مقدمه‌ای بر

اصول معماری نرم‌افزار

مرتضی نقی‌زاده «رگانتو»

reganto.github.io

t.me/cleanark

ورژن 0.1.0

مقیاس‌پذیری

درباره

افزایش یا کاهش ظرفیت یک سیستم با استفاده موثر از منابع به عنوان مقیاس‌پذیری شناخته می شود. یک سیستم مقیاس‌پذیر می‌تواند تعداد فزاینده‌ای از درخواست‌ها را بدون تاثیر نامطلوب بر زمان پاسخ‌دهی مدیریت کند.

در مهندسی نرم‌افزار، مقیاس‌پذیری ویژگی مطلوب یک سیستم، یک شبکه یا یک فرایند است که نشان‌دهنده توانایی آن در انجام حجم فزاینده کار به شیوه‌ای زیبا و روان است. به عنوان مثال، می توانیم به توانایی یک سیستم برای افزایش توان کل تحت یک بار افزایش یافته در هنگام اضافه شدن منابع (معمولا سخت افزار) اشاره کنیم.

مقیاس‌پذیری، به عنوان یک ویژگی سیستم‌ها، به طور کلی به سختی قابل تعریف است و در هر مورد خاص لازم است که الزامات خاصی برای مقیاس‌پذیری در ابعادی که مهم تلقی می‌شوند، تعریف شوند. مقیاس‌پذیری یک مسئله بسیار مهم در سیستم‌های الکترونیکی، پایگاه داده، روترها و شبکه است.

مقیاس‌پذیری اصولاً مفهومی برآمده از علم تجارت است که در مهندسی نرم‌افزار نیز استفاده می‌شود. (مشابه الگوهای طراحی در معماری و الگوهای طراحی در مهندسی نرم‌افزار) در هر صورت ما با یک مفهوم ثابت سروکار داریم- توانایی یک تجارت یا فناوری برای پذیرش حجم افزایش یافته بدون تاثیر بر حاشیه سود (= افزایش درآمد بدون تغییر در هزینه های متغیر). به عنوان مثال، یک قطعه معین از تجهیزات ممکن است از 1 تا 1000 کاربر ظرفیت داشته باشد، و بیش از 1000 کاربر، تجهیزات اضافی مورد نیاز است در غیر اینصورت منجر به کاهش عملکرد خواهد شد (هزینه‌های متغیر افزایش و حاشیه سود کاهش می‌یابد).

یک مثال عملی

فرض کنید یک کارخانه تولید کفش در هفته ۵۰۰ جفت کفش تولید می‌کند اما به یکباره تقاضا برای کفش‌های تولیدی این کارخانه افزایش یافته و کارخانه باید در هفته ۱۵۰۰ جفت کفش تولید کند در این صورت اگر اصول مقیاس‌پذیری در کارخانه رعایت شده باشد، صاحبان کارخانه می‌توانند بدون هزینه اضافی (یا با کم‌ترین هزینه اضافی) تولید را افزایش دهند. (افزایش سود بدون تغییر در هزینه‌های متغیر)

و اما دو نکته

۱. مقیاس پذیری به طور موازی کار می کند.

۲. پیچیدگی و رویکردهای متکی‌به‌انسان مقیاس‌پذیر نمی‌شوند. سادگی و رویکردهای الگوریتم‌محور مقیاس‌پذیر می‌شوند.

مقیاس پذیر به چه معناست؟

تعریف عملیاتی

- در گذشته: "حتی اگر داده ها در حافظه اصلی جا نشوند کار می کند"
- اکنون: "می توان از 1000 کامپیوتر ارزان استفاده کرد"

تعریف الگوریتمی

- در گذشته: اگر N آیتم داده دارید، نباید بیش از N^M عملیات انجام دهید (الگوریتم زمانی چندجمله‌ای)
- اکنون: اگر N آیتم داده دارید، برای مقادیر k بزرگ نباید بیش از N^M عملیات انجام دهید (با افزایش k تعداد عملیات‌ها همچنان ثابت می‌ماند)
- به زودی: اگر N آیتم داده دارید، نباید بیش از (N * LOG(N عملیات انجام دهید (الگوریتم زمانی لگاریتمی)

سخت‌افزار/سیستم

سیستمی که عملکرد آن پس از افزودن سخت افزار، متناسب با ظرفیت اضافه شده، بهبود می یابد، سیستمی مقیاس پذیر گفته می شود.

الگوریتم/برنامه

به یک الگوریتم، طراحی، پروتکل شبکه، برنامه یا سایر سیستم‌ها گفته می‌شود که در شرایطی که برای موقعیت‌های بزرگ به کار می‌رود، کارآمد و کاربردی باشد (مثلاً مجموعه داده‌های ورودی بزرگ یا تعداد زیادی گره‌های مشارکت‌کننده در مورد یک سیستم توزیع‌شده) مقیاس‌پذیر هستند. اگر طراحی با افزایش مقدار شکست بخورد، مقیاس نمی شود.

مثال

یک سیستم پردازش تراکنش آنلاین مقیاس‌پذیر یا سیستم مدیریت پایگاه داده، سیستمی است که می‌تواند برای پردازش تراکنش‌های بیشتر با افزودن منابع سخت‌افزاری جدید (پردازنده‌ها، دستگاه‌ها و ذخیره‌سازی) ارتقا یابد. این ارتقا به سادگی و بدون خاموش شدن سیستم اتفاق می‌افتد.

مقیاس‌بندی

مقیاس‌بندی فرآیند افزایش یا کاهش ظرفیت سیستم با تغییر تعداد فرآیندهای موجود برای درخواست خدمات است. مقیاس‌بندی یک سیستم ظرفیت اضافی را فراهم می کند، در حالی که کاهش مقیاس در یک سیستم ظرفیت را کاهش می دهد. مقیاس بندی نیز بخش مهمی از پیکربندی استقرار برای دسترسی بالا است.

روش‌های افزودن منابع بیشتر برای یک برنامه خاص به دو دسته کلی تقسیم می‌شوند:

- مقیاس عمودی (scale up)
- مقیاس افقی (scale out)

مقیاس عمودی (Scale Up)

"Scale up" زمانی است که شما یک ماشین را به یک ماشین قدرتمندتر ارتقا می دهید (مثلاً CPU سریعتر، GPU سریعتر، موتور با HP بیشتر، و غیره...) تا قدرت پردازش بیشتری داشته باشید.

مقیاس عمودی (scale vertically - scale up) به معنای اضافه کردن موارد زیر به یک گره در یک سیستم است:

- منابع (افزودن CPU یا حافظه)
- یا فرآیند (افزودن همان فرآیند)

یکی از مزایای عمده این روش، استفاده از فناوری مجازی‌سازی به شکل موثر آن است زیرا در سیستم‌هایی که به صورت Scale Up مقیاس‌بندی شده‌اند می‌توانیم منابع بیشتری برای مجازی‌سازی داشته باشیم.

مثال

فرض کنید یک CPU 800 مگاهرتز در رایانه خود داشته باشید. برای افزایش قدرت پردازش، تصمیم می گیرید یک کامپیوتر جدید با یک CPU 8 هسته ای 3.4 گیگاهرتزی بخرید تا جایگزین کامپیوتر قدیمی خودتان کنید.

مقیاس افقی (Scale Out)

"Scale Out" زمانی است که تعداد ماشین های پردازش (رایانه ها، پردازنده ها، سرورها و غیره) را برای افزایش قدرت پردازش افزایش می دهید.

مقیاس افقی (Scale Out) به معنای افزودن گره های بیشتری به یک سیستم است، مانند افزودن یک کامپیوتر جدید به یک برنامه نرم افزاری توزیع شده. یک مثال ممکن است افزودن دو نمونه از یک وب سرور به وب سرور موجود باشد.

با کاهش قیمت کامپیوترها و افزایش کارایی آن‌ها، سیستم‌های ارزان قیمت می‌توانند برای اجرای برنامه‌های محاسباتی با کارایی بالا استفاده شوند. این برنامه‌ها در گذشته فقط توسط ابررایانه‌ها قابل مدیریت بودند. صدها کامپیوتر کوچک ممکن است در یک **خوشه** (Cluster) پیکربندی شوند تا قدرت محاسباتی کل را بدست آورند.

مدل scale out تقاضای فزاینده‌ای برای ذخیره‌سازی داده‌های مشترک با عملکرد ورودی/خروجی بسیار بالا ایجاد کرده است، به‌ویژه در مواردی که پردازش مقادیر زیادی از داده‌ها مورد نیاز است، مانند تحلیل لرزه‌ای. این امر به توسعه فن آوری‌های ذخیره‌سازی جدید مانند دستگاه‌های ذخیره‌سازی اشیا دامن زده است . هنگامی که می خواهید یک **محیط بسیار در دسترس** را پیکربندی کنید، باید فرآیندها را برای دستیابی به افزونگی اصطلاحا scale out کنید.

مثال: واگن شما در حال حاضر توسط یک اسب کشیده می شود. برای افزایش قدرت واگن 10 اسب دیگر می خرید تا واگن را بکشند.

تریدآف‌ها

تریدآف‌هایی بین این دو مدل وجود دارد.

تعداد بیشتر کامپیوترها به معنای افزایش پیچیدگی مدیریت و همچنین مدل برنامه نویسی پیچیده‌تر و مسائلی مانند توان عملیاتی و تأخیر بین گره‌ها است. همچنین، برخی از برنامه‌ها را نمی‌توان در چارچوب یک مدل محاسباتی توزیع شده ارائه کرد. Scale Up به سرعت گران تمام می‌شود. یک راه حل دیگر این است که سرورهای ارزان خریداری کنید تا به جای توسعه عمودی، برنامه (پایگاه داده، ...) را در صدها یا حتی هزاران سرور توزیع کنید.

در گذشته، تفاوت قیمت بین این دو مدل، مقیاس‌پذیری افقی را برای آن دسته از برنامه‌هایی که با پارادایم آن مطابقت داشتند به گزینه‌ای مناسب تبدیل می‌کرد اما پیشرفت‌های اخیر در فناوری مجازی‌سازی این مزیت را محو کرده است. استقرار (deploy) یک سیستم مجازی جدید بر روی یک هایپروایزر (در صورت امکان) تقریباً همیشه ارزان‌تر از خرید و نصب واقعی یک سرور است.

مقیاس پذیری پایگاه داده

تعدادی از رویکردهای مختلف پایگاه‌های اطلاعاتی را قادر می‌سازد تا به اندازه بسیار بزرگ رشد کنند و در عین حال از نرخ فزاینده تراکنش‌ها در ثانیه پشتیبانی می‌کنند. البته سرعت سریع پیشرفت‌های سخت‌افزاری در سرعت و ظرفیت دستگاه‌های ذخیره‌سازی انبوه و همچنین پیشرفت های مشابه در CPU و سرعت شبکه قابل کاهش نیست. فراتر از آن، معماری‌های مختلفی در پیاده‌سازی پایگاه‌های داده در مقیاس بسیار بزرگ به کار گرفته شده‌اند.

پارتیشن‌بندی (partitioning) و خوشه (cluster)

یکی از تکنیک هایی که توسط اکثر محصولات اصلی DBMS پشتیبانی می شود، پارتیشن‌بندی جداول بزرگ بر اساس محدوده مقادیر در یک فیلد کلیدی است. به این ترتیب، پایگاه داده را می توان به صورت خوشه‌ای از سرورهای مجزا Scale Out کرد.

چند نخی (multi-threading)

همچنین، با ظهور ریزپردازنده‌های 64 بیتی، پردازنده‌های چند هسته‌ای و چند پردازنده‌های بزرگ SMP ، فروشندگان DBMS در خط مقدم پشتیبانی از پیاده‌سازی‌های چند نخی بوده‌اند که به طور قابل‌توجهی ظرفیت پردازش تراکنش را افزایش می‌دهند.

nas/san

ذخیره‌سازهای متصل به شبکه (NAS) و شبکه‌های منطقه ذخیره‌سازی (SAN) همراه با شبکه‌های محلی (LAN) سریع و فناوری کانال فیبر (Fiber Channel)، پیکربندی‌های بزرگ‌تر و منسجم‌تری (loosely coupled) از پایگاه‌های داده و قدرت محاسباتی توزیع‌شده را ممکن می‌سازند.

shared nothing/shared-everything

استاندارد X/Open XA که به طور گسترده پشتیبانی می شود، از یک مانیتور تراکنش جهانی برای هماهنگ کردن تراکنش های توزیع شده در بین منابع پایگاه داده نیمه خودمختار سازگار با XA استفاده می کند. Oracle RAC از مدل متفاوتی برای دستیابی به مقیاس‌پذیری استفاده می‌کند که بر اساس معماری «shared-everything» است که بر اتصالات پرسرعت بین سرورها متکی است.

در حالی که فروشندگان DBMS در مورد شایستگی‌های نسبی طرح‌های مورد علاقه خود بحث می کنند، برخی از شرکت‌ها و محققان راجع‌به محدودیت‌های ذاتی سیستم‌های مدیریت پایگاه داده رابطه‌ای سوالات اساسی می‌پرسند و اصولا مدل رابطه‌ای را مدلی مناسب برای مقیاس‌پذیری نمی‌دانند. به عنوان مثال GigaSpaces ادعا می کند که یک مدل کاملاً متفاوت از دسترسی به داده‌های توزیع‌شده و پردازش تراکنش، به نام معماری مبتنی بر فضا(space base architecture)، برای دستیابی به بالاترین عملکرد و مقیاس پذیری مورد نیاز است. از سوی دیگر، Base One بدون انحراف از فناوری پایگاه داده اصلی(مدل رابطه‌ای)، نوعی مقیاس پذیری افراطی را مطرح می کند. به نظر می رسد نمی‌توان هیچ پایانی برای بحث‌ها راجع‌به مقیاس پذیری پایگاه داده متصور شد.

سخت افزار بیشتر برای مقیاس پذیری؟

اغلب توصیه می شود که طراحی سیستم را بر مقیاس پذیری سخت افزار متمرکز کنید تا ظرفیت . ما معمولا با دو پارادایم برای دستیابی به عملکرد بهتر رو به رو هستیم. افزودن یک گره جدید به سیستم یا درگیری مداوم در تنظیم عملکرد هر گره برای بهبود ظرفیت پردازشی گره مورد نظر. در اکثر موارد مورد اول راحت‌تر و ارزان‌تر است. (راحت‌تر از آن جهت که اگر زیرساخت‌ها از قبل مهیا شده باشد، افزودن یک گره جدید مانند کپی پیست کردن یک فایل است). اما این رویکرد می تواند بازدهی کاهشی داشته باشد.

به عنوان مثال فرض کنید بخشی از یک برنامه می تواند در صورت موازی سازی با افزایش سرعت 70 درصد اجرا شود و دیگر آن که برنامه‌ی مورد نظر به جای یک CPU روی چهار CPU اجرا شود.

اگر:

- α کسری از یک محاسبه است که متوالی است،
- 1 - α کسری است که می توان آن را موازی کرد،
- و P تعداد پردازنده

سپس حداکثر سرعتی که می توان به دست آورد طبق قانون امدال(Amdahl's Law) داده می شود:

$$\frac{1}{\alpha+\frac{application:1-\alpha}{P}}$$

با جایگزینی مقادیر با α = 0.3، درمی‌یابیم که:

- برای 4 پردازنده: 2.105.
- برای 8 پردازنده: 2.581.

دوبرابر کردن قدرت پردازش فقط یک پنجم سرعت را بهبود بخشیده است. اگر کل مشکل قابل موازی بود، البته ما انتظار دو برابر شدن سرعت را داشتیم. بنابراین، افزایش توان سخت افزاری شاید منجر به افزایش جزیی سرعت پردازشی شود اما نمی‌توان آن را لزوما رویکردی بهینه در نظر گرفت.

ترجمه‌ای آزاد از این مطلب همراه با اضافات