دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر

پروژه کارشناسی

طراحی و پیاده‌سازی سیستم برداشت از حافظه در سیستم عامل لینوکس

نگارش

علی رضایی

استاد راهنما

دکتر سید احمد جوادی

خرداد ۱۴۰۳


بسم الله الرحمن الرحيم

صفحه فرم ارزیابی و تصویب پایان نامه- فرم تأیید اعضاء کمیته دفاع

در این صفحه فرم دفاع یا تایید و تصویب پایان نامه موسوم به فرم کمیته دفاع- موجود در پرونده آموزشی- را قرار دهید.

نکات مهم:

- نگارش پایان نامه/رساله باید به زبان فارسی و بر اساس آخرین نسخه دستورالعمل و راهنمای تدوین پایان نامه‌های دانشگاه صنعتی امیرکبیر باشد.(دستورالعمل و راهنمای حاضر)
- رنگ جلد پایان نامه/رساله چاپی کارشناسی، کارشناسی ارشد و دکترا باید به ترتیب مشکی، طوسی و سفید رنگ باشد.
- چاپ و صحافی پایان نامه/رساله بصورت پشت و رو(دورو) بلامانع است و انجام آن توصیه می شود.

دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

به نام خدا

تعهدنامه اصالت اثر

تاریخ: خرداد ۱۴۰۳

اینجانب **علی رضایی** متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر است. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

علی رضایی

امضا

تقدیم به

آن‌که جز به فضلش امیدی نیست...

سپاس‌گزاری

از پدر و مادرم که همواره در مواجهه با سختی‌های این دنیا دلسوزانه همراهم بوده‌اند؛
از استاد بزرگوارم جناب آقای دکتر سید احمد جوادی که با حسن خلق و گشاده‌رویی، رهنمودهای شبانه‌روزی خود را از من دریغ نکرده‌اند؛
و از سایر عزیزانی که در کنارشان این نتیجه حاصل آمد کمال تشکر و قدردانی را دارم.

علی رضایی
خرداد ۱۴۰۳

چکیده

در این پایان‌نامه، به طراحی و پیاده‌سازی سامانه‌ای برای برداشت و پیش‌بینی مصرف حافظه پرداخته شده است. این سامانه شامل عاملی است که اطلاعات مرتبط با ماشین‌های مجازی تحت نظارت مجازی‌ساز را جمع‌آوری می‌کند. عامل مذکور مؤلفه‌هایی از جمله تعداد ماشین‌های روشن، مجموع حافظه تخصیص داده شده به آن‌ها و مصرف لحظه‌ای حافظه را استخراج کرده و هر سه دقیقه به‌روزرسانی می‌کند. با استفاده از مؤلفه‌ای که از سرور دریافت می‌کند، عامل میانگین مصرف سه دقیقه گذشته را محاسبه کرده و برای پیش‌بینی سه دقیقه آینده در آن ضرب می‌کند. همچنین، عامل تفاوت بین حافظه تخصیص داده‌شده و پیش‌بینی‌شده را محاسبه و به مدیر سامانه جهت استفاده از این حافظه اضافه اعلام می‌کند. سامانه علاوه بر این، تعداد مواردی که مصرف حافظه بیش از پیش‌بینی شده باشد (تناقض) را ثبت می‌کند. تمامی این مؤلفه‌ها در سامانه‌ی دیگری پایش می‌شوند. سرور این سامانه قادر است آدرس عوامل مختلف را ثبت کرده و مؤلفه را برای هرکدام تعیین کند. همچنین امکان مشاهده مجموع تناقض‌ها برای هر عامل و گروه‌بندی عوامل با تعداد تناقضی بالا وجود دارد. هدف نهایی این پروژه، تعیین مؤلفه مناسب برای گروهی از عامل‌ها است. به منظور ارزیابی، چند ماشین مجازی با بارکاری واقعی بر روی مجازی‌ساز اجرا شده و با چند مؤلفه متفاوت ارزیابی شده‌اند تا بهترین مقدار مؤلفه برای کاهش مجموع تناقض‌ها و مقدار بیشتر حافظه اضافه در شرایط واقعی مشخص شود. نتایج حاصل از این پژوهش می‌تواند به مدیران سامانه در بهینه‌سازی تخصیص حافظه و پیش‌بینی دقیق‌تر مصرف آن کمک شایانی نماید.

واژه‌های کلیدی:

سامانه برداشت حافظه، بهینه‌سازی تخصیص حافظه ، مصرف حافظه ،پیش‌بینی مصرف

فهرست مطالب

عنوان صفحه

فهرست تصاویر

شکل صفحه

فهرست جداول

جدول صفحه

فصل اول

مقدمه

۱-۱ مقدمه

در دنیای امروز، هنگام خرید سرور، مردم به حداکثر مصرف منابع توجه دارند و بر اساس آن، مقدار منابع مورد نیاز را تعیین می‌کنند. این روش باعث می‌شود که در بسیاری از مواقع، درصد بالایی از منابع سرور بلااستفاده بماند. به عنوان مثال، طبق مطالعه‌ای استفاده از سرورهای ابری اغلب زیر ۱۰ درصد است.[۴] یک مطالعه جدیدتر از مایکروسافت[۱] نشان می‌دهد که ماشین‌های مجازی ابری میزبانی شده در Azure نیز از منابع به‌طور کم استفاده می‌کنند، به طوری که ۶۰درصد از ماشین مجازی[۲]ها دارای میانگین واحد پردازش مرکزی[۳] کمتر از ۲۰درصد هستند[۱].

راه‌حل ارائه شده توسط شرکت‌های بزرگ که به یک مکانیزم پرکاربرد تبدیل شده است، برداشت از منابع بلااستفاده به منظور اجرای پردازش‌های غیرحساس به زمان اجرا و بازگرداندن سریع منابع به ماشین‌های اصلی در زمان نیاز است. برای اجرای این راه‌حل، باید بتوان مصرف منابع ماشین‌های مجازی را پیش‌بینی کرد تا بر اساس آن تخصیص منابع بهینه‌سازی شود. این پیش‌بینی‌ها می‌توانند با استفاده از الگوریتم‌های هوش مصنوعی[۴] مانند الگوریتم‌های یادگیری ماشین[۵] یا رویکردهای آماری صورت بگیرند.

در صورت پیش بینی اشتباه ممکن است کمبود حافظه برای برنامه‌های در حال اجرا رخ دهد. کمبود حافظه (RAM) می‌تواند تاثیرات مختلفی بر اجرای برنامه‌ها داشته باشد. هنگامی که یک سرور با کمبود RAM مواجه می‌شود، سیستم‌عامل ممکن است بخشی از داده‌های موجود در حافظه را به دیسک سخت (فضای swap) منتقل کند، که این کار باعث کاهش عملکرد برنامه‌ها می‌شود زیرا دسترسی به دیسک بسیار کندتر از دسترسی به RAM است. در موارد شدیدتر، اگر فضای swap نیز پر شود یا عملکرد آن نتواند نیازهای برنامه را برآورده کند، سیستم‌عامل ممکن است برنامه‌های در حال اجرا را متوقف کرده و با خطای "Out of Memory" مواجه شود. این می‌تواند باعث از کار افتادن برنامه‌ها و در نتیجه توقف خدماتی که ارائه می‌دهند شود. برای جلوگیری از این مشکلات، مدیریت بهینه حافظه و استفاده از سیاست‌های نظارتی و تنظیمی مناسب توسط مدیران سیستم و ارائه‌دهندگان خدمات ابری بسیار حیاتی است.

سامانه برداشت از حافظه تخصیص داده شده به ماشین‌های مجازی، یکی از موضوعات مهم در بهینه‌سازی منابع سخت‌افزاری است. وقتی یک سرور فیزیکی را به چندین ماشین مجازی تقسیم

[1] Microsoft
[2] VM
[3] CPU
[4] Artificial Inteligense
[5] Machine Learning

می‌کنیم، نیاز به یک سامانه مدیریت حافظه مناسب داریم تا منابع را به ماشین‌های مجازی تخصیص دهیم. این کار به ما کمک می‌کند تا بهره‌وری منابع را افزایش دهیم و هزینه‌های مربوط به سخت‌افزار را کاهش دهیم.

پروژه حاضر، طراحی و پیاده‌سازی سامانه‌ای برای برداشت و پیش‌بینی مصرف حافظه در سیستم عامل لینوکس[۶] را هدف قرار داده است. ایده این سامانه و نحوه پیش‌بینی آن از پژوهش‌های پیشین برآمده‌است[۳]. این سامانه شامل یک عامل[۷] نوشته شده با زبان گولنگ[۸] است که اطلاعات مرتبط با ماشین‌های مجازی تحت نظارت مجازی ساز[۹] KVM را با استفاده از کتابخانه Libvirt جمع‌آوری می‌کند. عامل مذکور مؤلفه هایی از جمله تعداد ماشین‌های روشن، مجموع حافظه تخصیص داده شده به آنها، و مصرف لحظه‌ای حافظه (مؤلفه RSS) را استخراج کرده و هر سه دقیقه به‌روزرسانی می‌کند.

همچنین، مؤلفه برداشت که تفاوت بین حافظه تخصیص داده شده و پیش‌بینی شده است را محاسبه و به مدیر سامانه[۱۰] اعلام می‌کند. سامانه علاوه بر این، تعداد مواردی که مصرف حافظه بیش از پیش‌بینی شده باشد (تناقض[۱۱]) را ثبت می‌کند. تمامی این مؤلفه ها به پرومتئوس[۱۲] ارسال شده و در گرافانا[۱۳] نمایش داده می‌شوند.

سرور این سامانه با استفاده از زبان پایتون[۱۴] و چارچوب Flask پیاده‌سازی شده و از دیتابیس[۱۵] SQLite برای ذخیره‌سازی داده‌ها استفاده می‌کند. سرور قادر است آدرس[۱۶] عوامل مختلف را ثبت کرده و مؤلفه آستانه را برای هر کدام تعیین کند. همچنین امکان مشاهده و تغییر مجموع تناقض‌ها برای هر عامل و گروه‌بندی عوامل با تعداد تناقض بالا وجود دارد.

هدف نهایی این پروژه، تعیین مؤلفه آستانه مناسب برای گروهی از ماشین‌های مجازی یک سرور در یک خوشه[۱۷] است. به منظور ارزیابی، سه ماشین مجازی با بارکاری واقعی data analytics از مجموعه Cloud Suite بر روی سه ماشین مختلف اجرا و با شش مؤلفه آستانه متفاوت ارزیابی شده‌اند تا بهترین

[6] Linux
[7] Agent
[8] Golang
[9] hypervisor
[10] System Administrator
[11] violation
[12] Prometheus
[13] Grafana
[14] Python
[15] Database
[16] IP
[17] Cluster

مقدار مؤلفه برای کاهش مجموع تناقض‌ها در شرایط واقعی مشخص شود.

۱-۲ تعریف مسئله

در دنیای امروز، با افزایش استفاده از سرورهای ابری و مجازی، بهینه‌سازی استفاده از منابع سخت‌افزاری از اهمیت بسیاری برخوردار است. یکی از مشکلات اساسی در مدیریت منابع سرورها، مصرف نامناسب حافظه است که می‌تواند منجر به هدررفت منابع و افزایش هزینه‌ها شود. معمولاً، تخصیص حافظه به ماشین‌های مجازی براساس تخمین‌های سنتی یا تجربی انجام می‌شود که ممکن است با نیازهای واقعی سامانه مطابقت نداشته باشد. در این پروژه، ما با یک مسئله کلیدی در مدیریت منابع سرورها در مواجهه هستیم، به نام "برداشت از حافظه" یا "تخصیص حافظه[۱۸]". این مسئله به اهمیت مدیریت دقیق حافظه تخصیص داده شده به ماشین‌های مجازی بر روی یک سرور اشاره دارد. در یک محیط مجازی‌سازی، چندین ماشین مجازی روی یک سرور فیزیکی اجرا می‌شوند و این سرور باید توانایی تخصیص حافظه بهینه به هر ماشین مجازی را داشته باشد.مسئله اصلی در اینجا این است که چگونه می‌توانیم حافظه را بهینه‌سازی کنیم تا همه ماشین‌های مجازی بتوانند به حداکثر کارایی و کاربردیت خود دست یابند، در حالی که هدررفت حافظه بلااستفاده را به حداقل برسانیم. این مسئله راه‌حل‌هایی از قبیل تخصیص دقیق حافظه براساس الگوریتم‌های هوش مصنوعی یا تخمین‌های آماری مطرح می‌کند.با توجه به اهمیت این مسئله و نیاز به روش‌های دقیق و موثر برای مدیریت حافظه در محیط‌های مجازی، پروژه حاضر به منظور طراحی و پیاده‌سازی یک سامانه برداشت از حافظه در سیستم عامل لینوکس ارائه شده است.

۱-۳ ضرورت و اهداف پروژه

اهمیت پروژه طراحی سامانه برداشت از حافظه را می‌توان در موارد زیر خلاصه کرد:

۱. **بهبود بهره‌وری منابع:** بهبود بهره‌وری منابع سخت‌افزاری سرورها از طریق بهینه‌سازی تخصیص حافظه به ماشین‌های مجازی، از جمله اهداف اصلی این پروژه است.

۲. **کاهش هدررفت منابع:** کاهش هدررفت حافظه بلااستفاده بر روی سرورها، منجر به کاهش هزینه‌ها و بهبود کارایی سامانه خواهد شد.

[18] Memory Harvesting

۳. **پیش‌بینی دقیق مصرف حافظه:** امکان پیش‌بینی دقیق مصرف حافظه توسط ماشین‌های مجازی، کمک می‌کند تا تخصیص منابع بهینه‌تری انجام شود و از زیان‌های احتمالی جلوگیری شود.

همچنین اهداف پروژه پیش‌رو را می‌توان در موارد زیر مطرح کرد:

۱. **طراحی سامانه برداشت از حافظه:** طراحی یک سامانه کامل برای برداشت از حافظه تخصیص داده شده به ماشین‌های مجازی در سیستم عامل لینوکس.

۲. **پیاده‌سازی عامل و سرور:** پیاده‌سازی عامل برای جمع‌آوری اطلاعات مربوط به ماشین‌های مجازی و سرور به منظور برقراری ارتباط با مدیر سامانه و ارائه اطلاعات به‌روز.

۳. **پیش‌بینی مصرف حافظه:** ارائه الگوریتم‌ها و روش‌هایی برای پیش‌بینی دقیق مصرف حافظه توسط ماشین‌های مجازی.

۴. **ارسال داده به پرومتئوس**[۱۹]: ارسال اطلاعاتی که توسط عامل جمع‌آوری شده‌اند به پرومتئوس برای ذخیره و نمایش آنها در گرافانا.

۱-۴ ابزارها و کتابخانه‌های مورد استفاده

۱. **گولنگ**[۲۰]: زبان برنامه‌نویسی که برای پیاده‌سازی عامل استفاده می‌شود. از سادگی و قدرت بالای آن برای توسعه برنامه‌های سامانه‌ای و ابزارهای کاربردی استفاده می‌شود. کتابخانه libvirt را برای ارتباط با مجازی ساز KVM استفاده می‌کند تا اطلاعات مربوط به ماشین‌های مجازی را جمع‌آوری کند.

۲. **پرومتئوس:** سامانه متن‌باز جمع‌آوری و پایش سرورها و خدمات. اطلاعات عملکرد سرور و ماشین‌های مجازی را جمع‌آوری و ذخیره می‌کند تا بتوان از آن‌ها برای مدیریت بهتر سامانه استفاده کرد.

[19] Prometheus
[20] Golang

۳. **گرافانا:** ابزار متن‌باز و قدرتمند برای نمایش و پایش داده‌های زمانی. داده‌هایی که توسط پرومتئوس جمع‌آوری شده‌اند را به‌صورت گرافیکی و تصویری نمایش می‌دهد تا به کاربران کمک کند تا الگوها و روندهایی را که در داده‌ها وجود دارد را تشخیص دهند.

۴. **Python Flask:** یک چارچوب وب سریع و سبک برای پیاده‌سازی سرورهای وب. برای پیاده‌سازی رابط کاربری سرور و ارتباط با مدیر سامانه برای تنظیمات و مدیریت استفاده می‌شود.

کتابخانه libvirt را برای ارتباط با مجازی ساز KVM استفاده می‌کند تا اطلاعات مربوط به ماشین‌های مجازی را جمع‌آوری

۵. **SQLite:** یک موتور پایگاه‌داده رابطی SQL بسیار کوچک و سبک که برای ذخیره‌سازی اطلاعات کوچک و محلی استفاده می‌شود. برای ذخیره‌سازی تنظیمات و داده‌های کاربر در این پروژه به‌کار می‌رود.

۱-۵ ساختار گزارش

در فصل ابتدایی این گزارش مقدمه‌ای بر روی مسئله، اهداف مدنظر، ضرورت اجرای این پروژه و تکنولوژی‌های مورد استفاده ارائه شد. در فصل دوم به طور تفصیلی به توضیح سامانه‌های برداشت حافظه و انواع مختلف آن پرداخته می‌شود. سپس مفاهیم مرتبط با تخصیص حافظه، مدیریت حافظه در ماشین‌های مجازی و تکنولوژی‌های مرتبط مورد بحث واقع می‌شود. در فصل سوم در خصوص طراحی و پیاده‌سازی پروژه صحبت خواهد شد. در این فصل ابتدا درباره‌ی مجموعه مؤلفه ها و استخراج ویژگی‌ها از آن مطالبی مطرح می‌شود و در ادامه به جزئیات پیاده‌سازی عامل و سرور پرداخته خواهد شد. همچنین توضیحاتی درباره‌ی استفاده از گولنگ،پرومتئوس، گرافانا، Python Flask و SQLite ارائه می‌شود. در فصل چهارم با استفاده از سناریوهای از پیش تعیین‌شده سامانه را مورد ارزیابی قرار می‌دهیم تا از صحت عملکرد آن اطمینان حاصل کنیم و در فصل پایانی نیز به بیان نتایج حاصل از انجام پروژه و پیشنهادات پرداخته خواهد شد.

فصل دوم

مفاهیم پایه

در این فصل به معرفی و بررسی ابعاد مختلف و مفاهیم مورد استفاده در این پروژه می‌پردازیم. آشنایی با این مفاهیم برای پیشبرد بهتر پروژه ضروری می‌باشد.

۲-۱ مفاهیم مجازی‌سازی

مجازی‌سازی تکنیکی است که به‌وسیله آن یک سرور فیزیکی به چندین ماشین مجازی مستقل تقسیم می‌شود. هر یک از این ماشین‌های مجازی قادر به اجرای سیستم عامل و برنامه‌های خاص خود هستند. مجازی‌سازی با استفاده از لایه‌ای به نام مجازی ساز انجام می‌شود که منابع فیزیکی سرور (مانند واحد پردازش مرکزی[۱]، حافظه و دیسک سخت) را بین ماشین‌های مجازی تقسیم می‌کند و مدیریت می‌کند. این تکنولوژی به سازمان‌ها امکان می‌دهد که بهره‌وری از منابع فیزیکی را افزایش دهند، هزینه‌های سخت‌افزاری را کاهش دهند و انعطاف‌پذیری و مقیاس‌پذیری را بهبود بخشند[۸].

اهمیت مجازی‌سازی در مدیریت منابع سرورها به‌طور خاص در موارد زیر قابل مشاهده است:

۱. **افزایش بهره‌وری منابع:** با استفاده از مجازی‌سازی، می‌توان منابع فیزیکی را به‌صورت بهینه‌تر مورد استفاده قرار داد و از هدررفت منابع جلوگیری کرد.

۲. **کاهش هزینه‌ها:** کاهش نیاز به سخت‌افزارهای فیزیکی جدید و صرفه‌جویی در هزینه‌های مرتبط با نگهداری و انرژی.

۳. **بهبود انعطاف‌پذیری:**

امکان ایجاد و حذف ماشین‌های مجازی به‌سرعت و بر اساس نیازهای متغیر کسب‌وکار.

۴. **مدیریت ساده‌تر:** مدیریت متمرکز و یکپارچه ماشین‌های مجازی و امکان اعمال سیاست‌های مختلف امنیتی و بهره‌برداری به‌صورت یکپارچه.

انواع مجازی‌سازی:

۱. **مجازی‌سازی کامل (Full Virtualization)**

در مجازی‌سازی کامل، مجازی ساز به‌گونه‌ای عمل می‌کند که هر ماشین مجازی می‌تواند مستقیماً سیستم عامل خود را اجرا کند بدون اینکه نیاز به تغییرات در سیستم عامل باشد. این نوع

[1] CPU

مجازی‌سازی به‌طور کامل منابع فیزیکی را بین ماشین‌های مجازی تقسیم می‌کند و آن‌ها را از یکدیگر ایزوله می‌سازد. مزیت این روش این است که امکان اجرای هر نوع سیستم عاملی بر روی ماشین مجازی وجود دارد. VMware ESXi و Microsoft Hyper-V مثال‌های بارز از این نوع مجازی‌سازی هستند.

۲. **پارا مجازی‌سازی (Paravirtualization)**

در پارا مجازی‌سازی، سیستم عامل‌های مهمان به‌گونه‌ای تغییر می‌کنند که با مجازی ساز همکاری کنند. این تغییرات باعث بهبود کارایی و بهره‌وری می‌شوند زیرا ارتباط مستقیم‌تر و بهینه‌تری بین مجازی ساز و سیستم عامل‌های مهمان برقرار می‌شود. در این روش، سیستم عامل‌های مهمان باید برای اجرای در محیط پارا مجازی‌سازی تطبیق داده شوند. Xen یکی از معروف‌ترین مجازی سازهای پارا مجازی‌سازی است.

۳. **مجازی‌سازی سطح سیستم عامل (Operating System-Level Virtualization)**

در این نوع مجازی‌سازی، سیستم عامل میزبان یک نسخه از سیستم عامل خود را در هر ماشین مجازی اجرا می‌کند. این روش به‌طور معمول به‌عنوان مجازی‌سازی کانتینر(container)ی شناخته می‌شود و به ماشین‌های مجازی اجازه می‌دهد که به‌طور مستقل عمل کنند اما از هسته سیستم عامل میزبان مشترک استفاده کنند. Docker و LXC نمونه‌هایی از این نوع مجازی‌سازی هستند. این روش بسیار کارآمد است و به‌خصوص در محیط‌های توسعه و استقرار اپلیکیشن‌های کوچک و مقیاس‌پذیر استفاده می‌شود.

۲-۲ مجازی‌ساز

مجازی ساز، یا ناظر مجازی‌سازی، نرم‌افزاری است که امکان اجرای چندین سیستم عامل را به‌طور همزمان بر روی یک سرور فیزیکی فراهم می‌کند. مجازی ساز این کار را با تقسیم منابع فیزیکی (مانند پردازنده، حافظه و ذخیره‌سازی) بین ماشین‌های مجازی انجام می‌دهد[۶]. دو نوع اصلی از مجازی ساز وجود دارد:

۱. **مجازی ساز نوع ۱**

در مجازی‌سازی کامل، مجازی ساز به‌گونه‌ای عمل می‌کند که هر ما مجازی سازهای نوع ۱، که به‌عنوان "ناظر بومی" یا "Bare-metal Hypervisors" نیز شناخته می‌شوند، مستقیماً بر روی

سخت‌افزار فیزیکی اجرا می‌شوند. این نوع مجازی ساز به‌صورت مستقیم با سخت‌افزار سرور در تعامل است و هیچ سیستم عامل میزبان بین مجازی ساز و سخت‌افزار وجود ندارد. به دلیل این تعامل مستقیم، مجازی سازهای نوع ۱ کارایی بالاتری دارند و برای محیط‌های تولیدی و سرورهای بزرگ بسیار مناسب هستند. مثال‌هایی از مجازی سازهای نوع ۱ شامل VMware ESXi، Microsoft Hyper-V، و Citrix XenServer هستند.

۲. مجازی ساز نوع ۲

مجازی سازهای نوع ۲، که به‌عنوان "ناظر میزبانی شده" یا "Hosted Hypervisors" شناخته می‌شوند، بر روی یک سیستم عامل میزبان اجرا می‌شوند. این نوع مجازی ساز به‌صورت یک برنامه کاربردی در سیستم عامل میزبان عمل می‌کند. مجازی سازهای نوع ۲ به دلیل اجرای بر روی یک سیستم عامل میزبان، معمولاً کارایی کمتری نسبت به نوع ۱ دارند، اما به‌راحتی بر روی سامانه‌های دسکتاپ و لپ‌تاپ قابل نصب هستند و برای محیط‌های توسعه و ارزیابی مناسب می‌باشند. VMware Workstation و Oracle VirtualBox نمونه‌هایی از مجازی سازهای نوع ۲ هستند.

نقش مجازی ساز در مدیریت ماشین‌های مجازی:

مجازی ساز نقشی حیاتی در مدیریت ماشین‌های مجازی ایفا می‌کند. برخی از وظایف اصلی مجازی ساز شامل موارد زیر است:

تخصیص منابع: مجازی ساز منابع فیزیکی سرور (مانند پردازنده، حافظه و ذخیره‌سازی) را بین ماشین‌های مجازی تقسیم می‌کند و تضمین می‌کند که هر ماشین مجازی به اندازه نیاز خود از منابع بهره‌برداری کند.

ایزوله‌سازی: مجازی ساز هر ماشین مجازی را از سایر ماشین‌های مجازی ایزوله می‌کند تا از تداخل و تهاجم جلوگیری شود. این ایزوله‌سازی به افزایش امنیت و پایداری سامانه کمک می‌کند.

مدیریت عملکرد: مجازی ساز نظارت و مدیریت عملکرد ماشین‌های مجازی را بر عهده دارد و از طریق ابزارهای پایش و مدیریت، بهینه‌سازی منابع را انجام می‌دهد.

پشتیبانی از مهاجرت: مجازی ساز امکان مهاجرت زنده ماشین‌های مجازی (Live Migration) را فراهم می‌کند، به‌طوری که ماشین‌های مجازی می‌توانند بدون توقف و با حداقل قطعی به سرورهای دیگر منتقل شوند.

معرفی مجازی‌ساز مورد استفاده در پروژه:

Kernel-based Virtual Machine (KVM) یک مجازی ساز نوع ۱ است که به‌صورت پودمان[۲] در هسته لینوکس عمل می‌کند و اجازه می‌دهد که لینوکس به یک مجازی ساز کامل تبدیل شود. KVM با استفاده از تکنولوژی‌های سخت‌افزاری مجازی‌سازی مانند Intel VT و AMD-V، امکان اجرای چندین ماشین مجازی با کارایی بالا را فراهم می‌کند.

ویژگی‌های کلیدی KVM شامل موارد زیر است:

یکپارچگی با لینوکس: KVM به‌عنوان بخشی از هسته لینوکس عمل می‌کند، بنابراین از تمامی مزایای امنیتی و عملکردی لینوکس بهره‌برداری می‌کند.

پشتیبانی از انواع سیستم عامل‌ها: KVM امکان اجرای انواع سیستم عامل‌ها (مانند ویندوز، لینوکس و BSD) را به‌صورت ماشین‌های مجازی فراهم می‌کند.

مدیریت منابع: KVM از ابزارهایی مانند libvirt برای مدیریت و پایش ماشین‌های مجازی استفاده می‌کند، که امکان تخصیص بهینه منابع و مدیریت آسان را فراهم می‌سازد.

مقیاس‌پذیری: KVM برای محیط‌های بزرگ و مقیاس‌پذیر مناسب است و به‌راحتی می‌تواند تعداد زیادی ماشین مجازی را مدیریت کند.

در پروژه "Memory Harvesting" ، از KVM به‌عنوان مجازی ساز اصلی استفاده می‌شود. این انتخاب به دلیل کارایی بالا، قابلیت‌های مدیریت قوی و یکپارچگی با هسته لینوکس انجام شده است. با استفاده از KVM و ابزار libvirt، می‌توان اطلاعات مربوط به ماشین‌های مجازی را جمع‌آوری کرده و سامانه برداشت حافظه را بهینه‌سازی کرد.

مؤلفه RSS در مجازی‌ساز:

در محیط KVM (Kernel-based Virtual Machine)، مؤلفه RSS به معنای "Resident Set Size" است که به حجم حافظه‌ای اشاره دارد که به یک فرایند خاص اختصاص داده شده و در حافظه اصلی(RAM)سامانه قرار دارد، بدون در نظر گرفتن حافظه‌ای که به حالت مبادله(Swap)رفته است.

برای بهتر فهمیدن مفهوم RSS، می‌توان آن را به عنوان مجموع حجم حافظه‌ای دید که به صورت فیزیکی در حافظه RAM می‌ماند و برای اجرای فرآیند لازم است. این شامل حافظه‌هایی است که به صورت خصوصی برای آن فرآیند اختصاص داده شده‌اند، از جمله داده‌های کد برنامه، داده‌های متغیر (مثل متغیرهای ایستا و پشته(stack))، حافظه‌هایی از کتابخانه‌های مشترک که در حال حاضر در حافظه

[۲]Module

RAM وجود دارند، و همچنین حافظه‌های استک و پشته(heap) که برای ذخیره اطلاعات موقتی در طول اجرای برنامه استفاده می‌شوند.

به طور کلی، RSS نشان دهنده حافظه‌ای است که فرایند در حال استفاده از آن است و برای ادامه اجرای برنامه نیاز دارد. این اطلاعات بسیار مهم برای مدیریت منابع سامانه هستند تا بتوانند بهینه‌سازی مصرف حافظه را انجام دهند و از واگذاری زیاد حافظه به برنامه‌هایی که آن را نیاز ندارند، جلوگیری کنند.

۲-۳ مدیریت حافظه در مجازی‌سازی

در محیط‌های مجازی، تخصیص و مدیریت حافظه یکی از جنبه‌های حیاتی و پیچیده است که به طور مستقیم بر عملکرد و کارایی سامانه‌های مجازی تأثیر می‌گذارد. مجازی ساز به عنوان ناظر مجازی‌سازی مسئولیت تخصیص حافظه به ماشین‌های مجازی (ماشین مجازی) را برعهده دارد. این فرآیند شامل اختصاص میزان معینی از حافظه فیزیکی سرور به هر ماشین مجازی است تا بتوانند به طور مستقل سیستم عامل و برنامه‌های خود را اجرا کنند.

مدیریت حافظه در محیط‌های مجازی به دو بخش اصلی تقسیم می‌شود:

۱. تخصیص اولیه: در زمان ایجاد یک ماشین مجازی، مجازی ساز مقدار معینی از حافظه فیزیکی را به آن اختصاص می‌دهد. این تخصیص می‌تواند ثابت یا دینامیک باشد. در تخصیص ثابت، حافظه به طور دائمی به ماشین مجازی اختصاص می‌یابد، در حالی که در تخصیص دینامیک، مجازی ساز می‌تواند میزان حافظه اختصاص‌یافته به ماشین مجازی را بر اساس نیازها و استفاده‌های واقعی تغییر دهد.

۲. مدیریت حافظه پویا: پس از تخصیص اولیه، مجازی ساز باید حافظه را به طور مداوم مدیریت کند تا اطمینان حاصل شود که ماشین مجازی‌ها به مقدار حافظه مورد نیاز دسترسی دارند و منابع به طور بهینه استفاده می‌شوند. این مدیریت شامل پایش استفاده از حافظه، تطبیق تخصیص‌ها و استفاده از تکنیک‌های مختلف بهینه‌سازی است.

مفهوم تعهد بیش از حد حافظه و چالش‌های مربوط به آن:

Overcommitment حافظه یکی از تکنیک‌های مهم در مدیریت حافظه مجازی است که به مجازی ساز اجازه می‌دهد تا بیش از مقدار حافظه فیزیکی واقعی، حافظه به ماشین مجازی‌ها اختصاص دهد. این تکنیک بر این فرض استوار است که همه ماشین مجازی‌ها در هر زمان مشخص به حداکثر مقدار حافظه

تخصیص‌یافته نیاز نخواهند داشت. به عبارت دیگر، مجازی ساز می‌تواند منابع حافظه را به طور مجازی بین ماشین مجازی‌ها به اشتراک بگذارد.

با وجود مزایای Overcommitment حافظه، این تکنیک با چالش‌های زیادی همراه است:

۱. ریسک کمبود حافظه: اگر چندین ماشین مجازی به طور همزمان به حافظه بیشتری از حافظه فیزیکی موجود نیاز داشته باشند، مجازی ساز با کمبود حافظه مواجه می‌شود که می‌تواند منجر به کاهش عملکرد یا حتی خرابی ماشین مجازی‌ها شود.

۲. مدیریت پیچیده: Overcommitment نیازمند سامانه‌های پیشرفته پایش و مدیریت حافظه است تا از بروز مشکلات احتمالی جلوگیری شود. مجازی ساز باید به طور مداوم استفاده از حافظه را رصد کرده و در صورت لزوم تخصیص‌ها را تطبیق دهد.

۳. تأثیر بر عملکرد: در صورت بروز کمبود حافظه، مجازی ساز ممکن است نیاز به استفاده از تکنیک‌هایی مانند مبادله کردن(swapping) داشته باشد که می‌تواند به طور قابل توجهی عملکرد ماشین مجازی‌ها را تحت تأثیر قرار دهد.

تکنیک‌های مختلف برای بهینه‌سازی مصرف حافظه:

برای مدیریت کارآمد حافظه و بهینه‌سازی مصرف آن در محیط‌های مجازی، مجازی سازها از تکنیک‌های مختلفی استفاده می‌کنند. برخی از این تکنیک‌ها عبارتند از:

۱. **Ballooning**

Ballooning یک تکنیک برای بازگرداندن حافظه از ماشین مجازی‌ها به مجازی ساز است. این فرآیند توسط یک درایور خاص که در ماشین مجازی اجرا می‌شود (balloon driver) انجام می‌گیرد. این درایور مقداری از حافظه ماشین مجازی را اشغال کرده و به مجازی ساز بازمی‌گرداند. مجازی ساز سپس می‌تواند این حافظه را به ماشین مجازی‌های دیگر که به حافظه بیشتری نیاز دارند تخصیص دهد. Ballooning به مجازی ساز امکان می‌دهد تا به طور پویا حافظه را بین ماشین مجازی‌ها توزیع کند بدون نیاز به خاموش کردن یا تغییر تنظیمات ماشین مجازی‌ها.

۲. **مبادله کردن(swapping)**

Swapping فرآیندی است که در آن مجازی ساز بخش‌هایی از حافظه ماشین مجازی‌ها را به دیسک‌های ذخیره‌سازی منتقل می‌کند تا حافظه فیزیکی بیشتری برای سایر ماشین مجازی‌ها

آزاد شود. این تکنیک به‌ویژه در شرایط کمبود حافظه مفید است، اما می‌تواند تأثیر منفی زیادی بر عملکرد ماشین مجازی‌ها داشته باشد، زیرا دسترسی به دیسک‌های ذخیره‌سازی به مراتب کندتر از دسترسی به حافظه فیزیکی است. بنابراین، مجازی ساز باید با دقت این تکنیک را مدیریت کند تا تأثیرات منفی آن به حداقل برسد.

۳. **کپی برداری(Deduplication)**

Deduplication حافظه تکنیکی است که داده‌های تکراری در حافظه را شناسایی و حذف می‌کند. با استفاده از این تکنیک، مجازی ساز می‌تواند نسخه‌های تکراری داده‌ها را در حافظه شناسایی کرده و تنها یک نسخه از آن‌ها را نگهداری کند. این کار باعث کاهش استفاده از حافظه و افزایش بهره‌وری منابع می‌شود. Deduplication معمولاً در محیط‌هایی که ماشین مجازی‌های مشابه با نرم‌افزارها و داده‌های مشابه اجرا می‌شوند، بیشترین بهره‌وری را دارد.

این تکنیک‌ها در کنار هم به مجازی ساز کمک می‌کنند تا مدیریت حافظه را بهینه کرده و عملکرد ماشین مجازی‌ها را در شرایط مختلف تضمین کند. با استفاده از این تکنیک‌ها، مجازی ساز می‌تواند منابع حافظه را به‌طور کارآمدتری تخصیص دهد و بهره‌وری کلی سامانه را بهبود بخشد.

۲-۴ معرفی کتابخانه جهت ارتباط با مجازی‌ساز

libvirt یک کتابخانه برنامه‌نویسی است که برای مدیریت و کنترل محیط‌های مجازی مورد استفاده قرار می‌گیرد. این کتابخانه به برنامه‌نویسان اجازه می‌دهد با استفاده از رابط‌های برنامه‌نویسی (API) آن، اقداماتی مانند ایجاد، مدیریت، حذف و پایش ماشین‌های مجازی را انجام دهند. libvirt از زبان‌های برنامه‌نویسی متعددی مانند C، پایتون[۳]، Perl، و Java پشتیبانی می‌کند و می‌تواند با انواع مختلف مجازی سازها مانند KVM، Xen، VMware و Hyper-V کار کند[۵].

کاربردهای اصلی libvirt عبارتند از:

۱. **مدیریت ماشین‌های مجازی:** libvirt به برنامه‌نویسان اجازه می‌دهد تا ماشین‌های مجازی را ایجاد، مدیریت و کنترل کنند. این شامل ایجاد ماشین مجازی‌ها، راه‌اندازی، متوقف کردن، حذف و مدیریت منابعی مانند حافظه، واحد پردازش مرکزی و شبکه است.

[3] Python

۲. **جمع‌آوری اطلاعات مربوط به ماشین‌های مجازی:** libvirt اطلاعات جامعی از وضعیت و عملکرد ماشین مجازی‌ها و مجازی سازها را فراهم می‌کند. این اطلاعات شامل اطلاعاتی مانند استفاده از واحد پردازش مرکزی و حافظه، وضعیت آنلاین بودن ماشین مجازی‌ها، و مشخصات سخت‌افزاری مرتبط با مجازی سازها است.

نحوه استفاده از کتابخانه و جمع‌آوری مؤلفه‌ها:

برای استفاده از libvirt برای مدیریت ماشین‌های مجازی و جمع‌آوری اطلاعات مربوطه، می‌توان از رابط‌های برنامه‌نویسی (API) آن استفاده کرد. این APIها در زبان‌های مختلفی مانند ، Cپایتون، و Java قابل دسترسی هستند.

برای مدیریت ماشین‌های مجازی با استفاده از libvirt، مراحل زیر را می‌توان دنبال کرد:

۱. **اتصال به مجازی ساز**: ابتدا باید به مجازی ساز مورد نظر متصل شوید. این مرحله شامل انتخاب مجازی ساز مورد نظر (مانند KVM یا Xen) و برقراری ارتباط با آن می‌شود.

۲. **ایجاد ماشین مجازی**: با استفاده از APIهای libvirt، می‌توانید یک یا چند ماشین مجازی ایجاد کنید، مؤلفه های مورد نیاز ماشین مجازی را تنظیم کنید و آن‌ها را راه‌اندازی کنید.

۳. **مدیریت ماشین مجازی**: پس از ایجاد ماشین مجازی، می‌توانید عملیات مختلفی مانند راه‌اندازی، متوقف کردن، حذف و تغییر منابع ماشین مجازی‌ها را با استفاده از libvirt انجام دهید.

جمع‌آوری اطلاعات مربوط به ماشین‌های مجازی:

برای جمع‌آوری اطلاعات مربوط به ماشین‌های مجازی و مجازی سازها با استفاده از libvirt، می‌توانید از توابع و روش‌های مربوط به پایش استفاده کنید. این شامل دریافت اطلاعاتی مانند استفاده از واحد پردازش مرکزی و حافظه، وضعیت ماشین مجازی‌ها و مجازی سازها، و مشخصات سخت‌افزاری است.

به طور کلی، با استفاده از libvirt و APIهای آن، می‌توانید به طور کامل محیط‌های مجازی خود را مدیریت کنید و اطلاعات جامعی از وضعیت و عملکرد آن‌ها را جمع‌آوری کنید.

۲-۵ پیش‌بینی مصرف حافظه

پیش‌بینی مصرف حافظه یکی از جنبه‌های حیاتی در مدیریت منابع سامانه‌های مجازی است. با توجه به محدودیت‌های منابع موجود بر روی هر سرور، بهینه‌سازی و استفاده بهینه از حافظه از اهمیت بالایی

برخوردار است. پیش‌بینی مصرف حافظه به اپراتورها اجازه می‌دهد تا منابع حافظه را به‌طور بهینه‌تر مدیریت کرده و مشکلاتی نظیر افزایش فشار بر حافظه و بار بالای سرور را پیش‌بینی و جلوگیری کنند. عدم پیش‌بینی مصرف حافظه می‌تواند منجر به مشکلاتی مانند افزایش اختلالات سامانه‌ای، کاهش عملکرد، و حتی از دست رفتن دسترسی به سرویس‌ها شود[۱۱].

روش‌های مختلف پیش‌بینی: کاربردهای اصلی libvirt عبارتند از:

برای پیش‌بینی مصرف حافظه، می‌توان از روش‌های مختلفی استفاده کرد که شامل روش‌های آماری و الگوریتم‌های یادگیری ماشین می‌شوند.

۱. **روش‌های آماری**: این روش‌ها بر اساس تحلیل داده‌های سابقه و الگوهای آماری اطلاعات مصرف حافظه را پیش‌بینی می‌کنند. مثال‌هایی از این روش‌ها شامل مدل‌های زمانی (مانند ARIMA و SARIMA) و مدل‌های رگرسیون (مانند مدل‌های خطی و غیرخطی) می‌باشند.

۲. **الگوریتم‌های یادگیری ماشین**: این روش‌ها بر اساس تجربه و داده‌های مشاهده شده، الگوهای پیچیده‌تری را برای پیش‌بینی مصرف حافظه ایجاد می‌کنند. این الگوریتم‌ها از شبکه‌های عصبی مصنوعی (مانند شبکه‌های LSTM و GRU)، درخت‌های تصمیم و ماشین‌های بردار پشتیبان (SVM) استفاده می‌کنند. این الگوریتم‌ها اغلب می‌توانند الگوهای پیچیده‌تری را کشف کرده و به دقت بالاتری در پیش‌بینی مصرف حافظه دست یابند.

با استفاده از این روش‌ها، می‌توان پیش‌بینی‌های دقیق‌تر و بهینه‌تری از مصرف حافظه در محیط‌های مجازی ارائه داد که به بهبود عملکرد و بهره‌وری سامانه‌های مجازی کمک می‌کند.

۲-۶ سامانه‌های پایش و جمع‌آوری داده

سامانه‌های پایش مانندپرومتئوس ابزارهای قدرتمندی هستند که برای جمع‌آوری، ذخیره و نمایش داده‌های عملکردی سامانه‌ها و برنامه‌های نرم‌افزاری استفاده می‌شوند.پرومتئوس یک سامانه متن باز است که به منظور پایش و هشداردهی طراحی شده و از زبان برنامه‌نویسی Go برای پیاده‌سازی آن استفاده شده است. این ابزار از مدل معماری Pull (استخراجی) استفاده می‌کند که در آن، سرورپرومتئوس داده‌ها را از انواع مختلف منابع جمع‌آوری می‌کند.

اهمیت جمع‌آوری داده‌های عملکردی و استفاده از آن‌ها برای بهینه‌سازی:

جمع‌آوری داده‌های عملکردی از اهمیت بسیاری برخوردار است، زیرا این داده‌ها اطلاعات مفیدی را ارائه می‌دهند که می‌توانند در فرآیند تصمیم‌گیری و بهینه‌سازی سامانه‌ها و برنامه‌های نرم‌افزاری موثر باشند. با جمع‌آوری داده‌های عملکردی مانند زمان پاسخ، تعداد درخواست‌های همزمان، میزان ترافیک شبکه، و سایر معیارهای عملکرد، می‌توان به راحتی الگوها و رفتارهای نامطلوب سامانه را شناسایی کرده و بهینه‌سازی‌های لازم را اعمال کرد.

از دیگر مزایای جمع‌آوری داده‌های عملکردی می‌توان به ارائه آمارها و گزارش‌های تحلیلی برای تصمیم‌گیری‌های مبتنی بر داده، پیش‌بینی تناقض‌ها و اختلالات در سامانه، مانیتور کردن عملکرد و عملکرد در طول زمان، و شناسایی نیازهای موردی سامانه برای بهبود عملکرد اشاره کرد.

در کل، جمع‌آوری داده‌های عملکردی و استفاده موثر از آن‌ها می‌تواند به بهبود عملکرد، کاهش هزینه‌ها، و افزایش بهره‌وری کمک کند و نقش بسیار مهمی در موفقیت و پایداری سامانه‌های نرم‌افزاری دارد.

۲-۷ نمودارها و داشبوردهای پایش

داشبوردهای پایش به عنوان رابط کاربری گرافیکی، نقش بسیار مهمی در تجزیه و تحلیل داده‌های جمع‌آوری شده از سامانه‌ها و برنامه‌ها دارند. این داشبوردها به مدیران و متخصصان اجازه می‌دهند تا به سادگی و با دسترسی به اطلاعات مفید، وضعیت کلی سامانه‌ها را درک کنند، عملکرد آن‌ها را مانیتور کنند، و در صورت نیاز به تصمیم‌گیری‌های اصلاحی بپردازند.

این داشبوردها معمولاً شامل نمودارها، گراف‌ها، جداول، و معیارهای کلیدی عملکردی (KPIs) هستند که به صورت زنده و به روز شده نمایش داده می‌شوند. با تمرکز بر اطلاعات مهم و مرتبط، داشبوردهای پایش به مدیران امکان می‌دهند تا با سرعت و دقت به موضوعات اساسی و اولیه دسترسی پیدا کنند و برای بهبود عملکرد سامانه‌ها اقدام کنند

معرفی گرافانا به عنوان ابزار نمایش داده و ایجاد داشبوردهای پایش:

گرافانا یکی از ابزارهای قدرتمند برای نمایش داده و ایجاد داشبوردهای پایش است. این ابزار از طراحی ساده و کاربرپسند برخوردار است و امکانات فراوانی برای ساخت و تنظیم داشبوردهای گرافیکی فراهم می‌کند. با استفاده از گرافانا، می‌توانید از نمودارها، نمودارهای متحرک، جداول داده، داشبوردهای متنوع،

و ابزارهای دیگر برای نمایش داده‌های عملکردی خود استفاده کنید.

گرافانا قابلیت اتصال به انواع مختلف منابع داده را دارد، از جمله پرومتئوس، InfluxDB، Elasticsearch، MySQL، و سایر منابع. این امکان به کاربران اجازه می‌دهد که اطلاعات مورد نیاز خود را از منابع مختلف جمع‌آوری کنند و آن‌ها را به صورت دلخواه در داشبوردهای گرافانا نمایش دهند.

با امکانات چندین کاربر، سطح دسترسی، و قابلیت‌های پیشرفته مانند ایجاد هشدارها، گرافانا یک ابزار قدرتمند برای پایش و تحلیل داده‌های عملکردی سامانه‌ها و برنامه‌ها است.

۲-۸ زبان برنامه‌نویسی گولنگ

زبان برنامه‌نویسی Go یا گولنگ یک زبان برنامه‌نویسی مدرن و متن‌باز است که توسط گوگل توسعه داده شده است. این زبان در سال ۲۰۰۹ به عنوان یک پروژه آزمایشی معرفی شد و سپس به عنوان یک زبان برنامه‌نویسی مناسب برای توسعه نرم‌افزارهای مقیاس‌پذیر و کارآمد شناخته شد[۲].

ویژگی‌های زبان:

سادگی و خوانایی: Go با تمرکز بر سادگی و خوانایی کد، به برنامه‌نویسان اجازه می‌دهد که به راحتی کدی را ایجاد کنند که قابل فهم و توسعه آن ساده باشد.

کارآیی بالا: زمان اجرای کدهای Go به دلیل استفاده از کامپایلر موازی و مدیریت بهینه حافظه بسیار سریع است.

پشتیبانی از همروندی: Go از طراحی از پیش برای همروندی و همچنین از ابزارهای برنامه‌نویسی مانند کانال‌ها پشتیبانی می‌کند که ایجاد برنامه‌های همروند را آسان می‌کند.

پورتابل(portable): Go روی اکثر سامانه‌های عامل اجرا می‌شود و می‌توان کد را به صورت مستقیم بر روی سامانه‌های مختلف کامپایل(compile) کرد.

کاربردهای زبان:

برنامه‌نویسی سامانه: Go برای توسعه نرم‌افزارهای سامانه‌ای که نیاز به کارایی بالا و مدیریت منابع می‌باشد، مورد استفاده قرار می‌گیرد.

توسعه وب: با استفاده از چارچوب‌هایی مانند Gin یا Echo، Go برای توسعه وب‌سایت‌ها و وب‌سرویس‌های قدرتمند و کارآمد استفاده می‌شود.

مجازی‌سازی و مدیریت منابع: به عنوان زبانی با کارایی بالا و پشتیبانی از همروندی، Go برای توسعه ابزارها و برنامه‌های مرتبط با مجازی‌سازی و مدیریت منابع استفاده می‌شود. به طور خاص، در پروژه‌هایی

که نیاز به عملیات پیشرفته مانند مدیریت حافظه یا مدیریت منابع سرورها دارند، Go می‌تواند یک انتخاب مناسب باشد.

با استفاده از زبان برنامه‌نویسی Go، برنامه‌نویسان می‌توانند به سرعت و به صورت کارآمد برنامه‌هایی با کیفیت بالا را ایجاد کرده و نیازهای مختلف پروژه‌های مرتبط با مجازی‌سازی و مدیریت منابع را برطرف کنند.

۲-۹ سیستم عامل لینوکس

سیستم عامل لینوکس یکی از پرکاربردترین سیستم عامل‌های رایانه‌ای است که بر اساس هسته لینوکس (Linux Kernel) توسعه یافته است. این سیستم عامل به صورت متن‌باز منتشر می‌شود و به عنوان یکی از پایه‌های مجازی‌سازی و سرورهای ابری مورد استفاده قرار می‌گیرد[۱۰].

معرفی سیستم عامل لینوکس و اهمیت آن در مجازی‌سازی و سرورهای ابری:

متن‌باز بودن: این سیستم عامل برای همه‌ی افراد و شرکت‌ها به‌راحتی قابل دسترس است و از طریق توسعه‌دهندگان متعددی ارتقاء می‌یابد.

پایداری و امنیت: با توجه به تعداد بالای کاربران و توسعه‌دهندگان، امنیت و پایداری این سیستم عامل بسیار بالاست که برای استفاده در سرورهای ابری بسیار اهمیت دارد.

پشتیبانی از ابزارهای متعدد: سیستم عامل لینوکساز طیف گسترده‌ای از ابزارها و پکیج‌های متن‌باز برای مدیریت، پایش، و اجرای برنامه‌های مختلف استفاده می‌کند که این امر برای مجازی‌سازی و سرورهای ابری بسیار مفید است.

۲-۱۰ بارکاری

بارکاری به مجموعه‌ای از فعالیت‌ها، پردازش‌ها، یا تراکنش‌هایی اطلاق می‌شود که بر روی یک سامانه یا شبکه اجرا می‌شوند. انواع مختلف بارکاری شامل بارکاری وب، بارکاری پایگاه داده، و بارکاری شبکه هستند. بارکاری وب مربوط به درخواست‌ها و پردازش داده‌های ورودی و خروجی در سرویس‌های وب است. بارکاری پایگاه داده شامل عملیات خواندن، نوشتن، و بروزرسانی داده‌ها در پایگاه داده می‌باشد، و بارکاری شبکه مرتبط با انتقال داده‌ها بین دستگاه‌ها، پهنای باند، و پردازش اطلاعات در شبکه است.

اهمیت شناخت و مدیریت بارکاری در محیط‌های مجازی‌سازی و سرورهای ابری:

شناخت و مدیریت بارکاری در محیط‌های مجازی‌سازی و سرورهای ابری امری بسیار حیاتی است. با شناخت بارکاری، می‌توان به کمک بهره‌وری منابع و پیش‌بینی نیازمندی‌ها کمک کرد. همچنین، مدیریت بارکاری از افزایش عملکرد، ارتقاء کارایی، و پیش‌بینی نیازمندی‌ها در زمان توسعه و بهره‌برداری سامانه حمایت می‌کند.

cloud suit data analytics به عنوان یک بارکاری:

cloud suit data analytics به عنوان یک بارکاری مشخصی در نظر گرفته می‌شود که در آن اطلاعات مورد نیاز از سرویس‌های ابری جمع‌آوری، ذخیره، و ارسال می‌شود تا به مشتریان ارائه داده‌ها و سرویس کند. این بارکاری بهینه‌سازی شده برای ارائه سرویس‌های مبتنی بر ابر است و نیازمندی‌های آن شامل پردازش داده‌ها، ذخیره‌سازی، و انتقال داده‌ها می‌باشد. اجرای این بارکاری نیازمند استفاده از منابع محاسباتی و شبکه در محیط‌های ابری و مجازی‌سازی است که باید با دقت و بهینه‌سازی انجام شود تا کارایی و عملکرد بهینه را فراهم کند[۷].

۲-۱۱ خلاصه

در بخش مفاهیم پایه، مفاهیم اساسی و ابزارهای کلیدی که در سامانه تخصیص حافظه[۴] به کار گرفته شده‌اند، مورد بررسی و توضیح قرار گرفت. این شامل مفاهیمی مانند مجازی سازی(Virtualization) و مدیریت ماشین‌های مجازی است که با استفاده از فناوری‌هایی مانند KVM و Libvirt، سرورها مجازی سازی شده‌اند. همچنین در این بخش به توضیحاتی درباره مدیریت منابع و پیش‌بینی پرداخته شده‌است که ابزارهایی مانند گولنگ و پیش بینی کننده[۵] برای این امور به کار گرفته‌شده‌اند. جمع‌آوری و ارسال داده‌ها نیز از اهمیت بالایی برخوردار بوده‌است و ابزارهایی مانند Exporter و Web API برای این منظور استفاده شده‌اند. در نهایت، بخشی از مفاهیم پایه این است که چگونه ارتباط و رابط کاربری بین عامل و سرور ایجاد شده‌است که از طریق زبان‌ها و ابزارهایی مانند پایتون و Flask و همچنین SQLite انجام شده‌است. این مفاهیم اساسی به توضیح و درک بهتر از ساختار و عملکرد سامانه کمک کرده‌اند و اساسی‌ترین بخش‌های آن را شکل داده‌اند.

[4]Memory Harvesting
[5]Predictor

فصل سوم

طراحی و پیاده‌سازی سامانه

در این فصل، به مراحلی که برای پیاده‌سازی این سامانه طی شده، پرداخته شده‌است. در ابتدا جزئیات پیاده‌سازی و نمودار سامانه را توضیح خواهیم‌داد و پس از عامل ، به توضیح سرور می‌پردازیم،

۳-۱ نمودار و پیاده‌سازی سامانه

سامانه به دو بخش اصلی تقسیم می‌شود: عامل و سرور. بخش عامل مسئول اجرای عملیات در سطح سرور فیزیکی است و شامل اجزایی مانند KVM (Kernel-based Virtual Machine) برای مجازی‌سازی سخت‌افزار سرور و مدیریت ماشین‌های مجازی است که بارهای کاری مختلف را اجرا می‌کنند. زبان برنامه‌نویسی گولنگ برای توسعه عامل انتخاب شده است، زیرا از قابلیت‌های بالا در مدیریت هم‌زمانی و کارایی بالا برخوردار است. Systemd نیز برای مدیریت و اجرای سرویس‌های مختلف استفاده می‌شود و تضمین می‌کند که سرویس‌ها به صورت صحیح و در زمان مناسب اجرا شوند. Libvirt به عنوان یک ابزار مدیریت ماشین‌های مجازی و جمع‌آوری اطلاعات مصرف منابع به کار می‌رود. پیش بینی کننده[۱] وظیفه پیش‌بینی مصرف حافظه آینده را بر عهده دارد، که این امکان را فراهم می‌کند تا سامانه منابع خود را بهینه‌تر مدیریت کند. Violent Counter تعداد مواردی را که مصرف حافظه از پیش‌بینی‌های انجام شده فراتر رفته است، شمارش می‌کند. صادر کننده[۲] داده‌های جمع‌آوری شده را به فرمتی قابل خواندن توسط پرومتئوس تبدیل می‌کند. در نهایت، Web API ارتباط بین عامل و سرور را برای ارسال داده‌ها و دریافت مؤلفه های تنظیم فراهم می‌کند.

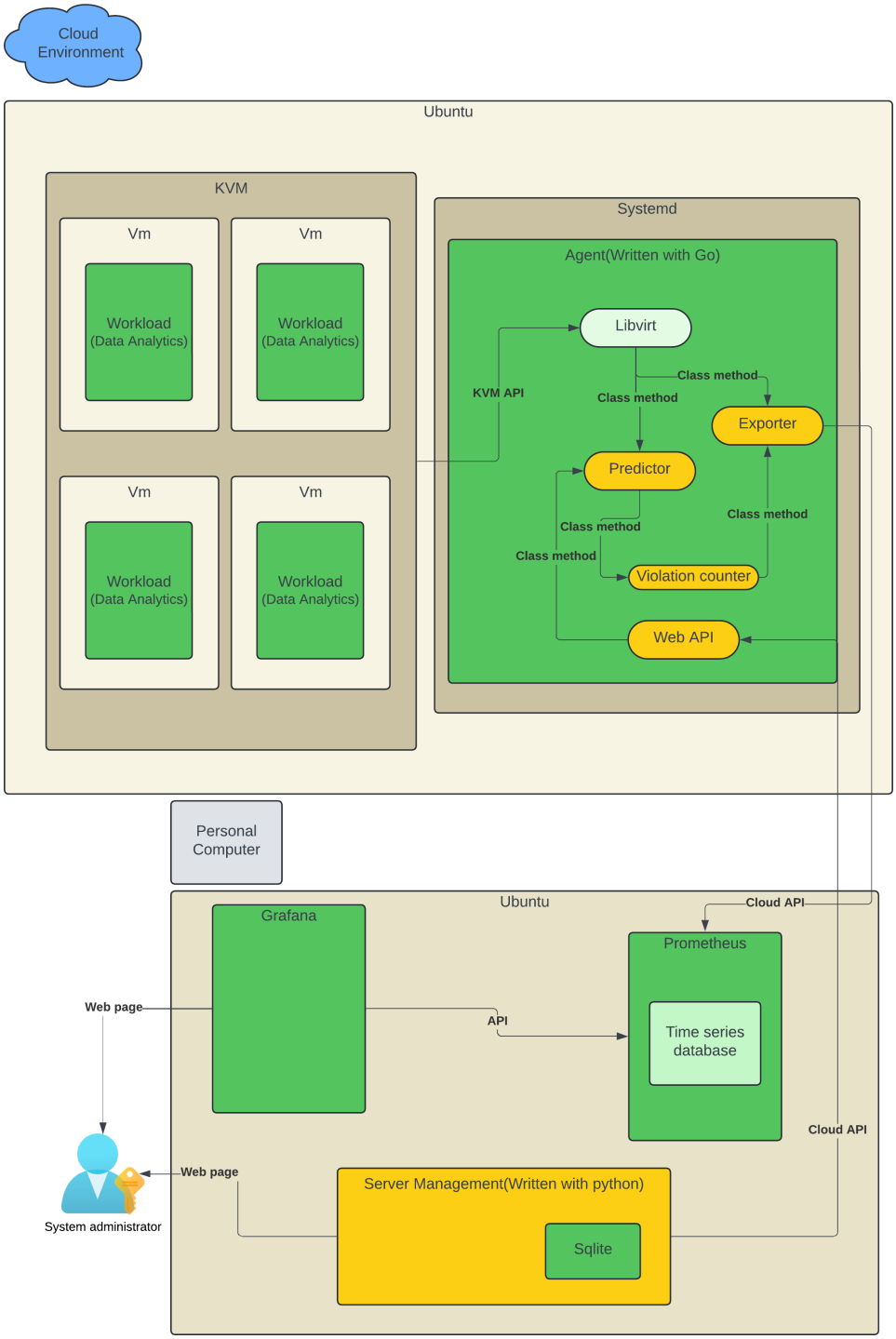
در بخش سرور،پرومتئوس برای جمع‌آوری و ذخیره‌سازی داده‌های مربوط به مصرف حافظه و عملکرد سامانه به کار می‌رود. این داده‌ها در یک پایگاه داده زمانی (Time Series Database) ذخیره می‌شوند که امکان تحلیل دقیق و سریع داده‌ها را فراهم می‌کند. گرافانا به عنوان ابزار نمایش داده و ایجاد داشبوردهای پایش مورد استفاده قرار می‌گیرد. این ابزار به مدیر سامانه(User) اجازه می‌دهد تا داده‌های مربوط به عملکرد و مصرف منابع را به صورت گرافیکی مشاهده کند و تصمیمات مدیریتی بهتری بگیرد. بخش سرور با استفاده از زبان پایتون و چارچوب وب Flask توسعه یافته است تا APIهای مورد نیاز برای ارتباط با عامل را فراهم کند. SQLite به عنوان یک پایگاه داده سبک برای ذخیره‌سازی تنظیمات و داده‌های مورد نیاز استفاده می‌شود. مدیر سامانه از طریق داشبوردهای پرومتئوس و گرافانا داده‌ها را مشاهده و تنظیمات مربوطه را مدیریت می‌کند، که این کار به بهینه‌سازی عملکرد و استفاده از منابع

[1] Predictor

[2] Exporter

سامانه کمک می‌کند.

که در ادامه میتوانیم نمودار بلوکی سامانه را مشاهده کنیم:



شکل ۳-۱: نمودار بلوکی سامانه

در این نمودار بخش‌های خاکستری رنگ پودمان‌های سخت‌افزاری سامانه و بخش‌های سبزرنگ پودمان‌های نرم‌افزاری را نشان می‌دهند و علاوه برآن بخش‌های زردرنگ توسط این پژوهش پیاده‌سازی شده‌است.

۳-۲ عامل

عامل یک بخش اصلی در سامانه "تخصیص حافظه[۳]" است که به سه بخش مجازی‌سازی و مدیریت ماشین‌های مجازی، مدیریت منابع و پیش‌بینی و جمع‌آوری و ارسال داده‌ها تقسیم میشود که مسئول اجرای بارهای کاری مختلف بر روی ماشین‌های مجازی می‌باشد. این بخش شامل تکنولوژی‌هایی مانند KVM برای مجازی‌سازی سخت‌افزار، گولنگ به عنوان زبان برنامه‌نویسی برای کارایی بالا و هم‌زمانی، و Libvirt برای مدیریت ماشین‌های مجازی و جمع‌آوری اطلاعات مصرف منابع می‌باشد. همچنین، از پیش بینی کننده[۴] برای پیش‌بینی مصرف حافظه آینده و از Violent Counter برای شمارش تناقض‌ها از پیش‌بینی مصرف حافظه استفاده می‌شود .

مجازی‌سازی و مدیریت ماشین‌های مجازی:

این بخش مسئول مجازی‌سازی سخت‌افزار و اجرای بارهای کاری مختلف بر روی ماشین‌های مجازی است که به بخش های زیر تقسیم میشود:

۱. **KVM (Kernel-based Virtual Machine)**

KVM یک فناوری مجازی‌سازی سطح سیستم عامل است که اجازه اجرای چندین ماشین مجازی را روی یک سرور فیزیکی با استفاده از هسته سیستم عامل فراهم می‌کند. این فناوری امکانات مربوط به مجازی‌سازی سخت‌افزاری را فراهم می‌کند و به عنوان قسمتی از هسته لینوکس ارائه شده است[۹].

۲. **Libvirt:**

Libvirt یک مجموعه ابزار و رابط برنامه نویسی (API) است که برای مدیریت ماشین‌های مجازی بر روی همه‌ی فناوری‌های مجازی‌سازی استفاده می‌شود. این ابزار امکاناتی مانند مهمان‌های

[3] Memory Harvesting
[4] Predictor

مجازی‌سازی را از طریق خط فرمان یا از طریق کتابخانه‌های برنامه‌نویسی (مانند libvirt-python) فراهم می‌کند.

۳. **ماشین مجازی(Virtual Machines)**

ماشین‌های مجازی محیط‌هایی هستند که توسط فناوری مجازی‌سازی ایجاد شده‌اند و به عنوان محیطی مستقل و کاملاً جداگانه از سرورهای فیزیکی عمل می‌کنند. هر ماشین مجازی می‌تواند سیستم عامل، برنامه‌ها، و منابع مورد نیاز خود را داشته باشد و به عنوان یک کامپیوتر مجازی عمل می‌کند.

۴. **Workload**

بارکاری(Workload) به مجموعه‌ای از برنامه‌ها، فرآیندها، و سرویس‌هایی اطلاق می‌شود که در ماشین‌های مجازی اجرا می‌شوند. این بارهای کاری ممکن است شامل برنامه‌های وب، بانک‌های اطلاعاتی، سرورهای فایل، و غیره باشند که هرکدام نیازمندی‌ها و منابع مختلفی را دارند.

مدیریت منابع و پیش‌بینی:

این بخش مسئول خواندن اطلاعات ماشین های مجازی و پیش بینی آینده آن و بررسی صحت آن می باشد که به بخش های زیر تقسیم میشود:

۱. **گولنگ**

گولنگ یا Go یک زبان برنامه‌نویسی با کارایی بالا است که برای توسعه عامل و مدیریت هم‌زمانی استفاده می‌شود. این زبان برای کارهایی که نیاز به عملکرد بالا، پردازش موازی، و کار با سامانه‌های متعدد دارند، مناسب است. استفاده از گولنگ به عنوان زبان اصلی برنامه‌نویسی عامل، به سرعت و کارایی بالا کمک می‌کند.

۲. **پیش بینی کننده**

پودما پیش بینی کننده وظیفه پیش‌بینی مصرف حافظه آینده را بر عهده دارد. این پودمان با تحلیل الگوهای مصرف حافظه گذشته و شناسایی روندها و پیش‌بینی‌های مربوط به مصرف آینده، به سامانه امکان می‌دهد تا منابع حافظه را بهینه‌تر تخصیص دهد و بازدهی سامانه را افزایش دهد.

برای پیش‌بینی مصرف حافظه در سه دقیقه آینده، میانگین ۱۲ نمونه گذشته گرفته می‌شود و سپس این مقدار در مؤلفه آستانه ضرب می‌شود. فرمول ریاضی آن به شرح زیر است:

فرض کنید M_i مصرف حافظه در نمونه i باشد، که $i = 1, 2, \ldots, 12$ است.

میانگین ۱۲ نمونه در سه دقیقه گذشته به این صورت محاسبه می‌شود:

$$\text{Avg}_{12}(t) = \frac{1}{12} \sum_{i=1}^{12} M_i(t)$$

سپس این میانگین در مؤلفه آستانه ضرب می‌شود تا پیش‌بینی مصرف حافظه برای سه دقیقه آینده به دست آید:

$$\hat{M}(t+3) = \text{Avg}_{12}(t) \times \text{آستانه}$$

به طور خلاصه، فرمول ریاضی پیش‌بینی مصرف حافظه به این شکل خواهد بود:

$$\hat{M}(t+3) = \frac{1}{12} \sum_{i=1}^{12} M_i(t) \times \text{آستانه}$$

در این فرمول:

- $\text{Avg}_{12}(t)$ میانگین مصرف حافظه در ۱۲ نمونه (هر ۱۵ ثانیه یک نمونه) در سه دقیقه گذشته است.
- آستانه مؤلفه تنظیمی شما برای تعیین پیش‌بینی مصرف حافظه در سه دقیقه آینده.
- $\hat{M}(t+3)$ پیش‌بینی مصرف حافظه در سه دقیقه آینده است.

این فرمول به صورت دوره‌ای هر سه دقیقه تکرار می‌شود تا پیش‌بینی‌های متوالی برای دوره‌های سه دقیقه‌ای آینده فراهم شود.

۳. Violent Counter

شمارشگر تعداد تناقض‌ها از پیش‌بینی مصرف حافظه است. این شمارشگر وظیفه شمارش تعداد بارهای کاری که از پیش‌بینی مصرف حافظه خارج می‌شوند و به این ترتیب ارزیابی کیفیت پیش‌بینی‌ها و مدیریت منابع حافظه را بهبود می‌بخشد.

جمع‌آوری و ارسال داده‌ها:

جمع‌آوری و ارسال داده‌ها در سامانه "تخصیص حافظه" دارای دو اجزا اصلی است:

۱. **Exporter**

این پودمان[5] مسئول تبدیل داده‌های جمع‌آوری شده توسط سامانه به قالبی قابل خواندن توسط پرومتئوس است. با تبدیل داده‌ها به فرمت مورد نظرپرومتئوس، این اطلاعات قابلیت استفاده و تحلیل توسط این سامانه را فراهم می‌کند.

۲. **Web API**

این رابط کاربری برای برقراری ارتباط بین عامل و سرور استفاده می‌شود. این رابط از طریق درخواست‌های HTTP اطلاعات را از عامل دریافت کرده و آن‌ها را به سرور ارسال می‌کند. همچنین، می‌تواند مؤلفه های تنظیمی را به عامل ارسال و پاسخ‌هایی را که از عامل دریافت می‌کند به سرور ارسال کند.

۳-۳ سرور

سرور در سامانه "تخصیص حافظه" نقش مرکزی را ایفا می‌کند و از جمع‌آوری، ذخیره، و نمایش داده‌های مربوط به مصرف حافظه و عملکرد سامانه مسئولیت دارد. این سرور از ابزارهایی مانندپرومتئوس برای جمع‌آوری داده‌ها و ذخیره‌سازی آن‌ها در پایگاه داده‌های سری زمانی استفاده می‌کند. سپس، با استفاده از ابزار نمایش داده مانند گرافانا، اطلاعات مربوط به مصرف حافظه و عملکرد سامانه را به شکل داشبوردها و نمودارها نمایش می‌دهد تا کاربران بتوانند عملکرد سامانه را به صورت گرافیکی مشاهده و مدیریت کنند. در کنار این‌ها، سرور از زبان برنامه‌نویسی پایتون و چارچوب Flask برای ایجاد APIهای مورد نیاز برای ارتباط با عامل استفاده می‌کند و از پایگاه داده SQLite برای ذخیره‌سازی تنظیمات و داده‌های مورد نیاز استفاده می‌کند.

جمع‌آوری داده‌ها و ذخیره‌سازی:

پرومتئوس[6] یک سامانه متن باز برای پایش و جمع‌آوری داده‌های مربوط به عملکرد سامانه است که از طریق ارسال درخواست‌های HTTP یا HTTPS به سرورهای هدف، اطلاعات را به صورت متن‌سازی شده دریافت می‌کند. این اطلاعات شامل متریک‌هایی مانند میزان مصرف حافظه، بار واحد پردازش

[5] Module
[6] Prometheus

مرکزی، تعداد درخواست‌های وب، و غیره می‌باشند.پرومتئوس این داده‌ها را بر اساس زمان ثبت شده ذخیره می‌کند و امکان جستجو، پرس‌وجو، و تحلیل آن‌ها را فراهم می‌کند.

Time Series Database یک نوع پایگاه داده است که برای ذخیره‌سازی داده‌هایی که به طور مداوم در طول زمان جمع‌آوری می‌شوند، استفاده می‌شود. این داده‌ها به صورت سری‌های زمانی مرتب شده و زمانی ثبت شده در پایگاه داده ذخیره می‌شوند. این پایگاه داده امکان جستجو، پرس‌وجو، و تحلیل داده‌ها بر اساس زمان را فراهم می‌کند و به عنوان مخزن اطلاعات برای سامانه‌های پایش و مدیریت عمل می‌کند.

نمایش و پایش:

شکل ۳-۲: داشبورد گرافانا

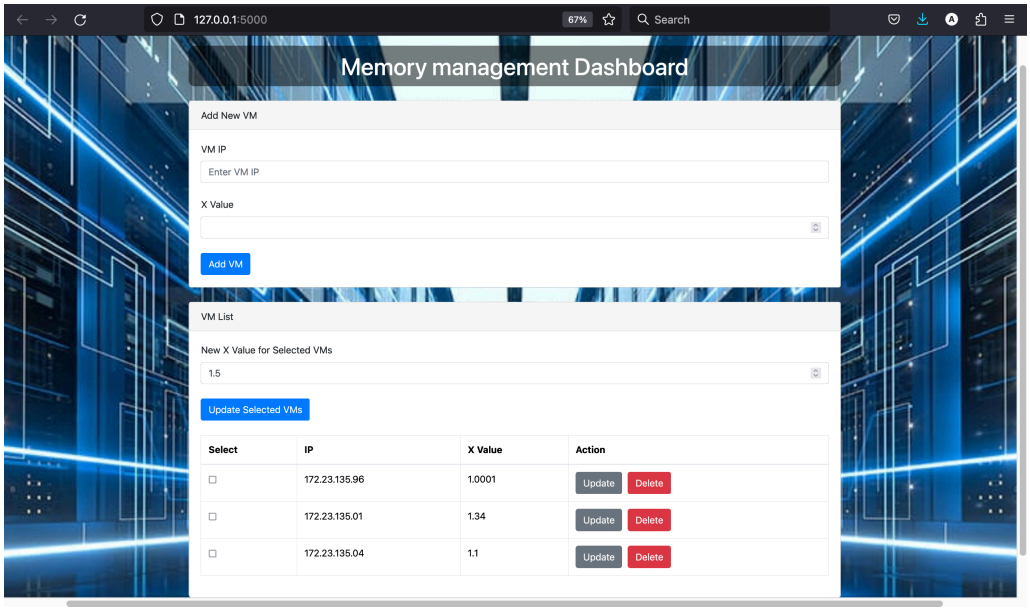
گرافانا یک ابزار متن باز و قدرتمند برای نمایش داده‌ها و ایجاد داشبوردهای پایش است. این ابزار به کمک افزونه‌ها و ابزارهای گرافیکی، به کاربران امکان می‌دهد تا داده‌های مختلف را به صورت گراف‌ها، نمودارها، و جدول‌های دینامیک نمایش دهند. با استفاده از گرافانا، می‌توانید داشبوردهای مختلفی را ایجاد کنید که شامل معیارها و متریک‌های مختلفی از سامانه باشند، از جمله مصرف حافظه، بار واحد پردازش مرکزی، ترافیک شبکه، و غیره. همچنین گرافانا قابلیت اتصال به پایگاه‌های داده‌های مختلف را دارد، از جمله پرومتئوس، InfluxDB، Elasticsearch و ... به منظور جمع‌آوری و نمایش داده‌های مختلف. از این رو، گرافانا به عنوان یکی از ابزارهای اصلی برای پایش و مدیریت سامانه‌ها و برنامه‌ها مورد استفاده قرار می‌گیرد.

ارتباط و رابط کاربری:

پایتون و Flask به عنوان ابزارهای اصلی برای توسعه رابط‌های برنامه‌نویسی (API) برای ارتباط با عامل استفاده می‌شوند. پایتون به عنوان یک زبان برنامه‌نویسی پرکاربرد و قدرتمند، و Flask به عنوان یک چارچوب سبک وزن برای توسعه وب‌سرویس‌ها، این امکان را فراهم می‌کنند که برنامه‌نویسان بتوانند با سرعت و کارایی بالا رابط‌هایی برای ارتباط با عامل ایجاد کنند.

SQLite به عنوان یک پایگاه داده سبک و قابل حمل برای ذخیره‌سازی تنظیمات و داده‌های مورد نیاز برای کار با سرور استفاده می‌شود. از آنجایی که SQLite به عنوان یک پایگاه داده توکار (embedded) عمل می‌کند، نیاز به سرور مرکزی ندارد و برای پروژه‌هایی که به یک پایگاه داده سبک و ساده نیاز دارند، مناسب است.

User، به عنوان مدیر سامانه [۷]، از طریق داشبوردهای سرور و گرافانا قادر است داده‌ها را مشاهده و تنظیمات مربوط به سامانه را مدیریت کند. این داشبوردها به کمک گرافیک‌ها و نمودارها، به مدیر سامانه کمک می‌کنند تا عملکرد و وضعیت سامانه را به‌طور کامل مشاهده و کنترل کند.



شکل ۳-۳: داشبورد مدیریت ماشین های مجازی

[7] System Administrator

۳-۴ خلاصه

این فصل به طراحی و پیاده‌سازی سامانه "تخصیص حافظه" پرداختیم. در این فصل، ابتدا به توضیح عامل پرداختیم که مسئول اجرای بارهای کاری مختلف بر روی ماشین‌های مجازی است. سپس به توضیح سرور پرداختیم که نقش مرکزی در جمع‌آوری، ذخیره، و نمایش داده‌ها دارد. در انتها، به پیاده‌سازی سامانه و نمودار آن پرداختیم که شامل دو بخش اصلی عامل و سرور است.

فصل چهارم

بررسی و ارزیابی سامانه

پس از بررسی جزئیات فنی، در این فصل به دریافت خروجی از سامانه و بررسی و ارزیابی نتایج سامانه پیاده‌سازی‌شده در این پروژه می‌پردازیم. بدون شک این مرحله اهمیت بسیار زیادی دارد و به ما این امکان را می‌دهد که عملکرد سامانه پیاده‌سازی‌شده را نظاره‌گر باشیم و اطمینان حاصل کنیم که این سامانه وظایف تعریف‌شده را به خوبی اجرا می‌کند.

۴-۱ دریافت خروجی از سامانه

آزمایش‌ها بر روی یک سرور فیزیکی HP ProLiant G8 با سیستم عامل Ubuntu انجام شدند. سه ماشین مجازی با استفاده از KVM اجرا شدند که هر یک از این ماشین‌ها به صورت مستقل عمل کرده و بارهای کاری مختلفی از نوع data analytics را که مشابه با ماشین‌های MapReduce در خوشه[1]های بزرگ هستند، اجرا می‌کردند. این ماشین‌های مجازی برای شبیه‌سازی بارهای واقعی سامانه‌های تولیدی طراحی شدند تا مدل تخصیص حافظه در شرایط نزدیک به واقعیت مورد آزمایش قرار گیرد. با این هدف، بارهای کاری انتخاب شده شامل پردازش داده‌های حجیم و عملیات سنگین بودند که به طور مداوم نیاز به مصرف حافظه و منابع محاسباتی داشتند.

عامل که با زبان گولنگ توسعه داده شده بود، به صورت یک فایل باینری کامپایل و بر روی ماشین‌های مجازی نصب شد. گولنگ به دلیل کارایی بالا و قابلیت‌های مدیریت هم‌زمانی انتخاب شد تا بتواند به خوبی نیازهای پردازشی و جمع‌آوری داده‌ها را در محیط‌های مجازی‌سازی تامین کند. برای مدیریت اجرای صحیح و به‌موقع سرویس‌ها، عامل با استفاده از Systemd راه‌اندازی و نظارت می‌شد. Systemd وظیفه داشت که اطمینان حاصل کند سرویس‌ها به درستی در حال اجرا هستند و در صورت نیاز مجددا راه‌اندازی شوند.

عامل با استفاده از Libvirt اطلاعات مصرف حافظه و منابع را از ماشین‌های مجازی جمع‌آوری کرد. داده‌های مصرف حافظه در بازه‌های زمانی ۱۵ ثانیه‌ای جمع‌آوری و هر سه دقیقه یک‌بار میانگین‌گیری شدند تا برای پیش‌بینی مصرف حافظه آینده استفاده شوند. این داده‌ها سپس به فرمتی قابل خواندن توسط پرومتئوس تبدیل شده و از طریق Web API به سرور ارسال شدند.پرومتئوس که بر روی یک لپ‌تاپ شخصی به عنوان سرور و سامانه مدیریت مرکزی نصب شده بود، این داده‌ها را دریافت و ذخیره کرد. این روش جمع‌آوری و ذخیره‌سازی داده‌ها کمک کرد تا اطلاعات دقیقی از وضعیت مصرف حافظه ماشین‌های مجازی در اختیار باشد.

[1] Cluster

در هر ساعت، مقدار مؤلفه آستانه تنظیم و داده‌ها شامل تعداد تناقض‌ها از پیش‌بینی مصرف حافظه (Violations) و مقدار حافظه برداشت شده (Harvesting Amount) ثبت شدند. این اطلاعات از طریق داشبوردهای گرافانا که به پرومتئوس متصل بود، به صورت گرافیکی نمایش داده شدند تا مدیر سامانه بتواند به راحتی تحلیل‌های لازم را انجام دهد. در پایان آزمایش‌ها، شش مقدار مختلف برای مؤلفه آستانه به همراه تعداد تناقض‌ها و مقدار حافظه برداشت شده برای هر مقدار در یک ساعت ارزیابی ثبت شدند. این داده‌ها به مدیر سامانه کمک کردند تا بهینه‌ترین مقدار برای مؤلفه آستانه را برای این نوع ماشین‌ها و بارهای کاری انتخاب کند. نتایج به‌دست‌آمده از این آزمایش‌ها به عنوان مبنایی برای تنظیمات دقیق‌تر و بهینه‌سازی بیشتر مدل در آینده مورد استفاده قرار گرفتند، که نشان‌دهنده‌ی نقاط قوت و ضعف مدل تخصیص حافظه در مدیریت منابع حافظه در محیط‌های مجازی‌سازی بود.

۴-۲ معیارهای ارزیابی

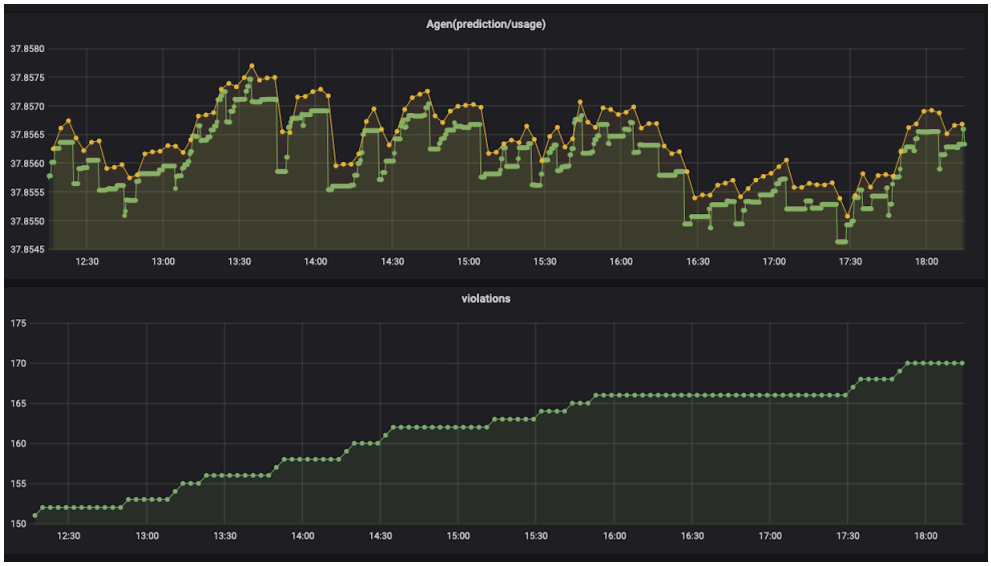
در این بخش، معیارهای ارزیابی برای سنجش عملکرد مدل تخصیص حافظه و تعیین بهترین مؤلفه آستانه توضیح داده می‌شود. این معیارها شامل تعداد تناقض‌ها از پیش‌بینی مصرف حافظه، مساحت زیر نمودار برداشت، مساحت زیر نمودار مصرف حافظه، مقدار کل حافظه تخصیص داده شده به ماشین‌های مجازی، و تعداد ماشین‌های مجازی می‌باشند.

تعداد تناقض‌ها(Violations):

تعداد تناقض‌ها به مواردی اشاره دارد که مصرف حافظه واقعی از پیش‌بینی‌های انجام‌شده توسط مدل بیشتر می‌شود. این مؤلفه به عنوان یک معیار مهم برای ارزیابی دقت پیش‌بینی مدل استفاده می‌شود. هر چه تعداد تناقض‌ها کمتر باشد، نشان‌دهنده این است که مدل پیش‌بینی بهتری از مصرف حافظه دارد و توانسته است به درستی نیازهای حافظه را تخمین بزند.

$$\text{Violation Count} = \sum_{i=1}^{n} \mathbb{I}(\text{Actual Memory Usage}_i > \text{Predicted Memory Usage}_i)$$

که در آن $\mathbb{I}$ یک تابع شاخص است که مقدار ۱ را در صورتی که شرط داخل آن صحیح باشد، برمی‌گرداند و n تعداد کل بازه‌های زمانی است.



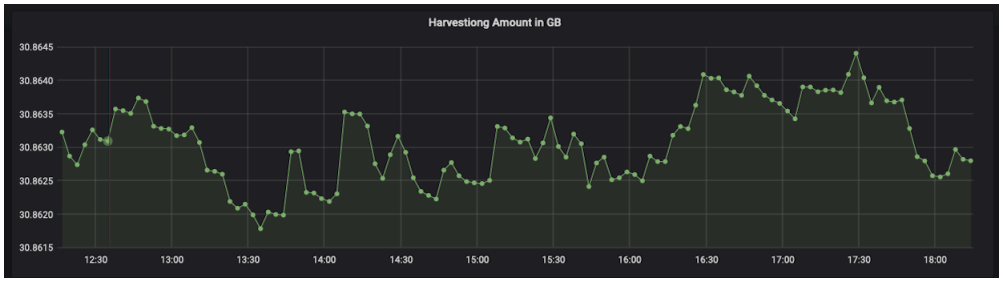
شکل ۴-۱: داشبورد گرافانا مربوط به Violations

مساحت زیر نمودار برداشت:

مساحت زیر نمودار برداشت نشان‌دهنده مقدار حافظه‌ای است که سامانه توانسته است از ماشین‌های مجازی برداشت کند. این مقدار هر چه بیشتر باشد، بهتر است زیرا نشان می‌دهد که سامانه به خوبی توانسته از منابع حافظه بهره‌برداری کند و حافظه‌های بلااستفاده را به دیگر بخش‌ها اختصاص دهد.

$$\text{Harvesting Area} = \int_0^T \text{Harvested Memory}(t)\, dt$$

که در آن $\text{Harvested Memory}(t)$ مقدار حافظه برداشت شده در زمان t و T کل مدت زمان بررسی است.



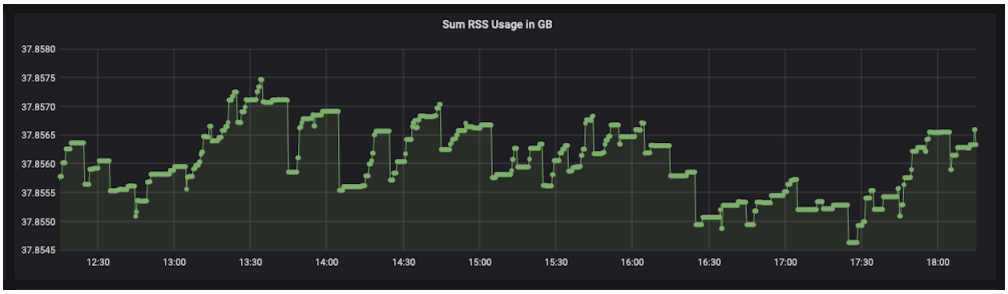
شکل ۴-۲: داشبورد گرافانا مربوط به برداشت

مساحت زیر نمودار مصرف حافظه:

این مؤلفه نشان‌دهنده مقدار کل مصرف حافظه توسط سامانه در طول دوره بررسی است. مساحت زیر نمودار مصرف حافظه بیانگر کارایی و بهره‌وری کلی سامانه در استفاده از منابع حافظه است.

$$\text{Memory Consumption Area} = \int_0^T \text{Memory Usage}(t)\, dt$$

که در آن $\text{Memory Usage}(t)$ مقدار حافظه مصرفی در زمان t و T کل مدت زمان بررسی است.



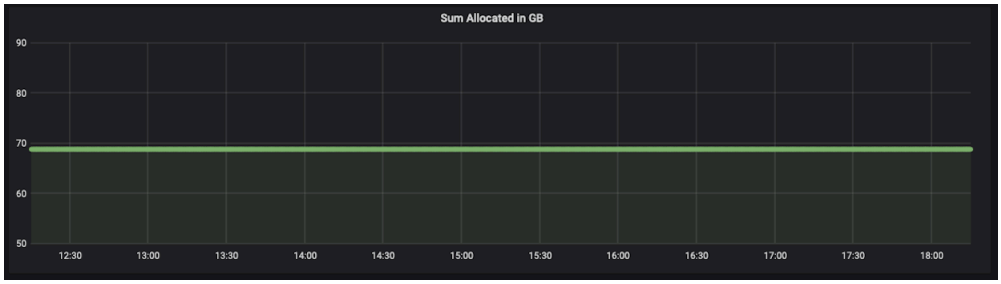
شکل ۴-۳: داشبورد گرافانا مربوط به Memory Usage

مقدار کل حافظه تخصیص داده شده:

این مؤلفه مقدار کل حافظه‌ای را که به ماشین‌های مجازی اختصاص داده شده است، نشان می‌دهد. این مقدار باید بهینه باشد تا از یک سو منابع هدر نرود و از سوی دیگر نیازهای بارهای کاری مختلف به درستی پوشش داده شود.

$$\text{Total Allocated Memory} = \sum_{i=1}^{m} \text{Allocated Memory}_{\text{ماشین مجازی}_i}$$

که در آن m تعداد کل ماشین‌های مجازی و $\text{Allocated Memory}_{\text{ماشین مجازی}_i}$ مقدار حافظه تخصیص داده شده به ماشین مجازی ماشین مجازی_i است.

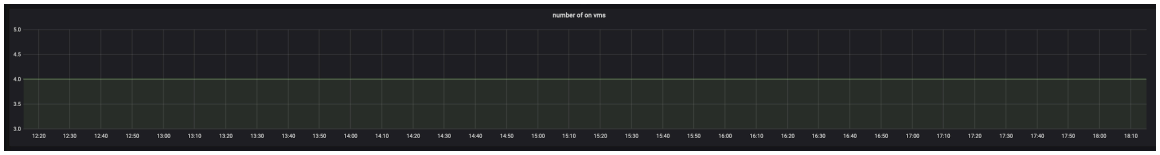


شکل ۴-۴: داشبورد گرافانا مربوط به مقدار کل حافظه تخصیص داده شده

تعداد ماشین‌های مجازی:

تعداد ماشین‌های مجازی نشان‌دهنده تعداد محیط‌های مستقل و جداگانه‌ای است که در سامانه وجود دارند. این مؤلفه نشان‌دهنده مقیاس‌پذیری و توانایی سامانه در مدیریت چندین ماشین مجازی به طور همزمان است.

$$\text{Number of Virtual Machines} = m$$



شکل ۴-۵: داشبورد گرافانا مربوط به تعداد ماشین‌های مجازی

ارزیابی بهترین مؤلفه آستانه:

برای تعیین بهترین مقدار برای مؤلفه آستانه، می‌توان از ترکیبی از معیارهای فوق استفاده کرد. یک روش ساده برای ارزیابی این مقادیر استفاده از یک تابع ارزیابی وزنی است که وزن‌های مختلفی به هر یک از این معیارها اختصاص می‌دهد. به عنوان مثال:

$$\begin{aligned}\text{Evaluation Score} = w_1 \times \left(1 - \frac{\text{Violation Count}}{\text{Max Violations}}\right) \\ + w_2 \times \frac{\text{Harvesting Area}}{\text{Max Harvesting Area}} \\ + w_3 \times \frac{\text{Memory Consumption Area}}{\text{Max Memory Consumption Area}} \\ + w_4 \times \frac{\text{Total Allocated Memory}}{\text{Max Allocated Memory}} \\ + w_5 \times \frac{\text{Number of Virtual Machines}}{\text{Max Virtual Machines}}\end{aligned}$$

که در آن w_1, w_2, w_3, w_4, w_5 وزن‌های اختصاص داده شده به هر معیار هستند و با استفاده از این تابع ارزیابی، می‌توان بهترین مقدار برای مؤلفه آستانه را پیدا کرد که بیشترین امتیاز را در میان مقادیر مختلف به دست آورد.

۴-۳ ارزیابی‌های مورد استفاده

در این بخش به بررسی ارزیابی[۲]های مختلفی که برای ارزیابی و اطمینان از عملکرد صحیح سامانه "Memory Harvesting" مورد استفاده قرار گرفته‌اند، پرداخته می‌شود. ارزیابی‌های مورد استفاده شامل موارد زیر هستند:

۱. **ارزیابی واحد (Unit Testing)**: ابتدا ارزیابی واحد (Unit Testing) برای بررسی عملکرد صحیح هر جزء یا ماژول به صورت جداگانه انجام شد. این ارزیابی‌ها برای اطمینان از عملکرد صحیح بخش‌های مختلف مانند Agent، Predictor و Exporter مورد استفاده قرار گرفتند.

۲. **ارزیابی یکپارچه‌سازی (Integration Testing)**: پس از اطمینان از عملکرد صحیح هر جزء، ارزیابی یکپارچه‌سازی (Integration Testing) صورت گرفت تا از عملکرد صحیح سامانه زمانی که تمام ماژول‌ها و اجزا با هم ترکیب شده و همکاری می‌کنند، اطمینان حاصل شود. این ارزیابی شامل بررسی ارتباط بین Agent و سرور و اطمینان از ارسال و دریافت صحیح داده‌ها از طریق Web API بود.

۳. **ارزیابی سامانه (System Testing)**: سپس ارزیابی سامانه (System Testing) استفاده شد تا بررسی شود که سامانه به عنوان یک واحد کامل به درستی کار می‌کند و با نیازمندی‌های اولیه مطابقت دارد.

۴. **ارزیابی عملکرد (Performance Testing)**: در مرحله بعد، بارکاری‌های مختلف روی سامانه اعمال شده و ارزیابی عملکرد (Performance Testing) انجام گرفت. این ارزیابی‌ها برای ارزیابی کارایی سامانه تحت بارهای کاری مختلف انجام شدند و سپس نتایج این ارزیابی‌ها یادداشت و مورد بررسی قرار گرفتند.

۴-۴ نتایج ارزیابی سامانه

نتایج ارزیابی الگوریتم‌ها بر اساس معیارهای گفته شده در جدول ۴-۱ آورده شده‌است. این نتایج پس از ارزیابی سامانه با مقادیر مختلف آستانه به دست آمده‌است. این جدول شامل مقادیر مؤلفه آستانه و معیارهای ارزیابی مختلفی است که برای هر مقدار آستانه، مقدار مربوط به تعداد تخلفات، مساحت

[۲] Test

زیر نمودار برداشت، مساحت زیر نمودار مصرف حافظه، مقدار کل حافظه تخصیص داده شده و تعداد ماشین‌های مجازی را نشان می‌دهد. این مقادیر به صورت مستقیم از ارزیابی عملکرد مدل برای هر مقدار آستانه بدست آمده‌اند.

وزن‌های اختصاص داده شده به هر معیار نیز به این صورت باشند:

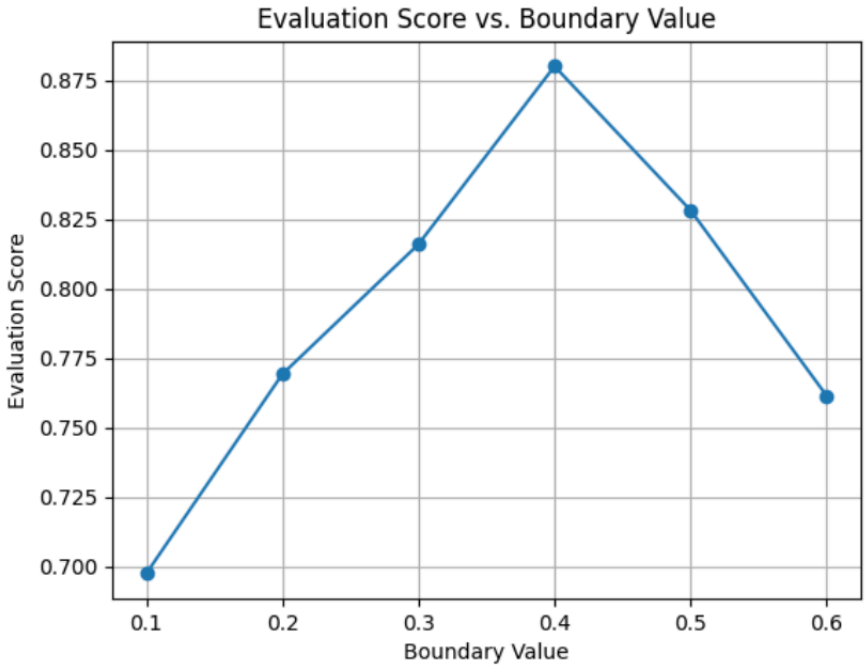
$$w_1 = 0.3, \quad w_2 = 0.3, \quad w_3 = 0.2, \quad w_4 = 0.2, \quad w_5 = 0.1$$

که به دلیل اهمیت تعداد تناقض‌ها و برداشت به آنها وزن بالاتری داده شده که این وزن‌ها با توجه به نظر مدیر سامانه و ملاحظاتش می‌تواند تغییر کند.

جدول ۴-۱: جدول ارزیابی با مقادیر مختلف آستانه

آستانه	تعداد تناقض‌ها	برداشت(GB)	مصرف حافظه(GB)	حافظه تخصیص داده شده(GB)	تعداد ماشین‌ها	تعداد ماشین‌هایی که میتوان بالا آورد
۱.۱	۱۶	۱۹.۶۲	۲۴.۱۲	۴۸	۳	۱
۱.۲	۱۳	۱۹.۴۵	۲۴.۲۱	۴۸	۳	۱
۱.۳	۱۱	۱۸.۲۳	۲۴.۷۵	۴۸	۳	۱
۱.۴	۹	۱۸.۷۴	۲۴.۲۰	۴۸	۳	۱
۱.۵	۷	۱۸.۴۱	۲۴.۴۹	۴۸	۳	۱
۱.۶	۶	۱۸.۳۲	۲۴.۵۷	۴۸	۳	۱

نمودار Evaluation Score نیز به صورت یک نمودار خطی نمایش داده شده است که محور x آن مقادیر مختلف آستانه و محور y آن امتیاز ارزیابی مدل به ازای هر آستانه است. این نمودار کمک می‌کند تا بهترین مقدار آستانه برای مدل را انتخاب کنیم، به طوری که امتیاز ارزیابی بیشینه باشد و عملکرد مدل بهینه شود.



شکل ۴-۶: نمودار مربوط به محاسبه Evaluation score برای آستانه های مختلف

۴-۵ تحلیل نتایج و نتیجه‌گیری

با توجه به جدول ۴-۲ و نتایج ارزیابی، مقدار آستانه=۱.۴ در آزمایش‌ها بهترین عملکرد را از خود نشان داده است، می‌توان این نتیجه را به عنوان پیشنهادی برای تنظیم بهینه مؤلفه آستانه در سامانه‌های مشابه در نظر گرفت. این انتخاب می‌تواند به مدیران سامانه کمک کند تا نیازهای مصرف حافظه را با دقت بیشتری پیش‌بینی و مدیریت کنند، که در نهایت به بهبود عملکرد و بهره‌وری سامانه‌ها منجر خواهد شد. این به معنای افزایش کیفیت خدمات ارائه شده توسط سامانه‌ها و کاهش هزینه‌های ناشی از نقصان عملکرد و نیاز به منابع اضافی می‌باشد.

با توجه به این نتایج، می‌توان نتیجه گرفت که الگوریتم تخصیص حافظه با تنظیم بهینه مؤلفه آستانه، می‌تواند به خوبی نیازهای مصرف حافظه در محیط‌های مجازی را مدیریت کند و باعث بهره‌وری بالاتری در استفاده از منابع حافظه شود. این نتایج می‌تواند به توسعه‌دهندگان و مدیران سامانه کمک کند تا در مدیریت حافظه و منابع سامانه‌های مجازی، تصمیمات بهتری بگیرند و عملکرد سامانه‌ها را بهبود بخشند.

۴-۶ خلاصه

در این فصل به بررسی و ارزیابی سامانه، پس از انجام آزمایش‌ها بر روی سامانه پیاده‌سازی‌شده پرداختیم. ما به دنبال دریافت خروجی از سامانه و ارزیابی عملکرد آن بودیم و با استفاده از معیارهایی مانند تعداد تناقض‌ها از پیش‌بینی مصرف حافظه و مساحت زیر نمودارهای برداشت و مصرف حافظه، عملکرد الگوریتم تخصیص حافظه را مورد بررسی قرار دادیم. نتایج نشان داد که مقدار آستانه=۱.۴ بهترین عملکرد را از خود نشان داده است. این پیشنهاد می‌تواند به مدیران سامانه کمک کند تا بهترین مقدار برای مؤلفه آستانه را انتخاب کنند و در نهایت بهبود عملکرد و بهره‌وری سامانه‌ها را به دست آورند.

فصل پنجم

جمع‌بندی، نتیجه‌گیری و پیشنهادها برای کارهای آتی

۵-۱ جمع‌بندی و نتیجه‌گیری

در این پروژه، یک سامانه برداشت حافظه برای ماشین‌های مجازی در محیط سیستم عامل لینوکس طراحی و پیاده‌سازی شد. هدف اصلی این سامانه، بهبود بهره‌وری منابع سخت‌افزاری سرورها از طریق بهینه‌سازی تخصیص حافظه به ماشین‌های مجازی و کاهش هدررفت منابع بود. برای دستیابی به این هدف، ابتدا مطالعه‌ای جامع بر روی مصرف منابع سرورهای ابری و ماشین‌های مجازی انجام شد تا اهمیت موضوع و نیاز به یک راهکار مناسب برای مدیریت حافظه مشخص شود.

سپس، با استفاده از زبان برنامه‌نویسی گولنگ و کتابخانه Libvirt، عاملی طراحی شد که اطلاعات مرتبط با ماشین‌های مجازی تحت نظارت مجازی ساز KVM را جمع‌آوری کند. این اطلاعات شامل تعداد ماشین‌های روشن، مجموع حافظه تخصیص داده شده به آنها، و مصرف لحظه‌ای حافظه (مؤلفه RSS) بود. عامل مذکور این اطلاعات را هر سه دقیقه به‌روزرسانی می‌کرد و به سرور ارسال می‌نمود. سرور این سامانه نیز با استفاده از زبان پایتون و چارچوب Flask پیاده‌سازی شد و داده‌های جمع‌آوری شده را در دیتابیس SQLite ذخیره می‌کرد.

با ارسال داده‌ها به پرومتئوس و نمایش آن‌ها در گرافانا، امکان پایش دقیق و لحظه‌ای مصرف حافظه ماشین‌های مجازی فراهم شد. سامانه همچنین مؤلفه برداشت را که تفاوت بین حافظه تخصیص داده شده و پیش‌بینی شده است، محاسبه و به مدیر سامانه اعلام می‌کرد. علاوه بر این، تعداد مواردی که مصرف حافظه بیش از پیش‌بینی شده باشد (تناقض) نیز ثبت و گزارش می‌شد.

در نهایت، هدف نهایی پروژه که تعیین مؤلفه آستانه مناسب برای گروهی از ماشین‌های مجازی در یک سرور بود، با اجرای سه ماشین مجازی با بارکاری واقعی data analytics از مجموعه Cloud Suite و ارزیابی با شش مؤلفه آستانه متفاوت، تحقق یافت. نتایج حاصل از این ارزیابی‌ها نشان داد که استفاده از روش‌های پیش‌بینی مصرف حافظه می‌تواند به طور قابل توجهی به کاهش مجموع تناقض‌ها و بهبود بهره‌وری منابع منجر شود.

۵-۲ پیشنهادها برای کارهای آتی

این پروژه قابلیت تکمیل و توسعه از ابعاد مختلفی را دارا است. در این بخش به برخی از قسمت‌هایی که می‌توان در نسخه‌های بعدی، آن‌ها را به این پروژه اضافه کرد و پروژه را بهبود بخشید، اشاره می‌کنیم.

۱. **بهبود الگوریتم‌های پیش‌بینی مصرف حافظه:** استفاده از الگوریتم‌های پیچیده‌تر و پیشرفته‌تر

هوش مصنوعی[1] و یادگیری ماشین[2] مانند شبکه‌های عصبی عمیق، جنگل‌های تصادفی، و الگوریتم‌های تقویت‌کننده می‌تواند دقت پیش‌بینی مصرف حافظه را افزایش دهد. الگوریتم‌های مبتنی بر داده‌های تاریخی و تحلیل الگوهای مصرف می‌توانند رفتار آینده ماشین‌های مجازی را با دقت بیشتری پیش‌بینی کنند و در نتیجه تخصیص منابع بهینه‌تر انجام شود.

۲. **توسعه قابلیت‌های پایش و تحلیل:** افزودن قابلیت‌های جدید به گرافانا و پرومتئوس برای تحلیل دقیق‌تر داده‌های جمع‌آوری شده می‌تواند به شناسایی الگوها و روندهای مصرف حافظه کمک کند. توسعه داشبوردهای تعاملی و گزارش‌های تفصیلی برای تحلیل و مقایسه مصرف حافظه در زمان‌های مختلف و تحت شرایط مختلف می‌تواند به مدیران سامانه در تصمیم‌گیری بهتر کمک کند.

۳. **پیاده‌سازی سامانه در مقیاس بزرگتر:** آزمایش و ارزیابی سامانه در محیط‌های با تعداد بیشتری از ماشین‌های مجازی و سرورها می‌تواند به بررسی کارایی و مقیاس‌پذیری سامانه کمک کند. اجرای ارزیابی‌های بارگذاری و شبیه‌سازی شرایط واقعی محیط‌های ابری با حجم بالای داده و تعداد زیادی ماشین مجازی می‌تواند نقاط ضعف احتمالی سامانه را شناسایی و به بهبود آن‌ها کمک کند.

۴. **ارتقاء امنیت سامانه:** با توجه به اهمیت امنیت در محیط‌های مجازی، افزودن ویژگی‌های امنیتی مانند احراز هویت دو مرحله‌ای، رمزگذاری داده‌ها، و دسترسی مبتنی بر نقش‌ها می‌تواند امنیت سامانه را بهبود بخشد. پیاده‌سازی مکانیزم‌های پایش و لاگ‌برداری امنیتی برای تشخیص و پاسخ به تهدیدات و حملات احتمالی نیز ضروری است.

۵. **ادغام با سامانه‌های مدیریت منابع دیگر:** ادغام سامانه برداشت حافظه با دیگر سامانه‌های مدیریت منابع و زیرساخت‌های ابری مانند Kubernetes، OpenStack و VMware می‌تواند به ایجاد یک راه‌حل جامع و یکپارچه برای مدیریت منابع سخت‌افزاری کمک کند. استفاده از APIهای استاندارد و پروتکل‌های ارتباطی برای تسهیل در ادغام سامانه‌ها و تبادل داده‌ها می‌تواند به افزایش کارایی و بهره‌وری کمک کند.

۶. **ارائه رابط کاربری کاربرپسندتر:** طراحی و توسعه رابط کاربری گرافیکی بهتر و کاربرپسندتر

[1] Artificial Inteligense
[2] Machine Learning

برای مدیران سامانه می‌تواند استفاده از سامانه را آسان‌تر و کارآمدتر کند. افزودن ویژگی‌های مانند نمودارهای تعاملی، فیلترها و ابزارهای جستجو، و داشبوردهای سفارشی‌سازی‌شده می‌تواند تجربه کاربری را بهبود بخشد.

کتاب‌نامه

[1] Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., and Bianchini, R. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP'17, pages 153–167, Shanghai, China, 2017.

[2] Donovan, Alan A. and Kernighan, Brian W. *The Go Programming Language*. Addison-Wesley Professional, 2015.

[3] Javadi, Seyyed Ahmad, Suresh, Amoghavarsha, Wajahat, Muhammad, and Gandhi, Anshul. Scavenger: A black-box batch workload resource manager for improving utilization in cloud environments. In *Proceedings of the ACM symposium on cloud computing*, pages 381–393. ACM, 2019.

[4] Liu, H. A measurement study of server utilization in public clouds. In *Proceedings of the 9th IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 435–442, 2011.

[5] Loizzo, Joseph. *Mastering libvirt: Manage your virtualized environments efficiently*. Packt Publishing Ltd, 2017.

[6] Loshin, Peter. Hypervisors: A brief history. *IEEE Software*, 31(2):24–26, 2014.

[7] Parsa, EPFL. Cloudsuite - data analytics, n.d.

[8] Raj, Rahul and Kant, Krishna. Virtualization techniques for managing and utilizing physical resources. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 1–6. ACM, 2013.

[9] Red Hat. Kernel-based virtual machine (kvm), n.d.

[10] Turnbull, James. *The Docker Book: Containerization is the new virtualization*. Independently Published, 2018.

[11] Veen, Peter et al. Predicting memory usage in virtualized environments using machine learning. *Journal of Systems and Software*, 123:123–135, 2016.

Amirkabir University of Technology
(Tehran Polytechnic)

Department of Computer Engineering

Bachelor Of Science Thesis

Design and Implementation of Memory Harvesting System for Linux Operating System

By

Ali Rezaee

Supervisor

Dr. Seyyed Ahmad Javadi

June 2024