مجازی به صورت پیچیده درآمده یا حتی غیرممکن است. راه دیگر دستیابی به توزیع کد برنامه گنجاندن تسهیلاتی برای جستجو و توزیع کد دینامیک در سطح لایه نرم افزاری است.

این مقاله به معرفی چارچوبی برای توزیع کد برنامه زمان اجرای بدون نقص می پردازد. چارچوب مبتنی بر پیش نمونه تحقیقاتی اولیه ما از CloudScale می باشد ولی به دلیل سطح عدم وابستگی بالا، می تواند به طور جداگانه در سایر سیستم ها نیز بکار رود. cloudScale یک لایه نرم افزاری است که به ساده سازی کاربرد جاوا در یک ابر IaaS می پردازد. در چارچوب کاری ما، توزیع کد برنامه کاملا با لایه نرم افزاری CloudSclae زیربنایی مورد رسیدگی واقع می شود. ما به ارزیابی گزینه های انتخابی مختلف پیکربندی و اجرا می پردازیم و نتایج عملکرد عددی را براساس یک مطالعه موردی مشروح ارائه می دهیم.

بقیه این مقاله به ترتیب ذیل ساختاربندی شده است. بخش دوم به شرح مطالعه موردی مشروح می پردازد که نشان دهنده مسئله توزیع کد مبتنی بر کاربرد دنیای واقعی است. بخش سوم به شرح تحقیقات مرتبط با کار ما می پردازد. بعد از آن، بخش چهارم زمینه ای از رهیافتی را ارائه می دهد که ما در این مقاله آورده ایم که مهمتر از همه چارچوب کاری CloudWare می باشد. بخش 5 به ارائه نقش علمی واقعی این مقاله می پردازد. که متعاقبا در بخش شش ارزیابی می شود. بالاخره مقاله در بخش 7 نتیجه گیری می شود.

2-مطالعه موردی توضیحی

در بقیه این مقاله ما از یک کاربرد انالیز گرایشی Web 2.0 برای اهداف توضیحی استفاده می کنیم. این کاربرد مطالعه موردی در اصل در رفرانس [14]معرفی شده و درباره سیستم نرم افزار به عنوان یک خدمات (SaaS)مبتنی بر ابر بحث می کند. نرم افزار کاربردی SaaSبه مشتریان اجازه می دهد تا به ثبت خدمات خود را پرداخته و شهرت آنلاین مارک ها و محصولات خود را تحت نظارت قرار دهند. برای این منظور، یک برنامه کاربردی جمع آوری کننده داده ها مطالب محتوای ارسالی به رسانه های اجتماعی مختلف(به عنوان مثال، توییتر، فیس بوک، status.net) را تجزیه و تحلیل خواهد کرد .این داده ها سپس توسط نرم افزار کابردی SaaSبرای تولید گزارش های دقیق بر اساس داده های زمان واقعی مربوطه به عقاید مورد استفاده قرار می گیرد .



تصویر 1-مرور کلی سناریوی انالیز عقاید(رفرانس 14)

یک مرور کلی سطح بالا روی این نوع تنظیمات در شکل 1 نشان داده شده است. مشتریان مختلف دسترسی به برنامه کاربردی SaaSبا درخواستهایی برای گزارشات دارند. به منظور دستیابی به انعطاف پذیری و مقیاس پذیری، نمایندگانی برای درخواست اعلام اجرای وظایف پردازش شدید، مانند تولید گزارش، به تعدادی از ماشین آلات مجازی مختلف وجود دارند، که در تقاضا از یک ابر IaaSاجاره ای، به عنوان مثال، آمازون EC2 وجود دارد به طور مشابه، بخش جمع آوری داده ها از مطالعه موردی استفاده از ماشین های مجازی IaaS را برای بازیابی، عادی سازی وذخیره اقلام فروشگاه ذخیره شده که از شبکه های اجتماعی بازیابی شده، دارد. عادی سازی شامل وظایفی مانند حذف کلمه ایست و ریشه یابی می باشد.

3-کار مرتبط

مسئله توزیع کد روی شبکه تقریبا در عین حال پدیدار می شود به نحوی که ارتباطات شبکه امکانپذیر گردد. ساده ترین راه حلی که نیاز به تلاش پدیدآوری حداقل برای به روز رسانی کد به طور دستی قبل از اعلام اجرا داشته باشد. این راه حل به اندازه کافی برای سیستم هایی خوب است که به ندرت به روز رسانی شده، و یا در شرایطی که سرعت شبکه برای انتقال کد و یا تایید نسخه در زمان اجرا ناکافی باشد .با این حال، با توسعه بیشتر شبکه بندی و برنامه های کاربردی برای آگاهی از شبکه، به روز رسانی خودکار کد متداول شده است. امروزه، این برنامه های کاربردی اغلب از لحاظ به روز رسانی به طور دوره ای و یا در هنگام شروع و راه اندازی بررسی انجام داده، و در صورت لزوم دانلود نسخه های به روز شده کد را صورت می دهند. این رویکرد مناسب است و یک استاندارد برای برنامه های کاربردی کاربر محور متداول شده است، اما در سایر موارد روش های پیچیده تری را می شود استفاده کرد.

یکی از زمینه های علمی که ذاتا مشکل توزیع کد دارد محاسبات کامپیوتری شبکه است. معمولا، وظایف پدید امده برای اجرای شبکه دارای محاسبات فشرده و مدت اجرای طولانی است. بنابراین، این مورد توزیع کد به گره های شبکه مناسب قبل از اعلام اجرا به صورت دستی یا به صورت اتوماتیک قابل کاربرد است. با این حال، برخی از روش ها برای کد برنامه توزیع و داده های اضافی هنگام اجرا در رفرانس [20]مطرح گردید. باز، توزیع کد در محاسبات کامپیوتری شبکه متفاوت از رویکرد ما در حال حاضر است، چون آنها مشکل اجرای طولانی مدت توزیع کد اولیه و یا انجام تکه کار مهم را حل کرده اند، اما ما چارچوبی برای اجرای وظیفه دربرگیرنده ارائه می دهیم، که اجازه می دهد اجرای نسخه های مختلف کد بر روی همان دستگاه صورت گیرد. این امر برای توسعه، تست و سناریوهای کاربرد چندتایی کاربران (چند-مستأجری) مهم است.

روشی که ارائه می دهیم شبیه به ایده عامل متحرک در محاسبات کامپیوتری مبتنی بر عامل است. با استفاده از این پارادایم، برنامه های کاربردی قادر به جابجایی یا مهاجرت از یک کامپیوتر به دیگری به طور مستقل بوده و اعلام اجرای آنها بر روی کامپیوتر مقصد ادامه خواهد یافت. توزیع کد یک مفهوم حیاتی برای چنین برنامه های کاربردی است و بسیاری از تحقیقات برای رسیدن به اهداف مختلف و بهبود مهاجرت کد

انجام شده است .با این حال، در مقایسه با رویکرد ما، عوامل متحرک فعال هستند و خود را برای جابجایی یا مهاجرت بین کامپیوترها در هر زمانی طی اجرای خود انتخاب می کنند. در این چارچوبی که ما ارائه می دهیم، برنامه به طور شفاف توزیع شده است و این آگاهی در دست نیست که کد توزیع شده است یا خیر. از این نقطه نظر، رویکرد ارائه شده بیشتر شبیه به ایده ارزیابی کد از راه دور هنگامی است که یک کار به سرور برای اجرا منتقل می شود .همچنین، رویکرد ما نشان دهنده برخی از ویژگی های کد در رویکرد تقاضاست، هنگامی که کد از دست رفته و فایل های مرتبط را می توان از مکان راه دور مورد تقاضا برداشت و آورد.

سرانجام اینکه، باید یاداوری کرد که راه حل ارائه شده در اینجا در طبقه بزرگتر تحرک کد ضعیف می گنجد چون هر دو کد و داده ها انتقال می یابد ولی انتقال به آن حالت کاربردی نیست.

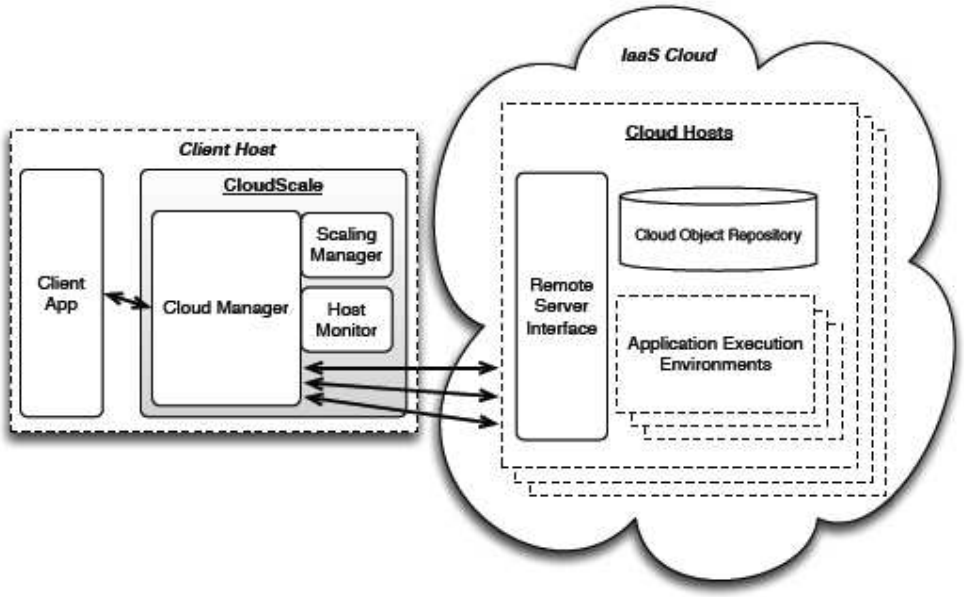
4- زمینه و بافت توزیع

کاری که در این مقاله درباره اش بحث می کنیم درون پروژه تحقیقاتی CloudScale بزرگتری انجام شده که در رفرانسهای 15 و 13 و14 در اغاز شرح داده شده است. در ذیل به معرفی CloudScale تا حد لزوم برای اهداف این مقاله خواهیم پرداخت و بیشتر چالشهایی را تفصیل می کنیم که در این مقاله مطرح گردیده اند.

الف-چارچوب کاری CloudScale

مفهوم کلی از CloudScale استفاده از تکنیک های برنامه نویسی مبتنی بر جنبه (AOP)به صورت پویا برای تغییر بایت کد برنامه های کاربردی مبتنی بر جاوا و حرکت شفاف قطعات تعیین شده از برنامه کاربردی در ابر (که ما به عنوان اشیاء ابر می نامیم) به منابع مجازی (به عنوان میزبان ابر نامیده می شود) می باشد. این فرایندی است که به طور کامل برای پدیده آورنده نرم افزار مشهود بوده و اتفاق آن به طور کامل در زمان اجرای آن برنامه ی کاربردی به طور خودکار می باشد .در پایان، برنامه های کاربردی ساخته شده در راس CloudScale مانند برنامه های کاربردی منظم (محلی) جاوا به نظر می رسد، اما در واقع در یک حالت به شدت توزیع شده اعلام اجرا می شود. مرور کلی معماری ساده سازی شده چارچوب CloudScaleدر شکل 2 نشان داده شده است. در اصل، برنامه های کاربردی سرویس گیرنده دسترسی به

ابر را از طریق یک جزء مدیر ابر محلی دارد. مدیر ابر حرکت اعلام اجرای بخشی از برنامه کاربردی را به میزبان های ابر مختلف در ابر IaaS انجام می دهد. برای این منظور، مدیر ابر نظارت بر عملکرد هر میزبان را انجام داده و برنامه ریزی درخواست ها را به نحوی انجام می دهد که عملکرد کل برنامه کاربردی بهینه باشد، که با توجه به سیاست های تعریف شده توسط پدیدآورنده نرم افزار می باشد.



تصویر 2-معماری ساده سازی شده CloudScale از رفرانس 15

به منظور اجرای این مدل CloudScale، مجموعه ای از چالشهای مورد نیاز هست که باید به آنها رسیدگی شود، از جمله مدیریت ماشین مجازی، برنامه موازی، برنامه ریزی درخواست به ماشین های مجازی، عملکرد و همچنین نظارت بر هزینه، و توزیع کد برنامه .چالش دوم تمرکز اصلی این مقاله است .افکار بیشتر در مورد چالش های دیگر را می توان در مقالات نشریاتی که زودتر درباره CloudScale چاپ شده است یافت.

ب-چالشهای توزیع کد برنامه

کد برنامه با توزیع پویا در یک ابر IaaS زندگی واقعی نیاز به تعدادی از جنبه های عنوان شده دارد. (1) در مرحله اول، این چارچوب نیاز به این تشخیص دارد که چه زمانی که کدی که قرار است اعلام اجرا شود اصلا در دسترس نیست، و درخواست کد را از یک سرور کد (دستگاهی که نسخه صحیح باینری برنامه را

دارد) می نماید. برای مثال، اگر برنامه کاربردی SaaS که در بخش II شرح داده شده بخواهد به نمایندگی و واگذاری تولید گزارشی به یک میزبان ابر واگذار کند، کد برنامه مورد نیاز باید در این ماشین مجازی در دسترس باشد. (2)اگر این مورد برقرار نیست، ابر میزبان باید ذخیره سازی کد مورد اعتماد را که در آن کد مناسب را می توان بازیابی کرد، پیدا کند. در مورد ما، معمولا خود برنامه کاربردی به شکل یک سرور کد عمل می کند و ارائه کد برنامه مورد نیاز مورد تقاضا را انجام می دهد، از جمله نسخه های صحیح و درست از تمامی وابستگی های از دست رفته که مورد نیاز برای اجرای این کد می باشد. در روش دیگر، امکانپذیر است که برای نصب یک سرور اختصاصی کد خاص در ابر اقدام کرد، که پس از آن این کار را از درخواست مشتری برای کاهش بار آن به عهده می گیرد. بدیهی است، امکانپذیر است که کدبندی سخت افزاری تمام کد برنامه مورد نیاز را به طور مستقیم به تصاویر ماشین مجازی استفاده شده توسطCloudScale صورت داد، اما این امر به شدت انعطاف پذیری سیستم را کاهش می دهد و باعث میشود که سیستم خراب شده و زمان بر شود.

جدول 1-خلاصه چالشهای توزیع کد

ردیف	نام چالش	سیناپس چالش
1	شناسایی کد ازدست رفته	میزبانهای ابر باید قادر به شناسایی دینامیک داشته باشند اگر کد برنامه نیاز به بارگذاری طبق تقاضا داشته باشد.
2	ذخیره سازی کد اعتماد شده	میزبانهای ابر باید قادر به مکان یابی خدمات ذخیره کد باشند (معمولا درخواست مشتری یا یک سرور کد اختصاص یافته در ابر)
3	سخت افزار میانی ارتباطاتی	میزبانهای ابر نیاز به داشتن دسترسی به یک لایه نرم افزاری ارتباطاتی مناسب دارند که به آنها امکان بارگذاری دینامیک کد را می دهد.
4	پروتکل ارتباطاتی	میزبانهای ابر و درخواست مشتری نیاز به استفاده از یک پروتکل کارامد برای به حداقل رسانی سرریز ارتباطاتی واقع شده با بارگذاری کد دینامیک دارند
5	نسخه برداری کد	میزبانهای ابر نیاز به اگاهی از این امر دارند که کد برنامه بتواند تغییر کند و اینکه کد برنامه بارگذاری شده به طور معینی روایی نداشته باشد.

(3)ثالثا، برخی وسایل ارتباطاتی نیاز به استقرار دارند که اجازه می دهد کد برنامه در زمان اجرا از ذخیره سازی کد مورد اعتماد به ماشین های مجازی ابر منتقل شود. این ارتباط می تواند یا به حالت نقطه به نقطه (به عنوان مثال، مانند SOAPاز طریق وب سایت فن آوری خدمات) و یا از طریق یک لایه نرم افزاری پیام بر مدیریت شود. (4) چهارم اینکه، به دلایل عملکرد عملی، لایه نرم افزاری به بهینه سازی پروتکل های ارتباطی بین کاربرد و ابر میزبان وجود دارد.

به عنوان مثال، مسئله آن است که به طور معمول امکان پذیر نیست تا شروع روال های تبادل کد پویا به صورت جداگانه برای هر کلاس از دست رفته از کد برنامه صورت گیرد. در عوض، لایه نرم افزاری نیاز به تصمیم گیری هوشمندانه در این مورد دارد که کد های اضافی و منابع غیر فعال (به عنوان مثال، تصاویر، فایل های پیکربندی) نیز علاوه بر کد از دست رفته شناسایی شده برای کار لازم است یا خیر. این وابستگی ها باید روی ابر در عین حال برای به حداقل رساندن سربار توزیع کد توزیع شود. (5) پنجم اینکه، پس از بارگذاری دینامیک و پویا یا Loadingکد برنامه،لایه نرم افزاری نیاز به تصمیم گیری در این باره دارد که چه مدت این کد و وابستگیهای آن را هم اکنون می توانید به عنوان معتبر در نظر بگیرید. برای این منظور، لازم است که لایه نرم ا فزاری CloudScaleقادر به تشخیص زمانی باشد که نسخه دیگری از کد برنامه مورد نیاز است و از آن استفاده کند.

این چالش ها در جدول 1خلاصه شده است. در بقیه این مقاله ، ما رویکردی برای حل این چالش های درون پروژه CloudScale را مورد بحث قرار می دهیم.

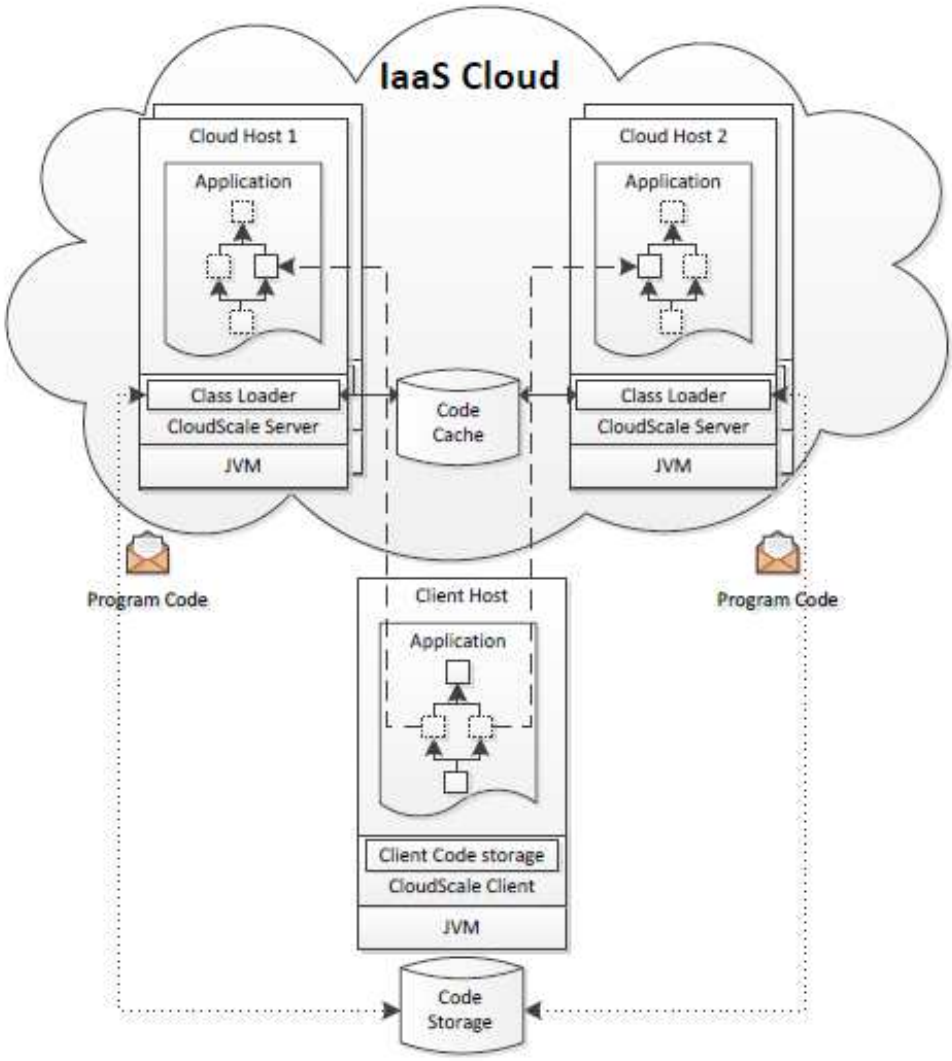
5-توزیع کد برنامه دینامیک

هر گاه یک میزبان ابر لایه نرم افزاری CloudScale برای اجرای یک کار جدید ملزم شده باشد، سیستم به این اطمینان نیاز دارد که تمام منابع لازم در دسترس هستند و برنامه اجرای کار زمانبندی شده است. در این بخش ما به شرح رویکردمان و نشان دادن اینکه چگونه سعی می کنیم به توزیع کد کارآمد و بدون اشکال با حل چالش های توصیف شده در بخش ب از قسمت چهارم برسیم، می پردازیم.

الف-مرور کلی سیستم

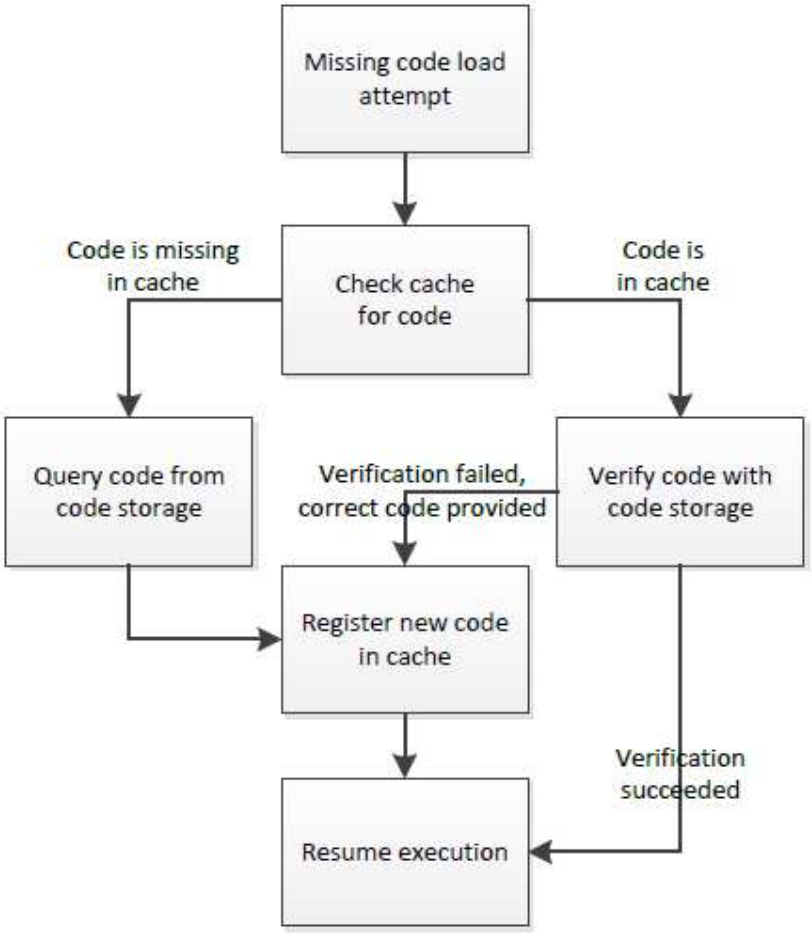
هنگامی که برنامه سرویس گیرنده به یک بخش کد می رسد که می تواند به ابر واگذار گردد، آن برنامه به اعلام اجرا بر روی میزبان ابر زمانبندی انجام می دهد که در حال حاضر در دسترس هستند .اگر هیچ میزبانی وجود ندارد، ماشین آلات ابر های اضافی می توانند آغاز به کار شوند .جزئیات بیشتر در این فرآیند در رفرانس [15]شرح داده شده است .در روی میزبان ابر، کد برنامه ریزی شده شروع به اجرا می کند، در حالی که یک بار کننده طبقه خاص روی سطح پلت فرم نگهداری و بازخوانی تمام کد برنامه مورد نیاز و دیگر منابع مربوطه را، مانند فایل های پیکربندی انجام می دهد. معماری راه حل ما در شکل 3 تصویرسازی شده، که در آن شما می توانید ببینید برنامه از میزبان مشتری آغاز شده و کار توزیع شده به مجموعه ای از ماشین آلات ابر صورت می گیرد که بازیابی کد لازم را از حافظه پنهان کد ابری یا به طور مستقیم از مشتری صورت می دهد. با توجه به این معماری، کدی که در حال اجرا ست لازم نیست که در مورد در دسترس بودن کد و نسخه نگران باشد ، چون زیرساخت بنیانی این مشکلات را بدون عیب و بطور شفاف مورد رسیدگی قرار می دهد.

1-شناسایی کد از دست رفته: در اغلب زبان های برنامه نویسی (از جمله جاوا، که توسط CloudScale استفاده می شود)، تشخیص منابع از دست رفته (هر دو هم فایل کد دار و هم غیر کد دار) را می توان توسط پدید آورنده از طریق رابط کاربردی برنامه نویسی (API) مورد رسیدگی قرار داد. با این حال، به منظور اجتناب از سوء رفتار، حل چالش های اعلام شده و توانایی کنترل در دسترس بودن کد و توالی بار، ما اجرا را براساس API های موجود، یعنی یک مدول خاص در لایه نرم افزاری مان برای رهگیری تمام درخواست ها برای کد برنامه در میزبان ابر انجام داده ایم. مشخصا، ما رهگیری مکانیسم بارگذاری کلاس زبان برنامه نویسی را برای بررسی و مقابله با مجموعه ای از کلاس های در حال حاضر حل و فصل شده انجام داده ایم. اگر کد مورد نیاز در حال حاضر پر شده است، می توان آن را دوباره بدون هیچ کار اضافی مورد نیاز را از بارکننده کلاس فراهم کرد. اگر کد مورد نیاز قبلا طی این اعلام اجرا بارگذاری نشده باشد، بارکننده کلاس یک حافظه پنهانی کد را برای جستجوی آن بررسی می کند، همانطور که در شکل 4 نشان داده شده است. جزئیات این مکانیسم



تصویر 3-مرور کلی مدل توزیع کد برنامه

بعدا توضیح داده خواهد شد. اگر کد در حافظه پنهان یافت نشد، بارکننده کلاس درخواست آن کد را از مشتری (و یا از یک ذخیره سازی کد مورد اعتماد) می نماید، و منتظر پاسخ می ماند (شکل 3).



تصویر 4- راهکار بارگذاری کد

2) لایه نرم افزاری ارتباطاتی. سیستم بارگذاری کد ما هیچ گونه الزامات خاصی برای یک کانال ارتباطی خاص ندارد، و معمولا می تواند روی همان امکانات ارتباطی استفاده شود که توسط بقیه لایه نرم افزاری CloudScaleاستفاده می شود. بارگذاری منابع براساس تماس های مسدود ساده کار می کند و ممکن است توانایی برای شروع ارتباط با تسهیلات ذخیره سازی کد مورد اعتماد را داشته باشد. تنها خواص کانال ارتباطی که برای این مورد استفاده مهم می باشند قابلیت اطمینان و سرعت انتقال داده های مناسب است. سرعت کانال بسیار مهم است چون تاخیر ارتباطی به طور مستقیم مؤثر بر عملکرد نرم افزار در ابرمی باشد. بدیهی است، هر گونه ارتباطات شبکه ای نسبت به بازیابی کد محلی کند است، و هرچه ارتباطات کندتر باشد، مزایای عملکرد که می تواند با توزیع نرم افزار کاربردی روی ابر در وهله اول حاصل شود پایین تر

است. قابلیت اطمینان نیز حیاتی است. خطاهای انتقال، که ابر میزبان می تواند با کمک حاصل جمع بررسی ها شناسایی کند، می تواند به طور چشمگیری بر سرعت ارتباطات به دلیل ارسال مجدد کد اثر بگذارد. در صورت عدم برقراری ارتباط، برنامه نرم افزاری مجبور به خاموش کردن آرام می شود، به طوریکه هیچ کدی به ادامه اجرای وظیفه وجود ندارد.

در نسخه فعلی ما از CloudScale، ما نیاز به یک صف پیام JMSسازگار (به عنوان مثال، Apahce ActiveMQ2) برای ارائه کانال های ارتباطی استفاده شده برای تمام ارتباطات مشتری میزبان، از جمله بارگذاری کد پویا داریم.

3) مکان ذخیره سازی کد اعتماد شده. در حالی که ایجاد سرور کد اختصاصی ممکن است قابلیت اطمینان و عملکرد چارچوب توزیع کد را بهبود بخشد، برای برخی از موارد این راه حل ترجیح داده نمی شود .گاهی اوقات لازم است که قادر به بردن و آوردن کد واقعی به طور مستقیم از برنامه کاربردی CloudScaleباشیم. به عنوان مثال، در طول تدوین نرم افزار و یا آزمایش آن، معنی دار تر است که از دستگاه اولیه راه اندازی برنامه برای توزیع کد به جای سرور اختصاصی استفاده کنیم که قبل از هر اجرا باید به روزرسانی شوند .در این مواقع، ارائه خدمات کد باید از قبال درخواست مشتری فراهم شود. علاوه بر این، برنامه سرویس گیرنده به طور معمول قابل اطمینان ترین منبع کد است چون به عنوان پایگاه کدهای برنامه مشتری یا سرویس گیرنده شامل دقیقا کدی است که پدید آورنده انتظار می رود اجرا کند. بنابراین، به طور پیش فرض، برنامه سرویس گیرنده همیشه اجرای خدمات توزیع کد را انجام می دهد، حتی در شرایطی که یک سرور کد اختصاصی بنابه انتظار استفاده می شود. این امر ساده سازی پیکربندی چارچوب را انجام داده و اجازه می دهد تا با استفاده از نرم افزار کاربردی مشتری به روز رسانی و یا تأیید کد در سرور فعال انجام شود، و یا به عنوان یک گزینه یدکی در مورد سرور کد آفلاین یا سرریز شده استفاده شود.

خدمات ارائه کد در داخل نرم افزار سرویس گیرنده باید قادر به ارائه کد به ماشین آلات ابر بدون ایجاد وقفه در موضوع برنامه اصلی باشد. برای رسیدن به این مهم، خدمات در موضوع اختصاص داده شده آغاز می شود. هنگامی که سرویس توزیع کد دریافت درخواستی کند، آن را برای در دسترس بودن کد درخواستی بررسی

میکند و تصمیم میگیرد که چه ارسال کند .کد ارائه شده توسط منبع مورد اعتماد آنگاه در حافظه نهانی ذخیره شده، و نقشه برداری آن به کار مناسب برای فعال کردن چند اجاره ای انجام شده و اعلام اجرا از سر گرفته می شود.

4) نسخه برداری کد: برای حل این چالش های مربوط به کنترل نسخه کد و انتشار کد به روز رسانی شده، ما یک سیستم تایید کد را به عنوان بخشی از مکانیسم بارگذاری کلاس CloudScaleاجرا کردیم. درصورتی که کد در حافظه نهانی موجود باشد بارکننده کلاس باز باید اطمینان حاصل کند که کد همان نسخه های مشابه مورد انتظار مشتری را دارد. بنابراین، بارکننده کلاس انجام راستی تایید کد را بر اساس تاریخ آخرین اصلاح انجام شده و اندازه فایل های کد، همانطور که در شکل 4 نشان داده شده را صورت می دهد. بدیهی است، برخی جایگزین های دیگر برای اجرای تایید کد نیز امکان پذیر است (به عنوان مثال، با استفاده از هش کدها، نسخه برداری صریح و روشن از طریق شماره های نسخه، و یا انتقال جزئی)، اما ما تلقی می کنیم که رویکرد اکتشافی انتخاب شده سریع ترین است، در حالی که هنوز به قدر کافی قابل اعتماد برای برنامه های کاربردی عملی است. این نقطه نظر با این حقیقت حمایت می شود که رهیافتهای مشابه در سایر راه حل های پیشرفته مانند RSync, Apache Ant, GNU Make و سایرین بکار می رود.

چون کد در حافظه نهانی ذخیره می شود، نه تنها خود کد مورد نیاز برنامه، بلکه تمام فایل هایی که در آن زمان قبلی برای درخواست همان کد ارائه شده بودند، تأیید شده باشد. برای هر فایل در این مجموعه، مشتری یا تایید می کند که این کد مورد انتظار است و یا فایلی را فراهم می کندکه باید مورد استفاده قرار گیرد (شکل 4 را ببینید. بعد از این، بارکننده طبقه ارائه کد صحیح را برای اجرا و، در صورت لزوم، به روز رسانی نسخه دریافت شده به عهده می گیرد.

ب-گرفتن کد

در CloudScale، میزبان ابر هر درخواست جداگانه را در یک الگوریتم Sandbox را اجرا کند. برای این منظور، زیرساخت های بازیابی کلاس در هر میزبان ابر حل تمام منابع را برای هر درخواست به صورت

جداگانه انجام می دهد. این امر اجازه می دهد تا اعلام اجرای درخواست های مختلف با استفاده از پایگاه های کد های مختلف انجام شود، و هرگونه تاثیر احتمالی یک درخواست به دیگران را محدود می کند. با این حال، ظاهرا این رویکرد یک انتقال کد زیادی را ارائه می کند، چرا که اگر همان کد برنامه باید بیش از یک بار استفاده شود، آن را باز هم به طور جداگانه برای هر درخواست منتقل کند. برای جلوگیری از این کار زیادی، یک مکانیسم کد ذخیره هوشمند معرفی می کنیم.

برای درخواست کد برای اولین بار، زمانی که کد مورد نیاز هنوز کسب نشده، باید از ذخیره سازی کد مورد اعتماد بارگذاری شود، در حالی که هر یک از درخواست های بعدی تنها با استفاده از کد های موجود از حافظه نهانی (در صورتی که تایید کد موفق آمیز باشد) انجام می شود. هنگامی که تغییرات در طول تأیید تشخیص داده شود، کد منسوخ شده یا جایگزین شده یا به موازات نسخه به روز شده استفاده می شود که بسته به استفاده از حافظه نهانی و سیاست پیکربندی است. وقتی که هیچ تغییری وجود ندارد، کد پنهان سازی شده را می توان بدون انتقال از طریق کانال های ارتباطی استفاده نمود.

جدول 2-تبادل انتخاب آرایش حافظه نهانی

	Host Private Cache	Cloud Cache
Cloud-Based Code Storage	+ code access speed - low cache hit rate	- no speed up + good cache hit rate
External Code Storage	+ code access speed - low cache hit rate	+ code access speed + good cache hit rate

وظیفه اصلی مکانیسم ذخیره ساز ارائه کد سریعتر در شرایط زمانی است که همان کد چندین بار درخواست شده است. بنابراین، کد از حافظه نهانی باید در دسترس سریعتر بسته به ذخیره سازی کد مورد اعتماد (به عنوان مثال، درخواست مشتری)قرار گیرد. سریعترین محل ممکن حافظه نهانی هارد دیسک و یا حتی حافظه میزبان ابر است. این کار سرعت دسترسی ایده آل را فراهم می کند، اما میزان درستی کار حافظه نهانی را کاهش می دهد، چون هر میزبان ابر مجبور به حفظ حافظه نهان خود می شود. در مورد برخی برنامه های کاربردی توزیع شده، این روش ممکن است اصلاً هیچ فایده ای نداشته باشد همانگونه که

در جدول 2 نشان داده شده است. یکی دیگر از روش های ممکن برای ایجاد یک حافظه نهانی سرور اختصاصی و یا به اشتراک گذاشتن یک حافظه نهانی بین میزبان ابر های متعدد است. این یک راه حل خوب است اگر کد در ابتدا از طریق یک کانال غیر قابل اعتماد یا آهسته منتقل شود، اما اگر برنامه کاربردی در حال حاضر از یک سرویس کد اختصاصی استفاده کند، یک حافظه نهانی مشترک در ابر به ندرت معنی دارد، چون سرعت دسترسی تقریبا همانند سرور کد می باشد.

از وضعیتی که در بالا توضیح داده شده، روشن است که ما با یک تبادل طبق بحث در جدول 2 روبرو هستیم. بسته به پیکربندی محیطی و وضعیت، روش های مختلفی را می شود کارآمد تر کرد و، از این رو، ترجیح داده شده ا ست. بنابراین، برای رسیدن به بهترین عملکرد، معنی دارد که به برنامه امکان تصمیم گیری در استراتژی ذخیره ارجح را بدهیم.

ج-بارگذاری دسته

هنگامی که زیرساخت بارگذاری کلاس درخواست برای کلاس و یا منابع جدید برای بارگذاری دریافت می کند، اطلاعات زیادی در دسترس نیست تا برخی از فرضیات در مورد داده هایی که باید بارگذاری شود، صورت گیرد. تنها چیزی که در دسترس است منابع است که باید بازیابی شود. بنابراین، ابر میزبان باید درخواستی به مرکز ذخیره سازی با تنها نام منبع مورد نیاز مشخص شده ارسال کند (همانطور که در بالا شرح داده شد، وضعیت کمی متفاوت است زمانی که کد در حال حاضر موجود در حافظه نهانی وجود داشته باشد، که ما این مورد را در حال حاضر به خاطر ساده سازی حذف می کنیم).

زمانی که درخواست بازیابی کد به مرکز ذخیره سازی می رسد، خدمات مناسب باید پیدا کردن قطعه مورد نیاز کد را انجام دهد و تصمیم بگیرد که چه چیزی را برای ارسال با آن همراه کند. البته، ساده ترین سناریو این خواهد بود که تنها منبع درخواستی را ارسال نمود، اما این کار هزینه بار کد پویا را افزایش می دهد و کم کردن سرعت برنامه، به خصوص در هنگام راه اندازی را به همراه دارد. مورد افراطی دیگر ارسال همه کد برنامه بنا به اولین درخواست است: این کار مقدار پیام ها را کاهش می دهد، اما حتی ممکن است ایجاد تاخیر بیشتری در معرفی همان اولین درخواست بنماید، زمانی که کل مجموعه کتابخانه ها و پایگاه کد

منتقل می شود. با توجه به این واقعیت است که معمولا تمام کد را بر روی هر یک میزبان ابر مورد نیاز نداریم، این گزینه ممکن حتی سرباری بیشتری از مورد اولی ارائه دهد.

یکی از گزینه های ممکن برای حل این تبادل این خواهد بود که اجازه دهد تا برنامه کاربر نهایی برای پیکربندی مقدار کد که باید برای هر درخواست منتقل شده اقدام کند. با این حال، این رویکرد تا اندازه ای دست و پا گیر برای پدیدآورندگان و مطابق با اهداف طرح اولیه CloudScale(که ساخت آن را برای کاربردهای برنامه های ابر اسان می سازد، است). یکی دیگر از انتخاب ها استفاده اکتشافی است، که راه حل رضایت بخشی برای حالات کاربرد عمومی مطرح می کند.

برای مثال، اگر کلاس درخواستی متعلق به یک کتابخانه (به عنوان مثال، یک فایلjar) باشد، معنی دار است که ارسال کل کتابخانه به جای آن صورت گیرد چون شانس اینکه منابع دیگر از آن کتابخانه درخواست خواهد شد بالا می باشد. به طور مشابه، اگر کلاس متعلق به یک بسته کامل است، معنی دار به نظر می رسد که ارسال کل بسته صورت گیرد. همچنین، اگر کلاس برخی وابستگی ها دارد و یا متعلق به سلسله مراتب طبقات و یا رابط هاست، کلاس های دیگری می باشد که به احتمال بسیار زیاد مورد نیاز است.

همه ی این فن آوری هوشمند دارای مزایا و مشکلات خاص خود است و تعیین اینکه کدامیک از آنها باید به عنوان رفتار پیش فرض استفاده شود، پیچیده است. برای تعیین تاثیر این عوامل در برنامه های کاربردی در زندگی واقعی، ما تعدادی از الگوریتم های مختلف بارگذاری دسته را در نتایج ارزیابی گنجانده ایم و نتایج ارزیابی را در بخش های زیر ارائه می دهیم.

6-ارزیابی

برای حمایت از استدلالمان و ارزیابی هزینه ها و مزایای روش توزیع کد مان، ما از یک پیاده سازی نرم افزار تجزیه و تحلیل عقاید که در بخش 2 آمده است استفاده کرده ایم. این نرم افزار دارای تعدادی از ویژگی های است که به ما اجازه می دهد ارزیابی بهتر استراتژی های مختلف توزیع کد را انجام دهیم. در مرحله اول، نرم افزار کاربردی به راحتی قابل موازی سازی است، بنابراین ما می توانیم اندازه کار مناسب برای

رسیدن به بار موثر بر مقدار متفاوتی از میزبان ها را برای بررسی تأثیر راهکار توزیع کد انتخابی بر راه اندازی نرم افزار را بیابیم. در مرحله دوم، پایگاه کد این نرم افزار کاربردی است از اندازه قابل توجهی (8 MBدر 6 تا فایل JAR) برخوردار است و همچنین باعث می شود از تعدادی از منابع غیرکدی بزرگ(42MB در 14 فایل) استفاده کند. این امر اجازه می دهد تا ما اندازه گیری نفوذ شبکه های ارتباطی و تاخیر ذخیره سازی بر روی عملکرد نرم افزار را صورت دهیم.

الف-تنظیم ارزیابی

ما ارزیابی های خود را در یک ابر IaaSخصوصی بر اساس OpenStackانجام دادیم. ابر خصوصی ما شامل 8دستگاه های فیزیکی (سرور های تیغه ای دل با دو پردازنده های Xeon اینتل E5620پردازنده در حال اجرا در 2.4گیگاهرتز چهار هسته ای و 32 GB RAMاز هر کدام) بود که از طریق متصل شدن به اترنت گیگابیت اختصاص یافته صورت گرفته است. نرم افزار تجزیه و تحلیل عقاید در یک لپ تاپ معمولی، که به ابر خصوصی از طریق LANمتصل می شد میزبانی شد.کد برنامه کاربردی در جاوا اجرا شد، و پیکربندی برای کار با چارچوب CloudScale، و ابر خصوصی ما انجام شد. در بررسی مان، ما یک نمونه متوسط ابر را (2 تا پردازنده مجازی، 3.75 GBاز RAM) برای سرور های ارتباطی ActiveMQ، و بین 1 تا 5 نمونه کوچک (1 CPUمجازی و 1.7 MBاز RAM هر کدام) به عنوان میزبان اجرای نرم افزار استفاده کردیم .تمام میزبان ها در حال اجرایUbuntu Linux 12.04 و جاوای 1.7 بودند.

اولین مرحله از ارزیابی ما این بود که برای ایجاد یک خط پایه برای ارزیابی توزیع کد اقدام کنیم. با استفاده از 1، 3و 5نمونه ابر کوچک به عنوان منابع محاسباتی، برای انجام این کار، ما برنامه تجزیه و تحلیل عقاید را با تمام کد های مورد نیاز و فایل ها به ماشین آلات ابر در 3تنظیم اجرا کردیم. این رویکرد به هیچ توزیع کد نیاز ندارد، بنابراین به ما اجازه می دهد تا به صورت کمّی سربار استراتژی های مختلف توزیع کد را اندازه گیری کنیم. پس از آن، ما همان آزمون ها را با استفاده از3 استراتژی مختلف توزیع کدی که در بالا توضیح داده شد تکرارکردیم:

• استراتژی یا راهکار توزیع کد مبتنی بر کلاس، زمانی که همه کد برنامه کاربردی طبق درخواست برای اولین بار در هنگام راه اندازی ارائه شده است

•بر اساس استراتژی توزیع کد مبتنی بر کلاس، زمانی که فقط کلاس و یا منبع درخواست ارائه شده است، و میزبان ابر باید هر یک از منابع را به طور جداگانه درخواست کند.

•استراتژی توزیع کد با دسته بندی هوشمند، هنگامی که کد در دسته مربوط به دسته های به شدت مرتبط تحویل داده شده است، در نتیجه بهینه سازی مقدار درخواستهای لازم و به حداقل رساندن انتقال کد های غیر ضروری صورت می گیرد.

در سمت میزبان های ابر، ما ارزیابی عملکرد 2 استراتژی های مختلف کد ذخیره را (خصوصی برای هر یک از میزبانها و مشترک در داخل ابر)، همانطور که در بخش 5 شرح داده شده است، صورت داده ایم استراتژی ذخیره سازی خصوصی با استفاده از هارد دیسک های خصوصی هر دستگاه به عنوان ذخیره سازی انجام می شود، در حالی که ذخیره به اشتراک گذاشته در راس یک پایگاه داده کلیدی ارزشمند Riak میزبانی روی یک نمونه ابر جداگانه اجرا شده است.

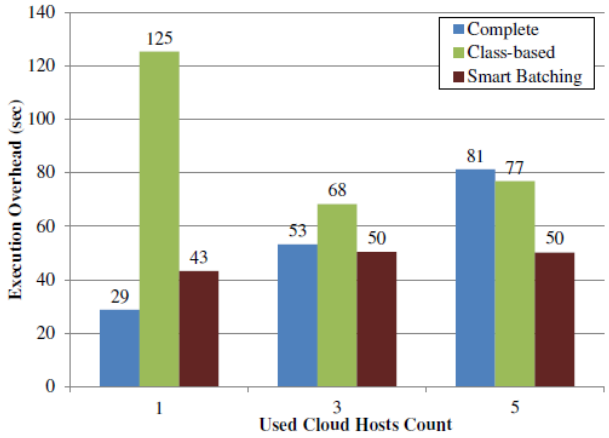
هر آزمایش چندین بار انجام شد و مقادیر میانگین در زیر استفاده می شود. با این حال، ما برخی از ناهنجاری های اعلام اجرا را که توسط سرعت های مختلف اعلام اجرای نرم افزار و شرایط محیطی تا اندازه ای به دلیل محدودیت میزان درخواست توییتر ایجاد شده بود مواجه شدیم.

ب-نتایج ارزیابی

میانگین زمان اجرای آزمایش پایه در حدود 1.5 دقیقه بود، در حالی که استفاده از الگوریتم کد توزیع این زمان به 22.5دقیقه افزایش یافته است. برای تمرکز بر هزینه های عملکرد هر یک از استراتژی های بارگذاری کلاس، ما تصمیم گرفتیم مقایسه سربار مدت زمان اجرای هر استراتژی در هر پیکربندی محیطی را انجام دهیم. در شکل 5 می توانید میانگین سربار اعلام اجرای (یعنی، زمان اجرای واقعی منهای مقدار پایه قبل، که برقرار شده تا با اجرای برنامه بدون توزیع کد برقرار شده) استراتژی های مختلف را (در تنظیم

های مختلف1 ، 3و 5میزبان های ابر) با حافظه نهانی خصوصی استفاده شده روی هر میزبان مشاهده کنید.

با استفاده از این ارزیابی اجرا ما می خواستیم تعیین کنیم که چگونه استراتژی توزیع کد دسته بندی هوشمندمان در مقایسه با دو حد نهایی دیگر رفتار می کند یعنی توزیع کد کامل و توزیع کد مبتنی بر کلاس Classbased.



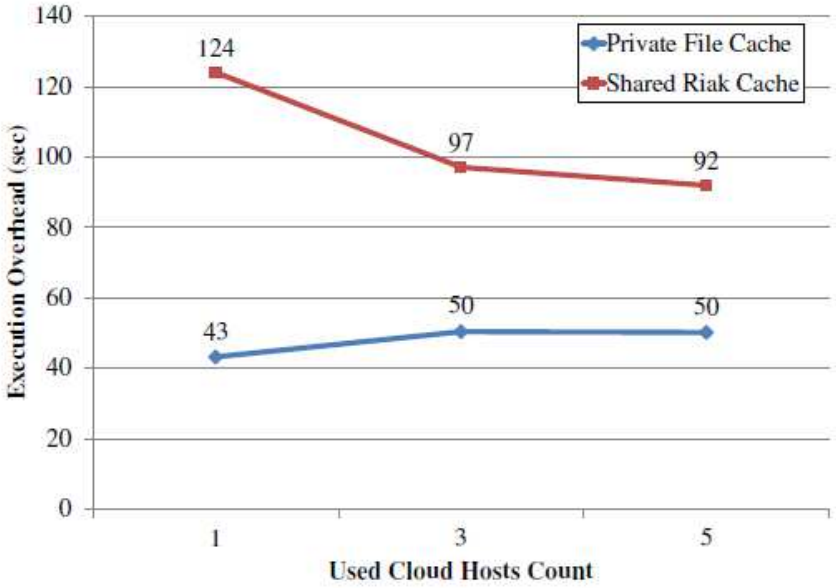
تصویر 5-سربار زمان اجرای کاربرد تحلیلی عقاید در اثر راهکارهای توزیع طبقه ای مختلف با حافظه پنهانی محلی

با مقایسه استراتژی های ذخیره خصوصی و مشترک، ما دوست داریم که در شکل 6، که در آن سربار اجرای استراتژی توزیع کد دسته بندی هوشمند را با دسته بندی خصوصی و مشترک استفاده شده بر روی میزبان ابر نشان داده ایم، تمرکز کنیم.

ج-بحث

در شکل 5، ما می توانیم ببینیم که اگر تنها داریم از یک ابر میزبان استفاده کنیم، استراتژی توزیع کد کامل در واقع سریع ترین می باشد، در حالی که بر اساس کلاس-کندترین است. دلیل آن این است که پهنای باند شبکه در راه اندازی ارزیابی ما به اندازه کافی خوب بوده که انتقال تمام کد لازم چند باری باعث تاخیر کمتری نسبت به ایجاد تعداد زیادی از تعاملات برای دسته های کد کوچکتر شود. در این تنظیم ارزیابی،

ارائه دهنده کد کامل در مجموع بیش از 110 MB از اطلاعات را فرستاده و کوچکترین سربار زمان اجرا را نشان داد، در حالی که دسته بندی هوشمند و بر اساس کلاس فقط نزدیک به 50 MBارسال شده است. علاوه بر این، می توان دید که استراتژی دسته بندی هوشمند قابل مقایسه با استراتژی کامل است، اما بسیار سریع تر از مبتنی بر کلاس است. دلیل آن این است که استراتژی دسته بندی هوشمند تقریبا همان مقدار از کد را که استراتژی مبتنی کلاس می فرستد، منتقل می کند اما با استفاده از درخواست به طور قابل توجهی کمتر اینکار را می کند. با این حال، لطفا توجه داشته باشید که در راه اندازی های مختلف، زمانی که ارتباط بین برنامه سرویس گیرنده یا مشتری و ابر آهسته تر است، و مقدار داده ی منتقل شده از مقدار درخواستی مهم تر است، استراتژی دسته سازی هوشمند از هر یک از رقبا کارامدتر خواهد بود .



تصویر 6-مقایسه سربارهای اجرایی حافظه پنهانی خصوصی و مشترک با راهکار توزیع کد دسته سازی هوشمند

با افزایش میزان میزبان های ابر استفاده شده، مشاهده می شود که هزینه استراتژی مبتنی برکلاس کاهش می یابد، در حالی که مقدار درخواست به طور منطقی همان مقدار باقی می ماند و یا حتی افزایش می یابد. این اتفاق به این دلیل می افتد که این درخواست را می توان به صورت موازی به کار گرفت و کاهش نفوذ در زمان اجرای برنامه به طور کلی صورت گرفته می شود. در همان زمان، سربار استفاده از استراتژی کامل را

افزایش می دهد چون به حد پهنای باند شبکه های ارتباطی بین سرویس گیرنده و ابر میزبان ضربه می زند. این امر را می توان دید که زمان اجرا در این مورد تقریبا به طورخطی با تعداد میزبان افزایش می یابد. این بدان دلیل است که تمام میزبان ها نیاز به کد تقریبا در همان زمان (در ابتدای اعلام اجرا) دارد که باعث می شود برنامه سرویس گیرنده برای ارسال همان کد به هر میزبان در همان زمان اقدام کند و کم کردن سرعت اجرای کلی برنامه را باعث می شود. برای استراتژی دسته ای هوشمند این هزینه به عنوان بالا نیست و آن را نگه می دارد در حدود عملکرد مشابه، از این رو به سرعت تبدیل شدن به بهترین استراتژی اگر میزبان ابر چند استفاده می شود.

با بحث در مورد شکل 6، می توان دید که هنگامی که میزبان با استفاده از یک حافظه پنهانی استفاده می کنند، هر یک از آنها برسر کانال های ارتباطی بین سرویس گیرنده و ابر رقابت می کنند، در حالی که تنظیم حافظه نهانی به اشتراک گذاشته کافی است که داده های مورد نیاز تنها با 1 میزبان بارگذاری شده و آن را برای هر میزبان در ابر در همان زمان قابل دسترس یافت. از این رو، با تعداد میزبان ابر افزایش یافته، سربار راه اندازی حافظه پنهانی خصوصی افزایش یافته، در حالی که بالای سربار حافظه پنهانی مشترک کاهش می یابد .با این حال، می توان به وضوح از روی شکل دید که سربار کلی از راه اندازی مشترک هنوز هم به طور قابل توجهی بالاتر است. این امر به دلیل مقدار بیشتری از ارتباطات و انتقال داده ها مورد نیاز برای راه اندازی حافظه پنهانی مشترک ایجاد می شود. در راه اندازی حافظه پنهانی خصوصی، میزبان ابر دانلود اطلاعات را به طور مستقیم از مشتری انجام داده، در حالی که در مورد حافظه نهانی مشترک، این میزبان است که برای دانلود کد از مشتری، و دادن آن به حافظه نهانی و سپس میزبان های دیگر می توانند به آن دسترسی داشته باشند. از این رو، در واقع وجود دارد 3 انتقال به جای فقط یکی، که در مورد ما قابل توجه است، وجود دارد آنهم زمانی که سرعت انتقال از مشتری به ابر قابل مقایسه با سرعت ارتباط در داخل ابر باشد.

براساس نتایج ارزیابی مان، می توانیم بگوییم که دسته بندی استراتژی توزیع کد هوشمند معمولا راه مرجح برای رسیدن به توزیع بدون عیب کد دینامیک و پویا با حداقل سربار نسبت به دیگر گزینه های ساده تر می

باشد. علاوه بر این، از این ارزیابی، ما دیدیم که استراتژی ذخیره پنهانی می تواند عملکرد برنامه های زیادی را تحت تاثیر قرار دهد. بنابراین معنی دار است که به کاربران امکان پیکربندی این پارامتر را با توجه به ویژگی های کانال ارتباطی بدهیم.

7-نتیجه گیری ها

با افزایش محبوبیت محاسبه کامپیوتری به شیوه ابری، پدیدآورندگان و شرکتها هرچه بیشتر در مورد نسخه های ابر آگاه برنامه های کاربردی خود فکر می کنند. با تدوین برنامه های مبتنی بر ابر کاربردی، مجموعه ای از مشکلات ناشی شده که وجود نداشته و یا قبلا معنی دار نبوده است. یکی از آنها مشکل توزیع کد به میزبان ابر است.

چارچوب توزیع کد بدون عیب که در این مقاله معرفی شد، اجازه می دهد تا کد توزیع به ابر مورد تقاضا به برنامه به طور بی عیب توزیع شود. چارچوب معرفی شده در یک کاربرد برنامه زندگی واقعی و سربار راهکارهای مختلف توزیع کد و ذخیره پنهانی که در آن در مقایسه و تجزیه و تحلیل شده است را مورد بررسی قرار داد. بررسی نشان داد که روش توزیع کد انتخاب شده یک لیست از مزیت ها را به راههای جایگزین و حداقل سربار برای کاربران فراهم می کند، در حالی که نیاز به مقدار بسیار ناچیزی از زمان برای پیکربندی و استفاده دارد.

در آینده، ما برنامه ریزی برای بهبود روش توزیع کد دسته ای هوشمند را داشته و پیاده سازی برخی از ترفند های دیگر عملکرد را انجام دهیم. به عنوان مثال، در معماری کنونی مان، فرصت پیش بینی کد مورد نیاز و ارسال درخواست های متعدد را به مشتری با همان بسته و یا اجرای برخی از الگوریتم های کد هوشمند برای بردن و آوردن مجدد که سرعت اجرای کد را بهبود بخشد و اجازه می دهد کد پس زمینه بارگذاری شود. علاوه بر این، اطلاعات تاریخی می تواند در نظر گرفته شود. به عنوان مثال، تاریخ درخواست اجراهای موازی مشابه یا قبلی می تواند برای پیشگویی درخواست های اتی و درخواست کلیه اطلاعات لازم از قبل استفاده شود.