پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

افزودن پشتیبانی از محاسبات روی کارت‌های گرافیکی به کلاستر محاسباتی دانشکدگان علوم

نگارنده

آرمین شوشتری

استاد راهنما:دکتر محمد گنج‌تابش

پایان‌نامه برای دریافت درجه کارشناسی
در رشته علوم کامپیوتر

زمستان ۱۴۰۰

چکیده

سیستم پردازش موازی دانشکدگان علوم یک مکان مناسب برای رفع نیاز محاسباتی دانشجویان است. برخی از تحقیقات دانشجویی نیاز به استفاده از پردازش کامپیوتری دارند اما استفاده از سیستم های مرسوم و شخصی برای این کار علاوه بر بهینه نبودن، هزینه‌بر نیز هستند. به همین منظور سیستم پردازش موازی دانشکدگان علوم راه اندازی شده است تا دانشجویان از سخت افزار های تخصصی در راستای کارهای محاسباتی خود استفاده کنند. در این پروژه تلاش شده است تا مشکلات سیستم پردازش موازی را با استفاده از تکنولوژی های جدید رفع کرده و استفاده از آن را برای دانشجویان راحت تر کند. یکی از چالش های این کار اضافه کردن پشتیبانی استفاده از سیستم پردازش گرافیکی به سیستم جدید بود.

سپاسگزاری

سپاسگزاری

با تشکر از دکتر محمد گنج‌تابش، دکتر مجتبی مجتهدی، و دیگر مسئولین دانشکدگان علوم که فرصت بهبود ساختاری کلاستر پردازش اشتراکی دانشکدگان علوم را در اختیار ما قرار دادند.

پیشگفتار

در این پروژه سعی شده است که یک سیستم نرم افزاری جدید برای کلاستر محاسباتی دانشکدگان علوم طراحی و پیاده‌سازی شود. کلاستر پردازشی دانشکدگان علوم در حال حاضر از سیستم rocks-cluster استفاده میکند که مشکلات متعددی دارد.
این گزارش یک مرور مختصر بر کارهای انجام شده و تصمیم‌های گرفته شده و دلایل آنها است.

در طول مدتی که سیستم rocks-cluster بر روی کلاستر در حال کار بود، نگه دارندگان سیستم به مشکلاتی برخورد میکردند که به دلیل قدیمی بودن سیستم rocks-cluster و عدم پشتیبانی مناسب از آن در بین توسعه دهندگان سیستم حل کردن این مشکلات بسیار دشوار میشد.
در گزارش ابتدا به ساختار سیستم قدیمی و مشکلات آن میپردازیم و سپس به تصمیماتی که برای ساخت سیستم جدید گرفته‌ایم نگاه مختصری میکنیم. در نهایت نیز راه های اضافه کردن سیستم پردازش گرافیکی به سیستم جدید را واکاوی میکنیم.

فهرست مطالب

فصل ۱

مفاهیم مقدماتی

این پروژه در رابطه با کلاستر های پردازشی میباشد به همین منظور تعدادی از مفاهیم استفاده شده در ادامه این گزارش در این بخش تعریف میشوند.

تعریف ۱.۱. کلاستر پردازش با کارایی بالا *(High Performance Computing)*
به کلاستر هایی گفته میشود که با کمک همدیگر توانایی انجام مقادیر زیادی از محاسبات را دارند. کلاستر فعلی دانشکدگان علوم تا حدی در این دسته قرار میگیرد.

یک کلاستر شامل تعدادی نود است که هر نود یک کامپیوتر مستقل از دیگریست. برای مدیریت یک کلاستر و نگه‌داری از آن نیاز به یک نرم افزار مخصوص است که به آن نرم‌افزار مدیریت کلاستر گفته میشود.

تعریف ۲.۱. نرم افزار مدیریت کلاستر *(Cluster Managemant Software)*
نرم افزار هایی که کارهای مدیریت یک کلاستر مانند (مدیریت منابع، توافق بین نودها، برنامه‌ریزی وظایف و ...) را انجام میدهند. تعدادی از نرم افزار هایی که برای این کار استفاده می شوند: Rocks-Cluster-Distribution Apache-Mesos Nomad Kubernetes

یکی از مهمترین کارهایی که یک نرم افزار مدیریت کلاستر باید انجام بدهد برنامه ریزی وظایف موجود با توجه به منابع آماده است.

تعریف ۳.۱. برنامه‌ریزی وظایف *(Task Scheduling)*
به فرآیند نگاشت بین منابع و کار ها و نحوه انجام این کار برنامه‌ریزی وظایف گفته میشود.

در ادامه تعدادی از مفاهیمی که برای درک ویژگی های یک کلاستر نیاز است تعریف شده است.

تعریف ۴.۱. جداسازی *(isolation)*
جداسازی به فرآیند مشخص کردن مرز برای استفاده از منابع برای تسک های اجرایی گفته میشود. این منابع از هر نوعی میتوانند باشند.

تعریف ۵.۱. مقیاس پذیری *(scalability)*
به افزایش منابع سخت افزاری سیستم های کامپیوتری گفته میشود. مقیاس پذیری به دو صورت انجام میگیرد: افقی و عمودی. مقیاس پذیری افقی به افزایش منابع کلی با اضافه کردن واحد های جدید گفته میشود در حالی که مقیاس پذیری عمودی به افزایش منابع واحد های فعلی اطلاق میشود.

تعریف ۶.۱. نگهداری *(maintenance)*
به نگهداری یک سیستم نرم افزاری و یا سخت افزاری در طول زمان گفته میشود. در هر سیستمی با گذر زمان مشکلاتی از قبیل باگ های نرم افزاری و یا سخت افزاری به وجود میاید. به مجموعه حل کردن این باگ ها و کار هایی که برای نگه داری انجام میشود مانند ارتقا و مقیاس دادن اصطلاحا هزینه نگهداری اطلاق میشود.

تعریف ۷.۱. مجازی سازی *(virtualization)*
مجازی سازی به فرایندی گفته میشود که یک محیط مجازی برای یک پردازش و یا کاربر ساخته میشود که با محیط فیزیکی و واقعی فرق دارد. یکی از راه حل های isolation استفاده از virtualization است زیرا به کلی محیط جدید و ایزوله ای از محیط های دیگر ساخته میشود.

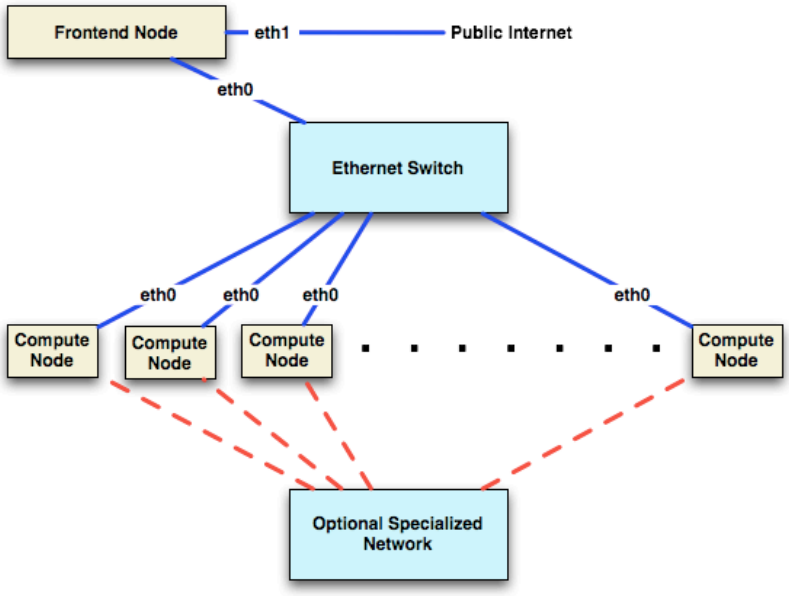
فصل ۲

سیستم قدیمی

کلاستر محاسباتی دانشکدگان علوم در سال ۱۳۹۵ تاسیس شد و از آن زمان به دانشجویان مختلفی توان محاسباتی ارائه میکند. در طول این زمان نرم افزار مدیریت کلاستر rocks-cluster-distribution بر روی آن اجرا میشد.
این نرم افزار بر روی مدیریت آسان کلاستر ها و مقیاس پذیری افقی کلاستر تمرکز میکند. برای نصب ابزار های مختلف بر روی این کلاستر میتوان از roll های از پیش آماده شده استفاده کرد. همچنین این پروژه در حال حاضر از سیستم عامل cent-os که یک توزیع لینوکس محبوب و رایج در دنیا است استفاده میکند.

۱.۲ ساختار

شمای کلی سیستم rocks-cluster در تصویر زیر خلاصه شده است:



تصویر ۱: شمای کلی ساختار rocks-cluster

در تصویر ۱ یک کلاستر rocks نشان داده میشود که متشکل از یک front-node و تعدادی نود compute است.
نحوه نصب این کلاستر به این صورت است که اول با استفاده از چند roll-disk که شامل تعدادی از ابزار های ضروری هستند، یک front-node را بالا میاوریم و سپس پس از بوت شدن front بقیه نود ها با استفاده از فناوری pxe بوت میشوند. برای نصب یک ویژگی جدید روی سیستم عامل باید از roll های از پیش آماده شده استفاده کرد. در کنار rocks-cluster یک roll مخصوص وجود دارد که وظیفه آن نصب sge است که این نرم افزار برای scheduling تسک ها استفاده میشود.

۲.۲ نحوه استفاده از سیستم

در این سیستم یک نود مخصوص برای storage وجود دارد که کار آن نگهداری اطلاعات کاربران است و با استفاده از پروتکل nfs به بقیه سیستم ها وصل میشود. کاربران برای استفاده از این سیستم باید ابتدا از طریق یک وبسایت اقدام به ثبت نام کنند و سپس یک یوزر برای آن ها در کلاستر ساخته میشود. کاربران سپس باید از طریق پروتکل ssh به نود فرانت که به شبکه دانشگاه باز است وصل شوند و بعد از آن اقدام به ثبت جاب خود در سیستم sge کنند.
سیستم sge یک scheduler ابتدایی است که در ساده ترین شکل خود متشکل از چند

صف است. در حال حاضر در سیستم فعلی یک صف cpu برای تسک هایی که قصد استفاده از gpu را ندارند وجود دارد که در نود های بدون gpu اجرا میشود و یک صف gpu وجود دارد که در یک نود دارای gpu اجرا میشوند و میتوانند از gpu استفاده کنند. ثبت یک جاب به سادگی اجرای یک دستور qsub است که در فرانت نود اجرا میشود و در نهایت توسط یکی از compute ها اجرا میشود.

۳.۲ نقاط قوت سیستم فعلی

سیستم فعلی دارای نقاط قوتی است که در اینجا به آن ها میپردازیم.

۱.۳.۲ راحتی در نصب

با توجه به ساختار cluster rocks نصب نود جدید به راحتی امکان پذیر است و در مدت زمان کمی یک نود جدید به سرور اضافه میشود. خیلی از نیازهایی که برای یک کلاستر وجود دارد به صورت خودکار در cluster rocks وجود دارد و نیازی به کار اضافه تر نیست.

۲.۳.۲ مقیاس پذیری

به دلیل راحت بودن نصب نود جدید، به راحتی میتوان به صورت عمودی یا افقی سرور را scale کرد. تنها کافیست hardware را داخل سیستم قرار دهیم و بقیه کارها و تنظیمات به صورت خودکار انجام میشود.

۳.۳.۲ قابلیت all in one

این نرم افزار مجموعه تمام کار هایی که برای اجرای یک کلاستر نیاز است را انجام میدهد.

۴.۲ نقاط ضعف سیستم فعلی

سیستم فعلی همچنین دارای ضعف هایی هم هست که میتواند استفاده از آن را سخت کند.

۱.۴.۲ سطح دسترسی زیاد کاربران

در حال حاضر کاربران برای اجرای یک جاب از پروتکل ssh استفاده میکنند که عملا یک shell با یک user-account به آن ها میدهد. یعنی آن ها میتوانند به نود front وارد شوند و دستورات خود را اجرا کنند که همین موضوع باعث خسارت های جدی شده است. به عنوان مثال تعداد زیادی از کاربران به دلیل عدم اطلاع از نحوه اجرای جاب بجای اینکه جاب خود را در داخل sge اجرا کنند، بعد از وارد شدن به front-node شروع به اجرا جاب خود میکنند که چندین بار باعث مختل شدن کارآیی سیستم شده است. همینطور در یک اتفاق نادر یک کاربر یک فایل نسبتا حجیم را با ویرایشگر nano باز کرد که این کار باعث شد front-node از دسترسی خارج شود زیرا ویرایشگر nano کل فایل را در حافظه بارگذاری میکند و تمام ram نود بخاطر آن دستور پر شد.
به علاوه این دسترسی زیاد باعث مشکلات امنیتی میشود. به عنوان مثال یک کاربر میتواند لیست کاربران دیگر را داشته باشد و یا اینکه با استفاده از front-node به یک سیستم داخلی دانشگاه حمله ای انجام دهد و رد کار خود را پاک کند. به طور خلاصه دسترسی ssh به سرور بسیار بیشتر از دسترسی مورد نیاز کاربر است که به دلیل نحوه کار سیستم sge نمی توان این دسترسی را از کاربر گرفت.

۲.۴.۲ سخت بودن نصب نرم افزار های درخواستی

هر روزه تعداد زیادی از دانشجویان درخواست استفاده از سیستم کلاستر محاسباتی را دارند و این دانشجویان از یک نرم افزار و ورژن مربوط به کار خود استفاده میکنند. در عین حالی که در بخش قبل ذکر شد که کاربران دسترسی زیادی برای استفاده از سیستم دارند، همچنان دسترسی کافی برای نصب نرم افزار خود را ندارند و این کار باید توسط یک administrator که دسترسی کامل دارد انجام شود. به علاوه تعدادی از این نرم افزار ها با هم دیگر تداخل دارند و اصلا نمیتوان هر دو را داخل کلاستر داشت. نصب نرم افزار سخت ترین و وقت گیر ترین کاری است که system-admin ها با آن درگیر هستند.

۳.۴.۲ جریمه برای مصرف غیر مجاز منابع

با وجود اینکه در sge مفهوم slot وجود دارد ولی در حال حاضر هیچ محدودیتی برای اجرای تسک ها وجود ندارد. یعنی یک پردازه میتواند تا هر مقداری که بخواهد از cpu مصرف کند و یا اینکه هر مقداری از حافظه را اتخاذ کند. ولی برای جلوگیری از سو استفاده یک مکانیزم جریمه وجود دارد که در هر لحظه اگر یک پردازه از حدی که برای خود رزرو کرده بیشتر استفاده کند جریمه میشود. عدم آگاهی کاربران از این سیاست و سخت بودن تنظیم کردن برنامه‌ها برای تعیین سقف مصرف مجاز یکی از مشکلات فعلی است.

۴.۴.۲ باگ های نادر و عدم وجود community خوب برای sge

با وجود اینکه سیستم rocks-cluster شناخته شده تر است و جامعه بزرگتری از آن استفاده میکنند، نرم افزار sge که یکی از بخش های مهم کلاستر فعلی است از نبود جامعه مناسب رنج میبرد. به نظر میرسد development روی sge مدتی است که متوقف شده و باگ هایی که هر از گاهی باعث ایجاد مشکل میشود به سختی قابل حل میباشند. همچنین sge به دلیل اینکه fault-tolerance پایینی دارد وقتی در سیستم ناپایداری به وجود آید، از کار میافتد.

همه این دلایل ما را به سمت استفاده از یک تکنولوژی جدید سوق داد که مشکلات ذکر شده را برطرف کند.

فصل ۳

سیستم جایگزین

برای حل مشکلات ارائه شده نیاز به استفاده از یک تکنولوژی جدیدتر است. مسائل فعلی که در این سیستم وجود دارند (مانند isolation و scheduling) سال هاست که در cloud-computing وجود داشته اند و راه حل های مناسبی برای آنها ارائه شده است. در ادامه به مراحلی که منجر به تصمیم ما برای سیستم جدید شد می پردازیم.

۱.۳ مهمترین مسئله: isolation

مشکل دسترسی زیاد و در عین حال ناکافی و مشکل عدم محدودیت در ریسورس استفاده شده با یک راه حل isolation مناسب حل میشود.
برای isolate کردن دو نوع راه حل کلی وجود دارد: containerization و hardware-virtualization.

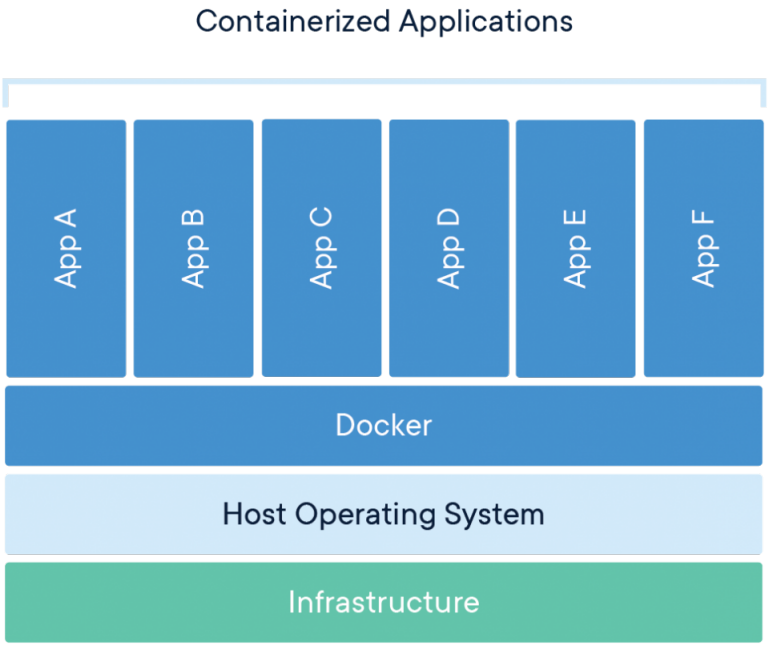
در containerization کاربر با یک سیستم عامل مجازی کار میکند در حالی که در hardware-virtualization کاربر یک سخت افزار مجازی را در اختیار دارد. برای مجازی سازی سخت افزار باید مقدار بیشتری پردازش انجام داد ولی در طراحی محیط دست باز تری داریم(میتوان از هر سیستم عامل و یا تنظیماتی استفاده کرد).

بهترین گزینه برای hardware-virtualization استفاده از نرم افزار vmware-esxi است. این نرم افزار یک hypervisor قدرتمند است که توانایی اداره ماشین مجازی در اندازه های بسیار بزرگ را داراست. در صورت استفاده از vmware کاربر برای اجرای تسک خود اول یک ماشین مجازی در کلاستر ما ساخته سپس بعد از استفاده از یک image آماده اقدام به نصب نرم افزار های خود می کند و سپس جاب خود را اجرا میکند.

این راه حل علاوه بر پیچیده و پرهزینه بودن نیاز به دانش زیادی از سمت کاربر دارد که

عملا استفاده از آن را غیر ممکن میکند. به علاوه استفاده از یک ماشین مجازی برای یک کلاستر پرفورمنس بالا کار مناسبی نیست زیرا بخش قابل توجهی از cpu برای شبیه سازی سخت افزار خرج میشود و عملا اتلاف منابع زیادی رخ میدهد.

راه حل دوم یک مفهوم نسبتا جدید به اسم containerization است. در این راه حل بجای اینکه سخت افزار شبیه سازی شود با استفاده از تعدادی از قابلیت های سیستم عامل برای یک پردازه میتوان یک محیط isolated از بقیه پردازه ها به وجود آورد.
تصویر زیر شمای کلی این مفهوم است:



تصویر ۲: مفهوم containerization و پیاده ساختار کلی آن

در این راه حل چون از ویژگی های خود سیستم عامل استفاده می شود هیچ پردازش اضافی ای برای اجرای پردازه ها انجام نمیشود.
بهترین گزینه برای استفاده از containerization نرم افزار docker است. این تکنولوژی تمام کار های مربوط به containerization را انجام میدهد و قابلیت های زیادی نیز ارائه میدهد. برای اجرای یک پردازه داخل سیستم داکر باید ابتدا یک image ساخته که ابزار

های مورد نیاز آن پردازه در آن نصب شده باشد و سپس با فرمان دادن به داکر میتوان آن را اجرا کرد.

در نهایت با توجه به خوبی ها و بدی های این دو راه حل ما به سراغ container-ization رفتیم.
ابتدا تلاش کردیم داکر را بر روی cent-os نصب کنیم و سپس با استفاده از sge اقدام به اجرای نرم افزار ها با استفاده از داکر کنیم. در این راه حل یک مشکل وجود داشت. داکر نرم افزار قدرتمندی است و قابلیت های زیادی دارد و اگر کاربر دسترسی کامل به داکر داشته باشد عملا دسترسی root در داخل سیستم دارد. این سطح از دسترسی بسیار خطرناک است و باید مکانیزمی وجود داشته باشد که جلوی این دسترسی را بگیرد. پروژه هایی مانند docker rootless وجود دارند که هنوز در مراحل آزمایشی هستند و استفاده از آن ها می‌تواند مشکل‌زا باشد. علاوه بر این همچنان مشکلات مربوط به sge باقی میماند. در واقع docker یک راه حل برای virtualization است و کار های مربوط به scheduling را انجام نمیدهد.

برای حل مشکل scheduling باید از یک تکنولوژی دیگر استفاده کرد. تکنولوژی های مرسوم برای کلاستر های HPC متشکل از cluster rocks معمولا نرم افزار های غیر ha هستند و به دلیل استفاده کمتر community کمتری برای مشکلات آنها وجود دارد. به همین دلیل استفاده از یک نرم افزار که شناخته شده تر باشد بسیار به حل مشکلات احتمالی کمک میکند. سوالی که به وجود میاید این است که آیا نرم افزار های دیگر توانای اداره کردن یک کلاستر HPC را دارند.
مهمترین کاری که rocks-cluster انجام میدهد توانایی scheduling آن است. در حال حاضر scheduling در این کلاستر به صورت یک صف fifo انجام میشود. این نوع از schedule کردن در عین حالی که ساده است برای کاربرد ما مناسب میباشد. از طرفی این نوع schedule کردن مقدار زیادی هدر رفت ریسورس دارد.

۲.۳ جایگزین ها

با توجه به مسائل گفته شده دو تا از محبوب ترین cms های موجود فعلی kubernetes و hashicorp-nomad هستند. کوبرنیتیز محصول شرکت گوگل بوده که در کلاستر های بیشماری در حال حاضر وجود دارد و نتیجه خوبی ارائه داده است. Nomad از طرف دیگر یک cms کمتر شناخته شده است ولی همچنان به دلیل سادگی بسیار محبوب است. هر دوی این cms ها از نوع ha بوده و توانایی مدیریت کلاستر های بسیار بزرگ و multi region را دارند. به علاوه هر دو قابلیت integration با docker را نیز دارا بوده و به دلیل open-source بودن بسیار آزمایش شده اند و میتوان به آنها اتکا کرد.

۱.۲.۳ کوبرنیتیز

این محصول نتیجه سالها تجربه گوگل در نگهداری سرویس های خود است و بر اساس معیار های گوگل طراحی شده. ایده کلی این محصول flexibility است. به راحتی میتوان هر کدام از بخش های این محصول را با روش های مرسوم بازنویسی کرد و نتیجه customize شده را به دست آورد. از طرفی این محصول پیچیدگی زیادی در نگهداری و بالا آوردن دارد که نیاز به یک فرد متخصص برای این کار است.

۲.۲.۳ نومد

ایده اصلی شرکت hashicorp برای ساخت این محصول سادگی و در عین حال پر قدرت بودن است. از کوچکترین کلاستر ها تا بزرگترین آنها میتوانند از nomad استفاده کنند. همچنین nomad قابلیت های بیشتری در همگام سازی با محصولات شرکت nvidia دارد. از این رو گزینه بهتری برای کلاستر فعلی دانشکدگان علوم است.

۳.۲.۳ نتیجه

به دلیل اینکه نیازی به customize کردن زیادی برای کار ما وجود ندارد و از طرفی سادگی برای کار بسیار اهمیت دارد nomad انتخاب بهتری از kubernetes است.

۳.۳ دیزاین جدید

پس از آنکه ما تصمیم به استفاده از nomad و docker گرفتیم، نیاز به یک دیزاین برای سیستم جدید داشتیم.
سیستم جدید باید ویژگی های زیر را دارا باشد:

۱. قابلیت ساپورت gpu
۲. دسترسی حداقلی برای کاربر
۳. مدیریت فضای ذخیره سازی
۴. مدیریت سیستم مالی
۵. صفحه admin
۶. مدیریت کاربران
۷. سیستم نظارتی(monitoring)

۸. عدم تعارض با سیستم قدیمی(برای انتقال تدریجی)

نومد و داکر توانایی مدیریت کلاستر را دارند. برای انجام کار هایی مثل مدیریت کاربران و یا monitoring باید یک سیستم جداگانه وجود داشته باشد که توانایی انجام کار های از این دست را داشته باشد. به عنوان مثال برای اینکه کاربر یک جاب را ثبت کند و نیاز به دسترسی ssh برای اینکار نداشته باشد باید یک صفحه طراحی شود که کاربر بتواند اطلاعات تسک خود را آنجا وارد کند.

۴.۳ اجزای سیستم جدید

- داکر و نومد: بر روی هر نود کامپیوت داکر و نومد برای اداره و تنظیم و اجرا کردن جاب های مشتریان اجرا میشود.
- سرور مدیریت: بر روی نود فرانت یک سرور اجرا میشود که مسئول کار های مدیریتی است. همچنین کار های مربوط به تحویل گرفتن جاب کاربران و سیستم مالی را بر عهده دارد.
- سیستم مانیتورینگ: برای مانیتور کردن وضعیت کلاستر که در نود مدیریتی اجرا میشود.
- یک نود storage که برای ذخیره سازی اطلاعات کاربران از آن استفاده میشود.

۱.۴.۳ سیستم نظارتی(monitoring)

در هر سیستمی اولین قدم برای حل یک مشکل یک سیستم مانیتورینگ مناسب است. در سیستم قدیم سیستم مانیتورینگ ganglia وجود داشت که به صورت پیشفرض تعدادی از مشخصه های سیستم را نظارت میکرد. عدم انعطاف پذیری ganglia استفاده از آن را سخت میکرد و همچنین به علت مشکلات ساختاری خیلی از اوقات دچار مشکل میشد.

سیستم های نظارتی در حال حاضر به دو صورت وجود دارند:

- سیستم های push-based :در این سیستم ها مشخصه هایی که باید داخل سیستم نمایش داده شوند از داخل سیستم های دیگر به سیستم مانیتورینگ ریخته میشوند.
- سیستم های pull-based :در این سیستم ها برعکس سیستم های قبلی سیستم های نظارتی شروع به بدست آوردن مشخصه ها میکنند.

سیستم ganglia یک سیستم push-based است. در هر نود یک پردازه مخصوص ganglia وجود دارد که اطلاعات را به نود اصلی منتقل میکند. برای سیستم جدید ما از ترکیب prometheus و grafana استفاده میکنیم. در این سیستم prometheus به عنوان سیستم نظارتی اطلاعات را از منابع مختلف جمع آوری میکند و grafana به نمایش مشخصه ها میپردازد. به دلیل زبان promql که یک زبان منعطف در زمینه time-series-queries هست به راحتی میتوان هر مشخصه ای از سیستم را در سیستم نظارتی به دست آورد.

۲.۴.۳ پشتیبانی gpu

یکی از مفیدترین ویژگی های کلاستر دانشکده استفاده از gpu برای انجام محاسبات است. در حال حاضر پردازه ها بر روی نود اجرا میشوند که بدون هیچ isolation ای به تمام اجزا سیستم از جمله gpu دسترسی دارند. یکی از مشکلات سیستم جدید استفاده پردازه ها از gpu در محیط isolated است. خوشبختانه هم داکر و هم نومد پشتیبانی مناسبی از gpu دارند که باعث میشود به راحتی بتوان از آن ها استفاده کرد.

همچنین یکی از برتری های سیستم جدید درک scheduler از سخت افزار gpu است که برخلاف سیستم قبلی که همه gpu ها در یک نود نصب شده اند در سیستم جدید gpu ها میتوانند در نود های مختلف نصب شوند.
برای استفاده از gpu در محیط داکر دو پیش نیاز وجود دارد:

- نصب درایور gpu
- نصب nvidia-container-runtime

درایور gpu برای شناخت host از سخت افزار gpu نیاز است. nvidia-container-runtime یک قطعه کد کوچک است که شرکت nvidia برای استفاده gpu در داکر از آن استفاده میکند. به طور مختصر این قطعه کد دقیقا قبل ساخته شدن یک محیط مجازی اجرا شده و gpu مربوطه را به آن محیط اضافه میکند

برای اضافه کردن توانایی شناسایی gpu در نومد نیاز به نصب nvidia-nomad-plugin است. این پلاگین به درایور nvidia وصل شده و اطلاعات تمام gpu ها را از نود میگیرد تا بتواند جاب ها را بر حسب نیازشان schedule کند.

Bibliography

[1] IBM Cloud Education. *Containerization*. 2021. URL: `https://www.ibm.com/cloud/learn/containerization` (visited on 03/12/2022).

[2] PB GALVIN and G GAGNE. "Operating system concepts 10th edition". In: (2018).

[3] Martin Kleppmann. *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems.* " O'Reilly Media, Inc.", 2017.

Abstract

The College of Science's Parallel Computing Cluster(PCC) provides a convenient environment for those researchers who need to run a heavy computational task. Computation is needed by many researches however, using personal computers for these tasks is not only wasteful but also expensive. By this means College of Science's PCC is launched and ready to help researchers with their tasks using specialized hardware. In this project we managed to solve common problems of the PCC by using new technologies and making it more comfortable for the users. One of the challenges we had was to add GPU Computing Support to this system.

College of Science
School of Mathematics, Statistics, and Computer Science

Adding Support for GPU Computing to College of Science's Parallel Computing Cluster

Armin Shoushtari

Supervisor: Dr. Mohammad Ganjtabesh

A thesis submitted in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science

2022