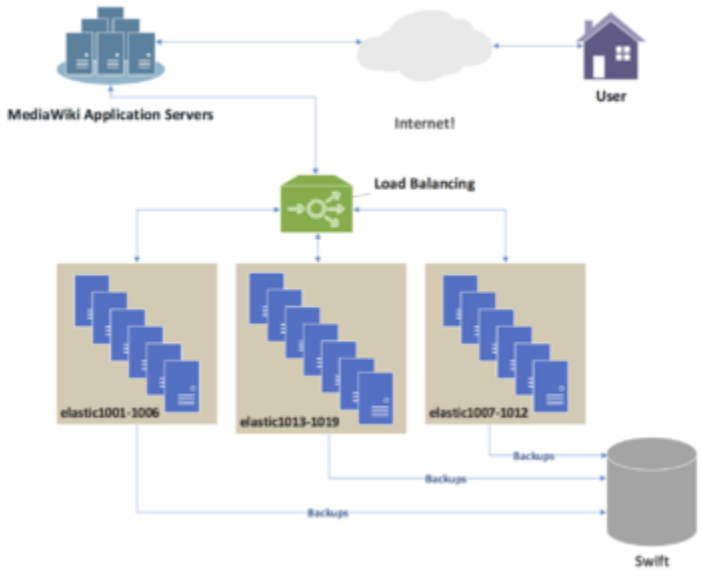
متعادل نمودن بار ترافیکی

در محاسبات ، **تعادل بار** به فرآیند توزیع مجموعه ای از وظایف بر روی مجموعه ای از منابع (واحدهای محاسباتی)، با هدف کارآمدتر کردن پردازش کلی آنها اشاره می کند . متعادل‌سازی بار می‌تواند زمان پاسخ را بهینه کند و از بارگذاری ناهموار برخی گره‌های محاسباتی در حالی که سایر گره‌های محاسباتی بی‌حرکت می باشند ، جلوگیری کند.



نموداری که درخواست های کاربر به یک خوشه Elasticsearch را نشان می دهد که توسط یک متعادل کننده بار توزیع می شود. (نمونه ای برای ویکی پدیا .)

تعادل بار موضوع تحقیق در زمینه کامپیوترهای موازی است . دو رویکرد اصلی وجود دارد: الگوریتم‌های استاتیک، که وضعیت ماشین‌های مختلف را در نظر نمی‌گیرند ، الگوریتم‌های پویا، که معمولاً عمومی‌تر و کارآمدتر هستند ، ولی نیازمند تبادل اطلاعات بین واحدهای محاسباتی مختلف، در معرض خطر ضرر هستند.

بررسی اجمالی مشکل

یک الگوریتم متعادل کننده بار همیشه سعی می کند به یک مشکل خاص پاسخ دهد. از جمله، ماهیت وظایف، پیچیدگی الگوریتمی، معماری سخت افزاری که الگوریتم ها بر روی آن اجرا می شوند و همچنین تحمل خطای مورد نیاز باید در نظر گرفته شود. بنابراین مصالحه باید پیدا شود تا به بهترین وجه نیازهای ویژه برنامه برآورده شود.

ماهیت وظایف

کارایی الگوریتم های متعادل کننده بار به طور بحرانی به ماهیت وظایف بستگی دارد. پس هرچه اطلاعات بیشتری درباره وظایف در زمان تصمیم‌گیری در دسترس باشد، پتانسیل بهینه‌سازی بیشتر می‌شود.

اندازه وظایف

دانش کامل از زمان اجرای هر یک از وظایف به شما امکان می دهد به توزیع بار بهینه برسید (به الگوریتم جمع پیشوند مراجعه کنید). متأسفانه، این در واقع یک مورد ایده آل است. دانستن زمان دقیق اجرای هر کار یک موقعیت بسیار نادر است.

به همین دلیل، تکنیک های مختلفی برای دریافت ایده از زمان های مختلف اجرا وجود دارد. از سوی دیگر، اگر زمان اجرا بسیار نامنظم است، باید از تکنیک های پیچیده تری استفاده کرد. یک تکنیک این است که به هر کار مقداری ابرداده اضافه کنید. بسته به زمان اجرای قبلی برای ابرداده های مشابه، می توان استنباط هایی را برای یک کار آینده بر اساس آمار انجام داد. [۱]

وابستگی ها

در برخی موارد، وظایف به یکدیگر بستگی دارد. این وابستگی های متقابل را می توان با یک نمودار غیر چرخه ای جهت دار نشان داد. برخی از کارها نمی توانند شروع شوند تا زمانی که برخی دیگر تکمیل شوند.

با فرض اینکه زمان مورد نیاز برای هر یک ازوظایف از قبل مشخص باشد، یک دستور اجرای بهینه باید منجر به به حداقل رساندن کل زمان اجرا شود. اگر چه این یک مشکل NP-hard است و بنابراین می تواند به طور دقیق حل شود.
الگوریتم‌هایی مانند زمان‌بندی کار وجود دارند که توزیع وظایف بهینه را با استفاده از روش‌های فراابتکاری محاسبه می‌کنند.

تفکیک وظایف

یکی دیگر از ویژگی های وظایف حیاتی برای طراحی یک الگوریتم متعادل کننده بار، توانایی آنها در تجزیه به وظایف فرعی در طول اجرا است. الگوریتم «محاسبات درختی شکل» که بعداً ارائه شد از این ویژگی بهره می‌برد.

الگوریتم های استاتیک و پویا

استاتیک

یک الگوریتم متعادل کننده بار زمانی "ایستا" است که وضعیت سیستم را برای توزیع وظایف در نظر نگیرد. بنابراین، وضعیت سیستم شامل معیارهایی مانند سطح بار (و گاهی اوقات حتی اضافه بار) پردازنده های خاص است. در عوض، مفروضاتی در مورد سیستم کلی از قبل ساخته شده است، مانند زمان رسیدن و منابع مورد نیاز وظایف ورودی. علاوه بر این، تعداد پردازنده ها، قدرت مربوطه و سرعت ارتباط آنها مشخص است. بنابراین، متعادل‌سازی باراستاتیک به منظور به حداقل رساندن یک عملکرد عملکرد خاص، مجموعه‌ای از وظایف شناخته‌شده را با پردازنده‌های موجود مرتبط می‌کند. ترفند در مفهوم این تابع عملکرد نهفته است.

تکنیک های متعادل کننده بار استاتیک معمولاً در اطراف یک روتر یا Master متمرکز می شوند که بارها را توزیع می کند و عملکرد عملکرد را بهینه می کند. این کمینه سازی می تواند اطلاعات مربوط به وظایفی که باید توزیع شود را در نظر بگیرد و زمان اجرای مورد انتظار را بدست آورد.

مزیت الگوریتم های ایستا این است که راه اندازی آسان و بسیار کارآمد در موارد نسبتاً معمولی (مانند پردازش درخواست های HTTP از یک وب سایت) است. با این حال، هنوز مقداری واریانس آماری درانتساب وظایف وجود دارد که می تواند منجر به بارگذاری بیش از حد برخی از واحدهای محاسباتی شود.

پویا

الگوریتم‌های توزیع بار استاتیک، الگوریتم‌های پویا بار جاری هر یک از واحدهای محاسباتی (که گره‌ها نیز نامیده می‌شوند) درسیستم را در نظر می‌گیرند. در این رویکرد، وظایف را می توان به صورت پویا از یک گره بارگذاری شده به یک گره کم بار منتقل کردتا پردازش سریعتر دریافت کند. در حالی که این الگوریتم‌ها برای طراحی بسیار پیچیده‌تر هستند، اما می‌توانند نتایج عالی را تولید کنند، به‌ویژه زمانی که زمان اجرا از یک کار به کار دیگر بسیار متفاوت است.

معماری متعادل کننده بار پویا می تواند مدولارتر باشد زیرا داشتن یک گره خاص برای توزیع کار اجباری نیست. هنگامی که وظایف به طور منحصربه‌فرد با توجه به وضعیت آنها در یک لحظه خاص به یک پردازنده اختصاص داده می‌شوند، این یک انتساب منحصر به فرد است. از طرف دیگر، اگر بتوان وظایف را بر اساس وضعیت سیستم و تکامل آن به طور دائمی توزیع کرد، به این امر تخصیص پویا می گویند. [۲] بدیهی است که یک الگوریتم متعادل کننده بار که برای رسیدن به تصمیمات خود به ارتباطات بیش از حد نیاز دارد، خطر کند کردن حل مشکل کلی رادارد.

معماری سخت افزار

ماشین های ناهمگن

زیرساخت‌های محاسباتی موازی اغلب از واحدهایی با توان محاسباتی مختلف تشکیل شده‌اندکه باید برای توزیع بار درنظر گرفته شوند.

به عنوان مثال، واحدهای کم مصرف ممکن است درخواست هایی را دریافت کنند که نیاز به محاسبات کمتری دارند، یا در مورد اندازه های درخواستی همگن یا ناشناخته، درخواست های کمتری نسبت به واحدهای بزرگتردریافت کنند.

حافظه مشترک و توزیع شده

کامپیوترهای موازی معمولاً به دو دسته کلی تقسیم می‌شوند: آنهایی که در آن‌ها همه پردازنده‌ها یک حافظه مشترک دارندکه روی آن به صورت موازی می‌خوانند و می‌نویسند (مدل PRAM)، و آنهایی که هر واحد محاسباتی حافظه خاص خود را دارد (مدل حافظه توزیع شده) و اطلاعات از طریق پیام ردوبدل می شود.

برای رایانه‌های دارای حافظه مشترک ، مدیریت تضادهای نوشتن، سرعت اجرای جداگانه هر واحد محاسباتی را تا حد زیادی کاهش می‌دهد. با این حال، آنها می توانند به طور موازی به خوبی کار کنند. برعکس، در مورد تبادل پیام، هر یک از پردازنده ها می توانند با سرعت کامل کار کنند. از سوی دیگر، وقتی صحبت از تبادل پیام جمعی به میان می آید، همه پردازنده ها مجبورند منتظر بمانند تا کندترین پردازنده ها فاز ارتباطی را شروع کنند.

در واقع، تعداد کمی از سیستم ها دقیقاً در یکی از دسته بندی ها قرار می گیرند. به طور کلی، پردازنده ها هر کدام یک حافظه داخلی برای ذخیره داده های مورد نیاز برای محاسبات بعدی دارند و در خوشه های متوالی سازماندهی می شوند. اغلب، این عناصر پردازش از طریق حافظه توزیع شده و ارسال پیام هماهنگ می شوند. بنابراین، الگوریتم متعادل کننده بار باید به طور منحصر به فرد با یک معماری موازی تطبیق داده شود. در غیر این صورت، این خطر وجود دارد که کارایی حل مسائل موازی تا حد زیادی کاهش یابد.

سلسله مراتب

با انطباق با ساختارهای سخت افزاری که در بالا مشاهده می شود، دو دسته اصلی از الگوریتم های متعادل کننده بار وجود دارد. از یک طرف، کاری که در آن وظایف توسط «استاد» محول می شود و توسط «کارگران» اجرا می شود که استاد را از پیشرفت کار خود مطلع می کنند، و سپس استاد می تواندمسئولیت تخصیص یا تخصیص مجدد حجم کار را بر عهده بگیرد. الگوریتم پویا ادبیات به این معماری «استاد-کارگر» اشاره می‌کند. از سوی دیگر، کنترل را می توان بین گره های مختلف توزیع کرد. سپس الگوریتم متعادل کننده بار بر روی هر یک از آنها اجرا می شود و مسئولیت تخصیص وظایف (و همچنین تخصیص مجدد و تقسیم در صورت لزوم) به اشتراک گذاشته می شود. دسته آخر یک الگوریتم متعادل کننده بار پویا را فرض می کند.

از آنجایی که طراحی هر الگوریتم متعادل کننده بار منحصر به فرد است، تمایزقبلی باید واجد شرایط باشد. بنابراین، امکان داشتن یک استراتژی میانی نیز وجود دارد، به عنوان مثال، گره‌های «مستر» برای هر زیرخوشه، که خود مشمول یک

«مستر» جهانی هستند. همچنین سازمان‌های چند سطحی، با تناوب بین استراتژی‌های کنترل اصلی و توزیع‌شده وجود دارند. استراتژی های اخیر به سرعت پیچیده می شوند و به ندرت با آنها مواجه می شویم. طراحان الگوریتم هایی را ترجیح می دهند که کنترل آنها راحت تر باشد.

انطباق با معماری های بزرگتر (مقیاس پذیری)

در زمینه الگوریتم هایی که در طولانی مدت اجرا می شوند (سرورها، ابر...)، معماری کامپیوتر در طول زمان تکامل می یابد. با این حال، ترجیح داده می شود که مجبور نباشیم هر بار یک الگوریتم جدید طراحی کنیم.

یکی از پارامترهای بسیار مهم یک الگوریتم متعادل کننده بار، توانایی آن برای انطباق با معماری سخت افزاری مقیاس پذیر است. به این مقیاس پذیری الگوریتم می گویند. یک الگوریتم زمانی مقیاس پذیر برای یک پارامتر ورودی نامیده می شود که عملکرد آن نسبتاً مستقل از اندازه آن پارامتر باقی بماند.

هنگامی که الگوریتم قادر به تطبیق با تعداد متغیری از واحدهای محاسباتی باشد، اما تعداد واحدهای محاسباتی باید قبل از اجرا ثابت شود، به آن قالب‌پذیر می‌گویند. از سوی دیگر، اگر الگوریتم بتواند در طول اجرای خود با مقدار متغیری از پردازنده ها مقابله کند، الگوریتم چکش خوار گفته می شود. اکثرالگوریتم های متعادل کننده بار حداقل قابل قالب گیری هستند. [۲]

تحمل خطا

به خصوص در خوشه های محاسباتی در مقیاس بزرگ، اجرای یک الگوریتم موازی که نتواند در برابر شکست یک جزء واحد مقاومت کند، قابل تحمل نیست. بنابراین، الگوریتم‌های تحمل خطا در حال توسعه هستند که می‌توانند خرابی پردازنده‌ها را شناسایی کرده و محاسبات را بازیابی کنند. [۴]

رویکردها

توزیع ایستا با آگاهی کامل از وظایف: مجموع پیشوند

اگر وظایف مستقل ازیکدیگر باشند و زمان اجرای مربوطه و وظایف را بتوان تقسیم بندی کرد، یک الگوریتم ساده و بهینه وجود دارد.

با تقسیم وظایف به گونه ای که مقدار محاسبات یکسانی به هر پردازنده داده شود، تنها کاری که باید انجام شود این است که نتایج را با هم گروه بندی کنیم. با استفاده از یک الگوریتم جمع پیشوند ، این تقسیم را می توان در زمان لگاریتمی با توجه به تعداد پردازنده ها محاسبه کرد.

الگوریتم تعادل بار بسته به تقسیم پذیری وظایف

با این حال، اگروظایف را نتوان تقسیم بندی کرد (یعنی اتمی هستند)، اگرچه بهینه سازی تخصیص کار مشکلی دشوار است، هنوز هم می توان به طور تقریبی توزیع نسبتاً منصفانه ای از کارها را تخمین زد، مشروط بر اینکه اندازه هر یک از آنها بسیار کوچکتر باشد. از مجموع محاسبات انجام شده توسط هر یک از گره ها.

بیشتر اوقات، زمان اجرای یک کار ناشناخته است و فقط تقریب های تقریبی در دسترس است. این الگوریتم، اگرچه کارآمدی ویژه ای دارد، اما برای این سناریوها قابل اجرا نیست.

توزیع بار استاتیک بدون اطلاع قبلی

حتی اگر زمان اجرا از قبل مشخص نباشد، توزیع بار استاتیک همیشه امکان پذیر است.

برنامه ریزی رفت و برگشت

در یک الگوریتم گردی، اولین درخواست به سرور اول، سپس درخواست بعدی به سرور دوم و به همین ترتیب تا آخرین سرور ارسال می شود. سپس دوباره شروع می شود و درخواست بعدی را به سرور اول اختصاص می دهد و به همین ترتیب.

این الگوریتم را می توان به گونه ای وزن کرد که قدرتمندترین واحدها بیشترین تعداد درخواست را دریافت کرده وابتدا آنها را دریافت کنند.

استاتیک تصادفی شده

توازن بار استاتیک تصادفی شده به سادگی یک موضوع اختصاص دادن تصادفی وظایف به سرورهای مختلف است. این روش کاملاً خوب عمل می کند. از سوی دیگر، اگر تعداد وظایف از قبل مشخص باشد، محاسبه جایگشت تصادفی از قبل کارآمدتر است. این از هزینه های ارتباطی برای هر تکلیف جلوگیری می کند. دیگر نیازی به یک توزیع اصلی نیست زیرا هر

پردازنده می داند که چه وظیفه ای به آن محول شده است. حتی اگر تعداد وظایف ناشناخته باشد، باز هم می توان از ارتباط با یک نسل تخصیص شبه تصادفی شناخته شده برای همه پردازنده ها اجتناب کرد.

عملکرد این استراتژی (که در زمان اجرای کل برای یک مجموعه مشخص از وظایف اندازه گیری می شود) با حداکثر اندازه وظایف کاهش می یابد.

دیگران

البته روش های دیگری نیزبرای انتساب وجود دارد:

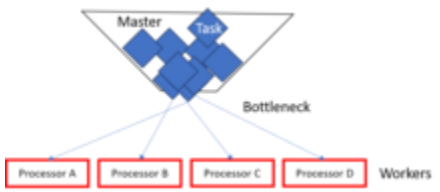
- کار کمتر: با انجام کارهای کمتر، وظایف بیشتری را به سرورها اختصاص دهید (روش را می توان وزن کرد).
- Hash: پرس و جوها را بر اساس جدول هش تخصیص می دهد.
- قدرت دو انتخاب: دو سرور را به صورت تصادفی انتخاب کنید و از بین دو گزینه بهتر را انتخاب کنید. [۵] [۶]

طرح استاد-کارگر

طرح‌های Master-Worker از ساده‌ترین الگوریتم‌های موازنه بار پویا هستند. یک استاد حجم کار را بین همه کارگران توزیع می کند (که گاهی اوقات به آنها "برده" نیز می گویند). در ابتدا، همه کارگران بیکار هستند و این را به استاد گزارش می دهند. استاد به درخواست های کارگران پاسخ می دهد و وظایف را بین آنها توزیع می کند. وقتی دیگر وظیفه ای برای دادن ندارد، به کارگران اطلاع می دهد تا از درخواست تکلیف منصرف شوند.

مزیت این سیستم این است که بار را بسیار منصفانه تقسیم می کند. در واقع، اگر زمان مورد نیاز برای انتساب را در نظر نگیریم، زمان اجرا با مجموع پیشوندی که در بالا مشاهده می شود قابل مقایسه خواهد بود.

مشکل این الگوریتم این است که به دلیل حجم بالای ارتباطات لازم، برای تطبیق با تعداد زیادی پردازنده مشکل دارد. این عدم مقیاس پذیری باعث می شود که به سرعت در سرورهای بسیار بزرگ یا رایانه های موازی بسیار بزرگ غیر قابل اجرا باشد. استاد به عنوان یک گلوگاه عمل می کند.



استاد-کارگر و تنگنا

با این حال، کیفیت الگوریتم را می توان با جایگزین کردن Master با لیست وظایف که می تواند توسط پردازنده های مختلف استفاده شود، تا حد زیادی بهبود بخشد. اگرچه اجرای این الگوریتم کمی دشوارتر است، اما نوید مقیاس پذیری بسیار بهتری را می دهد، اگرچه هنوز برای مراکز محاسباتی بسیار بزرگ کافی نیست.

منابع

1. Liu, Qi; Cai, Weidong; Jin, Dandan; Shen, Jian; Fu, Zhangjie; Liu, Xiaodong; Linge, Nigel (30 August 2016). "Estimation Accuracy on Execution Time of Run-Time Tasks in a Heterogeneous Distributed Environment" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038664). Sensors. **16** (9): 1386. Bibcode:2016Senso..16.1386L (https://ui.adsabs.harvard.edu/abs/2016Senso..16.1386L). doi:10.3390/s16091386 (https://doi.org/10.3390%2Fs16091386). PMC 5038664 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038664). PMID 27589753 (https://pubmed.ncbi.nlm.nih.gov/27589753).

2. Alakeel, Ali (November 2009). "A Guide to Dynamic Load Balancing in Distributed Computer Systems" (https://www.researchgate.net/publication/268200851). International Journal of Computer Science and Network Security (IJCSNS). **10**.

3. Asghar, Sajjad; Aubanel, Eric; Bremner, David (October 2013). "A Dynamic Moldable Job Scheduling Based Parallel SAT Solver". 2013 42nd International Conference on Parallel Processing: 110–119. doi:10.1109/ICPP.2013.20 (https://doi.org/10.1109%2FICPP.2013.20). ISBN 978-0-7695-5117-3.

4. Punetha Sarmila, G.; Gnanambigai, N.; Dinadayalan, P. (2015). "Survey on fault tolerant — Load balancing algorithmsin cloud computing". 2nd International Conference on Electronics and Communication Systems (ICECS): 1715–1720. doi:10.1109/ECS.2015.7124879 (https://doi.org/10.1109%2FECS.2015.7124879). ISBN 978-1-4799-7225-8.

5. "NGINX and the "Power of Two Choices" Load-Balancing Algorithm" (https://web.archive.org/web/20191212194243/https://www.nginx.com/blog/nginx-power-of-two-choices-load-balancing-algorithm/). nginx.com. 2018-11-12. Archived from the original (https://www.nginx.com/blog/nginx-power-of-two-choices-load-balancing-algorithm/) on 2019-12-12.

6. "Test Driving "Power of Two Random Choices" Load Balancing" (https://web.archive.org/web/20190215173140/https://www.haproxy.com/blog/power-of-two-load-balancing/). haproxy.com. 2019-02-15. Archived from the original (https://www.haproxy.com/blog/power-of-two-load-balancing/) on 2019-02-15.

آخرین ویرایش ۶ ماه پیش توسط Alirezaabarkhordari انجام شده