دوفصلنامه محاسبات و سامانه های توزیع شده
سال چهارم، شماره اول، شماره پیاپی ۷، صفحه ۱ تا ۱۱، سال ۱۴۰۰

ارائه روش نوین فشرده سازی ترکیبی برای بهینه سازی مصرف حافظه وسرعت دسترسی درپایگاه داده ابری

سید بهنام حسینی[1]
کارشناسی ارشد، دانشکده علوم کامپیوتر، دانشگاه پویش، قم.

چکیده

شرکت‌ها و ارائه‌کنندگان خدمات ابری افزایش چشمگیری در میزان حجم داده ذخیره‌شده در فضای ابر عمومی و خصوصی را داشته‌اند. بنابراین هزینه ذخیره‌سازی داده‌ها در حال رشد است. زیرا آن‌ها از تنها عملکرد لایه ذخیره‌سازی برای ذخیره‌سازی تمام داده‌های ابری استفاده می‌کنند. این موضوع قابل توجه است که برای کاهش هزینه‌های ابری، داده‌ها را به دو دسته فعال (گرم) و غیرفعال (سرد) طبقه‌بندی می‌کنند. با توجه به پژوهش در حافظه اصلی پایگاه داده‌ها، آثار اخیر در مورد روش‌های متمرکز برای شناسایی داده‌های گرم و داده‌های سرد است. روش پیگیری تاپل شناسایی است و به شناسایی گرم و سرد شناخته می‌شود. در مقابل، ما یک رویکرد جدید به نام روش LOAD DATA INFILE که آهنگ هر دو تاپل‌ها در دسترسی ذخیره‌سازی ثانویه پایگاه‌های داده است را معرفی می‌کنیم. هدف ما به منظور افزایش عملکرد در شرایط سه بعد است که عبارتند از: فضای ذخیره‌سازی، زمان سپری‌شده پرس‌وجو، و مصرف CPU. به منظور تأیید اثربخشی روش‌مان، پیاده‌سازی دقیق آن را بر روی رویکرد LOAD DATA INFILE Approach (LDA) متوجه شدیم که سطرها را از یک فایل متنی داخل یک جدول با سرعت بسیار بالا با استفاده از معیار شناخته شده qps و TPC-H می‌خواند. نتایج تجربی نشان می‌دهد که عملکرد رویکرد LOAD DATA در مقایسه با روش prepare_data، رویکرد بهتری را نسبت به دو بعد عملکرد دارد. به طور خاص، در عملکرد LOAD DATA کاهش فضای ذخیره‌سازی ۱۴ الی ۶۲ درصد به طور متوسط صورت گرفته و کاهش متوسط زمان پرس‌وجوی سپری‌شده نیز ۲۸۰ الی ۴۴۰ برابر نسبت به رویکرد سنتی پایگاه داده است.

کلمات کلیدی: رایانش ابری، ذخیره‌سازی ابری، حافظه اصلی پایگاه داده، داده گرم/سرد، مدیریت داده سرد.

The presentation of new hybrid compression techniques to optimize memory usage and speed of access in cloud database

Seyed Behnam Hoseini[1]

[1]Pooyesh University, Qom, Iran.

Abstract

Enterprises and cloud service providers face dramatic increase in the amount of data stored in private and public clouds. Thus, data storage costs are growing hastily because they use only one single high-performance storage tier for storing all cloud data. There,s considerable potential to reduce cloud costs by classifying data into active (hot) and inactive (cold). In the main-memory databases research, recent work focus on approaches to identify hot/cold data. Most of these approaches track tuple accesses to identify hot/cold tuples. In contrast, we introduce a novel LOAD DATA INFILE that tracks both tuples and columns accesses in secondary storage databases. Our objective is to enhance the performance in terms of three dimensions: storage space, query elapsed time and CPU usage. In order to validate the effectiveness of our approach, we realized its concrete implementation on LOAD DATA INFILE Approach (LDA) that reads rows from a text file into a table at a very high speed by using the well-known qps and TPC-H benchmark. Experimental results show that the proposed LOAD DATA approach outperforms prepare_data in respect of two performance dimensions. In specific, LOAD DATA reduces the storage space by average of 14-62% and reduces the query elapsed time by average of 280-440 times compared to the traditional database approach.

Keywords: Cloud computing, Cloud storage, Main-memory database, Hot/cold data, Cold data management.

تاریخچه مقاله:

تاریخ ارسال: ۱۴۰۰/۰۱/۱۶

تاریخ اصلاحات: ۱۴۰۰/۰۳/۰۷

تاریخ پذیرش: ۱۴۰۰/۰۳/۲۳

تاریخ انتشار: ۱۴۰۰/۰۶/۳۱

Keywords:

Cloud computing,
Cloud storage,
Main-memory database,
Hot/cold data,
Cold data management

*ایمیل نویسنده مسئول:
Behnam.hoseini1989@gmail.com

۱ – مقدمه

حافظه اصلی پایگاه داده (MMDB)[1]، یک مدیر سیستم پایگاه داده است که برای ذخیره‌سازی داده‌های کامپیوتر بر روی حافظه اصلی تکیه می‌کند. سیستم‌های مدیریت پایگاه داده با استفاده از مکانیزم ذخیره‌سازی دیسک مقایسه می‌شوند. پایگاه داده‌های حافظه اصلی، سریع‌تر از پایگاه داده‌های مبتنی بر دیسک هستند. چونکه دارای الگوریتم بهینه‌سازی داخلی ساده‌تری هستند و دستورالعمل‌های CPU کم‌تر بر روی آن‌ها اجرا می‌شوند. زمانی که داده‌ها را جست‌وجو می‌کنید، دسترسی به داده‌ها در حافظه از بین می‌رود. و کارایی پیش‌بینی شده دیسک سریع‌تر و بیش‌تر فراهم می‌شود [۱].

با توجه به ارزیابی‌های اخیر که در مورد اندازه حافظه اصلی صورت گرفته است، افزایش حجم عظیم داده‌ها در سیستم‌های پایگاه داده، موجب نگهداری کل پایگاه داده در حافظه شده است. با این وجود، حافظه اصلی هنوز یک منبع کم‌یاب و گران‌قیمت در مقایسه با دیسکت (ذخیره‌سازی ثانویه) است [۲]. عمده اهداف تحقیقات اخیر، بهینه‌کردن فضای ذخیره‌سازی حافظه اصلی است. حافظه آزاد بیش‌تر برای سیستم‌های بزرگ‌تر، کارایی و عملکرد ذخیره‌سازی را در پایگاه داده بهبود می‌بخشد و موجب بهره‌وری هزینه نیز می‌گردد. هدف ما این است که داده‌ها را به دو دسته داده‌های گرم (فعال) و داده‌های سرد (غیرفعال) جدا کنیم. داده‌های گرم در حافظه اصلی باقی می‌مانند و آن‌هایی که سرد هستند نیز به محل ذخیره‌سازی کم ارزش‌تر سرد نقل مکان می‌کنند و در آنجا ذخیره می‌شوند [۳]. تفاوت اصلی در تکنیک‌های موجود، سطح دقیق‌بودن اطلاعات در میزان سطح دسترسی و طبقه‌بندی داده‌ها به دو دسته گرم و سرد می‌باشد.، که در برخی از پایگاه‌های داده در سطح تاپل[2] ودر برخی دیگر در سطح صفحه[3] است.

رایانش ابری یکی از ابتکارات مهم عرصه فناوری اطلاعات و ارتباطات مدرن و خدمت‌رسانی برای برنامه‌های سازمانی است. و در حال حاضر تبدیل به یک معماری قدرتمند، برای به اجرا درآوردن در مقیاس بزرگ و محاسبات پیچیده شده است. از مزایای استفاده از رایانش ابری می‌توان استفاده از منابع مجازی، پردازش موازی، امنیت و یکپارچه‌سازی داده به همراه مقیاس‌پذیری ذخیره‌سازی داده‌ها را نام برد. رایانش ابری نه تنها می‌تواند هزینه و محدودیت برای هدایت و کنترل به صورت خودکار و کامپیوتری‌شدن توسط افراد و سیستم کسب و کار را به حداقل خود برساند، بلکه می‌تواند هزینه‌های مربوط به تعمیر و نگهداری زیرساخت را کم کند و مدیریت کارآمد و دسترسی کاربر را نیز فراهم‌سازی نماید [۴]. برخی از اولین پذیرندگان داده‌های بزرگ در رایانش ابری، کاربرانی هستند که خوشه‌های Hadoop را در محیط‌های محاسباتی بسیار مقیاس‌پذیر و انعطاف‌پذیر ارائه‌شده توسط فروشندگان، مانند IBM، Microsoft Azure و Amazon AWS مستقر کرده‌اند [۵]. مجازی‌سازی یکی از فناوری‌های پایه قابل اجرا برای پیاده‌سازی رایانش ابری است. اساس بسیاری از ویژگی‌های پلت‌فرم مورد نیاز برای دسترسی، ذخیره، تجزیه و تحلیل و مدیریت اجزای محاسباتی توزیع‌شده در یک محیط کلان‌داده از طریق مجازی‌سازی به دست می‌آید.

رایانش ابری بیش‌ترین تکنولوژی سازوار پذیر[4] را در زمان‌های اخیر به خود اختصاص داده است. و پایگاه داده نیز به سمت رایانش ابری گرایش پیدا کرده است، پس ما به جزئیات پایگاه داده به عنوان یک سرویس و عملکرد آن نگاهی می‌اندازیم. این مبحث شامل تمامی اطلاعات اساسی درباره پایگاه داده به عنوان یک سرویس است. کار پایگاه داده به عنوان یک سرویس و چالش‌های مواجه با آن درخور بحث ما می‌باشد. ساختار پایگاه داده در رایانش ابری و کار آن در همکاری با گره‌ها تحت عنوان پایگاه داده به عنوان یک سرویس اشاره شده است. این موضوع هم‌چنین نکات جالب و مهم را قبل از اتخاذ یک پایگاه داده به عنوان یک سرویس به یاد داشته است. و بهترین را در میان دیگر موارد فراهم می‌آورد. مزایا و معایب پایگاه داده به عنوان یک سرویس به شما اجازه می‌دهد در این مورد تصمیم بگیرید که از پایگاه داده به عنوان یک سرویس استفاده کنم و یا خیر. پایگاه داده به عنوان یک سرویس، در حال حاضر توسط بسیاری از شرکت‌های تجارت الکترونیک اتخاذ گردیده است و این شرکت‌ها در حال گرفتن سود حاصل از این سرویس هستند [۶].

بعضی اوقات پایگاه داده MySQL نیاز به کار سخت دارد. ما روی یک اسکریپت وارداتی کار کرده‌ایم که مقدار زیادی داده INSERT می‌کند. معمولا سرور پایگاه داده ۱۰۰۰، INSERT را در هر ثانیه مدیریت می‌کند. باز این تعداد کافی نبود. بنابراین، به دنبال روش‌هایی برای بهبود سرعت ورودی MySQL رفتیم و در نهایت توانستیم این تعداد را به ۲۲،۲۳۲ INSERT در هر ثانیه افزایش دهیم.

مشارکت‌های این مقاله را می توان به صورت زیر خلاصه کرد:

[1] Main-Memory Database
[2] tuple-level
[3] page-level
[4] adaptive

۱. طراحی سه جدول با استفاده از دو روش سنتی و مدرن در ایجاد پایگاه داده ابری و تمرکز به روی موتور ذخیره‌سازی InnoDB.
۲. معرفی رویکرد LOAD DATA و مطالعه موردی دقیق آن.
۳. بهبود عملکرد و کارایی پایگاه داده MySQL و افزایش سرعت با رویکرد LOAD DATA.

در ادامه این مقاله به شرح زیر سازماندهی شده است. بخش ۲ به صورت کلی به بررسی کارهای مربوطه اخیر می‌پردازد. بخش ۳ رویکرد LOAD DATA را معرفی می‌کند. بخش ۴ مطالعه موردی دقیقی را برای نشان‌دادن گردش کاری روش پیشنهادی ارائه می‌کند. بخش ۵ ارزیابی تجربی رویکرد پیشنهادی را گزارش می‌کند. در نهایت، بخش ۶ مقاله را با نتیجه‌گیری به پایان می‌رساند.

۲ – کارهای مربوطه

توسعه‌های اخیر در سخت‌افزار باعث شده است که ما شاهد سرعت بالای کاهش قیمت‌ها در بازار حافظه اصلی در سال‌های گذشته باشیم.این توسعه از لحاظ اقتصادی برای استفاده از حافظه اصلی امکان‌پذیر بوده که به عنوان داده‌های اولیه در DBMS[5] ذخیره می‌شوند.، که ویژگی اصلی آن حافظه اصلی DBMS است. بیش‌تر تحقیقات اخیر بر روی ذخیره‌سازی حافظه اصلی DBMS صورت گرفته است.

سیستم‌های تجاری شامل Oracle's Times Ten [7]، IBM's solideDB [8] و VoltDB [9] هستند. از سوی دیگر، HYRISE [10]، H-Store [11]، HyPer [12] و MonetDB [13] سیستم‌های تحقیق می‌باشند. این سیستم‌ها مناسب برای پایگاه داده‌ها هستند که حجم کوچکی از مقدار فیزیکی در دسترس حافظه است. اگر حافظه وجود داشته باشد، بسیاری از مشکلات به راحتی حل می‌شود. این مشکل از محدودیت ظرفیت DBMS حافظه اصلی است که توسط تعدادی از آدرس‌دهی‌ها در کارهای پیشین صورت گرفته است.

ستونی در سیستم مدیریت حافظه اصلی پایگاه داده (DBMS)ما به اسم HANA SAP [14] وجود دارد که پردازش تراکنش آنلاین[6] (OLTP) پردازش تحلیلی آنلاین[7] (OLAP)هر دو صورت می‌گیرد. این رویکرد مسئولیت رسیدگی به داده گرم و پیر را برعهده دارد [۱۵]، که به راحتی قابل دسترسی توسط مدیریت حافظه و پایگاه داده اصلی است. اما داده سرد در حافظه اصلی لود نشده و در دیسکت ذخیره می‌شود. در آن برای تمایز کردن داده گرم و سرد از روش LRU[8] استفاده می‌کند.

پایگاه داده اوراکل در بخشی از حافظه، داده‌ها را به صورت دو فرمته[9] ذخیره می‌کند [۱۶]، که مناسب برای استفاده زمان پاسخ‌دهی برنامه‌های کاربردیOLTP و هم‌چنین برنامه‌های کاربردیOLAP برای تصمیم‌گیری بلادرنگ می‌باشد. اوراکل برای پرکردن اطلاعات ستون‌هایش تکنیکی را به واسطه روش LRU برای شناسایی داده‌های گرم و سرد به کمک می‌گیرد.

HyPer ترکیبی از حافظه اصلی سیستم OLTP و OLAP می‌باشد [۱۲]. این یک رویکرد مبتنی بر فشرده‌سازی اطلاعات است که برای رسیدگی به داده‌های گرم و داده‌های سرد، مورد استفاده قرار می‌گیرد [۱۷]. در این رویکرد، نویسندگان با استفاده از قابلیت سیستم‌های سرویس‌دهنده مدرن به ردیابی و دسترسی به اطلاعات می‌پردازند. داده‌های ذخیره‌شده در یک طرح ستونی به صورت افقی تقسیم‌بندی‌شده و هر یک از پارتیشن‌ها توسط فرکانس در دسترس، طبقه‌بندی شده است. داده‌های به ندرت دیده‌شده[10] در دسته بسیار سرد قرار می‌گیرند که هنوز هم در حافظه اما به صورت فشرده، و برای استفاده بهتر از حافظه اصلی در صفحات بزرگ ذخیره می‌شوند. HyPer کار انجام طبقه‌بندی داده‌های گرم و سرد در سطح صفحه ماشین مجازی (VM) را برعهده دارد.

در [۱۸]، نویسندگان یک تکنیک ساده و کم‌سربار را پیشنهاد کرده‌اند که پایگاه داده حافظه اصلی را قادر می‌سازد تا به طور مؤثر مهاجرت داده‌های سرد به ذخیره‌سازی ثانویه با تکیه بر سیستم‌عامل (OS) مجازی‌ساز و صفحه‌بندی حافظه صورت گیرد. صفحات داغ در حالی که به حافظه متصل شده، صفحات سرد توسط سیستم‌عامل به ذخیره‌سازی سرد منتقل می‌شوند.

در [۱۹]، نویسندگان در پایگاه داده حافظه اصلی H-Store، داده‌های گرم و سرد را به صورت مجزا از هم پیاده‌سازی کرده‌اند. نویسندگان این روش را اصطلاحا "ضد ذخیره‌ساز[11]" نام نهاده‌اند که بر این موضوع تأکید دارد که داده گرم در حافظه اصلی ذخیره شده و داده سرد در حافظه ثانویه ذخیره می‌شود. برای دسترسی به هر تاپل نیز، تاپل‌ها در زنجیره LRU در هر جدول ذخیره می‌شوند.

یک روش مقایسه به نام Hekaton ارائه شده است [۲۰]، که یک حافظه بهینه‌شده SQL server، توسط موتور OLTP کار مدیریت تاپل‌های گرم و سرد را برعهده دارد. در Hekaton از اصل پایگاه داده کپی گرفته شده و در حافظه

[5]Database Management System
[6]Online Transaction Processing
[7]Online Analytical Processing

[8]Least Recently Used
[9]dual-format
[10]rarely accessed
[11]Anti-Caching

اصلی ذخیره می‌شود. داده‌های گرم در حافظه اصلی باقی مانده و داده‌های سرد به ذخیره‌سازی ثانویه انتقال پیدا می‌کنند [۲۱].

جدول ۱ به طور خلاصه، روشی را برای مدیریت و مقایسه‌ای مابین داده‌های گرم و سرد در حافظه اصلی پایگاه داده نشان می‌دهد. ما مشاهده می‌کنیم که SAP HANA [15]به صورت عمودی، داده‌ها را در یک طرح ستونی فیلتر کرده، که در زمینه‌های مختلف در طرح سطری نیز در رویکرد HFA توسعه پیدا می‌کند [۲۲]. اوراکل c12 دو فرمته[12] [۱۵] کار ذخیره‌سازی و کپی اولیه از داده‌ها را بر روی دیسکت به عهده دارد و سپس مفهوم استفاده از روش فیلترینگ ترکیبی[13] را متوجه خواهید شد. با این حال، فیلترینگ افقی[14] و فیلترینگ عمودی[15] بر روی دیسکت اعمال می‌شود، و پس از آن داده‌های گرم به سمت حافظه، به صورت ستونی حرکت کرده و ذخیره می‌شوند. در مقابل، ما رویکرد load data infile approach (LDA) را در محل اقامت داده‌ها در حافظه ثانویه، اعمال می‌کنیم.

HyPer [17, 18]کار انجام طبقه‌بندی داده‌ها (گرم/سرد) را در سطح صفحه ماشین مجازی (VM)برعهده دارد، که متفاوت از دامنه‌های ما است. در این جا است که ما ثابت کردیم که [۱۹] بهترین راه برای طبقه‌بندی داده‌ها در همان سطح از دانه دانه بودن داده‌های در دسترس، که همان سطح تاپل می‌باشد، در نظر گرفته می‌شود. در مقایسه با ضد ذخیره‌ساز [۱۹]، با استفاده از تکنیک روش LRU، به صورت افقی پایگاه داده ما فیلتر می‌شود. درحالیکه، رویکرد فیلترینگ ترکیبی (HFA) از روش کلیدی فیلترینگ "datetime" استفاده می‌کند. در نهایت، Hekaton [20]نزدیک به روش کاری ما است که از همان متولوژی فیلترینگ افقی به همراه تاپل‌های گرم استفاده می‌کند، که الگوی نرم‌افزاری ما در اینجا همان "datetime می‌باشد که کار تقسیم‌کردن داده‌ها را برعهده دارد [۲۲]. بنابراین، ما تصمیم به ساخت در کار و گسترش معماری خود، به منظور پیاده‌سازی رویکرد LDA را داریم.

پرس‌وجوی LOAD DATA سطرها را از یک فایل متنی در یک جدول با سرعت بسیار بالا می‌خواند. فایل را می‌توان از میزبان سرور[16] یا میزبان کلاینت[17] خواند، بسته به اینکه آیا اصلاح‌کننده LOCAL داده شده است. LOCAL هم‌چنین بر تفسیر داده‌ها و رسیدگی به خطا تأثیر می‌گذارد.

(جدول-۱): روش‌های مدیریت داده‌های گرم و سرد درپایگاه‌داده‌های حافظه اصلی.

Main-memory database approach	Main-memory physical layout	Hot/cold data classification	Horizontal Filtering (HF) by hot tuple	Vertical Filtering (VF) by hot column	Hybrid filtering
SAP HANA [15]	Columnar	Hot columns	NO	YES	NO
Oracle 12c dual-format [16]	Both	Hot tuples & hot columns	YES	YES	YES
HyPer [17]	Both	Hot pages	YES	YES	YES
Stoica et al. [18]	Row	Hot pages	YES	NO	NO
Anti-caching [19]	Row	Hot tuples	YES	NO	NO
Hekaton [20]	Row	Hot tuples	YES	NO	NO
Proposed HFA [22]	Row	Hot tuples & hot columns	YES	YES	YES
LOAD DATA	Row	Hot pages	YES	YES	YES

[12]Oracle 12c dual-format
[13]Hybrid Filtering Approach
[14]Horizontal Filtering
[15]Vertical Filtering
[16]OLAP workloads
[17]OLAP workloads

۳- روش پیشنهادی ترکیبی LOAD DATA INFILE

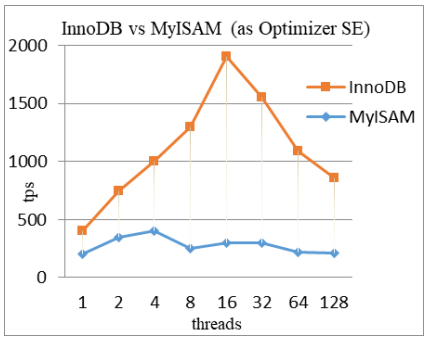
این کار نه تنها معنای درونی ما را بهبود بخشیده است، بلکه باعث قوی‌تر شدن و قابلیت اعتماد هر چه بیش‌تر پایگاه داده MySQL شده است، اما کارایی را نیز بهبود بخشیده است! با استفاده از گزینه

internal_tmp_disk_storage_engine،

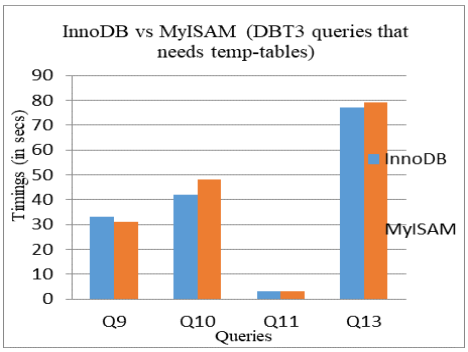
می‌توانیم به راحتی عملکرد پایگاه داده MySQL 5.7.6 را به هنگام استفاده از موتور ذخیره‌سازی MyISAM یاInnoDB برای جدول‌های موقت مبتنی بر دیسک داخلی محک بزنیم.

بیایید ابتدا به چند معیار ساده با استفاده از دو مجموعه محبوب برای محک‌زدن پایگاه داده MySQL نگاهی کنیم، برای مثال:

sysbench[18] همانطور که در شکل ۱ نشان داده شده است [۲۳] و DBT-3[19] همانطور که در شکل ۲ مشاهده می‌شود.



(شکل-۱): سیستم پردازش تراکنش در موتورهای ذخیره‌سازی MySQL با اجرای یک میلیون رکورد و حجم‌های کاری مختلف در هنگام خواندن اطلاعات.



(شکل-۲): تجزیه و تحلیل DBT3 با اندازه ۱۰ گیگابایت رکورد.

باز هم، این تست‌ها به طور مداوم نشان می‌دهند که پایگاه داده MySQL زمانی که از موتور ذخیره‌سازی InnoDB برای جدول‌های موقت مبتنی بر دیسک داخلی استفاده می‌شود، به همان اندازه خوب یا بهتر عمل می‌کند.

این کار نقش بزرگی در تلاش بزرگتر ما برای تبدیل موتور ذخیره‌سازی InnoDB به موتور ذخیره‌سازی ساده MySQL ایفا می‌کند. منتظر بهبودهای بیش‌تر MySQL باشید، زیرا ما MySQL را با یک سرور سازگار با MVCC و تراکنش‌های ACID[20] از end-to-end می‌سازیم (در حالی که به کاربران اجازه می‌دهیم تا به استفاده از موتور ذخیره‌سازی MyISAM و سایر موتورهای غیر رابطه‌ای[21] ادامه دهند، اما فقط در صورت درخواست صریح خود قادر خواهند بود).

۱-بهینه‌سازی پرس‌وجوی INSERT: برای بهینه‌سازی سرعت پرس‌وجوی INSERT، با ترکیب‌کردن بسیاری از عملیات کوچک به یک عملیات بزرگ میسر می‌گردد. در حالت ایده‌آل، شما یک اتصال واحد را ایجاد

می‌کنید، شما داده‌ها را در چندین سطر به یک باره ارسال می‌کنید. بدین ترتیب زمان تأخیر تمامی شاخص‌ها به روز رسانی می‌شود و تا زمانی که تراکنش به پایان می‌رسد، هم‌خوانی داده‌ها نیز کنترل می‌شود. زمان مورد نیاز برای وارد کردن یک سطر توسط عوامل زیر تعیین می‌شود، که در آن اعداد به صورت نسبی و تقریبی به دست آمده‌اند:

- اتصال: (۳)
- ارسال پرس‌وجو به سرور: (۲)
- تجزیه پرس‌وجو: (۲)
- وارد کردن سطر: (۱*اندازه سطر)
- وارد کردن شاخص‌ها: (۱*تعداد شاخص‌ها)
- بسته شدن: (۱)

این باعث نمی‌شود سربار ابتدایی و اولیه برای بازکردن جدول‌ها ایجاد شود. بلکه برای هر پرس‌وجو که به صورت هم‌زمان اجرا می‌شود را در نظر می‌گیرد.

اندازه جدول با وارد کردن شاخص‌ها توسط log N، با فرض شاخص‌های B-tree، کاهش پیدا می‌کند.

۲-پرس‌وجوی LOAD DATA: هنگام بارگیری جدول از یک فایل متنی، از LOAD DATA استفاده کنید. این معمولا ۲۰ برابر سریع‌تر از استفاده از دستورات INSERT است. پرس‌وجوی LOAD DATA [23] سطرها را از یک فایل متنی

[18] OLAP workloads
[19] OLAP workloads
[20] OLAP workloads
[21] OLAP workloads

در یک جدول با سرعت بسیار بالا می‌خواند. فایل را می‌توان از میزبان سرور یا میزبان کلاینت خواند، بسته به اینکه آیا اصلاح‌کننده LOCAL داده شده است. LOCAL هم‌چنین بر تفسیر داده‌ها و رسیدگی به خطا تأثیر می‌گذارد.

```
LOAD DATA
    [LOW_PRIORITY | CONCURRENT]
[LOCAL]
INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[PARTITION],(partition_name
partition_name]...)]
[CHARACTER SET charset_name]
[{FIELDS | COLUMNS}
[TERMINATED BY 'string']
 [OPTIONALLY] ENCLOSED BY 'char']
 [ESCAPED BY 'char']
 ]
 [LINES
 [STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[(col_name_or_user_var
[, col_name_or_user_var]...)]
[SET col_name={expr | DEFAULT},
[, col_name={expr | DEFAULT}]...]
```

(جدول-۲): جدول آیتم‌ها در ذخیره‌سازی ثانویه.

productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP
11	Plastic	23	good	ALBORZ	Italy	203	2.1$	3$
12	Shirt	47	good	BAS	India	650	19$	25$
13	Glass	91	good	SIMA	Iran	421	1.6$	4$
14	pen	67	good	BIC	France	270	1$	2$
15	Lamp	55	good	LIGHT	China	118	3$	4$

۴ – مطالعه موردی

در این بخش، یک مطالعه موردی دقیقی به منظور نشان‌دادن گردش کاری LOAD DATA پیشنهادی ارائه گردیده است. همانطور که در جدول ۲ مشاهده می‌کنید، جدول آیتم‌ها را نشان می‌دهد که از ۹ ویژگی تشکیل شده است.

به طور پیش‌فرض، جدول‌ها در پایگاه داده پیش‌فرض با استفاده از موتور ذخیره‌سازی InnoDB ایجاد می‌شوند. اگر جدول وجود داشته باشد، اگر پایگاه داده پیش‌فرض وجود نداشته باشد و یا اگر پایگاه داده وجود نداشته باشد، خطا رخ می‌دهد. جدول products طبق دستورات زیر ایجاد می‌شود:

```
CREATE TABLE `products` (
 productCode`VARCHAR(15)NOT NULL,
 `productName` VARCHAR(70)  NOT NULL,
 productLine` VARCHAR(50)  NOT NULL,
 productScale` VARCHAR(10)  NOT NULL,
`productVendor` VARCHAR(50)  NOT NULL,
 `productDescription`  TEXT NOT NULL,  --
64KB
```

```
`quantityInStock` SMALLINT NOT NULL,  -- Allow negative
`buyPrice` DECIMAL(8,2) UNSIGNED  NOT NULL,
    `MSRP`  DECIMAL(8,2) UNSIGNED  NOT NULL,
    PRIMARY KEY (`productCode`),
    INDEX (`productName`),
    INDEX (`productVendor`),
    INDEX (`productLine`)  -- needed to be indexed to be used as foreign key

) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

این قوانین مکان فایل ورودی LOAD DATA را تعیین می‌کنند:

اگر LOCAL مشخص نشده باشد، فایل باید بر روی میزبان سرور قرار بگیرد. و به صورت دایرکتوری توسط سرور خوانده می‌شود. سرور برای تعیین محل فایل از قوانین زیر استفاده می‌کند:

اگر نام فایل به عنوان یک مسیر مطلوب باشد، سرور از آن استفاده می‌کند.

اگر نام فایل به عنوان یک مسیر نسبی با یک یا چند اجزای عمده باشد، سرور فایل مربوط به دایرکتوری داده خود را جست‌وجو می‌کند.

اگر نام فایل بدون اجزاء باشد، سرور به دنبال فایل در دایرکتوری پایگاه داده که حالت پیش‌فرض پایگاه داده را در نظر می‌گیرد، می‌گردد.

اگر LOCAL مشخص شده باشد، برنامه کلاینت بر روی میزبان کلاینت خوانده می‌شود.، محل قرارگیری آن به شرح زیر است:

اگر نام فایل به عنوان یک مسیر مطلوب باشد، برنامه کلاینت از آن استفاده می‌کند.

اگر نام فایل به عنوان یک مسیر نسبی باشد، برنامه کلاینت به دنبال فایل مربوط به دایرکتوری فراخوانی آن می‌گردد.

۵ – ارزیابی تجربی روش LOAD DATA INFILE

به منظور اعتبار دادن به تأثیرات سیستم رویکرد LOAD DATA INFILE خود، آن و روش سنتی را پیاده‌سازی کرده‌ایم. در این بخش ما جزئیات آزمایش

صورت‌گرفته از قبیل تنظیمات آزمایش، حجم کاری، سناریوی آزمایش و در نهایت مطالعه عملکرد گزارش‌شده را خواهیم داشت.

۱- تنظیمات آزمایش:

پلت‌فرم سخت‌افزار: هسته پردازنده Intel® Core(™)2 Duo CPU (@2.53GHZ) باRAM 2GB به روی ویندوز ۷ ۶۴ بیتی در حال اجرا.

ابزارهای نرم‌افزار: MariaDB 10.1.25 به عنوان یک ابزار نرم‌افزاری برای ساخت پایگاه داده درون‌حافظه[۲۲] و جدول‌ها و اجرای پرس‌وجوها. علاوه بر این، ما از ابزار وضعیت سرور برای نظارت و مقایسه عملکرد پرس‌وجوهای خود استفاده می‌کنیم.

۲. حجم کاری

ما برای همه آزمایش‌ها، از معیار شناخته شده qps و TPC-H [24]استفاده می‌کنیم که در کارهای تحقیقاتی معتبر استفاده شده است [۱۲, ۱۴, ۱۶, ۲۲, ۲۵]. حجم کاری شامل سه جدول customers، employees و products در پایگاه داده store است. ما جدول‌ها را با داده‌ها با تجزیه و تحلیل qps پر کردیم. جدول customers دارای ۳,۱۹۹,۹۰۴ ردیف، جدول employees دارای ۶,۴۹۹,۹۰۴ ردیف و جدول products نیز دارای ۶,۴۹۹,۸۰۴ ردیف است. شکل ۳ طراحی جدول‌ها را نشان می‌دهد. جدول customers شامل ۱۱ ستون، جدول employees شامل ۱۹ ستون و جدول products نیز شامل ۹ ستون است.

[22] in-memory

store		
Customers	employees	Products
CustomerID	EmployeeID	productCode
CompanyName	LastName	productName
ContactName	FirstName	productLine
ContactTitle	Title	productScale
Address	TitleOfCourtesy	productVendor
City	BirthDate	productDescription
Region	HireDate	quantityInStock
PostalCode	Address	buyPrice
Country	City	MSRP
Phone	Region	
Fax	PostalCode	
	Country	
	HomePhone	
	Extension	
	Photo	
	Notes	
	ReportsTo	
	PhotoPath	
	Salary	

(شکل-۳): طراحی جدول‌ها

۱- سناریوی آزمایش

ما پایه تمام آزمایش‌ها را بر اساس دو نوع پرس‌وجوی زیر قرار می‌دهیم:

روش اول:

```
DELIMITER $$
CREATEPROCEDURE prepare_data()
BEGIN
 DECLARE i INT DEFAULT 100;
 WHILE i < 1700000 DO
  INSERT INTO products (productCode, productName, productLine, productScale, productVendor, productDescription, quantityInStock, buyPrice, MSRP) VALUES (i);
    SET i = i + 1;
  END WHILE;
 END$$
 DELIMITER ;
 CALL prepare_data();
```

رویکرد دوم:

```
LOAD DATA INFILE '100000 Sales Records.csv' INTO TABLE table;
```

فایل متنی Sales Records.csv حاوی اطلاعات زنده و خامی است که با استفاده از روش LOAD DATA INFILE، به سرعت اطلاعات به صورت مجموعه‌ای از پرس‌وجوها وارد می‌شوند. جریان کاری LDA به صورت خلاصه در زیر آمده است:

• ایجاد سه جدول با استفاده از تولید داده‌های تصادفی به روش عملکرد سنتی پایگاه داده برای تولید جدول‌ها و حافظه بهینه‌شده از طریق روش ترکیبی Transaction و LOAD DATA INFILE و آن‌ها را به طور کامل بر روی هارد دیسک ذخیره می‌کند.
• بررسی میزان سرعت پردازش پرس‌وجو به ازای کل زمان به‌دست‌آمده و مقایسه‌کردن آن با روش فیلترینگ ترکیبی [۲۲] و آزمایش‌های جناب آقای دکتر Kev van Zonneveld [25].
• تجزیه و تحلیل فشرده‌سازی اطلاعات و میزان مصرف CPU به ازای دریافت حجم عظیمی از سطرها.

۴- مطالعه عملکرد

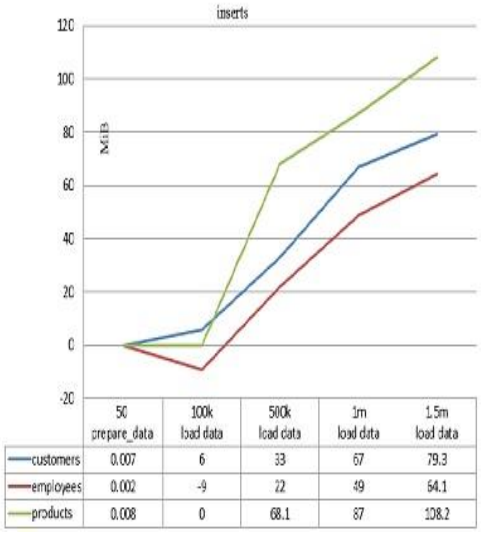
در این بخش، ما به طور آزمایشی اثربخشی رویکرد LOAD DATA INFILE را در مقایسه با رویکرد سنتی در سه عملکرد فضای ذخیره‌سازی، مدت زمان سپری‌شده پرس‌وجو و میزان مصرف CPU در موارد مختلف بررسی کرده‌ایم. در

نتیجه، در جدول‌های customers، employees و products دو روش برای واردکردن داده‌های گرم و سرد در نظر گرفته شده است. در این جدول‌ها، اطلاعات به صورت تصادفی با استفاده از تابع generate_data نوشته می‌شوند. در ادامه، با توجه به رویکرد LOAD DATA INFILE، ایجاد و با یکدیگر مقایسه شده و با وارد کردن پرس‌وجوهای متفاوت ارزیابی می‌شوند.

A-بعد فضای ذخیره‌سازی

در این بخش، ما به جزئیات میزان فضای ذخیره‌شده حاصل از سطرهای ایجاد شده می‌پردازیم. همانطور که در شکل ۴ مشاهده می‌کنید، حجم فضای ذخیره‌شده برای یکصد هزار، پانصد هزار و یک و نیم میلیون رکورد در جدول customers عبارتست از ۶، ۳۳ و ۷۹.۳ مگابایت. این در حالی است که این اعداد در جدول‌ها با استفاده از تابع prepare_data به ۰.۰۰۷، ۰.۰۰۲ و ۰.۰۰۸ مگابایت می‌رسد. یکی از مزایای استفاده از پایگاه داده رابطه‌ای MySQL این است که با وجود حجم عظیمی از سطرها، هیچ گونه حجم سربار اضافی را نداریم.

در شکل ۴، با توجه به نتایج به‌دست‌آمده، در رویکرد LOAD DATA INFILE، فضای ذخیره‌سازی به‌طور متوسط ۱۴ الی ۶۲ درصد نسبت به رویکرد سنتی پایگاه داده بهینه‌سازی شده است.



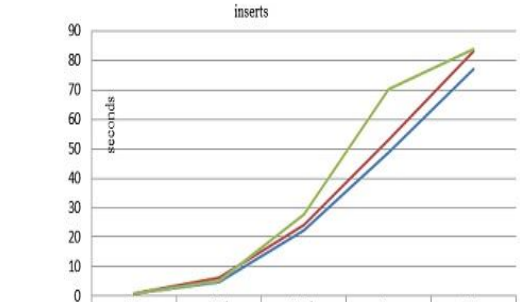
	50 prepare_data	100k load data	500k load data	1m load data	1.5m load data
customers	0.007	6	33	67	79.3
employees	0.002	-9	22	49	64.1
products	0.008	0	68.1	87	108.2

(شکل-۴): بهبود فضای ذخیره‌سازی برای همه جدول‌ها

B. بعد زمان سپری‌شده پرس‌وجو

همانطور که در شکل ۵ مشاهده می‌کنید، مدت زمان سپری‌شدن پرس‌وجو برای یکصد هزار، پانصد هزار و یک و نیم میلیون رکورد در جدول customers عبارتست از ۴.۹۱، ۲۲.۴۹ و ۷۶.۹۶ ثانیه. این در حالی است که با استفاده از رویکرد سنتی پایگاه داده برای تهیه جدول‌ها، به طور میانگین و تقریبی در هر ثانیه ۵۰ پرس‌وجو پردازش می‌شود.

همانطور که در شکل ۵ نشان داده شده است، Load Data این اجازه را به MySQL می‌دهد که به صورت متناوب حدود ۲۲،۲۳۲ پرس‌وجوی INSERT را در هر ثانیه پردازش نماید. در روش Load Data هیچ میزان حجم سربار قابل توجهی وجود ندارد. در شکل ۵، با توجه به نتایج به دست آمده، در رویکرد LOAD DATA INFILE، مدت زمان سپری‌شده پرس‌وجو به طور متوسط ۲۸۰ الی ۴۴۰ برابر نسبت به رویکرد سنتی پایگاه داده بهینه‌سازی شده است.

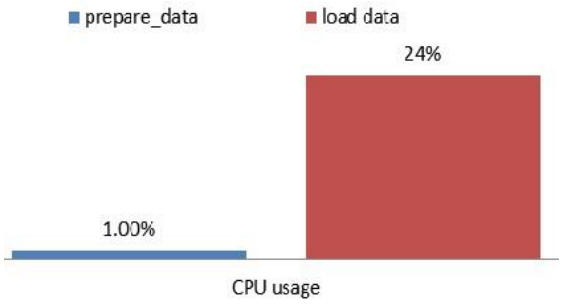


	50 prepare_data	100k load data	500k load data	1m load data	1.5m load data
customers	1	4.91	22.49	48.74	76.96
employees	1	6.24	23.98	53.03	83.18
products	1	5.25	27.74	70.19	83.86

(شکل-۵): بهبود مدت زمان سپری‌شده برای همه جدول‌ها.

C. بعد مصرف CPU

در این آزمایش، ما میزان مصرف CPU با استفاده از LOAD DATA INFILE پیشنهادی را در مقایسه با روش سنتی در ذخیره‌سازی ثانویه پایگاه داده بررسی می‌کنیم. همانطور که در شکل ۶ مشاهده می‌کنید، میانگین استفاده از CPU برای تهیه جدول در رویکرد LOAD DATA پیشنهادی بیست و چهار درصد و در رویکرد سنتی تنها یک درصد است. آنچه مشخص است این است که در روش‌های سنتی، به دلیل سرعت پایین‌تر، عمدتاً شاهد استفاده کم‌تر از CPU نسبت به رویکردهای مدرن بودیم.



(شکل-۶): بهبود مصرف CPU برای همه جدول‌ها.

۶ – نتیجه‌گیری و پژوهش‌های آتی

با توجه به چالش‌های بودجه‌ای ذخیره‌سازی عظیم داده‌ها در فضای ابری، به صورت گرم و یا سرد شناسایی شده و روند قابل توجهی را در پیش گرفته است. برای همکاری و مشارکت در این تحقیق، هدف ما از بهینه‌سازی فضای مورد نیاز برای ذخیره‌سازی، بررسی کاهش هزینه‌ها در حافظه ثانویه پایگاه داده‌های ابری است.

ما یک تجزیه و تحلیل جامع از ذخیره‌سازی ثانویه پایگاه داده‌های موجود با تمرکز بر روی مدیریت داده‌های سرد انجام دادیم. ما یک رویکرد جدید LOAD DATA INFILE Approach (LDA) پیشنهاد کردیم که با استفاده از موتور ذخیره‌سازی InnoDB، سطرها را از یک فایل متنی در یک جدول با سرعت بسیار بالا می‌خواند. ما جریان کاری آن را از طریق یک مطالعه موردی دقیق نشان دادیم و اثربخشی LDA را با استفاده از معیار qps و TPC-H مورد ارزیابی قرار دادیم. نتایج تجربی ثابت کرد که رویکرد LDA پیشنهادی در مقایسه با روش prepare_data، رویکرد بهتری را برای ذخیره سطر پایگاه‌داده در ذخیره‌سازی ثانویه نسبت به دو بعد کارایی دارد. LDA پیشنهادشده، فضای ذخیره‌سازی را به طور متوسط ۱۴ الی ۶۲ درصد کاهش می‌دهد و زمان سپری‌شده پرس‌وجو را نیز به طور متوسط ۲۸۰ الی ۴۴۰ برابر نسبت به رویکرد سنتی پایگاه داده کاهش می‌دهد.

۷- مراجع

[1] Gupta, M.K., V. Verma, and M.S. Verma, "In-memory database systems-a paradigm shift." arXiv preprint arXiv:1402.1258, 2014.

[2] Arora,ف I. and A. Gupta, "Improving performance of cloud based transactional applications using in-memory data grid." International Journal of Computer Applications, 2014. 107(13).

[3] Boissier, M. "Optimizing main memory utilization of columnar in-memory databases using data eviction." in Proc. VLDB Ph. D. Workshop. 2014.

*[4] Hashem, I.A.T., et al., "The rise of "big data" on cloud computing: Review and open research issues." Information systems, 2015. **47**: p. 98-115.*

[5] Liu, C., et al. "Public auditing for big data storage in cloud computing--a survey." in 2013 IEEE 16th International Conference on Computational Science and Engineering. 2013. IEEE.

*[6] Al Shehri, W., "Cloud database database as a service." International Journal of Database Management Systems, 2013. **5**(2): p. 1.*

*[7] Lahiri, T., M.-A. Neimat, and S. Folkman, "Oracle TimesTen: An In-Memory Database for Enterprise Applications." IEEE Data Eng. Bull., 2013. **36**(2): p. 6-13.*

*[8] Lindström, J., et al., "IBM solidDB: In-Memory Database Optimized for Extreme Speed and Availability." IEEE Data Eng. Bull., 2013. **36**(2): p. 14-20.*

*[9] Stonebraker, M. and A. Weisberg, "The VoltDB Main Memory DBMS." IEEE Data Eng. Bull., 2013. **36**(2): p. 21-27.*

*[10] Grund, M., et al., "Hyrise: a main memory hybrid storage engine." Proceedings of the VLDB Endowment, 2010. **4**(2): p. 105-116.*

*[11] Kallman, R., et al., "H-store: a high-performance, distributed main memory transaction processing system." Proceedings of the VLDB Endowment, 2008. **1**(2): p. 1496-1499.*

[12] Kemper, A. and T. Neumann. "HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots." in 2011 IEEE 27th International Conference on Data Engineering. 2011. IEEE.

[13] Boncz, P.A., M. Zukowski, and N. Nes. "MonetDB/X100: Hyper-Pipelining Query Execution." in Cidr. 2005. Citeseer.

*[14] Färber, F., et al., "SAP HANA database: data management for modern business applications." ACM Sigmod Record, 2012. **40**(4): p. 45-51.*

[15] Archer, S., "Data-aging strategies for SAP NetWeaver BW focusing on BW's new

NLS offering for Sybase IQ." SAP BW Product Management, 2013.
[16] Colgan, M., J. Kamp, and S. Lee, "Oracle database in-memory." Oracle White Paper, 2014.
[17] Funke, F., A. Kemper, and T. Neumann, "Compacting transactional data in hybrid OLTP & OLAP databases." arXiv preprint arXiv:1208.0224, 2012.
[18] Stoica, R. and A. Ailamaki. "Enabling efficient OS paging for main-memory OLTP databases." in Proceedings of the Ninth International Workshop on Data Management on New Hardware. 2013.
*[19] DeBrabant, J., et al., "Anti-caching: A new approach to database management system architecture." Proceedings of the VLDB Endowment, 2013. **6**(14): p. 1942-1953.*
[20] Diaconu, C., et al. "Hekaton: SQL server's memory-optimized OLTP engine." in Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. 2013.
[21] Delaney, K., "SQL Server in-memory OLTP internals overview." White Paper of SQL Server, 2014.
*[22] Afify, G.M., A. El Bastawissy, and O.M. Hegazy, "A hybrid filtering approach for storage optimization in main-memory cloud database." Egyptian Informatics Journal, 2015. **16**(3): p. 329-337.*
[23] Sasmito, G.W. and M. Nishom, "An Efficient Method for Speeding up Large-Scale Data Transfer Process to Database: A Case Study."
[24] Revision, T.B.H.S.S., 2.17. 1. Transaction processing performance council, 2014.
[25] Mardan, A., "Database, Keys and Stream Tips," in Pro Express. js. 2014, Springer. p. 161-170.

سید بهنام حسینی فارغ التحصیل ممتاز مقاطع کارشناسی و کارشناسی ارشد در رشته مهندسی کامپیوتر گرایش نرم افزار به ترتیب از دانشگاه آزاد اسلامی تهران شرق و موسسه آموزش عالی پویش. ایشان در حال حاضر شروع به یادگیری دوره ISMV4 در موسسه آموزش عالی آزاد ارژنگ نموده‌اند. علاقه مندی های پژوهشی ایشان عبارتند از سیستم های بانک اطلاعاتی، رایانش ابری و امنیت شبکه.

رایانامه ایشان: Behnam.hoseini1989@gmail.com

روش ارجاع به مقاله : ب. حسینی. ارائه روش نوین فشرده سازی ترکیبی برای بهینه سازی مصرف حافظه وسرعت دسترسی در پایگاه داده ابری. دوفصلنامه محاسبات و سامانه های توزیع شده سال چهارم، شماره اول، شماره پیاپی ۶، صفحه ۱ تا ۱۱، سال ۱۴۰۰.

How to cite: Seyed Behnam Hoseini. The presentation of new hybrid compression techniques to optimize memory usage and speed of access in cloud database. Journal of Distributed Computing and Systems(JDCS), Vol 4, Issue 1, Page 1-11, 2021.