عنوان مقاله:

بهینه سازی پرس و جو در محیط های توزیع شده

مترجم:

رضا فاریابی

۱۳۹۳-۹۴

چکیده:

بهینه سازی پرس و جو قبل از اجرای پرس و جو مهم است و این اصل در پایگاه داده های توزیع شده اهمیت بسیار بیشتری دارد. در پایگاه داده های توزیع شده با اندازه بزرگ، مسأله بهینه سازی پرس و جو ماهیتی NP-hard پیدا می کند و حل آن بسیار مشکل است. به همین دلیل ما به دنبال روش هایی می‌گردیم تا در زمان کمتر و با هزینه کمتر جوابی نزدیک به بهینه بدهند و در این راستا از الگوریتم ژنتیک(GA) و الگوریتم بهینه سازی کلنی مورچه ها برای محاسبه راه حل بهینه استفاده می کنیم. همچنین برای هر پایگاه داده در این سیستم توزیع شده یک حافظه نهان محلی در نظر می گیریم که تاثیر زیادی در کم کردن زمان پاسخ و کم کردن هزینه ها دارد. اگر پرس و جو تکراری باشد، با استفاده از حافظه نهان پاسخ داده می شود که بسیار سریع تر است و اگر پرس و جو جدید باشد، پردازش می شود و نتایج آن از پایگاه داده برگشت داده می شوند و همچنین حافظه نهان نیز به روز می شود.

فهرست مطالب

فهرست تصاویر

فهرست جداول

مقدمه:

مسأله بهینه سازی پرس و جو ها در پایگاه داده های توزیع شده با اندازه بزرگ ماهیتی NP-hard دارد و حل آن بسیار مشکل می باشد. همانطور که تعداد ارتباط ها و پیوندها در یک پرس و جو افزایش می یابد، پیچیدگی بهینه ساز نیز افزایش می یابد. در اینجا برای به دست آوردن یک الگوریتم مناسب برای جستجوی راه حل بهینه مخصوصا برای زمانی که اندازه پایگاه داده افزایش پیدا می کند تحقیق انجام می دهیم. در این مقاله روش های بهینه سازی مختلفی مورد بررسی قرار گرفته است و مطالعات نشان می دهد که در پایگاه داده های توزیع شده هنگامی که از الگوریتم کلنی مورچه ها به طور متحد با دیگر الگوریتم ها استفاده شده است کارایی بهبود یافته است.

در پایگاه داده های توزیع شده، معمولا هزینه بهینه سازی با افزایش منابع داده زیرساختی در پردازش پرس و جو های بهینه سازی زیاد می شود. ممکن است بهینه ساز پرس و جو منابع داده و ترفند های بهینه سازی مورد نیاز را مدیریت کند تا همه ی هزینه های مربوطه را با کمترین مقدار ارتباط به کار بگیرد.

این را می دانیم که مساله ی بهینه سازی پرس و جو به ارتباطات مانند کاردینالیتی، اندازه تاپل و خُردشدگی تاپل های شرکت کننده در پیوند با دیگر ارتباطات وابسته است. البته به نظر می رسد که مساله بهینه سازی پرس و جو ها به صفات کاردینالیتی قلمرو[1]، تعداد حقیقی مقادیر ناهمگون و فرض های رایج وابسته است. مساله بهینه سازی پرس و جو شامل ترتیب ها و زیرترتیب ها در امتداد با هزینه ترتیب ها و زیرترتیب ها است. در بسیاری از موارد هزینه اجرای پرس و جو ها در واحدهای مفهومی اندازه گیری می شوند. اما در پایگاه داده های توزیع شده، هزینه های اجرای پرس و جو باید به ابعاد مختلفی تقسیم شود. در این مقاله قصد داریم تا به بهینه سازی پرس و جو نزدیک شویم و ترفند های جستجو در محیط های توزیع شده را برشماریم. همانطور که به نظر می رسد انتخاب ترفند بهینه سازی برای اجرای پرس و جو ها NP-hard است. در تعدادی از ارتباطات ممکن است الگوریتم ژنتیک برای انتخاب مقدار، زمان CPU، هزینه ترتیب پرس و جو و زیرترتیب ها شامل هزینه I/O به کار بسته شود. هدف، اندازه گیری هزینه datasetها هنگام برخورد با datasetهای با اندازه بزرگ در محیط های توزیع شده است.

بهینه‌سازی پرس و جو در پایگاه داده های توزیع شده به طور صریح نیازمند جنبه های زیادی از پروسه ی بهینه‌سازی، اغلب تحکیم آن برای بهینه ساز برای منابع داده اساسی هنگام انجام بهینه سازی بر اساس هزینه است. این نه فقط هزینه های بهینه سازی را افزایش می دهد، اما همچنین trade-off درگیر در پردازه بهینه سازی را افزایش می دهد. بهینه ساز فقط می تواند موجب کمی گردی پیام ها به منابع داده under-lying شود و

[1] domain

از اینجا تکنیک های بهینه سازی در این محیط باید ابزاری در جهت گردآوری اطلاعات همه ی هزینه های مورد نیاز با کمترین ارتباط باشد. در این مقاله، یک مدل بهینه سازی پرس و جو بر اساس حافظه ی نهان پیشنهاد شده است که نسبتِ به هدف رسیدن بهتری برای مقداردهی پرس و جو های ساخته شده از زمان حافظه نهان محلی که در عوض حافظه نهان کلی استفاده شده اند نشان می دهد. یک حافظه نهان بین بهینه ساز محلی و پایگاه داده محلی به کار برده شده است. هر زمان که یک پرس و جو به بهینه ساز محلی داده می شود، بهینه ساز محلی ابتدا حافظه نهان ترجیحی را برای واکشی مستقیم داده از پایگاه داده بررسی می کند. در این مورد، اگر پاسخ پرس و جو بتواند از حافظه ی نهان به دست بیاید، این باعث صرفه جویی عظیمی در زمان محاسبه می شود که دسترسی به حافظه نهان سریع تر از دسترسی به پایگاه داده است. مدل بهینه سازی پیشنهادی برای هزینه روی چهار فاکتور مختلف از جمله، دوردستی سرویس دهنده، گنجایش سرویس دهنده، بارگیری(Load) سرویس دهنده و طول صف جاری برای فراهم کردن نقطه بهینه در جایی که پرس و جو باید اجرا شود بستگی دارد.

کلمات کلیدی:

طرح اجرای پرس و جو، پایگاه داده های توزیع شده، الگوریتم کلنی مورچه ها، ترفند های جستجو، بهینه سازی پرس و جو، طرح پرس و جو، کاردینالیتی، هزینه دستورات، هزینه CPU، NP-hard، هزینه I/O، پایگاه داده توزیع شده، بهینه ساز هزینه، بهینه سازی پرس و جو، حافظه نهان، حافظه نهان محلی، بارگیری سرویس دهنده.

فصل ۱

استراتژی های بهینه سازی پرس و جو در پایگاه داده های توزیع شده[*۱]

۱,۱ پیشگفتار

با متصل شدن به شبکه های پرسرعت، مهم ترین پژوهش روی شناخت ترفندهای با کارایی بالا که روی هزینه اثر دارند در پایگاه داده های توزیع شده خواهد بود.

یک پایگاه داده توزیع شده مجموعه ای از پایگاه داده هایی است که زیاد با هم مرتبط هستند و روی کامپیوترهای مختلف یک شبکه برای بهبود کارایی، قابلیت اطمینان، در دسترس بودن و ماژولاریتی سیستم توزیع شده قرار گرفته اند.

پردازش پرس و جو ها در یک محیط توزیع شده بسیار مشکل تر از یک محیط متمرکز است. از آنجا که داده ها از نظر جغرافیایی در مکان های مختلف توزیع شده اند، پردازش پرس و جو ها سختی انتقال اطلاعات از پایگاه های دیگر را نیز دارد. بازیابی داده ها از پایگاه های مختلف با نام "پردازش پرس و جوی توزیع شده"(DQP[۲]) شناخته می شود.

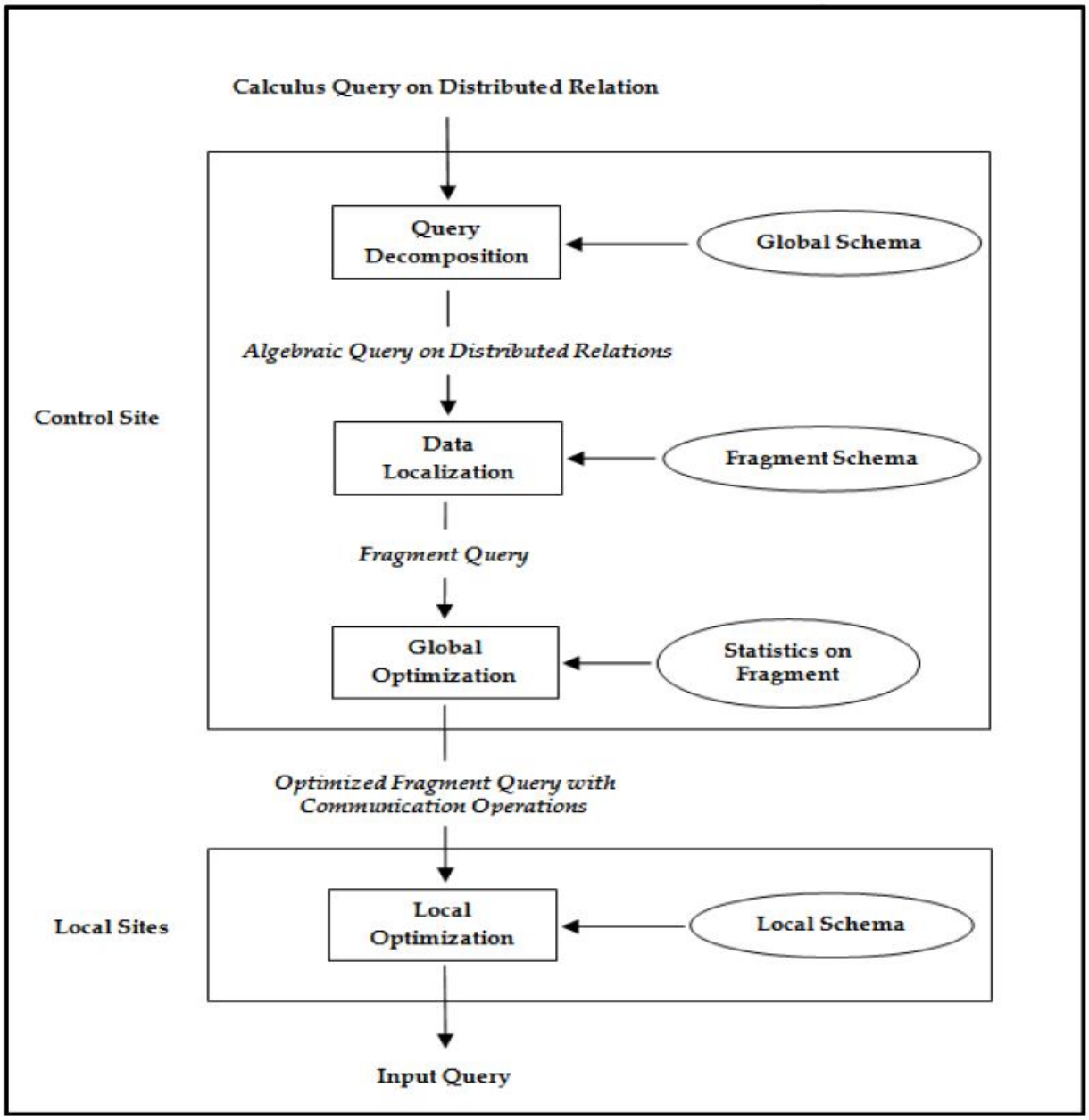
پردازشگر پرس و جو، داده ها را از پایگاه داده هایی که روی پایگاه های چندگانه در یک شبکه قرار گرفته اند روی پردازنده‌های چندگانه پردازش می کند تا یک نتیجه واحد برای پرس و جو به دست بیاورد. سه مرحله پیچیده در پردازش پرس و جو های توزیع شده وجود دارد.

مرحله پردازش محلی[۳] : در این مرحله پرس و جوی جبری روی ارتباطات کلی به بخش های مختلف تغییر داده می شود(تجزیه داده ها) و آماده پردازش توسط پایگاه های مربوطه می شود. (محلی سازی داده ها) مانند انتخاب‌ها[۴] و پرتوهای[۵] محلی.

مرحله کاهش : ترتیبی از پیوندها و نیم پیوندها برای کاهش داده ها استفاده می شود. با توجه به حجم داده هایی که برای به انجام رساندن پیوندها باید منتقل شود، ترتیبی از پیوندها و نیم پیوندها برای کاهش داده ها استفاده می شود که در روال هزینه ها تاثیرگذار است.

پردازش نهایی یا مرحله گردآوری : در این مرحله همه ی فایل های پردازش شده به پایگاهِ گردآور منتقل می‌شوند تا خروجی نهایی تولید شود.

[2] Distributed Query Processing
[3] local
[4] selection
[5] Projection



شکل ۱ : مدل بهینه سازی پرس و جوی توزیع شده

کارایی پرس و جو های توزیع شده به طور بحرانی به بهینه ساز پرس و جو وابسته است که بتواند به طور مؤثر از ترفندهای پردازش پرس و جو استفاده کند. بهینه ساز پرس و جو یکی از مهم ترین و گران ترین عرصه ها در اجرای پرس و جو های توزیع شده است. پیچیدگی بهینه سازی پردازش ها با تعدادی ارجاع به انواع روش های دسترسی به مقداردهی پرس و جو ها و مجموعه ای از قوانین پیچیده برای تولید درخت های پرس و جو یا گراف های پرس و جو مصمم شده است.

هنگامی که کاربر یک پرس و جو وارد می کند، ابتدا به فرم استاندارد ارتباطات جبری تبدیل می شود و بهینه‌ساز، ترتیب اجرای بهینه را جستجو می کند. تعداد ترتیب های فرعی برای اجرا، با افزایش تعداد ارتباطات مورد نیاز برای پردازش پرس و جو به طور نمایی افزایش می یابد. بهینه ساز پرس و جو باید فضای بزرگی را برای تولید ترتیب بهینه پرس و جو جستجو کند.

مسأله بهینه سازی پرس و جو در پایگاه داده های توزیع شده اندازه بزرگ از نوع NP-hard است. در واقعیت حل این مشکل با مرور همه ی راه های پرس و جو در این فضای جستجوی بزرگ امکان پذیر نیست.

این مسأله در پایگاه داده های توزیع شده، یک مسأله بهینه سازی ترکیبی است و با ترفندهای مختلفی مانند simulated annealing و iterative improvement و two-phase optimization و Deterministic و الگوریتم Greedy and Heuristic برای پیدا کردن راه حل بهینه با در نظر گرفتن پیچیدگی زمانی و هزینه برای اجرای این پرس و جو ها مورد مطالعه قرار گرفته است.

در این مقاله سعی شده مطالعه ای روی ترفندهای مختلف جستجو صورت بگیرد که بتواند روش اجرای بهینه پرس و جو را در پرس و جو های توزیع شده مشخص کند.

۱,۲ اجزاء بهینه سازی پرس و جو های توزیع شده

بهینه سازی پرس و جو در محیط های توزیع شده کار مشکلی است زیرا به پارامترهای زیادی مانند مکان داده‌ها، سرعت کانال ارتباطی، فهرست بندی، در دسترس بودن حافظه، اندازه پایگاه داده، نتایج ذخیره سازی میانی، حجم داده هایی که باید منتقل شوند و pipelining وابسته است. قوانین بهینه ساز پرس و جو برای این است تا پرس و جو ها به روشی اجرا شوند تا کمترین هزینه را داشته باشند(QEP[۶]). یک بهینه ساز، یک ماژول نرم افزاری است که بهینه سازی پرس و جو ها را روی سه جزء اصلی و پایه ای پرس و جو ها که جستجوی فضا، جستجوی ترفندها و جستجوی مدل های هزینه است انجام می دهد.

۱,۲,۱ فضای جستجو

این به تولید QEPهای هم معنی اشاره می کند که از یک پرس و جوی ورودی با اِعمال قوانین تبدیل مختلفی که در ترتیب اجرای دستورات با هم متفاوتند ایجاد می شوند. QEPها معمولا به درخت عملگرها یا درخت پیوندها که در آنها عملگرها از انواع مختلف پیوندها یا ضرب های کارتزین هستند اشاره می کنند. این می تواند به صورت یک گراف پرس و جو به نمایش در بیاید، به صورت G=(N,A) که در آن N مجموعه ی گره ها در گراف پرس و جو است و A مجموعه لبه ها است. هر گره مجموعه ای از Base File(BF) را در مشخصات پیوند پرس و جو

[6] Query Execution Plans

مشخص می کند. دو گره با یک لبه به هم متصل شده اند اگر دو فایل همخوانی داشته باشند و پرس و جو آن دو فایل را با هم پیوند کرده باشد، هر گره در پرس و جو، یک وابستگی به مجموعه پایگاه[7] دارد. این گره های برگ نوع نتیجه پردازش محلی را نمایش می دهند و این یک فایل کاهش یافته است. گره ریشه مرحله نهایی را نمایش می دهد و جایی است که نتیجه تولید می شود. هر گره والد از فایل های نتیجه فرزندانش به عنوان ورودی استفاده می کند.

به عنوان مثال: فایل BF0 در پایگاه های S1 و S2 ذخیره شده است.

فایل BF1 در پایگاه S3 ذخیره شده است.

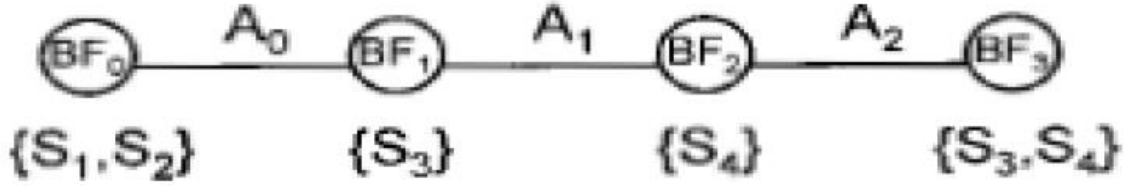
فایل BF2 در پایگاه S4 ذخیره شده است.

و فایل BF3 در پایگاه های S3 و S4 ذخیره شده است.

یک پرس و جو در نظر بگیرید که یک توالی پیوندها روی این ۴ فایل انجام می دهد. BF0 با BF3 ارجاع داده شده است. صفات پیوند A0 را همراه با A2 در نظر می گیرند. (چهار فایل با سه عملیات پیوند، با هم پیوند شده اند.) در SQL این پرس و جو اینگونه مشخص می شود.

```
select A1, <non-join projection attribute list>
from BF0, BF1, BF2, BF3
where BF0.A0 = BF1.A0 and BF1.A1 = BF2.A1 and
BF2.A2 = BF3.A2
```

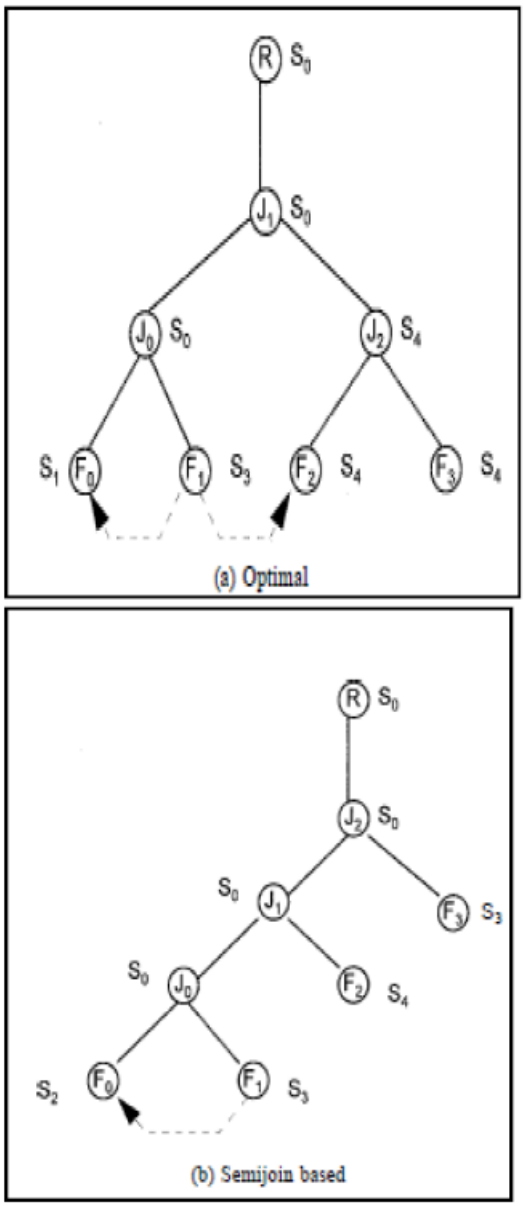
گراف پرس و جو برای این پرس و جوی توزیع شده با مجموعه پایگاه برای هر فایل در زیر نمایش داده شده است.



شکل ۲: گراف پرس و جو برای پرس و جوی توزیع شده

[7] site

از آنجا که گراف پرس و جو مدل سازی گوناگون از عملیات های پیوند به صورت درخت پرس و جوی پیچیده است، QEP برای پرس و جوی داده شده می تواند به صورت زیر نمایش داده شود.



شکل ۳: ترتیب اجرای پرس و جو برای پرس و جوی بالا

در یک پرس و جوی پایگاه داده توزیع شده که تعداد زیادی از ارتباطات و پیوندها را در بر می گیرد به علت وابستگی و جابه‌جاپذیری خواص عملیات پیوند، تعداد QEPها به صورت نمایی افزایش می یابد. (هنگامی که N تعداد ارتباطات باشد) بهینه ساز پرس و جو نیاز دارد تا برای تولید ترتیب بهینه پرس و جو فضای جستجوی بزرگی را بررسی کند.

کارهایی که بهینه ساز پرس و جو انجام می دهد عبارتند از:

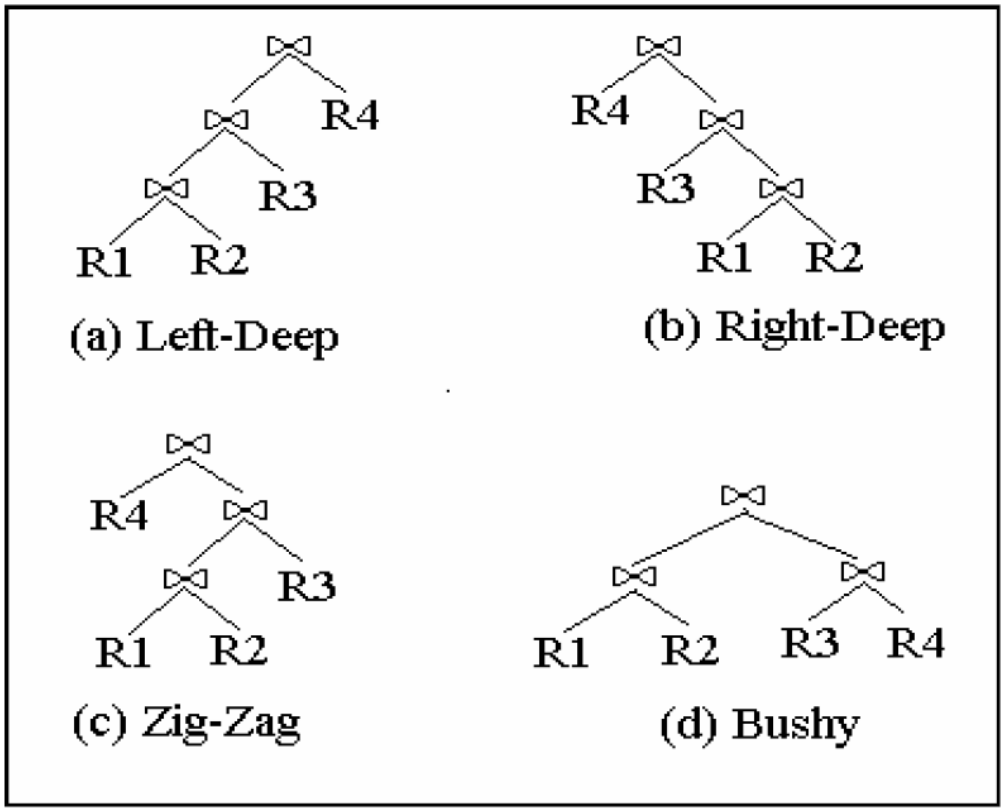
الف) مشخص کردن ترتیب اجرای عملیات پیوند.

ب) مشخص کردن روش های دستیابی عملیات پیوند.

ج) مشخص کردن شکل گراف پرس و جوی پیوند در فضای جستجوی داده شده با به کارگیری ترفندهای جستجو. مانند : اندازه‌گیری کارایی پاسخگویی پرس و جو ها در ترتیب اجرا بهینه است.

این می تواند درخت های خطی باشد، درخت های Left Deep، درخت های Right Deep، درخت های Bushy join، درخت های Binary یا درخت های Zig-Zag.

د) مشخص کردن ترتیب انتقال داده ها بین پایگاه های پیوند[8] برای کم کردن حجم داده های انتقالی و هزینه ارتباطات شبکه.



شکل ۴: شکل گراف های پیوند پرس و جو ها

[8] Join sites

برای بررسی همه ی ترتیب های پرس و جو در این فضای جستجوی بزرگ، ترفندهای جستجو روی یافتن ترتیب بهینه مورد بررسی قرار گرفته اند.

۱,۲,۲ ترفندهای جستجو

این به الگوریتم های بررسی فضای جستجو و مشخص کردن بهترین ترتیب اجرای پرس و جو (QEP) بر پایه‌ی انتخاب پیوندها و پیوند پایگاه ها برای کم کردن هزینه بهینه سازی پرس و جو اشاره می کند. در اینجا به طور پایه ای دو کلاس از ترفندها وجود دارد که مشکل زمانبندی پیوند برای بهینه سازی پرس و جو ها را حل می کند.

اولین پیشنهاد مشخص کردن ترفندی است که با ساخت ترتیب سودآور است، از ارتباطات پایه شروع می کند، یک یا چند ارتباط را پیوند می دهد که هر مرحله تا تمام شود ترتیب ها به دست می آیند.

برای کم کردن هزینه بهینه سازی، ترتیب هایی که راه حل بهینه نمی دهند هرس می شوند[۲].

برنامه‌نویسی پویا همه ی این ترتیب ها را با استفاده از روش BFS[۹] بررسی می کند در حالی که الگوریتم های حریصانه[۱۰] از DFS[۱۱] استفاده می کنند.

پیشنهاد بعدی ترفندهای تصادفی است[۱۱][۱۷]. آنها راه حل بهینه را روی برخی از نقاط جداگانه جستجو می‌کنند. این ترفندها ترتیب بهینه را تضمین نمی کنند اما بهینه سازی بالایی در هزینه ها، Terms of memory و زمان دارند. بهینه سازی بازگشی و شبیه سازی شده الگوریتم های رایجی در این ترفندها هستند.

یکی از سختی هایی که محققان در این زمینه با آن مواجه شدند، انعطاف پذیری است. فضای حل مسأله هنگامی که تعداد ارتباطات و داده های توزیع شده پایگاه های ذخیره سازی افزایش می یابد به صورت نمایی رشد می‌کند. توان بهینه ساز در آدرس دهی برون ریزی ترتیب پیوند، روش پیوند، پرس و جوی کاهش اندازه داده ها و کاهش در هزینه های پرس و جو از سخت ترین کارها هستند.

اگرچه الگوریتم های تکاملی مانند بهینه سازی کلنی مورچه ها، الگوریتم های ژنتیک و بهینه سازی Particle swarm اکنون برای پیدا کردن راه حل های بهینه سازی و بهینه سازی جزئی در پرس و جو های Large join در فضای جستجوی داده شده در پایگاه داده های پیوندی و توزیع شده در حال مطالعه هستند. این الگوریتم ها

[9] Breadth first search
[10] Greedy
[11] Depth first search

به خاطر قابلیت جستجوی کلی، طبیعت قوی، و قابلیت مدیریت به هم آمیختگی های مختلف در بهینه سازی مسائل، به طور فوق العاده ای موفق هستند.

۱,۲,۳ مدل هزینه

واقعیت بهینه سازی پرس و جو در محیط پایگاه داده های توزیع شده، کم کردن هزینه کلی منابع کامپیوتری است. یک بهینه ساز مدل هزینه شامل توابع هزینه برای پیش بینی هزینه متصدی ها و توابع برای تخمین اندازه نتایج است[۱۶] تابع هزینه می تواند با رعایت هریک از موارد "زمان کلی" یا "زمان پاسخ" بیان شود. زمان کلی با احتساب هزینه پردازش محلی[۱۲] (زمان CPU + هزینه I/O)، هزینه ارتباط (زمان آماده سازی پیام + زمان انتقال داده ها) به دست می آید.

به حداقل رسانی زمان کلی به این معنی است که به کارگیری منابع در صورتی افزایش پیدا می کند که کارایی سیستم افزایش پیدا کند. زمان پاسخ به عنوان زمان سپری شده میان مقداردهی و اتمام یک پرس و جو به صورت موازی شده برآورد می شود. در انتقال موازی، زمان پاسخ با افزایش اجرای موازی کمینه می شود[۱۱].

اساسا هزینه یک پرس و جو به اندازه ارتباطات میانه ای که در حین اجرا تولید می شوند و آنچه که باید روی شبکه برای عملیات پرس و جو به یک پایگاه داده دیگر منتقل شود بستگی دارد. تاکید روی تخمین اندازه ارتباطات میانی بر پایه ی ترتیب پیوندها و روش های پیوند برای کم کردن حجم اطلاعات انتقالی و از آنجا کم کردن هزینه کلی و زمان کلی در اجرای پرس و جو های توزیع یافته است.

۱,۳ الگوریتم های راه حل

بخش مرکزی یک بهینه ساز پرس و جو ترفند جستجوی آن یا الگوریتم تعیین شماره آن است. در این بخش تحقیقاتی روی روش های بهینه سازی پرس و جو روی تعدادی از الگوریتم های بهینه سازی در پایگاه داده های توزیع شده صورت گرفته است. یکی از نخستین سیستم های پایگاه داده توزیع شده سیستم SPP-1 بود که برای شبکه های کم سرعت گسترده طراحی شده است که از نیم پیوند[۱۳] برای کم کردن هزینه ها با تولید ترتیب پرس و جو ها بدون در نظر گرفتن پارگی[۱۴] داده ها به صورت افقی یا عمودی در پایگاه داده های توزیع شده استفاده می کند[۱۳]. سیستم R* توسط بهینه ساز پرس و جوی DDB استفاده شده است و همچنین در شبکه های

[12] Local
[13] semijoin
[14] Fragment

سریع تر، پرس و جوی استاتیک دگرگونی پذیر ایجاد کرده است ولی از هیچ کدام از نیم پیوندها یا مدیریت پارگی افقی[۱۵] یا عمودی[۱۶] استفاده نکرده است[۲۱].

Distributed-INGRES ها قادر بودند تا QEPهای پویا را در زمان اجرا روی سریع ترین ارتباطات شبکه ای تولید کنند. نیم پیوند هیچگاه برای کاهش اندازه پرس و جو مورد استفاده قرار نگرفت اما سیستم قادر بود تا پارگی افقی را بدون همتاسازی مدیریت کند. کار زیادی روی تولید حل بهینه برای ترتیب پیوند پرس و جو ها صورت گرفته است. ترفندهای تصادفی شده[۱۸] به طور مشخص هزینه بهینه سازی پرس و جو ها را کم می کنند اما آنها مشکل سربار ثابت دارند و کندتر از فرایند های کاوشی هستند. ترفندهای جبری[۱۹] راه حل های پویا را در زمان اجرا تولید می کنند اما هنگامی که در پرس و جو های توزیع شده تعداد ارتباط ها افزایش پیدا کند آنها پیچیدگی نمایی برای زمان و فضا دارند.

الگوریتم بهینه سازی Two-phase[۲۰] ترکیبی از بهینه سازی موازی و Simulated Annealing است. این یک قدم تصادفی روی راه حل های مختلف فضای جستجو بر می دارد، یک راه حل را بهینه می کند اما سربار بهینه سازی پرس و جو را افزایش می دهد. یک برنامه نویسی پویا بر پایه تولید راه حل برای کمینه سازی مجموع هزینه های ارتباطات و هزینه های پردازش های محلی با مشخص کردن ترتیب بهینه پیوند و ترفندهای پیوند (sort-merge یا nested-loop) و پایگاه های پیوند توسط دانشمندان پیشنهاد شده است. اگرچه آنها وانمود می کنند داده ها باید بدون افزونگی ذخیره شوند[۲۱][۲۲].

"چِن[۱۷]" و "یو[۱۸]" یک الگوریتم کاوشی ارائه کرده اند که ترتیب پیوند و پیوند پایگاه ها با یک قاطعیت که کپی فایل ها پیش انتخاب هستند هنگامی که کپی های چندگانه وجود دارد[۱۴]. تعداد زیادی پژوهش روی فاز کاهش پردازش پرس و جو های توزیع یافته جایی که مقصد پیدا کردن هزینه کمینه ترتیب نیم پیوند است صورت گرفته است که به طور کامل فایل های ارجاع داده شده با پرس و جو را کاهش می دهد. این کار با کاهش اندازه نتایج میانی پرس و جو برای کم کردن هزینه عملیات هم انجام می شود. یک الگوریتم بر پایه برنامه نویسی پویا که نیم‌پیوندها و ترتیب پیوندها و پایگاه های پیوند سود بخش را مشخص می کند توسط دانشمندان کار شده است[۲۳]. دو برداشت جدید در فاز کاهش پایگاه داده های توزیع شده سودآوری نیم پیوندها و ضعف ویژگی های پیوندها پیشنهاد شده است. می زیفینگ[۱۹] و فَن[۲۰] الگوریتم جدیدی بر پایه الگوریتم های بهینه سازی معمول

[15] horizontal
[16] vertical
[17] Chen
[18] Yu
[19] Mi Xifieng
[20] Fan

برای کم کردن مقدار داده های میانی و هزینه ارتباطات شبکه و افزایش کارایی طراحی از آنجا که برنامه نویسی پویا راه حل بهینه سازی ممکنی برای پرس و جو های توزیع شده نیست زیرا پیچیدگی فضا و پیچیدگی توابع هدف آنها بالا است. اکنون ترفندهای جدیدی مانند الگوریتم های ژنتیک، الگوریتم کلنی مورچه ها و الگوریتم های ترکیبی در حال مطالعه برای پرس و جو های توزیع شده هستند. الگوریتم های ژنتیک از خانواده مدل های محاسباتی هستند و از فرایند تکامل طبیعی ایده گرفته شده اند. معنی کلی الگوریتم GA توسط John Holland پیشنهاد شده است که به طور تصادفی جواب هایی برای مسأله ایجاد می کند که کروموزوم ها هستند و این کروموزوم ها اجازه دارند تا مجموعه جدیدی از فرزندان را با کروموزوم های بهتر با استفاده از عملیات همبری کروموزوم ها و جهش ایجاد کنند[۲۸].

این الگوریتم همچنین قادر است هزینه درخت پرس و جوی توزیع شده را کم کند.

الگوریتم ACO[۲۱] یک الگوریتم کاوشی مناسب مسائل بهینه سازی ترکیبی است، مانند بهینه سازی پرس و جو در پایگاه داده های توزیع شده، به خاطر خصوصیت مشابه تکنیک های جستجوی هوشمند، بهینه سازی کلی، قدرتمندی، محاسبات توزیع شده و قابلیت گردآوری با روش های دیگر[۳۱]. روش ACO برای اولین بار توسط سه دانشمند ایتالیایی به نام دوریگوم[۲۲]، کولورنی‌اِی[۲۳] و مانیزو وی[۲۴] در سال ۱۹۹۲ پیشنهاد شده است این یک الگوریتم بهینه سازی بیولوژیکی است که از زندگی مورچه ها ایده گرفته شده است و از روش های احتمالی برای حل مسائل محاسباتی استفاده می کند. این بر مکانیزم بازخورد مثبت ساخته شده است، بنابراین خیلی نیرومند است، جستجوی هوشمند فراهم می کند و می تواند برای مسائل بهینه سازی کلی استفاده شود[۳۲].

علاوه بر این الگوریتم بهینه سازی کلنی مورچه ها مَنِشی خاص، مشابه محاسبات توزیع شده دارد و طبیعتاً قدرتمند است و مکانیزم بازخورد مثبت دارد. الگوریتم ACO یک سری نقص نیز دارد.

الف) مقداردهی اولیه ACO هیچ روش سازمان یافته ای برای شروع ندارد.

ب) سرعت همگرایی ACO ابتدا خیلی کمتر است اما سرعت همگرایی به سوی پاسخ بهینه به خاطر مکانیزم بازخورد های مثبت افزایش می یابد.

[21] Ant Colony Optimization
[22] Dorigom
[23] Colorni A
[24] Maniezzo V

در دستور کار دانشمندان یک الگوریتم ژنتیک(GA) بر پایه بهینه سازی پرس و جو که NGA نامیده می شود قرار گرفت که QEPهای داده شده را با در نظر گرفتن همه ی جوانب ترتیب پیوندهای پایگاه های پیوند و نیم پیوندها بهبود می بخشد.

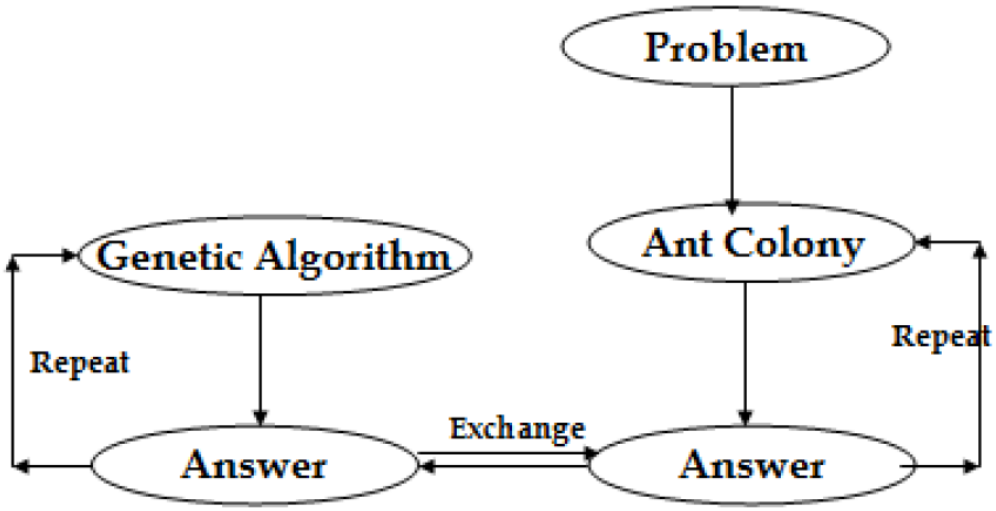
این الگوریتم قادر بود که هزینه پردازش های محلی و هزینه ارتباطات شبکه را با مقادیر جدید جهش و همبری و جستجوهای فراگیر کاهش دهد. قدرت الگوریتم ژنتیک برای بهینه سازی پرس و جو های توزیع شده روی مسائل کم کردن همه ی موارد در یک پرس و جو روی ساختار درخت داده در رأس قرار دارد[۲۶]. یک الگوریتم اتوماتیک سازی از الگوریتم ژنتیک و یادگیری های ماشینی[۲۷] برای تولید ترتیب اجرای پرس و جو های بهینه بر پایه ترتیب اجرای پیوند و انتخاب پایگاه پیوند در پایگاه داده توزیع شده توسط دانشمندان پیشنهاد شده است.

همچنین در اهداف نویسنده[۲۸]، همبری الگوریتم ژنتیک و عملیات کاوشی برای حل مسائل ترتیب پیوند مانند فروشنده دوره گرد در پایگاه داده های بسیار بزرگ و آزمایش های محاسباتی آن را بهبود داد تا همچنین به عنوان راه حلی موفق برای سیستم های توزیع شده نیز باشد. ترکیب الگوریتم بهینه سازی ژنتیک و بهترین-بدترین-کلنی‌مورچه‌ها(BWACO)[۲۵] [۲۹] ابزاری شد برای پیدا کردن ترتیب اجرای بهینه پرس و جو و ترتیب پیوند با کم کردن زمان اجرای پرس و جو برای بهینه سازی پرس و جو های چند پیونده[۲۶] در پایگاه داده. این الگوریتم از مکانیزم بازخورد مثبت الگوریتم کلنی مورچه ها با قابلیت جستجوی کلی الگوریتم ژنتیک بهره برده است.

نوع دیگری از ترکیب الگوریتم ژنتیک و کلنی مورچه ها[۳۰] در مسائل ترتیب پیوند در پایگاه داده (فقط حلقه حلقه کردن پیوندها در نظر گرفته شده است) با غلبه بر کاستی های هر دو الگوریتم به کار برده شده است. این الگوریتم از الگوریتم ژنتیک برای دادن فرومون به توزیع کمک می گیرد، سپس با استفاده از الگوریتم کلنی برای موشکافی راه حل استفاده می کند.

از قابلیت های الگوریتم ترکیبی GA-ACO در جستجوی دامنه گسترده ای برای پاسخ دادن پرس و جو های پیوند در پایگاه داده های مرتبط می توان برای بهینه سازی پرس و جو های پیوند در پایگاه داده های توزیع شده، زمانی که مسأله تولید بهترین QEP برای نتایج بهینه است استفاده کرد. روهت.آل[۲۷] [۸] یک الگوریتم ژنتیک برای مشخص کردن سریع QEP بهینه بر پایه ی شیوه حل پیشنهاد کرده است. این مدل شامل شناسایی کپی(افزونگی داده)، شناسایی نیم پیوندهای سودمند، انتخاب پایگاه پیوند، ترتیب اجرای پیوند و هزینه پردازش محلی و هزینه ارتباط است.

[25] Best-Worst Ant Colony
[26] MultiJoin
[27] Rhoet.al



شکل ۵: نموداری برای ترکیب GA-ACO [۳۰]

تانسل ای‌تی.ال[۲۸] [۳۳] الگوریتم بهینه سازی DP-ACO[۲۹] را برای بهینه سازی پرس و جو های چند راهه[۳۰] در محیط پایگاه داده های توزیع شده پیشنهاد داده است.

برنامه نویسی پویا از زمان اجرای طولانی و نیاز به حافظه بالا بر اساس اندازه ارتباطات و تعداد پیوندها در پرس و جو ها رنج می برد.

DP-ACO با موفقیت برای حل تولید ترتیب خوب برای اجرا با ۱۵ روش پیوند پرس و جو ها با زمان محدود و فضای حافظه بسیار محدود امتحان شده است. برتری دیگر DP-ACO این است که می تواند به آسانی با بهینه سازان پرس و جوی موجود که بر پایه ی الگوریتم های DP کار می کنند سازگار شود.

با استفاده از ویژگی های الگوریتم کلنی مورچه ها و بهینه سازی Particle Swarm، یک الگوریتم ترکیبی برای حل مسائل فروشنده دوره گرد پیشنهاد می شود [۳۵] الگوریتم ابتدا از روش آماری برای گرفتن چندمین شروع بهتر حل و مطابقت آنها، اطلاعات فرومون ها را برای توزیع می دهد، سپس از الگوریتم کلنی مورچه ها استفاده می کند تا چندین حل بر اساس اطلاعات انباشتگی یا بازآغازی فرومون ها به دست بیاورد. در نهایت با استفاده از عملیات عرضی و جهش بهینه سازی Particle Swarm راه حل مؤثر به دست می آید. الگوریتم ترکیبی ACO-PSO توسعه داده شده است تا مؤثر واقع شود.

[28] Tansel et.al
[29] Dynamic Programming- Ant Colony Optimization
[30] Multiway

با افزایش تعداد ارتباطات در یک پرس و جو، استفاده بیشتری از حافظه و پردازشگر، مورد نیاز است. هم اکنون DDBMS به عنوان یک DBMS استاندارد در همه ی برنامه های بازرگانی که داده هایی از پایگاه های مختلف در بر می گیرند استفاده می شود. رفتار علامت گذاری مسیر در مورچه ها باعث می شود مورچه ها بخش های بررسی نشده را نیز جستجو کنند و بدون دانستن توپولوژی گرافیکی، همه ی گره ها را ملاقات کنند تا حل بهینه را برای پرس و جو های پایگاه داده های توزیع شده تولید کنند. این مورچه ها زمان اجرای ترتیب های مختلف در پرس و جوی داده شده را محاسبه و نتایج سریع، با کارایی بالا و بهینه را در یک روش مؤثر در هزینه را محاسبه می کنند.

ترفندهای جستجوی اقتباس شده با بهینه ساز پرس و جو در سیستم های مدیریت پایگاه های داده های توزیع شده می تواند کمک کند تا زمان و هزینه اجرای پرس و جو ها را کم کند و از اینجا با انتخاب بهترین ترتیب برای اجرای پرس و جو ها کارایی را افزایش دهد. به کارگیری این الگوریتم های احتمالی راه حل های زیستی برای زمانی که اندازه پرس و جو و تعداد پیوندها افزایش پیدا می کند را توسعه داده است.

فصل ۲

به کارگیری ترفندهای بهینه سازی پرس و جو در محیط های توزیع شده به کمک الگوریتم ژنتیک[*۲]

۲,۱ پیشگفتار

تکنولوژی پایگاه داده های توزیع شده شامل تکمیل شِما، انتقال داده ها، پردازش پرس و جو های توزیع شده و بهینه سازی پرس و جو است. معمولا پردازش پرس و جو های توزیع شده به سه پایه نیاز دارند:

(۱) نیاز به پردازش پرس و جو: در یک سیستم توزیع شده با اندازه بزرگ، هر دو عمل دسترسی به داده ها و محاسبه ممکن است بین پایگاه های مختلف منتقل شوند.

(۲) نیاز به کارگزارهای هزینه: در DBMS های متمرکز، هزینه اجرای پرس و جو وابسته به واحد کارگزاران اندازه گیری مفهومی است. نویسنده مشابه، میشرا سامبیت کومار[۳۱]: در یک پایگاه داده توزیع شده، هزینه ها باید به ابعاد چندگانه تحت کنترل پایگاه داده های منطقی تکی تقسیم شوند.

(۳) نیاز به تخمین هزینه: به کارگیری مکانیزم پردازش بهینه سازی پرس و جو با نیاز تخمین هزینه در بخشی از بهینه ساز پرس و جو برانگیخته شده است. معمولا الگوریتم های بهینه سازی به سه قدم تقسیم می شوند.

قدم۱: انتخاب زیر طرح هایی که نیازمند تخمین هزینه هستند.

قدم۲: ارزیابی اندازه طرح ها و زیرطرح ها

قدم۳: محاسبه هزینه برای طرح ها یا زیرطرح ها. ارزیابی اجرای طرح ها برای پرس و جو.

برای کم کردن بازیابی پایگاه داده های بزرگ، اندازه گیری هزینه dataset های مورد نیاز ضروری است. هنگام بهینه سازی پرس و جو های چندگانه در یک زمان در یک محیط توزیع شده، مشاهده شده است که با تعداد بیشتری پرس و جو های مشابه موثرتر است.

مساله بهینه سازی پرس و جو روی ارتباطات وابسته است. مثلا برونش ناشی از پیچیدگی کاردینالیتی، اندازه تاپل‌ها و تکه تکه شدگی شرکت تاپل ها در یک پیوند با ارتباط پیوند دیگر همچنین مسأله بهینه سازی روی صفات متشکل با کاردینالیتی قلمرو، تعداد واقعی مقادیر ناهمگون و پندارهای رایج مانند استقلال میان مقادیر صفات مختلف و توزیع یکنواخت مقادیر صفات در قلمرو آنها وابسته است.

در حالت کلی ممکن است پنج نیازمندی برای پرس و جو های توزیع شده به وجود بیاید:

[31] Mishra sambit kumar

(۱) حفظ منابع: از آنجا که روی تعداد سطر هایی که یک پرس و جو ممکن است برگرداند هیچ محدودیتی وجود ندارد، منابع باید به طور واضح برای برآوردن امکان مجموعه نامحدود برای نتایج حفاظت شوند. حفاظت منابع، روی سرویس دهنده با نگه نداشتن روی هیچ پایگاه داده ای به انجام رسیده است.

(۲) پاسخ گویی در یک زمان منطقی: پرس و جو باید به اندازه یک پرس و جوی محلی زمان ببرد(با حجم مشابه) این باید با اولین سطر(های) اطلاعات، مشابه یک پرس و جوی محلی پاسخ دهد.

(۳) نیرومندی: دو مدل نیرومندی وجود دارد. مدل اول به نتایج بازگشتی مرتبط است. تا زمانی که سرویس گیرنده در حال انجام کار روی مجموعه نتایج است، آنها دیگر کاربران با نتایج برگشتی را نمی خواهند. مدل دومِ نیرومندی به منابع دور اشاره می کند. اگر اتصال شبکه به هر دلیلی قطع شد، اتکا روی خطاهای مبهم برای مدیریت منابع سرویس دهنده به طور واضح ناخوشایند است. اگرچه منابع سیستم از زمان درخواست های سرویس گیرنده بازنگاه داری نشده اند.

(۴) مجموعه نتایج جزئی اجازه داده نشده: این نیازمندی به سادگی مشخص می کند هر روش برای بازیابی اطلاعات، نباید تعداد سطر های برگشتی را محدود کند. همه ی سطر های پرس و جوی داده باید از سرور بازگشت داده شوند.

(۵) درون یافتی برای استفاده: توسعه دهنده کلاینت باید بازیابی داده ها را مشابه یا بهتر از شیوه پرس و جو های محلی فعلی پیدا کند. علاوه بر این، سرویس گیرنده ای که از روش پرس و جو استفاده می کند نباید نیازمند ساخت اجازه برای حقایقی باشد که آن پرس و جوی دور[۳۲] است. آنها باید قادر باشند تا از کد سرویس گیرنده مشابهی برای پرس و جو های محلی و پرس و جو های دور استفاده کنند.

۲,۲ بررسی نوشتار های وابسته

لِنزِلوت[۳۳] در مقاله اش درباره ترفندهای شمارشی کافی هنگام بهینه سازی پرس و جو های پیچیده مناسب برای تعداد زیادی طرح های اجرایی بحث کرده است. برای حل این مشکل جزئی، ترفندهای تصادفی مورد استفاده قرار گرفتند. به طور کلی، ناهماهنگی و خودمختاری منابع داده، سیستم های یکی سازی داده ها را مشخص می کند. منابع ممکن است درخور محدودیت واسط پرس و جو یا صفات خاص باید درخور دلایل محرمانگی مخفی شوند. برای مدیریت قابلیت های پرس و جوی محدود شده منابع داده، مانلِسکو[۳۴] اپراتور پیوند وابسته را معرفی کرد که در حقیقت نا متقارن است.

[32] Remote query
[33] Lenzelotte
[34] Manolescu

اُسزو[35] در مقاله اش بحث کرده است که هدف بهینه سازی پرس و جو پیدا کردن ترفند اجرای نزدیک به بهینه برای پرس و جو ها است. یک ترفند اجرا برای یک پرس و جوی توزیع شده ممکن است با عملوندهای "جبر ارتباطی" و ارتباط های اولیه برای انتقال داده توصیف شود. اُسزو در مقاله اش بحث کرده است که انتخاب ترفند بهینه، به طور کلی نیازمند اسناد هزینه اجرای حل کننده های داوطلب برای اجرای پرس و جو است. سنگینی ترکیب I/O، CPU و هزینه های ارتباطی.

اُنیدیس[36] در مقاله اش درباره تولید کد برای راه های اجرای پرس و جو بحث کرده است که ممکن است به حالت کامپایل شده[37] و یا سطر به سطر[38] برای تولید نتایج پرس و جو اجرا شوند.

کاسمن[39] در مقاله اش درباره استفاده جزئی از فهرست ها با استفاده از بهینه ساز برای اجرای یک پرس و جو و اینکه در چه ترتیبی از عملگرها باید اجرا شوند بحث کرده است. بهینه ساز راه های فرعی را بر می‌شمارد، با استفاده از مدل هزینه، هزینه هر روش را تخمین می زند و طرحی که کمترین هزینه دارد را بر می گزیند.

ایبراراکیت[40] در مقاله اش بحث کرده است که انتخاب ترفند بهینه برای یک پرس و جو در تعدادی از ارتباطات از نوع NP-hard است. برای مثال: پرس و جو های با تعداد ارتباطات زیاد، هزینه های هنگفت بهینه سازی را تحمیل می کند، اگرچه مأموریت حقیقی بهینه ساز، پیدا کردن ترفند نزدیک به بهینه و اجتناب از ترفندهای بد است.

جی.گرایفی[41] در مقاله اش بحث کرده است که اگرچه پردازش پرس و جو های توزیع شده یک مسأله Well-Studied است، معماری های مدرن چالش ها و فرصت های جدیدی را برای موازی سازی Fine-grained در همه ی سطوح اجرا از سطح intra-operator تا سطح کار با دستگاه مطرح می کند.

اُ.گُرلیتز[42] در مقاله اش بحث کرده است که بهینه سازی ممکن است به دو بخش کلی بهینه سازی محلی برای انتخاب ترفندهای دسترسی و بهینه سازی کلی[43] در جایی که ترتیب پیوند و انتخاب پایگاه[44]، کارهای اصلی هستند برای انتخاب پایگاه، دو گزینه اصلی وجود دارد: ترابری داده ها جایی که داده از پایگاه ذخیره سازی به

[35] Oszu
[36] Ioannidis
[37] compiled
[38] interpreted
[39] Kossmann
[40] Ibrarakit
[41] G.Graefe
[42] O.Gorlitz
[43] global
[44] site

پایگاه اجرای پرس و جو منتقل شده است و ترابری پرس و جو جایی که ارزیابی "زیر پرس و جو" به نمایندگی پایگاه ذخیره سازی است.

اظهارات مسأله

۱- مثال

```
SELECT ENAME
FROM EMP,ASG
WHERE EMP.ENO = ASG.ENO
AND DUR > 37
```

ترفند۱

ΠENAME(σDUR>37$\cap$EMP.ENO=ASG.ENO (EMP X ASG))

ترفند۲

ΠENAME(EMP X ENO (σDUR>37 (ASG)))

در این مورد، ترفند۲ از پیامدهای کارتزین اجتناب می کند. برای کمینه کردن تابع هزینه، هزینه I/O، هزینه CPU و هزینه ارتباط طرح پرس و جو باید محاسبه شود. طرح پرس و جو ممکن است در محیط های توزیع شده مختلف سنگینی های متفاوتی داشته باشد.

هزینه کلی طرح های پرس و جو = هزینه CPU + هزینه I/O + هزینه ارتباط

هزینه CPU = هزینه هر واحد دستورات*تعداد دستورات

هزینه I/O = هزینه هر واحد از I/O دیسک*تعداد I/Oهای دیسک

هزینه ارتباط = راه اندازی پیام + انتقال

زمان سپری شده بین مقداردهی و اتمام یک پرس و جو ممکن است با محاسبه زمان پاسخ، زمان I/O و زمان ارتباط برآورده شود.

زمان پاسخ = زمان CPU + زمان I/O + زمان ارتباط

زمان CPU = زمان هر واحد دستور*تعداد دستورات سری

زمان I/O = زمان هر واحد I/O * تعداد I/Oهای سری

زمان ارتباط = زمان مقداردهی هر واحد پیام * تعداد پیام های سری + زمان انتقال هر واحد * تعداد بایت های سری

۲,۳ پیچیدگی عملکردهای وابسته

ارتباطات کاردینالیتی n با پویش سریال را در نظر بگیرید.

جدول۱: پیچیدگی(عملیات وابسته)

Operation	**Complexity**
Select	O(n)
Project (without duplicate elimination)	
Project (with duplicate elimination)	O(nlog n)
Group	
Join	O(nlog n)
Semi-join	
Division	
Set Operators	
Cartesian Product	$O(n^2)$

هنگام حرکت دادن ارتباط داخلی به پایگاه ارتباط خارجی، ممکن است پیوند در لحظه‌ی ورود ممکن نباشد. برای ذخیره کردن هزینه کل تاپل های خارجی، ممکن است تشبیه کردن به تاپل های داخلی مورد نیاز باشد.

هزینه کل = هزینه(بازیابی تاپل های خارجی واجد شرایط) + تعداد تاپل های خارجی واکشی شده * هزینه(بازیابی تاپل های داخلی مشابه از حافظه ذخیره سازی موقت) + هزینه(بازیابی تاپل های داخلی مناسب) + هزینه(مرتب سازی همه ی تاپل های داخلی مناسب در حافظه موقت) + هزینه پیام * (تعداد تاپل های داخلی واکشی شده * میانگین اندازه تاپل داخلی) \ اندازه پیام.

۲,۴ مواد و روش ها

این مسأله ممکن است به عنوان یک مسأله الگوریتم ژنتیک نشان داده شده باشد، جایی که این برای مشخص کردن کروموزوم ها ، عملگرهای الگوریتم ژنتیک در نظر گرفته شده است، روش پرس و جوی شخصی ممکن است

به عنوان کروموزوم نمایش داده شود و زیر روش ها یا کارها ممکن است به عنوان ژن نمایش داده شوند. برای همبری، ممکن است یک نقطه در کروموزوم دیگر در نظر گرفته شود و سپس دنباله ی آن ممکن است عوض شود. پردازه جهش، برخی از بیت ها را برعکس می کند و اطلاعات جدیدی را به وجود می آورد. اگرچه بهترین سلایق ممکن است به جلو برای تولید بعدی پردازش شوند. بعد از به وجود آوردن عملیات ها، برخی از کروموزوم ها ممکن است در مرحله fitness راضی کننده نباشند و به عنوان یک نتیجه، الگوریتم این پردازه ها را دور می اندازد و q که (q<=n) است را فرزند کروموزوم ها می گیرد. سپس الگوریتم n کروموزوم با کمترین مقادیر شایستگی را از n + q کروموزوم انتخاب می کند(q فرزند و n والد) تا والد نسل های بعدی باشند. این پردازش تا زمانی که تعداد خاصی نسل تولید شوند ادامه می یابد، سپس بهترین کروموزوم انتخاب می شود.

۲,۵ فرمول بندی مسأله

۲,۵,۱ الگوریتم

قدم۱ : بیشترین مقدار تولید و ارتباط را تنظیم کن.

قدم۲ : بیشترین تعداد ارتباط را تنظیم کن.

قدم۳ : بیشترین تعداد پرس و جو را نسبت بده.

قدم۴ : طول مورد نیاز طرح پرس و جو را که طرح پرس و جو می تواند به عنوان یک کروموزوم باشد را نسبت بده.

قدم۵ : با استفاده از تعداد پرس و جو ها و طرح های پرس و جو، جمعیت را محاسبه کن.

قدم۶ : احتمال همبری را تنظیم کن(pc).

قدم۷ : احتمال جهش را تنظیم کن(pm).

قدم۸ : زمان عملیات I/O را تعیین کن.

قدم۹ : طرح انتخاب مقدار با گرفتن تعداد پرس و جو ها و طرح های پرس و جو در ضمانت اجرا را محاسبه کن.

قدم۱۰ : زمان CPU در طول پردازش را محاسبه کن.

قدم۱۱: تخمین هزینه طرح را با در نظر گرفتن طرح انتخاب مقدار، تعداد کلی پرس و جو ها در طول زمان و تعداد ارتباطات محاسبه کن.

قدم۱۲ : هزینه کلی طرح با در نظر گرفتن هزینه CPU، هزینه I/O و هزینه ارتباط در حساب را محاسبه کن. هزینه CPU ممکن است با ضرب هزینه واحد دستور با تعداد دستورات بدست بیاید. به طور مشابه هزینه I/O ممکن است با ضرب واحد هزینه I/O دیسک در تعداد I/Oهای دیسک بدست بیاید.

۲,۵,۲ تجزیه و تحلیل های آزمایشی

حداکثر فرآوری[۴۵] =۱۰۰

تعداد پرس و جو ها = ۱۰۰

تعداد ارتباطات = ۱۰۰

اندازه کروموزوم(طرح در یک پرس و جو) = ۵

احتمال همبری(pc) = ۰,۰۷

احتمال جهش(pm) = ۰,۰۰۲

جدول۲ : طول طرح پرس و جو یا هزینه

Sl.No.	Query Plan	Est_Cost	Total_Cost
1	7	0.018344	0.36288
2	15	0.018344	0.36688
3	18	0.018419	0.36838
4	24	0.018569	0.37138
5	25	0.018144	0.37188

[45] generation

۲,۶ مذاکره و رهنمون آینده

ما همچنین درباره ترفندهای بهینه سازی در شرایط احتساب هزینه طرح های پرس و جو در طول الگوریتم بحث کردیم. ممکن است اطلاعات در میان مجموعه استاندارد از نتایج در دسترس باشد که امکان پرس و جوی پایگاه داده اساسی را درباره آمار می دهد، یا با آمار نهان[۴۶] از پرس و جوی قبلی اجرا می شود.

معمولا هزینه های ارتباطات در طول بهینه سازی و اجرای پرس و جو باقی می مانند. کارها یا زیر طرح ها ممکن است برخی هزینه های همبسته داشته باشند که آنها هر دو هزینه های CPU و I/O مورد نیاز برای پردازش آنها را تحت الشعاع قرار می دهند. اینطور به نظر می‌رسد که هزینه ی دسترسی به یک طرح، هزینه ی پردازش کارهای اجزای آن است. در یک محیط توزیع شده، یک طرح دسترسی کلی برای مجموعه ای از پرس و جو ها با طرحی که راهی برای محاسبه قوانین همه ی پرس و جو ها فراهم می کند همخوان است. در این مورد، طرح دسترسی ممکن است با انتخاب یک طرح برای هر پرس و جو ساخته شود و سپس آنها با هم ادغام شوند. این به طور شفاف فهمیده می شود که این نوع مسأله، NP-hard است. یک الگوریتم غیر قطعی فقط یک طرح برای هر پرس و جو نیاز دارد و اینکه بررسی کند آیا هزینه طرح دسترسی کلی کسب شده با ادغام دسترسی محلی حدس زده شده کمتر یا برابر دسترسی ادغام شده طرح ها است که می توانند به آسانی در زمان چند فرمولی انجام شوند و سپس قدم های بررسی فقط زمان چند فرمولی طول می کشند.

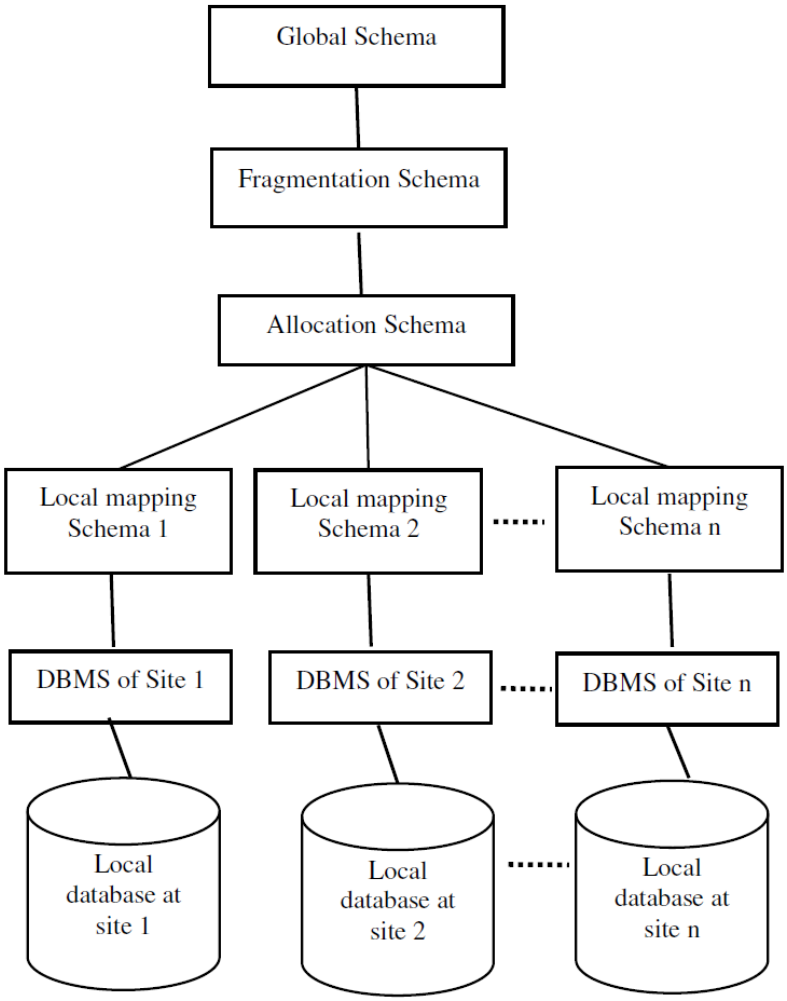
[46] cache

فصل ۳

نزدیک شدن بهینه سازی پرس و جو بر اساس حافظه نهان در پایگاه داده های توزیع شده[۳*]

۳,۱ پیشگفتار

در جهان همگانی وابسته روی سیستم های اطلاعاتی، مردم می خواهند به پایگاه داده های نقاط مختلف جهان دسترسی پیدا کنند. شرکت ها نیز می خواهند تجارتشان را در سطح جهانی توسعه دهند. در خور نظام تجارت جهانی، پایگاه داده های توزیع شده بسیار مشهور هستند و بیشتر در سطح جهان استفاده می شوند. یک پایگاه داده توزیع شده یک پایگاه داده در هر کدام از دستگاه های ذخیره سازی است که در واحدهای پردازش رایج مانند CPU با هم مشترک نیستند. این ممکن است در چند کامپیوتر که مکان فیزیکی یکسانی دارند باشند و یا روی یک شبکه روی کامپیوترهای متصل به هم، پخش شده باشند[۴۵].



شکل ۶: معماری سیستم پایگاه داده توزیع شده

مدیریت پردازش پرس و جو بسیار پیچیده می شود و زمان پردازش آن زیاد می شود، بنابراین، پردازش پرس و جو یک بحث کلیدی در سیستم های پایگاه داده های توزیع شده است. شکل۶ گردآوری داده ها را نمایش می‌دهد

(در یک پایگاه داده) این می تواند در مکان های فیزیکی چندگانه توزیع شده باشد، یک پایگاه داده توزیع شده می تواند روی سرویس دهنده های اینترنت ماندگار باشد، روی اینترانت ها یا اکسترانت ها یا دیگر شبکه های شرکتی. همتایی و توزیع شدگی پایگاه داده ها، کارایی پایگاه داده را در پایگاه های کاری کاربران بهبود می بخشد. بهینه سازی پرس و جو یک عملیات از سیستم های وابسته مدیر یک پایگاه داده است. در هر طرح، پرس و جوی چندگانه برای ایفا کردن یک پرس و جو بازرسی می شود و یک طرح خوب برای پرس و جو شناسایی می شود. این ممکن است بهترین ترفند باشد و یا نباشد. زیر روش های زیادی برای اجرای طرح ها وجود دارد[۴۸][۵۳]. یک trade-off میان زمان گذشته شده برای ایجاد طرح و مجموع اجرای طرح وجود دارد. مدل های مختلف سیستم های مدیریت پایگاه داده روش های مختلفی برای هم تراز کردن این دو عامل دارند. بهینه سازهای پرس و جویی که بر پایه هزینه هستند جای گیری منابع طرح های مختلف پرس و جو را محاسبه می کنند و از این به عنوان پایه ای برای انتخاب طرح استفاده می کنند[۴۹].

تکیه‌گاه این بخش به صورت زیر سازمان دهی شده است. در بخش۳,۲، ما نوشتار های وابسته را بررسی می کنیم، در بخش۳,۳، مدل بهینه سازی پرس و جوی پیشنهاد شده در بخش۳,۴ بر اساس هزینه نمایش داده شده است. در نهایت، در بخش۳,۵، الگوریتم بهینه سازی پرس و جو و قوانین آزمایشی مورد بحث قرار گرفته اند.

۳,۲ بررسی آثار وابسته

به عقیده اسواتی گوپتا[۴۷][۴۵] سیستم های پایگاه داده توزیع شده یک توسعه روی ارتباطات و پردازش داده ها، متناسب با توزیع شدگی داده هایش میان پایگاه های مختلف شبکه فراهم می کند. نه تنها دسترسی به داده ها سریع تر است، احتمال رخداد یک نقطه ی شکست[۴۸] نیز کمتر است و این امکان کنترل داده های را به صورت محلی برای کاربران فراهم می کند[۶۷].

فَن یانیان[۴۹][۴۶] و رضا قائمی[۵۰] [۴۷] تاکید کرده اند که متناسب با اپلیکیشن های پایگاه داده های توزیع شده پیچیدگی جستجو به طور ثابت افزایش پیدا می کند، مانند سیستم های عظیم پایگاه داده های استنتاجی[۵۱] و ما به الگوریتم های بهتری برای سرعت بخشیدن به پرس و جو های سنتی پایگاه های داده نیاز داریم. یک روش بهینه برنامه نویسی پویا برای پرس و جو های با ابعاد بالا کاستی ترتیب نمایی دارد و بنابراین ما علاقه‌مند به روش‌های مشابه بهینه و سریع تر هستیم.

[47] Swati Gupta
[48] Single-Point of failure
[49] Fan Yuanyuan
[50] Reza Ghaemi
[51] deductive

برتری استفاده از حافظه نهان در بهینه سازی پرس و جو در کاهش بار گذاری محتواهای یکسان مکرر از سرویس دهنده است. رزرو کردن گنجایش سرویس دهنده برای دیگر درخواست های غیر قابل استفاده در حافظه نهان برای سرویس گیرندگان یا کاربران و هزینه های اساسی ممکن است به خوبی صرفه جویی عملیاتی توسط بهینه ساز سرویس دهنده به کار گرفته شود.

دی.کاسمن[۵۲] [۵۰] توضیح داده است که سرمایه گذاری روی جزء حافظه نهان چگونه می تواند صورت بگیرد و بدون تغییر در اجزاء اصلی مانند طرح محیط، ترفند جستجو یا مدل هزینه، با بهینه ساز پرس و جو ادغام شود.

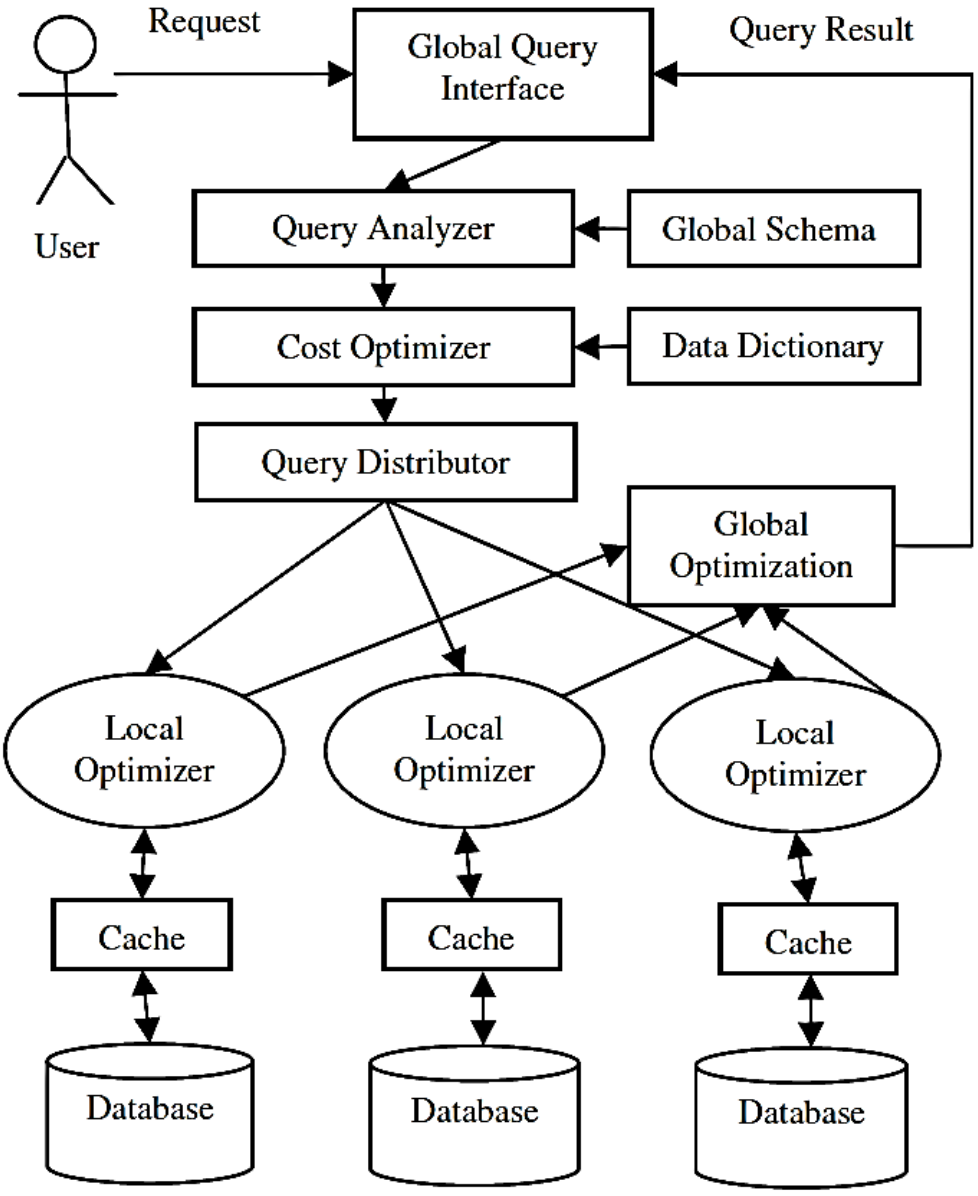
اس.ادالی[۵۳] [۴۸] از حافظه نهان توزیع شده استفاده می کند که ترافیک شبکه را به ازدحام کمتر جریان می دهد و همچنین اجازه اشتراک بار گذاری بهتری با پوشش خطای بیشتر می دهد.

کیو.لو[۵۴] [۵۶] حافظه های نهان پایگاه داده که در برنامه ی سرویس دهنده توسعه داده شده اند را توضیح می‌دهد. هنگامی که اظهارات SQL دلخواهی از یک درخواست، با استفاده از اپلیکیشنی تولید می شوند که برای پشتوانه سرویس دهنده پایگاه داده در نظر گرفته شده است، آنها می توانند پاسخ داده شوند: در حافظه نهان، در پشتوانه سرویس دهنده پایگاه داده، یا در هر دو مکان در یک رفتار توزیع شده. عامل هایی که توزیع شدگی میزان کار شامل انواع اظهارات SQL را مشخص می کنند، محتوای حافظه نهان، نیازمندی برنامه به تازه سازی داده ها و بهینه سازی بر اساس هزینه در حافظه نهان است.

۳,۳ مدل بهینه سازی پرس و جوی پیشنهاد داده شده

برای حل کردن مشکل پردازش پرس و جو در سیستم های پایگاه داده توزیع شده، یک مدل بهینه سازی پرس و جو در پایه ی حافظه نهان پیشنهاد داده شده است.

[52] D.Kossmann
[53] S.Adali
[54] Q.Luo



شکل ۷: بهینه سازی پرس و جو بر پایه حافظه نهان در پایگاه داده توزیع شده

در این مدل، حافظه نهان بین بهینه ساز محلی و سرویس دهنده پایگاه داده به کار برده شده است که هر حافظه نهان با سرویس دهنده پایگاه داده محلی اش پیوند شده است.

شکل۷، مدل پیشنهاد شده برای بهینه سازی پرس و جو بر پایه ی حافظه ی نهان را در پایگاه داده توزیع شده نمایش می دهد. این مدل ماژول های زیر را دارد:

- کاربر: یک کاربر یک نماینده است، نه فقط یک نماینده انسانی(end-user) یا نماینده نرم افزاری که از یک کامپیوتر یا سرویس های شبکه استفاده می کند. یک کاربر یک حساب کاربری دارد و با نام کاربری شناسایی شده است. دیگر لغات برای نام کاربری شامل نام ورود[۵۵]، نام نمایشی و نام دوستانه است. همچنین کاربران به طور گسترده به عنوان کلاسی از مردم که از یک سیستم بدون کامل کردن نیازمندی-های تکنیکی برای شناسایی کامل استفاده می کنند مشخص می شوند. در زمینه های وابسته به دزدان کامپیوتری[۵۶]، این کاربران همچنین به کاربران و کاربران حرفه ای تقسیم می شوند. در پروژه ها که هر فاعل سیستم یک سیستم دیگر و یا یک موتور نرم افزاری است[۵۷]، این امکان کاملاً ممکن است که سیستم هیچ end-userی نداشته باشد. در این مورد، end-userها برای سیستم، باید end-userهای غیر مستقیم باشند.
- واسط جهانی پرس و جو: واسط جهانی پرس و جو برای به دست آوردن یک اشاره گر به واسط دیگر، ارائه دادن یک واسط گرافیکی کاربر برای پایگاه داده (GUID[۵۷]) که به طور منحصر به فرد آن واسط را شناسایی می کند.
- تحلیلگر پرس و جو: تحلیلگر پرس و جو، پرس و جوی جهانی را از GUI دریافت می کند و شِمای جهانی را برای تحلیل پرس و جو رایزنی می کند و در رابطه با تخصیص گره ها تصمیم می گیرد جایی که نتایج پرس و جوی درخواست شده قرار دارند.
- بهینه ساز هزینه: یکی از سخت ترین مسائل در بهینه سازی پرس و جو، حدس دقیق هزینه های طرح‌های فرعی پرس و جو است[۶۳][۶۶]. هزینه طرح های پرس و جوی بهینه ساز با استفاده از مدل های ریاضی هزینه های اجرای پرس و جو که سنگینی واقعی روی تخمین کاردینالیتی یا تعداد تاپل ها در هر لبه در طرح پرس و جو دارند طرح می شود[۶۵]. تخمین کاردینالیتی در چرخش به تخمین فاکتور انتخاب اظهارات در پرس و جو وابسته است[۵۹]. به طور سنتی، سیستم های پایگاه داده گزینشگری به طور منصفانه بین توزیع یافتگی مقادیر در هر ستون تخمین می زنند، مانند هیستوگرام.
- توزیع کننده پرس و جو: این برای ارسال زیر پرس و جو ها به پایگاه های اختصاص داده شده پاسخگو است، بنابر این محاسبات واقعی روی آن زیر پرس و جو ها می تواند به بیرون ترابری شود[۵۵][۶۰].
- حافظه نهان: در علم کامپیوتر، یک حافظه نهان یک جزء است که به طور شفاف داده ها را ذخیره می‌کند که در آینده برای آن داده ها درخواست می شود که سریع تر به کار می روند[۵۴][۶۸]. داده هایی که در حافظه نهان ذخیره شده اند ممکن است مقادیری باشند که قبلا محاسبه شده اند یا کپی هایی از

[55] Login name
[56] hacker
[57] Graphical User Interface Database

مقادیر اصلی باشند که ذخیره شده اند. اگر داده های درخواست شده در حافظه نهان وجود دارند (cache hit)، این درخواست می تواند با خواندن حافظه نهان به آسانی پاسخ داده شود که بسیار سریع تر است. در غیر اینصورت (cache miss) داده ها باید دوباره محاسبه شوند و یا از مکان حافظه اصلی واکشی شوند که خیلی کندتر است. از این جا، تعداد درخواست هایی که بتواند با استفاده از حافظه نهان پاسخ داده شود، باعث بالا رفتن کارایی و سرعت سیستم می شود. برای تأثیرگذار بودن روی هزینه و تأثیرگذار بودن روی استفاده از داده ها، حافظه های نهان نسبتا کوچک هستند. با این حال، حافظه های نهان، خودشان را در عرصه های مختلفی از محاسبات رشد داده اند، به دلیل دسترسی به الگوها در برنامه های اصلی کامپیوتر که به صورت محلی آدرس دهی می کنند. بررسی ها نشان می دهد آدرس های بعدی که درخواست می شوند نزدیک جایگاه آدرس های قبلی هستند که اخیرا درخواست شده اند.

- پایگاه داده: یک پایگاه داده یک مجموعه سازمان یافته از داده ها است، که امروزه به طور کلی در فرم دیجیتال است. داده ها به طور کلی برای مدل کردن نمودارهای وابسته قابلیت اطمینان سازمان دهی شده اند. (به عنوان مثال، قابلیت دسترسی به اتاق های هتل)، در یک مسیر که از پردازش های مورد نیاز این اطلاعات پشتیبانی می کند. (برای مثال، پیدا کردن یک هتل با جای خالی). پایگاه داده آغازین به درستی به ساختار داده های پشتیبانی شده اش اعمال شده و نه به سیستم مدیریت پایگاه داده (DBMS). کاربران می توانند پایگاه داده را در واسط کاربر جهانی پرس و جو کنند. پرس و جو ها می توانند پرس و جو های وابسته باشند[۶۹]. پرس و جو های وابسته، پرس و جو های با درخواست دسترسی به بیش از یک پایگاه داده هستند. هنگامی که یک واسط کاربر جهانی، یک پرس و جوی همبسته دریافت می کند، این ابتدا پرس و جوی همبسته را به زیر پرس و جو ها تقسیم می کند و آن زیر پرس و جو ها را به پایگاه داده‌های مخصوص آنها ارسال می کند سپس پاسخ های برگشتی از هر پایگاه داده با هم ترکیب شده و نتایج در واسط کاربر جهانی نمایش داده می شود.

۳,۴ بهینه ساز هزینه

این پاسخگویی بهینه ساز هزینه برای انتخاب گره اختصاص داده شده برای پردازش پرس و جو است. تا زمانی که داده پایگاه های چندگانه جایگزین شده است[۶۱]، بهینه ساز هزینه روی پارامترهای اصلی انتخاب شده برای انتخاب گره اختصاص داده شده برای پردازش کار می کند.

- طول صف فعلی: این تعداد پردازش هایی است که در حال اجرا برای اجرا شدن روی سرویس دهنده هستند.

- دوردستی سرویس دهنده: دوردستی سرویس دهنده به فاصله جغرافیایی سرویس دهنده از سرویس گیرنده اشاره می کند. هر قدر که سرویس دهنده نزدیک تر باشد، هزینه واکشی داده ها از سرویس دهنده کمتر خواهد بود.
- گنجایش سرویس دهنده: این تعدادی از پردازش هایی است که می توانند بدون مختل کردن عملیات معمولی سرویس دهنده روی آن اجرا شوند.
- بارگیری[58]: این نسبت پردازش های واقعی در حال اجرا روی سرویس دهنده به گنجایش سرویس دهنده است. مثال: بارگیری= طول صف فعلی تقسیم بر گنجایش سرویس دهنده.

پارامترهایی که مورد بررسی قرار گرفته اند، طول صف، دوردستی سرویس دهنده و بارگیری است. از آنجایی که طول صف یکی از مهم ترین فاکتورها در تعیین کردن زمان انتظار است، یک پردازش برای اجرا در صف صبر خواهد کرد. مدل پیشنهاد شده با اولویت بندی عامل های مختلف به کار برده شده در بررسی برای بهینه سازی هزینه طراحی شده است. بالا ترین اولویت به طول صف در ترتیب بندی برای کم کردن ترافیک گلوگاه[59] در یک محیط تکراری و فاصله وابسته سرور، دومین اولویت بالا در ترتیب برای کم کردن هزینه های انتقال نادرست داده شده از اینجا، طول صف با فاکتور از ۱۰ تقسیم شده است در حالی که در محاسبات ریاضی، دوردستی سرویس دهنده با فاکتور از ۱۰۰ تقسیم شده است. در ترتیب بندی برای انتخاب گره بهینه برای ارتباط با به عهده گیری که طول صف معمولا در اندازه کوچک است هنگامی که با دوردستی سرویس دهنده مقایسه می شود. بنابراین کوچک تر کردن طول صف در سرویس دهنده، کم کردن هزینه و متعاقباً بالا بردن زمان پاسخ است.

بارگیری نسبت تعداد درخواست های پاسخ داده شده به وسیله سرویس دهنده به کل گنجایش درخواست هایی است که می تواند توسط سرویس دهنده مدیریت شود است. این عامل ها به متعادل کردن ازجمله بارگیری سرویس دهنده ها کمک می کند.

هماهنگی برای تخمین هزینه یک اجرا بر پایه پرس و جو بر فراز همه ی عامل های مطالعه شده در بالا مانند زیر است:

اگر به تعداد n تا سرویس دهنده داشته باشیم، برای هر سرویس دهنده i و j

[58] Load
[59] bottleneck

$$P_r = P_r + (QL_i - QL_j)/10$$

$$P_r = P_r + (SD_i - SD_j)/100$$

$$L_i = QL_i / SC_i$$

$$L_j = QL_j / SC_j$$

$$P_r = P_r + (L_i - L_j)$$

Where P_r = net priority of server

QL_i - queue length of server i

QL_j - queue length on server j

where $\{0 < i < n\}$ and $\{0 < j < n\}$

SD_i - server distance of server i from requesting node

SD_j -server distance of server j from requesting node

L_i - load on server i

L_j - load on server j

SC_i - Server capacity of server i

SC_j - Server capacity of server j

۳,۵ الگوریتم بهینه سازی صف

نزدیک شدن به بهینه سازی پرس و جو بر پایه ی حافظه نهان، بر پایه ی حقیقت وجود یک حافظه نهان در کنار پایگاه یا سرویس دهنده داده است. یک درخواست برای دسترسی به داده ها به وسیله کاربر با استفاده از واسط کاربر ساخته شده است، این درخواست (یا پرس و جو) به وسیله تحلیلگر پرس و جو تحلیل شده است و هزینه این پرس و جو توسط بهینه ساز هزینه، در ترتیب تصمیم گیری درباره گره های بهینه ای که زیر پرس و جو ها برای پردازش های بعدی به آنها ارسال شده اند[۶۲][۶۴] بهینه سازی شده است. توزیع کننده پرس و جو برای بیشتر توزیع های زیر پرس و جو ها به پایگاه های مخصوص پاسخگو است. بهینه ساز محلی در هر پایگاه بررسی می کند آیا نتایج وابسته به این حقیقت که آیا این یک درخواست تکراری است و یا درخواست جدید است از

حافظه نهان می توانند برگشت داده شوند یا باید داده ها از پایگاه داده واکشی شوند. اگر داده ها از پایگاه داده واکشی شوند، حافظه نهان طبیعتاً به روز رسانی می شود و نتایج در واسط کاربر به نمایش در می آید.

۳,۵,۱ الگوریتم پیشنهاد شده

قدم۱ : ورودی: پرس و جو

خروجی: نتایج بهینه سازی شده

قدم۲ : پایگاه[] = پایگاه هایی که روی هر زیر پرس و جو تکرار گرفته شده اند.

طول = پایگاه[] . طول

قدم۳ : اگر(if) (طول > ۱)

۳,۱ بهینه ساز هزینه را شروع کن

۳,۲ عامل های زیر را برای هر پایگاه i که زیر پرس و جو ها در آنها کپی گرفته شده اند بررسی کن

- طول صف فعلی (QL_i)
- دور دستی سرویس دهنده (SD_i)
- گنجایش سرویس دهنده (SC_i)
- بارگیری (L_i)

۳,۳ ارجحیت بندی همه ی پایگاه ها روی پایه ی مقادیر این پارامتر ها با استفاده از تساوی های زیر:

در نظر بگیرید n تعداد سرویس دهنده ها برای هر پایگاه i در پایگاه[] باشد.

$$P_r = P_r + (QL_i - QL_j)/10$$

$$P_r = P_r + (SD_i - SD_j)/100$$

$$L_i = QL_i / SC_i$$

$$L_j = QL_j / SC_j$$

$$P_r = P_r + (L_i - L_j)$$

$site_0$ = site with maximum priority

else

$site_0$ = site [0]

end if

قدم۴ : زیر پرس و جو را به پایگاه ۰ (صفر) ارسال کن

قدم۵ : حافظه نهان محلی پایگاه ۰ (صفر) را بررسی کن

قدم۶ : اگر (داده در حافظه نهان موجود بود) مثلا در پایگاه ۰ (صفر) نتایج را از حافظه نهان محلی برگردان

اگرنه

- داده ها را از پایگاه داده واکشی کن
- حافظه نهان را به روز رسانی کن
- نتایج را برگردان

پایان شرط اگر (if)

قدم۷ : پایان الگوریتم

۳,۵,۲ نتایج آزمایش

برآوردهای آزمایش از مدل پیشنهاد شده از مجموعه ای از حافظه کامپیوتر پایگاه داده های توزیع شده، سیستم عامل ویندوز سرور۷ و سیستم مدیریت پایگاه داده مایکروسافت اکسس استفاده می کند.

عامل هایی مانند دور دستی سرویس دهنده، طول صف سرویس دهنده و بارگیری سرویس دهنده برای رده بندی پایگاه ها برای تخمین هزینه بررسی شده اند. یک تصویر از فاکتورهای رده بندی برای بررسی در ادامه در شکل۸ نمایش داده شده است.

Tabular data

ID	QLength	SerDis	SerCap	Load	Priority
1.0	85.0	100.0	100.0	0.85	0.0
2.0	77.0	200.0	160.0	0.48125	0.16875
3.0	83.0	150.0	130.0	0.6384615384615384	-0.08846153846153842

شکل ۸ : رده بندی پایگاه ِ پایگاه داده

همانطور که در شکل بالا نشان داده شده است، پایگاه بهینه، پایگاه با آی‌دی(ID) ۲,۰ با بهترین حق تقدم ۰,۱۶۸۷۵ است. بنابر این توزیع کننده پرس و جو، زیر پرس و جو را به پایگاه با آی‌دی(ID) ۲,۰ برای پردازش های بعدی ارسال می کند.

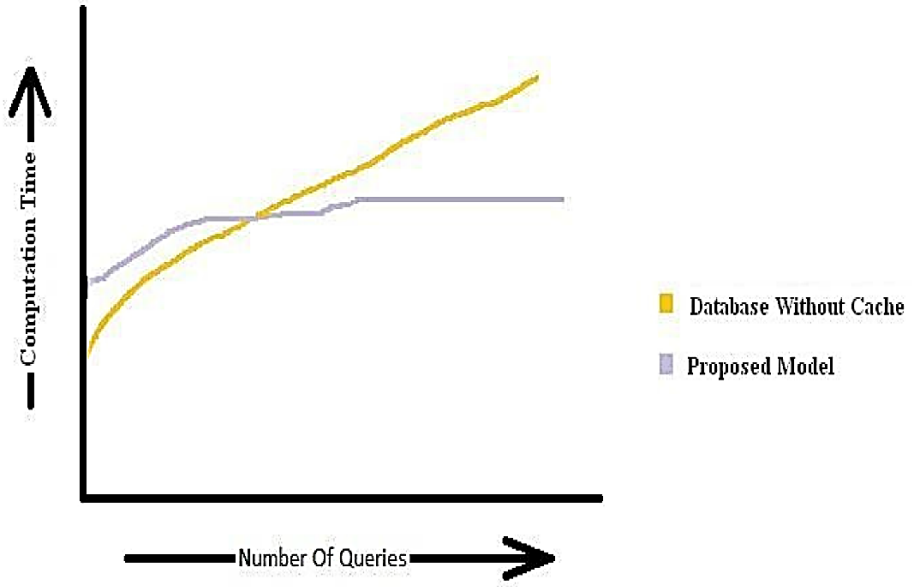


Figure 4. Computation time vs. number of queries

شکل ۹: زمان محاسبه در مقابل تعداد پرس و جو ها

شکل ۹ نشان می دهد که چگونه مدل پیشنهاد شده بر پایه حافظه نهان، در مقایسه با مدلی که از حافظه نهان محلی استفاده نمی کند بهینه سازی پرس و جو را بهبود می دهد.

در داستان نمایش داده شده، پایگاه داده تکراری برای محاسبه نتایج مدل پیشنهاد شده استفاده شده است، هنگامی که بدون حافظه نهان استفاده شده است به عنوان مدل فعلی در نظر گرفته شده و با مدل پیشنهاد شده بر پایه حافظه نهان مقایسه شده است.

همانطور که شکل ۹ نشان می دهد، زمان پاسخ نشان داده شده با مدل پیشنهاد شده بهتر از زمان استفاده از سیستم پایگاه داده بدون حافظه نهان نیست. این، به این حقیقت وابسته است که در ابتدا حافظه نهان خالی است و داده ها از پایگاه داده واکشی می شوند و همچنین زمان سربار برای به روز رسانی حافظه نهان نیز مورد نیاز است.

بعد از یک دوره زمانی خاص، همانطور که تعداد پرس و جو ها افزایش می یابد، زمان محاسبه نشان داده شده در هر دو مورد یکسان است. به این حقیقت پرداخته شده است که زمان با واکشی داده ها از حافظه نهان با زمان مورد نیاز برای به روز رسانی حافظه نهان خنثی می شود. بیشتر افزایش در تعداد پرس و جو ها شِمای پیشنهاد شده را تحت تأثیر قرار نمی دهد، از آنجایی که بیشتر نتایج ارسال شده برای درخواست های پرس و جو ها از حافظه نهان واکشی شده است و محاسباتی مورد نیاز نیست.

از اینجا، ترفند پیشنهاد شده به طور مهمی زمان پاسخ و در نتیجه دسترسی به داده ها توسط کاربر را سریع تر می‌کند. ما نتایج به دست آمده توسط خودمان را با نتایج وِن-سیان[۶۰][۶۸] که از حافظه نهان جهانی استفاده کرده است مقایسه کردیم. وِن-سیان نتایج را بر پایه ی دو فاکتور به دست آورد: تعداد پرس و جو ها و نوع پرس و جو ها. اما از آنجایی که هنگامی که پرس و جوی تکراری از طرف کاربران دریافت می شود نوع پرس و جو اهمیت کمی دارد ما فقط روی تعداد پرس و جو ها تأمل کردیم. مقایسه نتایج مدل پیشنهادی ما با نتایج به دست آمده توسط وِن-سیان[۲۰۰۳] در جدول۱ جدول بندی شده است.

[60] Wen-Syan

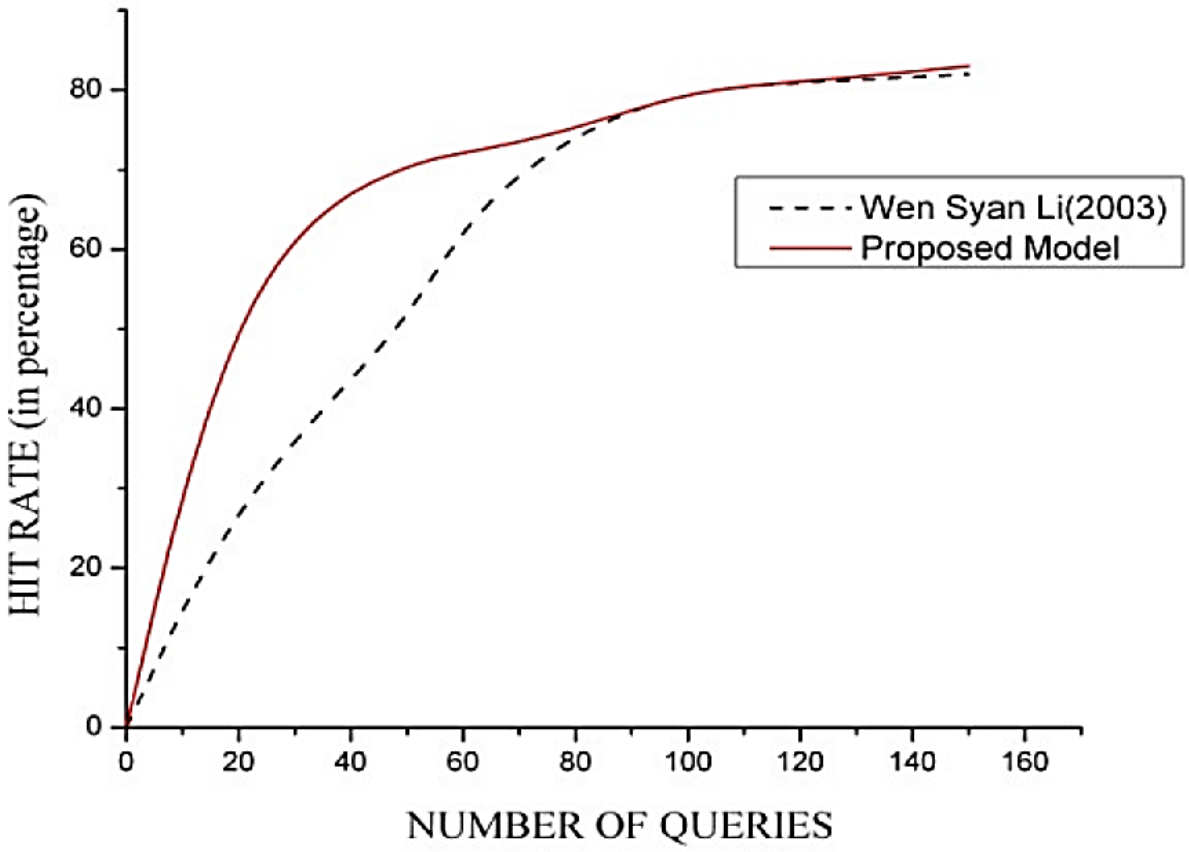
جدول۳: نتایج آزمایشی

No. of Query	Cache hit-rate (Wen-Syan[2003])	Cache hit-rate (Proposed Model)
0	0	0
20	0.3	0.6
50	0.5	0.71
60	0.64	0.72
80	0.75	0.75
100	0.8	0.76
120	0.81	0.81
150	0.82	0.83

جدول ۳ تعداد پرس و جو ها را در مقایسه با نرخ به هدف خوردن های[۶۱] حافظه نهان هنگامی که مدل پیشنهاد شده به کار برده شده است و مقایسه پارامتر ها با مقادیر انجام شده با ترفند حافظه نهان استفاده شده توسط وِن-سیان لی[۶۲][۲۰۰۳] که به عنوان یک ترفند کارا در بهینه سازی پرس و جو معرفی شده است استفاده شده به نمایش در آمده است.

شکل ۱۰ وابستگی میان تعدادی از پرس و جو ها و نرخ به هدف خوردن های ارائه شده در جدول ۳ را نمایش می‌دهد. همانطور که در شکل نمایش داده شده است، هنگامی که ۲۰ پرس و جو انتخاب شده است، نرخ به هدف خوردن ها نزدیک ۶۰٪ است و هنگامی که ۱۰۰ پرس و جو انتخاب شده است، نرخ به هدف خوردن ها نزدیک به ۸۰٪ است.

[61] hitrate
[62] Wen-Syan Li



شکل ۱۰ : نرخ به هدف خوردن ها در مقابل تعداد پرس و جوها

وِن-سیان لی[۲۰۰۳] همچنین از مفهوم حافظه نهان استفاده کرد اما از شکل نشان داده شده در مدل پیشنهاد شده می توان نتیجه گرفت که این ارتقاء تندی در نرخ به هدف خوردن ها در مقایسه با ترفند استفاده شده توسط وِن-سیان لی[۲۰۰۳] دارد. این موضوع با این حقیقت قابل توجیه است که، در ترفند پیشنهاد شده، حافظه نهان طرف پایگاه داده در هر پایگاه داده استفاده شده است یعنی حافظه نهان، محلیِ سرویس دهنده است در حالی که وِن-سیان لی از حافظه نهان بر پایه شبکه استفاده کرده است[۶۸].

نتیجه گیری

مفهوم ترکیب الگوریتم بهینه سازی کلنی مورچه ها برای بهینه سازی پرس و جو های پایگاه داده های توزیع شده هنوز یک گزینه مبتدی است. تحقیق برای ایجاد و به کار گیری ترکیب ACO برای حل انواع مختلف مسائل در حال پیشرفت است. نتایج تولید شده توسط ترکیب های ACO در مسائل بهینه سازی مؤثر هستند. تحقیقات نشان می دهد به کار گیری این الگوریتم های احتمالیِ تولید راه حل های موفق در حالت توزیع یافته، مانند مدیریت پایگاه داده های ارتباطی، زمانی که اندازه پرس و جو و تعداد پیوندها در پرس و جو رشد می کند را بهبود می‌بخشد. هنوز فرصت زیادی برای تولید راه حل های بهینه و مشخص کردن ترفندهای جستجو با استفاده از ترکیب ACO برای پرس و جو ها در پایگاه داده های توزیع شده مخصوصا زمانی که اندازه و پیچیدگی ارتباطات افزایش پیدا می‌کند با استفاده تعدادی از پارامترهای مؤثر پرس و جو وجود دارد.

ما مشاهده کردیم که هر پرس و جو ممکن است تعدادی طرح حل داشته باشد و همچنین هر طرح پرس و جو ممکن است شامل مجموعه ای از کارها باشد که هر کار ممکن است هزینه های مربوطه را داشته باشد. اگرچه مقدار هزینه ممکن است با اعداد مثبت نمایش داده شود. طرح های فرعی پرس و جو و دیگر پرس و جو های مجموعه ی پرس و جو ممکن است شامل چند کار باشند. این برای مشخص کردن مجموعه ای از کارها با کم ترین هزینه کل مورد نیاز است که ممکن است شامل همه ی کارها در حداقل یک طرح از هر پرس و جو باشد. ارزیابی طرح های پرس و جو، ارزیابی هزینه طرح های پرس و جو و به کار گیری های خاص بهینه سازی پرس و جو بسیار مشابه سیستم های پایگاه داده های توزیع شده است. در این مورد، بهینه ساز باید منابع داده را رایزنی کند تا هزینه عملیات را محاسبه کند. پردازش به کار برده شده در این مورد نشان می دهد که، هنگامی که طراحی پایگاه داده فیزیکی برای بهینه ساز شناخته شده است، این الگوریتم بهینه سازی به درستی کار می کند.

بهینه سازی بر پایه پیشنهاد حافظه نهان برای اینکه گزینه بهتری برای بهینه سازی پرس و جو ها در سیستم های پایگاه داده توزیع شده همگن باشد آزمایش شد. این می تواند توسعه یابد تا برتری بزرگی[در مقایسه با دیگر روش ها] باشد، زمانی که دسترسی به طور کلی نوع فقط خواندنی برای آن موارد است و به روز رسانی حافظه نهان به طور "متناوب" و "در یک زمان" حفظ می شود، جریان پایگاه داده پیامد بزرگی نیست و همچنین این زمان محاسبات را افزایش می‌دهد. این مدل می تواند مورد سنجش قرار بگیرد تا در بیشتر موارد در موقعیت هایی که کاربران فقط به طور کلی به بخش خاصی از پایگاه داده دسترسی پیدا می کنند، مانند موقعیت ارجاع ها، مفید باشد. از آنجایی که دسترسی به پایگاه داده به دلیل زمان جستجو و تأخیر دیسک زمان زیادی مصرف می کند، در چنین مواردی حافظه نهان می تواند در مقدار زیادی از دسترسی ها به پایگاه داده صرفه جویی کند. زمان

جستجو و تأخیر دیسک از حد مشخصی نمی توانند کمتر بشوند. از اینجا حافظه نهان توسعه داده می شود تا حل بزرگی باشد، که دسترسی به حافظه نهان بسیار سریع تر از دسترسی به پایگاه داده است. برای کار آینده، مدل پیشنهاد شده می تواند برای پایگاه داده های توزیع شده همگن به کار برده شود. اندیشه کلی از حافظه نهان جهانی همچنین می تواند برای نتایج بهتر به سیستم معرفی شود. حقیقت مشابه این صرفه جویی در گردش داده در سیستم های پایگاه داده در جایی که دسترسی خواندن و نوشتن مشابه هم است نیز می تواند معرفی شود.

منابع:

[*1] Query Optimization Strategies in Distributed Databases Ms. Preeti Tiwari-Research Scholar (Computer Science)-The IIS University, Gurukul Marg, SFS,-Mansarovar, Jaipur 302020, (Raj.) India-preetitiwari76@yahoo.co.in and Swati V. Chande-Dean and Professor (Computer Science)-International School of Informatics and Management,-Sector-12, Mahaveer Marg, Mansarovar,-Jaipur 302020, (Raj.) India-swatichande@rediffmail.com

[*2] Implementation of query optimization techniques in distributed environment through genetic algorithm,Mishra Sambit Kumar , Pattnaik Srikanta (Dr.), Patnaik Dulu(Dr.)

[*3] Cache Based Query Optimization Approach in Distributed Database-Mantu Kumar, Neera Batra and Hemant Aggarwal

[۱] C.Yu, Z M Ozoyoglu, K. Kam," Optimization of Distributed Tree Queries", J.Comput. Sys. Sci, Vol ۲۹, No ۳, pp ۴۴۵-۴۰۹, ۱۹۸۴.

[۲] S.Ceri, G. Pologatti, "Distributed Database Principles and Systems", Mc GrawHill.

[۳] A. Hameurlain, F. Morvan, "Evolution of Query Optimization Methods", Trans. on Large Scale Data and Knowledge Cent. Syst.I, LNCS ۵۷۴۰, pp۲۴۲-۲۱۱, ۲۰۰۹.

[۴] R. Ghaemi, AM Fard, Md. NB Sulaiman, " Towards Optimal Query Execution in Data Grids", Advanced Technologies, pp ۷۲-۵۷, ۲۰۰۵.

[۵] A. Aljanaby, E. Abuelrub, M.Odeh, "A Survey of Distributed Query Optimization", The International Arab Journal of Information Technology, Vo۲, No.۱, ۲۰۰۵.

[۶] Craig S. Mullins, "Distributed Query Optimization", Technical Support ۲۰۰۶.

[۷] D.Abdullah, "Query Optimization in Distributed Databases",

[۸] S. Rho, T. March, " Optimizing Distributed Joins Queries: A Genetic Approach", Annels, of Operations Research, Science Publishers, pp ۲۸۸-۱۹۹, .۱۹۹۷

[9] M.Chen, P.Yu, "Using Join Operations as Reducers in Distributed Query Processing", Proceedings of 2nd Intl. Symp. on Databases in Parallel and Distributed System, July 1990.

[10] PMG Apers, AR Henver, SB Yao, "Optimization Algorithms for Distributed Queries", IEE Transactions on Software Engineering, Vol. 9, no.1, pp 57-68, January 1983.

[11] E. Sevinc, A. Cosar, "An Evolutionary Genetic Algorithm for Optimization of Distributed Database Queries",The Computer Journal, Vol. 54, No.8, 2011.

[12] C.Wang, M.Chen, "On Complexity of Distributed Query Optimization", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 4, August 1996.

[13] PA Bernstein, N Goodman, E.Wong, C. Reeve, “Query Processing in a System for Distributed Database (SDD-1)”, ACM Trans. Database Sys., Vol. 6, No. 4, pp 602-625, December 1981.

[14] MS Chen, PS Yu, “Interleaving Join Sequence with Semijoins in Distributed Query Processing”, IEEE Transactions, Issue 5, Vol. 3, August 2005.

[15] S. Pramanik, D. Vineyard, “Optimizing Join Queries in Distributed Database” IEEE Trans, Software Engg. Vol. 14, pp 1391-1426, September 1988.

[16] D. Sukheja, Umesh kr. Singh, “ A Novel Approach of Query Optimization for Distributed Database System”, IJCSI, Vol. 8, Issue 4, No. 1, July 2011.

[17] AN Swami, A. Gupta, “Optimization of Large Join Queries in Distributed Database”, Proc. of ACM-SIGMOD Conference on Management of Data, pp 8-17, 1988.

[18] YC Kang, “Randomized Algorithms for Query Optimization”, PhD Thesis, University of Wisconsin-Madison, 1991, Technical Report # 1053.

[19] M. Hussein, F. Morvan, A. Hameurlain, “Dynamic Query Optimization”, 19th Intl. Conf. on Parallel and Distributed Computing Systems, ISCA, 2009.

[20] D. Kossman, “The State of Art in Distributed Query Processing”, ACM Computing Survey, pp 422-469, 2000.

[21] LF Lohman, GM Mackert, “R* Optimizer Validation and Performance Evaluation for Distributed Queries”, Proc. of 12th Intl. Conf. on VLDB, pp 149-159, 1986.

[22] D. Kossman, K. Stocker, “Iterative Dynamic Programming: A New Class of Query Optimization Algorithm”, ACM Transactions on Database Systems, 2000.

[23] La Fortune, Wong, “ Query Optimization>>>>

[24] M. Chen, P.Yu, “Combining Join and Semi Join Operations for Distributed Query Processing”, IEEE Transactions on Knowledge and Data Engineering., Vol. 5, No. 3, 1993.

[25] M. Xifeng, F.Yuanyuan, “Distributed Database System Query Optimization Algorithm Research”, IEEE Intl. Conf. on Computer Science and Information Technology”, Vol.8, pp 657- 660, 2010.

[26] M. Gregory, “Genetic Algorithm Optimization of Distributed Database Queries”, IEEE World Congress on Computational Intelligence, pp 271-276, 1998.

[27] M. Naohoghdem, “ Query Optimization in Distributed Database using Hybrid Evolutionary Algorithm, Intl. Conf. on Information retrieval and Knowledge Management, pp 125-130, 2010.

[28] Z. Zhou, “Using Heuristics and Genetic Algorithms for Large Scale Database Query Optimization”, Journal of Information and Computing Science, Vol.2, No. 4, 2007.

[29] Y. Zhou, W.Wan, J. Liu, “Multi-Joint Query Optimization of Database Based on the Integration of Best-Worst Ant Algorithm and genetic Algorithm”, Wireless Mobile Computing (CCWMC 2009), IET Intl. Communication Conference, pp 543, 2009.

[30] H. Kadhkhodaei, F. Mahmoudi, “A Combination Method for Joining Ordering Problem in Relational Database using Genetic Algorithm and Ant Colony”, IEEE Trans. On Granular Computing, pp 312-317, 2011.

[31] Enxiu Chen1 and Xiyu Liu, “Ant Colony Optimization – Methods and Applications-Multi-Colony Ant Algorithm”, Google Scholar, 2011,

http://cdn.intechweb.org/pdfs/13584.pdf

[32] Marco Dorigo, Mauro Birattari, and Thomas Suiitzle, “Ant Colony Optimization- Artificial Ants as a Computational Intelligence Technique”, IEEE Computational Intelligence Magazine, November 2006.

[33] Zar Chi Su Su Hlaing, May Aye Khine , “An Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem”, International Conference on Information Communication and Management, Singapore, IPCSIT vol.16 (2011), IACSIT Press.

[34] Tansel Dokeroglu, Ahmet Cosar, “Dynamic Programming with Ant Colony Optimization Metaheuristic for optimization of Distributed Database Queries”, ISCIS:26th International Symposium on Computer and Information Sciences, IEEE, Vol 2 , pp.107-113, 2011

[35] Gao Shang' 2, Jiang Xin-zil, Tang Kezong', Yang Jingyu, “Hybrid Algorithm Combining Ant Colony Optimization Algorithm with Particle Swarm Optimization”, Proceedings of the 25th Chinese Control Conference, Harbin, 7-11 August, 2006

[36] Lanzelotte, R.S.G., Zaït, M., Gelder, A.V.: Measuring the effectiveness of optimization. Search Strategies. In: BDA 1992, Trégastel, pp. 162–181 (1992).

[37] Manolescu, I., Bouganim, L., Fabret, F., Simon, E.: Efficient querying of distributed resources in mediator systems.In:Meersman, R., Tari, Z., et al. (eds.) CoopIS 2002, DOA 2002, and ODBASE 2002. LNCS, vol. 2519, pp. 468–485. Springer, Heidelberg (2002).

[38] Oszu, M. T. and Valduriez P., “Distributed and Parallel Database Systems,” in Trucker A. (Ed), The Computer Science and Engineering Handbook, CRC press, pp. 1093-1111, 1997.

[39] Oszu M. T. and Valduriez P., Principles of Distributed Database Systems, Prentice Hall International, NJ, 1999.

[40] Ioannidis Y. E., “Query Optimization,” in Trucker A. (Ed), The Computer Science and Engineering Handbook, CRC press, pp. 1038- 1054, 1996.

[41] Kossmann D. and Stocker K., “Iterative Dynamic Programming: A New Class of Query Optimization Algorithm,” ACM TODS, March 2000.

[42] Ibraraki T. and Kameda T. “Optimal Nesting for Computing N-Relational Joins,” ACM Transactions on Database Systems, vol. 9, no. 3, pp. 482-502, 1984.

[43] G. Graefe. Parallel Query Execution Algorithms. In Encyclopedia of Database Systems , pages 2030{2035. 2009.

[44] O. Gorlitz and S. Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VOID descriptions. In COLD'11 , 2011.

[45] Swati Gupta, Kuntal Saroba, Bhawna, “Fundamental Research of Distributed Databse”, International Journal of Computer Science and Management Studies, vol. 11, 2011, pp. 138-146.

[46] Fan Yuanyuan, Mi Xifeng, “Distributed Database System Query Optimization Algorithm Research”, IEEE international conference on Computer Science and Information Technology, 2011, vol. 8, pp.145-149.

[47] Reza Ghaemi, Amin Milani Fard, Hamid Tabatabee, Mahdi Sadaghizadeh, “Evolutionary Query optimization for Heterogeneous Distributed Database Systems” ,World Academy of Science, Engineering and Technology, vol. 45, 2008, pp. 43-49.

[48] Adali S., Candan K. S., Papakonstantinou Y., Subrahmanian V. S., “Query caching and optimization in distributed mediator systems”, ACM SIGMOD, vol. 2, 2006, pp. 45-118.

[49] Yannis. E. Loannidis and Youngkyung Cha Kang,“Randomized Algorithms for optimizing large Join Queries”, ACM SIGMOD, 1990, vol. 19, pp. 47-53.

[50] D. kossmann, M. J. Franklin, G. Drasch, “Cache Investment: Integrating Query Optimization and Distributed Data Placement”, ACM, vol. 25, 2000, pp. 517-558.

[51] D. Kossman, “The state of the art in distributed query processing” , ACM Computing Surveys, vol. 32, 2000, pp.422-469.

[52] P. Griffiths, Selinger, M. M. Astrahan, D. D. Chamberlin, R.A. Lorie, T. G. Price, “Access path selection in a rational database management system”, ACM SIGMOD, vol. 20, 1979, pp. 23-34.

[53] Stocker, Kossman, Braumandl, Kemper, “Integrating Semi Join Reducers into state of the art query processors”, ICDE, 2001, pp. 143-156.

[54] Neera Batra, A . K . Kapil, “Three Tier Cache Based Query Optimization Model in Distributed Database”, IJEST, vol. 2,2010, pp. 3206-3212.

[55] C. Damianos, K. A. Ross, “Partitioned Optimization of Complex Queries”, ELSEVIER, vol. 32, 2007, pp. 248-282.

[56] Q. Luo, S. Krishnamurthy, C. Mohan, H. Pirahesh, H. Woo,B. G. Lindsay, J. F. Naughton, “Middle-tier database caching for e-business”, ACM SIGMOD, 2002, pp. 600-611.

[57] A. Korzyk, “Towards XML As A Secure Intelligent Agent Communication Language”, the 23rd National Information Systems, 2000, pp. 134-138.

[58] Huang, Kuan – Tsae, Davenport, Wilbur B., “Query Processing in Distributed Heterogeneous Systems”, MIT Laboratory for information and Decesion Systems, 1981,pp. 1-17.

[59] J. Callan, “Distributed Information Retrieval ” , W. B.Croft, Ed. Kluwer Academic Publishers, 2000.

[60] Patricia G. Selinger, Michael E. Adiba, “Access Path Selection in Distributed Database Management Systems”,ICOD, 1980, pp. 204-215.

[61] Syam Menon, “Allocating fragments in distributed Database”, IEEE Transactions on Parallel and Distributed Systems, vol. 16, 2005, pp. 577-585.

[62] D. Kossmann, K. Stocker, “Iterative Dynamic Programming: A New Class of Query optimization Algorithms”, ACM Computing Surveys, vol. 25, 2000, pp.422-469.

[63] Peng-Yeng Yin, Shiuh-Sheng Yu,Pei-Pei Wang and Ye-Ti Wang, “A Hybrid particle swarm optimization algorithm for optimal task assignment id distributed systems”, Computer Standards and Interfaces, vol. 28, 2006, pp. 441-450.

[64] LEE Chiang, CHIH Chi-seng, CHEN Yaw-huei,“Optimizing large join queries using a graph based approach”, IEEE Transactions on Knowledge and Data Engineering, vol. 13, 2001, pp. 298-315.

[65] Tsai P S M, CHEN A L P, “Optimizing queries with foreign function in a distributed environment”, IEEE Transactions on Knowledge and Data Engineering, vol. 14, 2002, pp.809-824.

[66] S. Pramnaik, D. Vineyard, “Optimization Join Queries in Distributed Database” , IEEE Transaction on Software Engineering, vol. 14, 1998, pp.1319-1326.

[67] Pankti Doshi, Vijay Raisinghami, “Review of Dynamic Optimization Strategies in Distributed Database”, IEEE International Conference on Electronics And Computer Technology, 2011, vol. 6, pp. 145-149.

[68] Wen-Syan Li, Oliver Po, Wang-Pin Hsiung, K. Selc_uk Candan, Divyakant Agrawal , “Freshness-driven adaptive caching for dynamic content Web sites” , ELSEVIER, vol. 47, 2003, pp. 269-296.

[69] Wen-Syan Li, Dengfeng Gao, Haifeng Jiang, “Improving parallelism of federated query processing”, ELSEVIER, vol.64, 2008, pp.511-533.