دوفصلنامه محاسبات و سامانه‌های توزیع شده
www.jdcs.ir

بهینه سازی پرس‌وجو در موتور پرس‌وجوی کلان‌داده Hive

میثم پاسداری هریس[۱] (*)، داود محمدپورزنجانی[۲]
[۱]دانشجوی کارشناسی ارشد مهندسی نرم‌افزار، گروه کامپیوتر، دانشگاه زنجان، زنجان
[۲]عضو هیئت علمی، گروه کامپیوتر، دانشگاه زنجان، زنجان

چکیده

در این پژوهش سعی گردیده‌است تا چالش‌های مطرح در زمینه بهینه‌سازی پرس‌وجوهای محیط‌های کلان داده از جمله نحوه اجرای پرس‌وجو بر روی داده‌های با حجم زیاد مورد بررسی قرار گیرد. معماری و ساختار موتور پرس‌وجوی هایو به تفضیل مورد بررسی قرار گرفته شده است و به نحوه بهینه سازی پرس‌وجوها که به طور مشخص توسط واحدهای کامپایلر و بهینه‌ساز انجام می‌شود، اشاره گردیده است. در نهایت روش مبتنی بر همبستگی و همچنین روش پیشنهادی به صورت جزئی با مثال‌های مختلف مورد بررسی قرار گرفته است. همچنین در انتها، نتیجه کارآمدی روش‌های کلی در قالب یک نمودار تحلیلی آورده شده است.

کلمات کلیدی: کلان‌داده، موتور پرس‌وجو، بهینه‌سازی پرس‌وجو، Hive ،بهینه‌سازی مبتنی بر همبستگی

تاریخچه مقاله:

تاریخ ارسال: ۹۷/۴/۱
تاریخ اصلاحات: ۹۷/۵/۱
تاریخ پذیرش: ۹۷/۵/۵
تاریخ انتشار: **۹۷/۵/۱۵**

Keywords:

Big Data
Hive Query engine
Query Optimization
Correlation-based Query Optimization

Query Optimization in Hive Big Data Query Engine

Meysam Pasdari Heris[1] (*),
Davoud Mohammadpour Zanjani[2]
[1]M.Sc. Student, Department of Computer Engineering, University of Zanjan, Zanjan
[2]Faculty member, Department of Computer Engineering, University of Zanjan, Zanjan

Abstract

In this research, efforts have been made to address the challenges posed by query optimization in big data environments, including how to improve queries on large volume of data. The architecture of the Hive query engine are investigated in detail, and are referred to how to optimize queries that are specifically done by the compiler and optimizer units. The details of the optimization methods are pretty discussed and, at the end, the correlation-based approach is described in detail with various examples. Finally, the result of the efficiency of the general methods is presented in the form of an analytical chart.

روش ارجاع به مقاله: م.پاسداری هریس، د.محمدپور نجانی، بهینه سازی پرس‌وجو در موتور پرس‌وجوی کلان‌داده Hive، دوفصلنامه محاسبات و سامانه‌های توزیع شده، سال اول، شماره اول، ص ۹۲-۱۰۸، سال انتشار ۱۳۹۷.

* Corresponding author : نویسنده ی عهده دار مکاتبات

۱- مقدمه

پژوهشگران، سازمان‌ها و افراد فعال در حوزه کلان داده ویژگی‌های متفاوتی از کلان داده ارائه داده‌اند. برای مثال موسسه گارتنر سه ویژگی حجم، نرخ تولید و تنوع [1] را به عنوان خصوصیات محیط کلان داده معرفی می‌کند. شرکت IBM علاوه بر این سه خصوصیت، صحت [2] را نیز به عنوان یکی دیگر از خصوصیات محیط کلان داده معرفی کرده است. و یا در تعریفی دیگر علاوه بر خصوصیات بالا از ارزش نیز به عنوان یکی از ابعاد کلان داده یاد می‌شود. شرکت مایکروسافت نیز ویژگی‌های حجم، تنوع، نرخ تولید، صحت، تغییر پذیری و قابل نمایان [3] بودن داده‌ها را نیز به عنوان خصوصیات محیط کلان داده معرفی کرده است.

در یک تعریف مختصر کلان داده اصطلاحی است که برای مجموعه داده‌های بزرگ و با تنوع داده‌ای زیاد استفاده می‌شود که ذخیره و پردازش این مجموعه از داده‌ها، با تکیه بر ابزارهای قابل دسترس موجود مدیریت پایگاه داده و یا راه‌حل‌های سنتی پردازش داده، مشکل است. چالشی که کلان داده با آن روبه‌رو است ثبت کردن، سازمان‌دهی، ذخیره‌سازی، جستجو، اشتراک گذاری، انتقال، تجزیه و تحلیل و مصورسازی این داده‌ها می‌باشد.

در ادامه سعی خواهد شد به بحث پرس‌وجو بر روی حجم عظیم داده‌ها در فضای کلان‌داده پرداخته شود. این نگاه به پرس‌وجو با توجه به چالش‌های مطرح شده در حوزه کلان‌داده متفاوت از نگاه سنتی می‌باشد، چرا که در پایگاه‌داده‌های سنتی اغلب ماهیت داده‌ها مشخص بوده و در مقیاس کوچکی است اما داده‌های تولید شده در مقیاس‌های بزرگ در زمینه‌های مختلف از جمله صنعت، پزشکی، شبکه‌های اجتماعی و... معمولا با رویکردهای سنتی قابل مدیریت و پرس‌وجو با سرعت مطلوب نیستند. بنابراین تغییر مقیاس و نیازمندی‌ها بر معماری‌ها نیز اثرگذار بوده و رویکردهای نوین را می‌طلبد.

در همین راستا آپاچی هدوپ[1] یکی از محبوب‌ترین و متداول‌ترین راه‌حل‌هایی است که برای مدیریت و پردازش داده‌ها بکار می‌رود. هدوپ یک چارچوب برنامه‌نویسی متن‌باز و مبتنی بر جاوا است، که ذخیره‌سازی و پردازش مجموعه داده‌های بسیار بزرگ را در یک محیط محاسباتی توزیع شده[2]، محقق می‌کند. هدوپ بخشی از پروژه‌ی آپاچی است که توسط بنیاد نرم‌افزار آپاچی حمایت می‌شود. هدوپ با تکیه بر سیستم پردازشی توزیعی خود، حجم عظیمی از داده‌های ساختاریافته و بدون ساختار را، به شیوه‌ای بسیار بهینه‌تر از شیوه‌ی سنتی انبار داده‌ی سازمان‌ها، مدیریت می‌کند.

با توجه به چالش‌های مطرح شده و نیاز به پردازش پرس‌وجوها بر روی کلان داده‌ها، پروژه‌های مختلفی از جمله Pig، Hive، Presto، Spark، Impala توسط شرکت‌های مختلف تعریف گردیده است. لازم به ذکر است که همگی این پروژه‌ها با توجه به نیازمندی‌های واقعی ایجاد و توسعه داده شده است. به عنوان نمونه Presto موتور پرس‌وجویی است که توسط شرکت فیسبوک برای رفع نیازمندی‌های شبکه اجتماعی

[1] Apache Hadoop
[2] Distributed Computation Environment

فیسبوک ارائه شده است. به عنوان مثال امکان پرس‌وجوی تعاملی نیازمند توسعه ابزاری بود تا بشود بر روی حجم عظیم داده‌ها در مدت زمان کم، پاسخ مناسبی دریافت گردد، به همین دلیل این موتور پرس‌وجو توسعه داده شد.

یکی از موتورهای پرس‌وجوی معروف و مهم دیگر در زمینه کلان داده Hive می‌باشد که بر روی هدوپ توسعه داده شده است. در ادامه به بررسی دقیق تر این موتور پرداخته خواهد شد.

۲- موتور پرس‌وجوی Hive

این موتور پرس‌وجو در سال ۲۰۰۹ توسط فیسبوک توسعه داده شده است و هم اکنون تحت نظر بنیاد آپاچی قرار دارد. هایو به جهت SQL-Like بودن زبان پرس‌وجویی تحت عنوان HQL دارد و از پرس‌وجوهای رابطه‌ای پشتیبان می‌کند. در حقیقت هایو موتور پرس‌وجویی است که برای تحلیل داده‌ها و در واقع عملیات OLAP در بستری که داده‌ها به صورت حجیم و توزیع شده باشند، به کار می‌رود. اما برای تحلیل داده‌های OLTP گزینه مناسبی نیست.

اغلب، پایگاه‌داده‌های رابطه‌ای از نوع ساختار در زمان نوشتن، هستند، یعنی زمانیکه داده وارد جدول می‌شود از لحاظ ساختاری بررسی می‌شود تا مطابق ساختار باشد، بنابراین این اقدام موجب تحمیل سربار اضافی در زمان نوشتن داده شده و باعث کاهش سرعت در زمان درج داده‌ها در پایگاه‌داده‌های رابطه‌ای می‌شود. اما هایو به صورت ساختار در زمان خواندن می‌باشد بدین صورت که ساختار در زمان دریافت داده‌ها بررسی نمی‌گردد و از این نظر سرعت بارگزاری داده‌ها زیاد خواهد بود [1].

در مقایسه روش‌ها و سیستم‌های رابطه‌ای با توجه به معیار ساختاری سیستم، در تکمیل نکات گفته شده می‌توان به موارد زیر اشاره کرد:

- **ساختار در زمان نوشتن**[3]

در سیستم‌های رایج رابطه‌ای به هنگام درج داده جدید همخوانی ساختار با داده‌های دریافتی به صورت سخت‌گیرانه مورد بررسی قرار می‌گیرد و در صورت عدم همخوانی از ورود داده صرف‌نظر می‌شود. به این حالت ساختار در زمان نوشتن اطلاق می‌گردد.

مزیت این روش این است که با توجه به ایجاد شاخص‌ها بر روی ستون‌ها و رکوردها کارایی را از نقطه نظر سرعت اجرایی پرس‌وجوها ارتقا می‌دهد اما در مقابل، سرعت درج اطلاعات در سیستم را نیز به شدت کاهش می‌دهد. در واقع این روش برای سیستم‌هایی که نرخ نوشتن در آنها بالا است مناسب نیست.

- **ساختار در زمان خواندن**[4]

سیستم هایو به هنگام ذخیره سازی و ورود داده‌ها این سخت‌گیری را اعمال نمی‌کند بلکه به هنگام بازیابی اطلاعاتی ساختار را بررسی می‌کند که به این ویژگی ساختار در زمان خواندن گفته می‌شود. این نوع نگرش به ساختار اساساً به این دلیل است که بتوان

[3] Schema on write
[4] Schema on read

سرعت بارگذاری اولیه[5] بالایی را فراهم نمود. به عنوان مثال، دیگر نیازی به تجزیه، سریالی‌سازی موارد در سطح دیسک نیست، این سخت گیری کاهش یافته به نوعی انعطاف سیستم را بالا برده و برای مواردی که در آنها نرخ نوشتن بالاست مناسب است.

۳- مدل داده‌ای

نماشی از مدل داده‌ای Hive در شکل ۱ آورده شده‌است. در این بخش، توضیحات مختصری در خصوص اجزاء این مدل داده‌ای ارائه شده است.

جداول

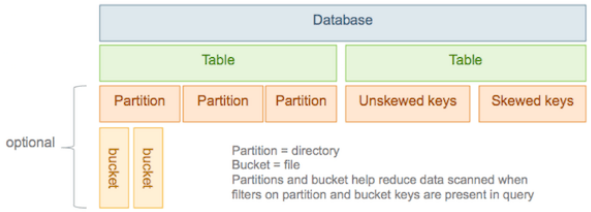
هر جدول یک دایرکتوری متناظر در سیستم فایل HDFS [6] دارد، بدین صورت که داده‌های جدول به صورت سریالی مرتب گردیده و در قالب فایل در دایرکتوری مورد نظر ذخیره می‌گردد. هایو عملیات سریالی کردن را به صورت داخلی و نهفته انجام می‌دهد، به طوری که فرمت این فرآیند سریالی سازی نیز در بخشی تحت عنوان کاتالوگ سیستم نگه‌داری می‌گردد. این فرمت سریالی سازی در فرآیند کامپایل و اجرای پرس‌وجو به کار برده می‌شود. هایو همچنین از جداول خارجی که بر روی HDFS، NFS و حتی دایرکتوری‌های محلی ذخیره شده اند، پشتیبانی می‌کند [1]. هر رکورد در این جدول‌ها بسیار به مدل‌های رابطه‌ای شبیه است که امکان اجرای عملیات DDL بر روی آن‌ها وجود دارد [6].

پارتیشن‌ها

هر جدول همچنین می‌تواند دارای یک یا چندین پارتیشن باشد، که این پارتیشن‌ها نحوه توزیع شدگی داده را در زیر دایرکتوری‌های دایرکتوری اصلی جدول نشان می‌دهد[4] .

باکت‌ها

همچنین داده‌ها موجود در باکت‌ها می‌توانند به بخش‌های تحت عنوان باکت‌ها بر اساس Hash یک ستون در جدول تقسیم بندی گردد. هر باکت نیز در دایرکتوری هر پارتیشن در قالب فایل ذخیره می‌گردد [4].



شکل ۱ مدل داده‌ای در Hive [1]

۴- بستر Hadoop

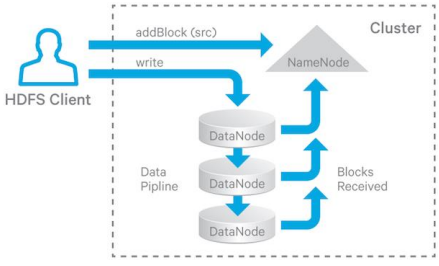
هدوپ یک بستر متن باز برای پردازش، ذخیره و تحلیل حجم عظیم داده های توزیع شده و بدون ساختار است . [9]منشاء اصلی پیدایش این چهارچوب پردازشی به شرکت‌های جستجوی اینترنتی یاهو و گوگل باز می گردد که برای شاخص‌گذاری صفحات وب و جستجوی آنها نیاز به ابزار و مدل های جدید پردازشی داشتند. هدوپ تلاش دارد تا فرآیندها را به صورت توزیع‌شده در کنار داده‌ها انجام دهد. بنابراین با توجه به رویکرد توزیع‌شدگی امکان اجرای فرآیندها و ذخیره سازی داده‌ها بر روی گره‌های مختلف فراهم می‌شود [10]. این رویکرد موجب شده است تا هدوپ علاوه بر اینکه بتواند حجم بالایی از داده‌ها و پردازش را اجرا و

[5] Fast initial load
[6] Hadoop Distributed File System

مدیریت کند، بلکه امکان بهره‌مندی از سخت افزارهای ارزان قیمت[7] را نیز فراهم کند . همچنین با بهره‌مندی از گره‌های توزیع شده قابلیت اطمینان[8] بالایی را نیز تضمین می‌کند. امروزه هدوپ بصورت وسیع در زمینه های بسیاری از فعالیت‌های دانشگاهی تا تجارت و صنعت ، از علوم تا نجوم مورد استفاده قرار می‌گیرد [9].

۵- سیستم فایل توزیع‌شده هدوپ[9]

هدوپ با توجه به معماری توزیع‌شده، از سیستم مدیریت فایل توزیع شده نیز برخوردار است. به صورتی که داده‌ها را در قالب گره‌هایی که بخشی از یک سیستم توزیع شده اند، ذخیره، بازیابی و مدیریت می‌کند. گره‌ها به صورت قویاً همبند[10] به یکدیگر متصل می‌باشند. فراداده‌ها بر روی یک سرور اختصاصی ذخیره تحت عنوان گره نام[11] ذخیره می‌گردند. داده‌ها نیز بر روی گره‌های دیگر تحت عنوان گره‌های داده[12] ذخیره می‌گردند. موارد گفته شده در تصویر ۲ نشان داده‌شده است.



تصویر ۲ معماری HDFS [10]

از ویژگی‌های بارز این سیستم فایل می‌توان به مقیاس پذیری بالای آن اشاره کرد.

۵-۱-۱- گره نام

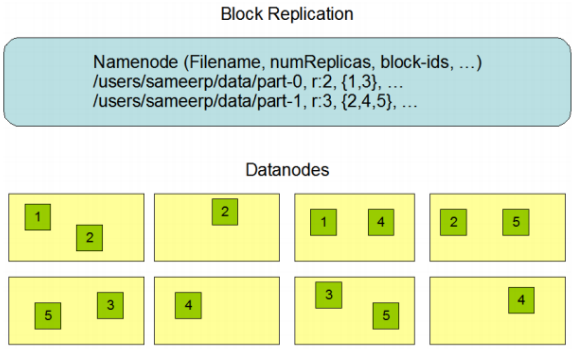
این گره فضاهای نام را مدیریت می‌کند که با مشتری HDFS مطابق تصویر ۳ در ارتباط است. به عنوان مثال وقتی درخواستی برای ذخیره یا بازیابی داده‌ها ایجاد می‌شود. این گره به دلیل اینکه با توجه به مکانیزم هایی همچون Heartbeat از وضعیت گره‌های داده‌ای اطلاع دارد و همچنین آدرس داده در گره‌های نام را نیز می‌داند، بنابراین عملیات را به سمت گره‌های نام هدایت می‌کند.

هدوپ برای تضمین کارایی رکوردهای این گره را در یک سرور اختصاصی بر روی حافظه اصلی تصادفی نگهداری می‌کند. همچنین در راستای تضمین قابلیت اطمینان ممکن است چندین کپی[13] از این گره‌ها را نیز ایجاد کرده باشد.

۵-۱-۲- گره داده

همانگونه که در تصویر ۳ مشخص گردیده‌است، گره‌های داده داده‌ها را ذخیره سازی می‌کنند. این گره‌ها تحت یک سری فضای نام ممکن است بر روی ماشین‌های مختلف توزیع شده باشد. داده‌ها نیز با توجه به ساختار توزیع شده این گره‌ها ذخیره می‌گردند. این سیستم فایل نیز برای تامین قابلیت اطیمنان چندین کپی (به صورت پیش‌فرض ۳ عدد) از داده‌ها را ایجاد می‌کند. به صورتی که رکوردهای اطلاعاتی ممکن است در چندین گره داده مختلف مطابق تصویر ۳ ذخیره شده باشند.

[7] Commodity hardware
[8] Reliability
[9] Hadoop Distributed File System
[10] Strongly connected
[11] Name node
[12] Data nodes
[13] Replication



تصویر ۳ توزیع شدگی داده‌ها در گره‌های نام

۵-۲- مدل برنامه نویسی نگاشت-کاهش [14]

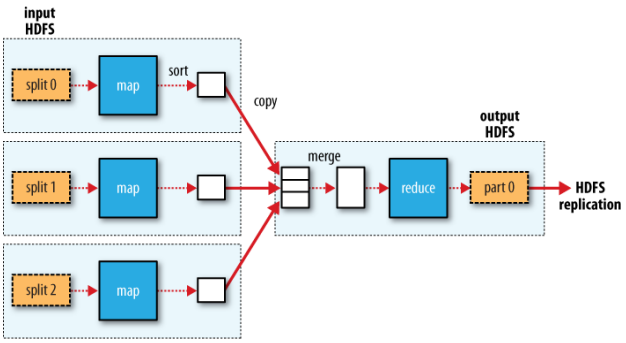
نگاشت-کاهش یک مدل برنامه نویسی است که اجازه اجرای پردازش موازی و توزیع شده بروی مجموعه بزرگی از داده‌ها در یک محیط توزیع‌شده را می‌دهد.

در مدل نگاشت-کاهش، کار میان چندین گره تقسیم شده و هر گره به طور همزمان بخشی از کار را انجام می‌دهد. بنابراین، نگاشت-کاهش براساس الگوی تقسیم و غلبه به پردازش داده‌های عظیم کمک می‌کند. همانطور که داده‌ها بوسیله چندین ماشین به جای یک ماشین بصورت موازی پردازش می‌شوند، زمان پردازش داده‌ها نیز به میزان بسیار زیادی کاهش می‌یابد [2].

یکی دیگر از مزیت‌های مدل نگاشت-کاهش محلی بودن دادهای پردازشی است. دراین مدل، داده‌های ورودی به بلاک‌هایی تقسیم شده و در گره‌های کلاستر ذخیره می‌شوند. در هنگام پردازش داده‌ها، بجای حرکت داده‌ها به سمت گره‌های پردازشگر، واحدهای پردازش به سمت داده‌ها حرکت می‌کنند. در رویکردهای سنتی، داده‌ها به سمت واحد پردازش ارسال و در آنجا پردازش می‌شدند. اما هنگام کار با داده‌های عظیم، بردن این حجم از داده‌ها به سمت واحد پردازش باعث بروز مسائل زیر می‌گردد:

- حرکت دادن داده‌های بزرگ به سمت واحد پردازش هزینه‌بر بوده و موجب اختلال در عملکرد شبکه می‌شود.
- به دلیل اینکه داده‌ها توسط یک واحد پردازش می‌شوند، این کار بسیار زمانبر شده و تبدیل به تنگنا [15] در سیستم می‌شود.
- گره پردازشگر ممکن است دارای اضافه بار شده و از کار بیفتد [16].

در تصویر ۴ جزئیات جریان داده‌ای یک فرآیند تک-کاهشی نشان داده شده است. به صورتی که فرآیند در قالب چندین نگاشت بر روی نگاشت‌گرهای مختلف پردازش شده و در نهایت در قالب یک کاهش‌گر اصطلاحا کاهش یافته است.

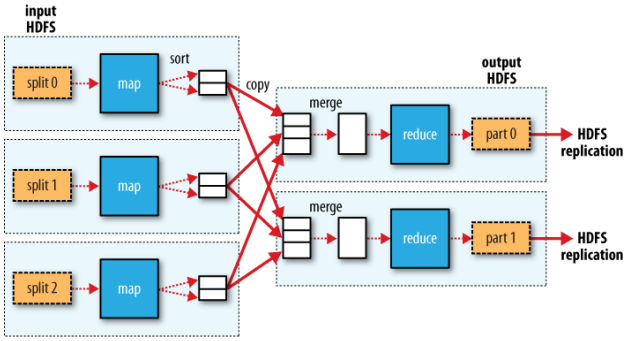


تصویر ۴ جریان داده نگاشت - کاهش مطابق فرآیندی با یک کاهش [9]

اجرای همین روال بر روی فرآیندی که نیازمند بیش از یک کاهش‌گر است به صورت زیر، نشان داده‌شده در تصویر ۵، می‌باشد.

[14] Map-reduce

[15] Bottleneck
[16] Single point of failure

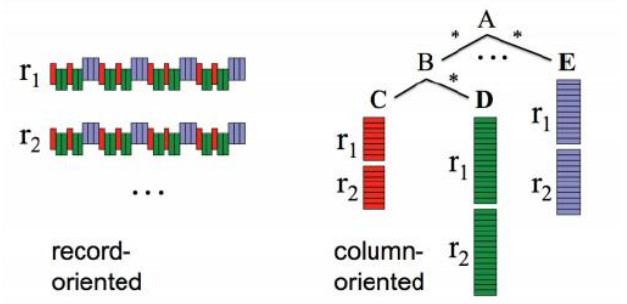


تصویر ۵ جریان داده نگاشت-کاهش مطابق فرآیندی با چندین کاهش [9]

۶- مدل ذخیره سازی

در کاربردهای امروزی بر خلاف سیستم‌های سنتی که همه مجموعه رکوردها در یک حجم از دیسک ذخیره می‌گردید، از روش ذخیره‌سازی بر مبنای ستون استفاده می‌گردد به این صورت که رکوردها در قالب ستون‌ها جدا گردیده و نهایتاً بر روی ذخیره‌سازی‌های مختلف ذخیره می‌گردند. به عنوان مثال Parquet [۱۷] با همکاری دو شرکت توئیتر و کلودرا به عنوان یک مدل ذخیره‌سازی ستونی با هدف افزایش کارآمدی در بستر هدوپ توسعه داده شد.

در تصویر ۶ دو نوع رویکرد ذخیره‌سازی رکوردی و ستونی نشان داده شده است.



تصویر ۶ رویکرد ذخیره سازی سطری و ستونی[8]

۶-۱- فایل‌های ORC [۱۸]

این نوع فایل‌ها تحت حمایت بنیاد آپاچی در راستای افزایش سرعت هایو و بهبود کارآمدی ساختار ذخیره سازی آن ارائه گردیده و تحت یک ساختار ستونی خاص در بستر HDFS ایجاد می‌شود. به عنوان مثال در حال حاضر فیسبوک با استفاده از این نوع ساختار ده‌ها پتابایت از داده‌های خود را ذخیره سازی می‌کند.

۶-۱-۱- ساختار ORC

مطابق تصویر ۷، این ساختار در اصل دارای ویژگی‌های زیر است به صورتی که از یک سری واحد ذخیره سازی تحت عنوان stripes تشکیل شده است. هر کدام از شیارها شامل یک سری از سطرها می‌باشد به طوریکه شاخص‌ها در ابتدا، سپس داده‌ها و قسمت انتهایی آن را زیربخش شامل می‌شود. زیر بخش شامل موارد زیر است:

- لیست شیارهای داده موجود در فایل
- میزان رکوردهای هر کدام از شیارها و همچنین نوع ستون‌ها
- مقادیر تجمیع‌شده در سطح ستون شامل count، min، max و sum

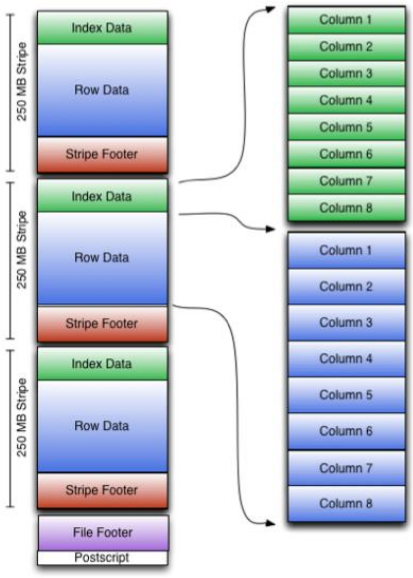
از ویژگی‌های بارز این ساختار می‌توان به موارد زیر اشاره کرد:

- کاهش زمان دسترسی
- کاهش میزان فضا برای ذخیره سازی
- جداسازی در سطح ستون

[17] https://parquet.apache.org

[18] Optimized Row Columnar

- ذخیره سازی آمار (مقادیر تجمعی max ،min و غیره)



تصویر ۷ ساختار فایل‌های ORC

۶-۱-۲- مشخصه‌های اصلی ORC

علاوه بر موارد اشاره شده در بخش قبلی، برخی از ویژگی‌های اصلی ORC به شرح زیر است:

- **پشتیبانی از ACID**

از اجرای اتمی، سازگار، ایزوله شده و پایدار تراکنش‌ها پشتیبانی می‌کند. همچنین امکان جداسازی تراکنش‌ها بر اساس روش دید لحظه‌ای را نیز فراهم می‌کند [8].

- **پشتیبانی از شاخص‌های نهفته**[19]

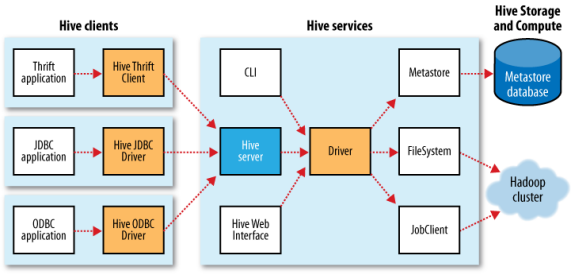
دسترسی مستقیم به رکورد ها را با استفاده از شاخص‌هایی بر اساس کمترین، بیشترین و فیلترهای دیگر برای هر کدام از ستون‌ها فراهم می‌کند. به گونه‌ای که بر هر کدام از ستون‌ها با سرعت بالایی می‌توان دسترسی داشت[8].

- **پشتیبانی از انواع پیچیده**

از انواع پیچیده‌ای همچون لیست‌ها، Mapها، Unionها و ساختارها[20] پشتیبانی می‌کند [8] [9].

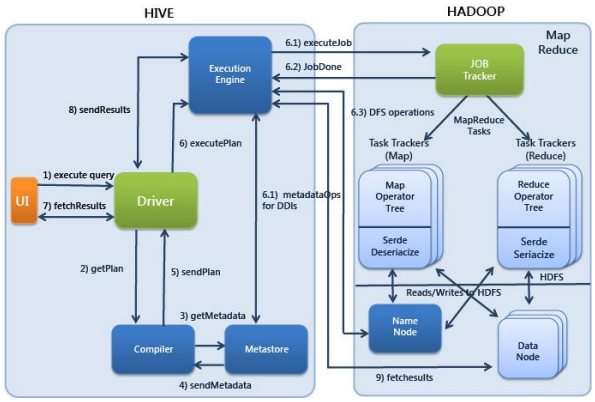
۷- معماری هایو

همانطور که در تصویر ۸ و به طور جزئی تر در تصویر ۹ مشخص شده است، هایو موتور پرس‌وجویی است که با سیستم هدوپ به صورت قوی در ارتباط است. این تصویر علاوه بر اینکه بیانگر معماری Hive بوده بلکه جریان اجرای یک پرس‌وجوی معمول را نیز نشان می‌دهد.



تصویر ۸ معماری سطح بالای هایو [9]

این موتور از مولفه‌های زیر تشکیل شده است:



تصویر ۹ فرآیند اجرای پرس‌وجو در هایو [3]

[19] Built-in indexes

[20] Structs

- **رابط کاربری[۲۱] (UI)**

رابط کاربری موتور اجرای پرس‌وجو که به عنوان رابطی برای اعمال یک پرس‌وجو می‌باشد. به عنوان مثال هم اکنون دو نوع رابط کاربری مبتنی بر خط دستور و رابط گرافی مبتنی بر وب وجود دارد [7] .

- **درایور[۲۲]**

مولفه‌ای هست که پرس‌وجوها را دریافت می‌کند. این مولفه مفهوم مدیریت Session، اجرای درخواست‌های رابط‌های JDBC/ODBC را بر عهده دارد [7] .

- **کامپایلر[۲۳]**

این مولفه فرآیند تجریه و تحلیل معنایی بر روی بلاک‌های پرس‌وجو را انجام می‌دهد. همچنین با توجه به فراداده‌های تولید شده توسط بخش ذخیره‌گر فراداده برنامه اجرایی پرس‌وجو را تولید می‌نماید. این بخش از نقطه نظر بهینه سازی بسیار مهم بوده و عملیات اصلی برای تولید پلن اجرایی[۲۴] بهینه در این بخش صورت می‌گیرد[7] . در ادامه به جزئیات این بخش بیشتر پرداخته شده است. برای پرس‌وجوهای درج و دیگر پرس‌وجوها کامپایلر یک گراف جهت‌دار بدون فرآیندهای نگاشت-کاهش را تولید می‌کند که در بستر هدوپ بتواند به خوبی اجرا گردد [2].

- **ذخیره‌گر فراداده[۲۵]**

این مولفه شامل تمام فراداده هایی است که در رابطه با جداول و پارتیشن‌ها در انبار داده HDFS وجود دارد. اطلاعاتی همچون ستون‌ها، نوع ستون‌ها، نحوه سریالی سازی و بلعکس برای عملیات خواندن و نوشتن در این قسمت قرار دارد [7] . این بخش به طور قوی با کامپایلر و موتور اجرایی در ارتباط است.

- **موتور اجرایی[۲۶]**

این مولفه پلن اجرایی تولید شده توسط کامپایلر را اجرا می‌کند. پلن اجرایی یک گراف جهت‌دار بدون دور[۲۷] از مرحله‌های مختلف پرس‌وجو است. موتور اجرایی وابستگی‌های بین مرحله‌های مختلف را برای اجرای بهتر در موقعیت مناسب در راستای افزایش سرعت مدیریت می‌کند.

۸- زبان پرس‌وجوی HiveQL

این زبان پرس‌وجو یک زبان شبیه به SQL می‌باشد که این امکان را می‌دهد که بتوان پرس‌وجوهای SQL ای را نوشت و نیز اجرا کرد همچنین امکان اجرای اسکریپت‌های نگاشت-کاهش را نیز فراهم می‌کند. این اسکریپت‌ها می‌توانند بر اساس یک رابط جریانی[۲۸]

[21] User Interface
[22] Driver
[23] Compiler
[24] Execution Plan
[25] Metastore
[26] Execution Engine
[27] Directed Acyclic Graph (DAG)
[28] Streaming Interface

مبتنی بر سطر[۲۹] به هر زبان دلخواه نوشته شوند، اما در نهایت این سطرها به حالت رشته‌ای تبدیل می‌شود، این تبدیل می‌تواند منجر به سربار کارایی گردد [2]. یکی از ویژگی‌های منحصر به فرد این زبان امکانی تحت عنوان درج چند جدولی[۳۰] می‌باشد. در این حالت کاربران می‌توانند چندین پرس‌وجو را بر روی یک داده ورودی یکسان در قالب تنها یک پرس‌وجوی HiveQL اجرا نمایند. هایو با بهینه‌سازی این پرس‌وجوها برای به اشتراک گذاری فرآیند پویش[۳۱] داده ورودی، بهره‌وری پرس‌وجوها را افزایش می‌دهد[7].

۸-۱- کامپایلر و روند کلی بهینه سازی پرس‌وجو

کامپایلر توسط درایور با یک رشته پرس‌وجو فراخوانی می‌شود، که این پرس‌وجو می‌تواند یکی از انواع DDL یا DML[۳۲] باشد. کامپایلر این پرس‌وجو را به یک طرح اجرایی تبدیل می‌کند. این پلن اجرایی برای پرس‌وجوهای از نوع DDL شامل فراداده عملیات و شامل عملیات HDFS از جمله عمل LOAD است. همانطور که در بخش‌های قبلی نیز اشاره شد، این فرآیند نهایتا منجر به این می‌گردد که یک گراف جهت‌دار بدون دور از فرآیندهای نگاشت-کاهش ایجاد شود. در ذیل هر کدام از بخش‌های کامپایلر مورد بررسی قرار می‌گیرد.

- **تجزیه‌گر**[۳۳]

یک رشته پرس‌وجو را به صورت درخت تجزیه بیان می‌کند.

- **تحلیل‌گر معنایی**[۳۴]

درخت تجزیه ایجاد شده را به یک پرس‌وجوی داخلی مبتنی بر بلاک[۳۵] تبدیل می‌کند. به این صورت که اطلاعات فراداده جداول ورودی را از ذخیره‌گر فراداده دریافت می‌کند. با استفاده از این اطلاعات تحلیل‌گر عنوان ستون‌ها و تبدیل نوع‌ها را در پرس‌وجو مورد صحت سنجی قرار می‌دهد.

- **مولد پلن منطقی**[۳۶]

پرس‌وجوی داخلی ایجاد شده را در قالب یک پلن منطقی که شامل درختی از عملگرهای منطقی است بیان می‌کند.

- **بهینه‌ساز**[۳۷]

این بخش مهم‌ترین عملیات را از نقطه نظر بهینه سازی انجام می‌دهد بدین صورت که چندین گام از عملیات را بر روی طرح منطقی انجام می‌دهد و این عملیات را به چندین طریق مختلف اعمال می‌کند:

  - ترکیب چندین Join در راستای استفاده از تنها یک کلید در یک Join چندین-راهه و در نتیجه ایجاد تنها یک فرآیند

29 Row-based
30 Multi-table Insert
31 Scan
32 Data Manipulation Language
33 Parser
34 Semantic Analyzer
35 Block-based
36 Logical Plan Generator
37 Optimizer

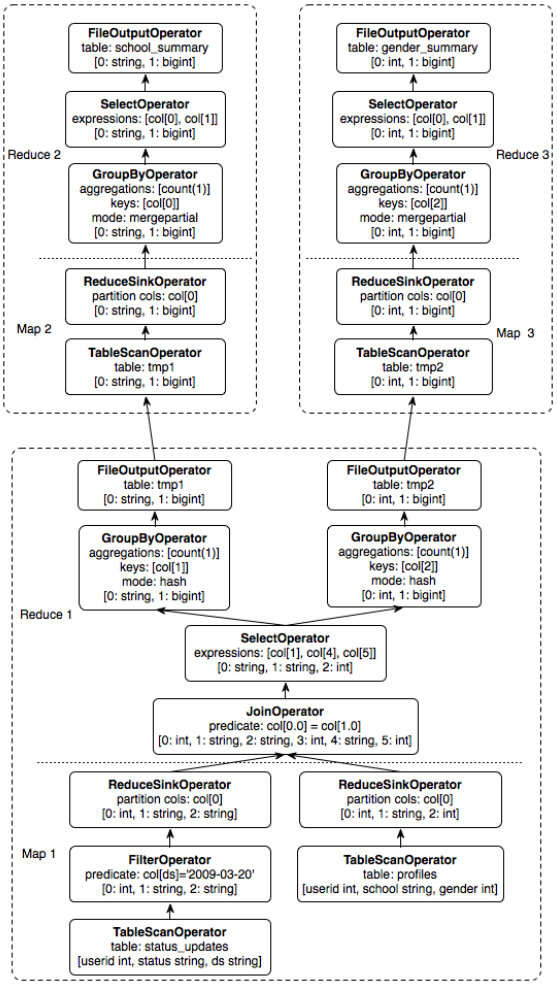
نگاشت-کاهش که منجر به بهینه سازی فرآیندهای نگاشت-کاهش می‌گردد.

- افزودن عملیات تقسیم و توزیع برای پیوندها، عملگرهای گروهی و همچنین عملیات نگاشت-کاهش دلخواه که ایجاد شده اند. این فرآیندهای تقسیم و توزیع حدود بین فازهای map و reduce را در طول ایجاد پلن اجرایی فیزیکی مشخص می‌کند.
- هرس کردن ستون‌ها در پرس‌وجوها و همچنین اعمال چندین جابه جایی به جهت به حداقل رساندن میزان داده‌ای که باید بین عملگرها انتقال داده شود.
- حذف کردن پارتیشن‌هایی که در پرس‌وجوی جدول‌های پارتیشن شده مورد نیاز نیستند.
- حذف کردن باکت‌ها در پرس‌وجوهای نمونه برداری.

- **مولد پلن اجرایی فیزیکی**[۳۸]

این بخش از کامپایلر هایو، پلن اجرایی منطقی را به پلن اجرایی فیزیکی تبدیل می‌کند بدین صورت که به ازای هر کدام از عملیات repartition و union all یک فرآیند نگاشت-کاهش جدید ایجاد می‌کند.

تصویر ۱۰ بیانگر بخشی از فرآیند بهینه سازی پرس‌وجوی درج چند جدولی مطابق با نکات گفته شده در بالا می‌باشد.



تصویر ۱۰ طرح اجرایی با سه فرآیند نگاشت-کاهش برای پرس‌وجوی درج چند-جدولی [5]

۹- روش‌های بهینه‌سازی پرس‌وجو

در ادامه توضیحات ارائه شده در بخش کامپایلر و روند کلی بهینه‌سازی پرس‌وجو، بهینه سازی را در موتوری همچون هایو با رویکرد رابطه‌ای، می‌توان به دو دسته فیزیکی و منطقی تقسیم بندی کرد.

رویکردهای بهینه سازی منطقی تاکید بر این دارد که پرس‌وجو با تاکید بر مسائلی همچون جبر رابطه‌ای و مستقل از جزئیات فیزیکی، چگونه اجرا می‌شود. با این هدف که پرس‌وجو را پیش از ارسال به لایه فیزیکی از

[38] Physical Execution Plan Generator

نظر معیارهای مختلفی همچون هزینه، میزان نگاشت-کاهش و... بهینه نماید. رویکردهای منطقی را می‌توان به صورت زیر دسته بندی نمود:

- هرس‌کردن پرتوها[۳۹]
- کاهش مبتنی بر پیش‌بینی‌های انتقالی[۴۰]
- کاهش Push down
- الحاق پرس‌وجوهای تودرتوی select-select به تنها یک عملگر[۴۱]
- پیوندهای چندراهه[۴۲]
- دوباره نویسی پرس‌وجوی پیوندی بر اساس مقادیر برخی ستون‌ها[۴۳]

رویکردهای بهینه‌سازی فیزیکی نیز روش‌هایی هستند که از جزئیات لایه فیزیکی آگاه می‌باشند. به عنوان مثال بهره‌گیری از فایل‌های ORC و همچنین جزئیات فرآیند نگاشت-کاهش از روش‌های فیزیکی برای بهینه‌سازی پرس‌وجو به شمار می‌روند. می‌توان دسته بندی زیر را برای اینگونه روش‌ها در نظر گرفت:

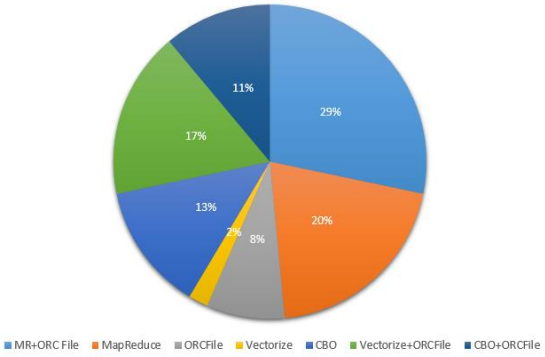
- هرس‌کردن پارتیشن[۴۴]
- هرس‌کردن روند پویش بر اساس پارتیشن‌ها و باکت‌ها[۴۵]
- هرس‌کردن پرس‌وجو در صورتی که پرس‌وجو از نوع sampling باشد
- اعمال فرآیند پیوند بر روی نگاشت‌گرها
- بهینه‌سازی unionها بر اساس رویه نگاشت
- حذف کاهش‌های اضافی[۴۶]

- اجتناب از نگاشت مستقیم پرس‌وجوهای ساده به فرآیندهای نگاشت-کاهش
- دوباره‌نویسی پرس‌وجوهای group by براساس جدول شاخص به جای جدول اصلی
- و چندین روش مطرح دیگر

البته در حالت کلی دسته‌بندی‌های زیر برای بهینه‌سازی مطرح است:

- بهینه‌سازی نگاشت-کاهش
- بهره‌مندی از فایل‌های ORC
- روش‌های مبتنی بر هزینه
- روش ترکیبی مبتنی بر هزینه و فایل‌های ORC
- برداری‌سازی
- بهینه‌سازی نگاشت-کاهش و فایل‌های ORC

نتایج سنجش میزان کارآمدی روش‌های ذکر شده در تصویر ۱۰ آورده شده است.



تصویر ۱۰ کارآمدترین روش‌های بهینه سازی پرس‌وجو در هایو[11]

مطابق تصویر ۱۰ یکی از روش‌های اساسی مطرح در بهینه‌سازی پرس‌وجوهای هایو، روش مبتنی بر هزینه[۴۷] می‌باشد. اساسا بیشتر روش‌های مطرح نیز بر هزینه اجرایی معطوف گریده‌اند. اما روش‌هایی که بیشتر بر

39 Projection pruning
40 Deducing transitive predicates
41 Merging select-select queries in to single operator
42 Multiway
43 Query Rewrite to accommodate for Join skew on some column values
44 Partition pruning
45 Scan pruning based on partitions and bucketing
46 Remove unnecessary reduce sink operators
47 Cost-based optimization

روی آن‌ها تاکید شده است، که متاثر از معماری هایو نیز می‌باشند، به حداقل رساندن تعداد عملیات کاهش در فرآیند نگاشت-کاهش است. مطابق نتایج به دست آمده بهینه‌ترین روش نیز ترکیب بهینه‌سازی نگاشت-کاهش با فایل‌های ORC می‌باشد.

برخی از تصمیماتی که می‌توان بر اساس روش‌های مبتنی بر هزینه اتخاذ نمود به شرح زیر است:

- نحوه جابجایی ترتیب پیوندها
- انتخاب الگوریتم متناسب با پیوند داده شده
- بررسی لزوم ذخیره سازی نتایج میانی
- درجه موازی‌سازی مطابق با عملگرهای مختلف
- نحوه ایجاد عملگرهای select شبه-پیوندی

در این پژوهش با نگرشی به روش‌های پیشین از رویکرد مبتنی بر همبستگی[48] بین پرس‌وجوها در راستای کاهش میزان فرآیند نگاشت-کاهش برای بهینه‌سازی استفاده گردیده است. یکی از مزیت‌های اساسی این روش تاکید آن بر بهینه‌سازی فیزیکی با در نظر گرفتن همبستگی منطقی میان پرس‌وجوها می‌باشد. در ادامه با چندین مثال از پرس‌وجوهای مختلف به جزئیات این روش پرداخته شده است.

شایان ذکر است که موتور هایو به تنهایی قادر به تشخیص همبستگی میان عملگرها نیست و به نوعی این آگاه‌سازی باید صورت بگیرد. در اثنای پرس‌وجوهای ذکرشده در ادامه، به بررسی جزئیات پرداخته شده است.

پرس‌وجوی ۱

```
SELECT t1.key, SUM(value)
FROM t1 JOIN t2 ON (t1.key = t2.key)
GROUP BY t1.key
```

کد اجرایی پرس‌وجوی ۱

با بررسی پرس‌وجوی ۱، ملاحظه می‌گردد که هر دو عملگر پیوند و group by نیازمند ترکیب داده هستند، به صورتی که داده ورودی عملگر group by خروجی عملگر پیوند می‌باشند، اما بهینه‌ساز باید تصمیمی اتخاذ کند که در کدام نقطه داده می‌بایست ترکیب[49] گردد. هایو به دلیل مطلع نبودن از همبستگی دو فرآیند نگاشت-کاهش را برای پرس‌وجوی بالا ایجاد می‌کند. اما در اصل نیازی به تولید فرآیند مجزا نیست، چرا که فیلدی که بر روی آن عمل group by انجام شده است همان فیلدی است که به عنوان کلید در شرط عملگر پیوند به کار برده شده است، که می‌تواند در قالب تنها یک فرآیند محاسبه گردد. البته حالت استثنایی هم ممکن است رخ دهد اگر فیلدهای هر دو عملگر اشتراکی با یکدیگر نداشته باشند.

پرس‌وجوی ۲

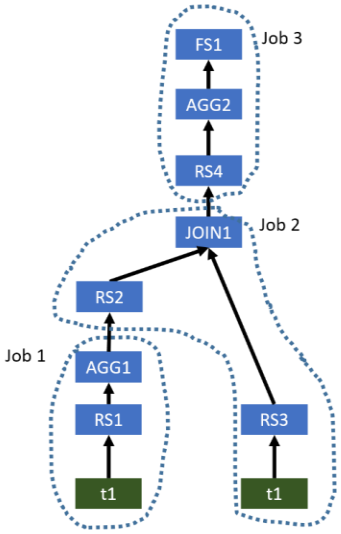
```
SELECT tmp1.key, COUNT(*)
FROM (SELECT key, AVG(value) AS avg
      FROM t1
      GROUP BY /*AGG1*/ key) tmp1
JOIN /*JOIN1*/ t1
ON (tmp1.key = t2.key)
WHERE t1.value > tmp1.avg
GROUP BY /*AGG2*/ tmp1.key
```

کد اجرایی پرس‌وجوی ۲

برای پرس‌وجوی ۲، درختی که پیش از عملیات بهینه‌سازی ایجاد می‌گردد به صورتی که در تصویر ۱۱ نشان داده‌شده می‌باشد.

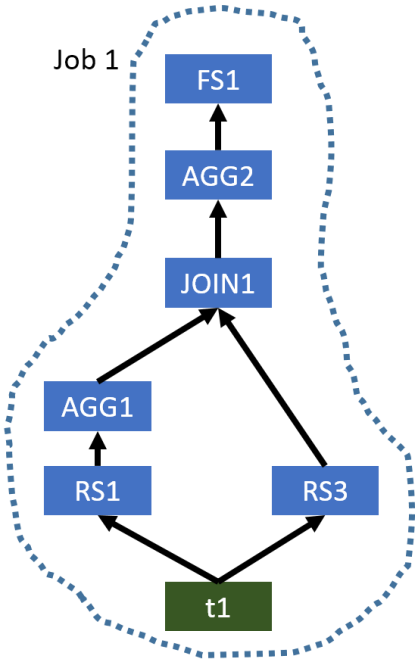
[48] Correlation-based optimization

[49] Shuffle



تصویر ۱۱ درخت فیزیکی پرس‌وجوی ۲ پیش از بهینه‌سازی

در تصویر ۱۱ ملاحظه می‌گردد که ۳ فرآیند نگاشت-کاهش برای پرس‌وجوی مورد نظر ایجاد شده است اما با تحلیل همبستگی میان پرس‌وجوهای داخلی و عملگر پیوند مشخص می‌گردد که نیازمند تنها ۱ فرآیند است، که در تصویر ۱۲ نشان داده شده است.



تصویر ۱۲ درخت فیزیکی پرس‌وجوی ۲ پس از بهینه‌سازی مبتنی بر همبستگی

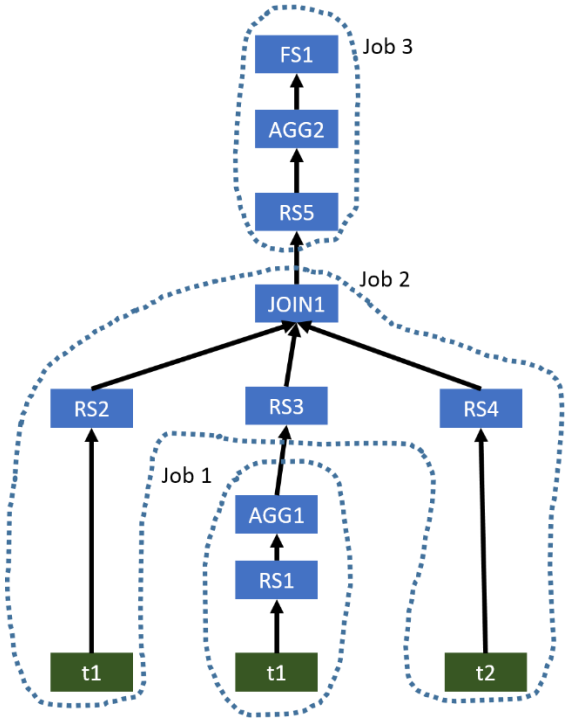
در این تصویر داده‌ها تنها یک بار ترکیب می‌گردد.

پرس‌وجوی ۳

```
SELECT tmp1.key,COUNT(*)
FROM  t1
JOIN /*JOIN1*/ (SELECT KEY, AVG(value) AS AVG
                FROM t1
                GROUP BY /*AGG1*/ key ) tmp1
                ON ( t1.key = tmp1.key )
JOIN /*JOIN1*/ t2 ON ( tmp1.key = t2.key )
WHERE t2.value > tmp1.avg
GROUP BY /*AGG2*/ t1.key;
```

کد اجرایی پرس‌وجوی ۳

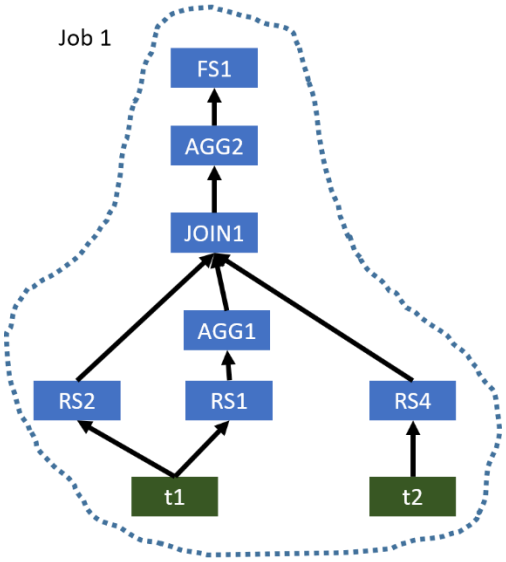
پرس‌وجوی ۳، مشابه پرس‌وجوی ۲ عمل می‌کند. ممکن است به نظر برسد که پرس‌وجوی پیچیده‌ای است و ممکن است به بیش از ۱ عملیات نگاشت-کاهش نیاز داشته باشد اما این پرس‌وجو نیز تنها با یک فرآیند نگاشت-کاهش به اتمام می‌رسد. درخت فیزیکی اجرا برای این پرس‌وجو پیش از بهینه‌سازی در تصویر ۱۳ آورده شده است.



تصویر ۱۳ درخت فیزیکی پرس‌وجوی ۳ پیش از بهینه‌سازی

با توجه به اینکه پرس‌وجوهای داخلی و عملیات پیوند بر روی کلید t1 تعریف گردیده است، و همچنین عملگر group by تنها بر روی کلید t1 عمل می‌کند،

بنابراین تنها با یک فرآیند نگاشت-کاهش می‌توان پرس‌وجو را اجرا کرد. حالت بصری توضیحات گفته شده در تصویر ۱۴ بر روی درخت فیزیکی بهینه شده، نشان داده شده است.



تصویر ۱۴ درخت فیزیکی پرس‌وجوی ۳ پس از بهینه‌سازی مبتنی بر همبستگی

بهینه سازی در سطح نوشتن پرس‌وجو

زبان HiveQL و موتور پرس‌وجوی آن، این امکان را به توسعه دهندگان می‌دهد که بتوانند پرس‌وجوهای خود را پیش از ارسال به موتور پرس‌وجو با استفاده از عملگرهای Hint بهینه کنند. در ادامه به دو عملگر پر کاربرد اشاره شده است.

- **عملگر STREAMTABLE**

این عملگر از بافرشدن جداول بزرگ در حافظه اصلی جلوگیری می‌کند و موجب می‌شود تا این جدول به صورت جریانی بسته به نیاز مورد بازیابی قرار گیرد. جدول‌های دیگر که در عملیات پیوند شرکت دارند، در حافظه اصلی بافر می‌گردند[7] .

```
SELECT /* STREAMTABLE(a) */
a.val, b.val, c.val, d.val, e.val
FROM a JOIN b ON (a.key = b.key1)
JOIN c ON (c.key = b.key1)
JOIN d ON (d.key = c.key)
JOIN a ON (e.key = d.key)
```

به عنوان مثال در پرس‌وجوی بالا جدول a به صورت جریانی مورد بازیابی قرار می‌گیرد.
این عملگر بیشتر بر روی جداول بزرگ اعمال می‌گردد.

- **عملگر MAPJOIN**

این عملگر برعکس عملگر قبلی، باعث cache شدن جدول‌های کوچک در حافظه شده و از این طریق میزان سرعت دسترسی به جدول را افزایش می‌دهد. به عنوان مثال در پرس‌وجوی زیر جدول‌های b، c، d و e در حافظه نگه‌داری می‌شوند [7] .

```
SELECT /* MAPJOIN(b,c,d,e) */
 a.val, b.val, c.val, d.val, e.val
FROM a JOIN b ON (a.key = b.key1)
JOIN c ON (c.key = b.key1)
JOIN d ON (d.key = c.key)
JOIN a ON (e.key = d.key)
```

۱۰- نتیجه‌گیری

با توجه به چالش‌های مطرح در حوزه کلان داده، امروزه نیازمندی‌های اساسی در حوزه تحلیل‌داده‌ها با اهداف مختلف تعریف شده است. بنابراین نیازمند روش‌هایی است که بتواند به صورت بهینه بر روی داده‌های با حجم بالا، متنوع و نیازمند صحت، عمل کند. حجم داده تولید شده در سیستم‌های امروز از اینترنت اشیا، شبکه‌های اجتماعی گرفته تا سیستم‌های بررسی و کنترل آب و هوا و سیستم‌های حوزه سلامت با میزان تولید آن در گذشته متفاوت است. بنابراین رویکردهایی که مورد استفاده قرار می‌گیرند باید پاسخگویی این چالش‌ها باشند. موتورهای پرس‌وجو یکی از بخش‌های مهمی هستند که در راستای تامین سرعت و دقت در تحلیل داده‌ها مورد استفاده قرار می‌گیرند. در این پژوهش سعی شد تا ضمن پرداختن به بحث‌های سیستمی بستر هدوپ، به جزئیات موتور پرس‌وجوی

هـایو نیـز اشـاره گـردد. همچنـین ضـمن معرفـی رهیافت‌های مطرح در این زمینه، رهیافـت مبتنـی بـر همبسـتگی بـه عنـوان یـک روش نـوین و کـارا بـرای بهینه‌سازی پرس‌وجوها به صورت جزئی مـورد بررسـی قرار گرفت و نتایج آن ارائه گردید.

۱۱- مراجع

[1] "Hive Website," [Online]. Available: http://hive.apache.org/.

[2] A. Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff and Raghotham Murthy, "Hive: a warehousing solution over a map-reduce framework," in *Proceedings of the VLDB Endowment 2, no. 2*, 2009.

[3] Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu and Raghotham Murthy, "Hive-a petabyte scale data warehouse using hadoop," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on, pp. 996-1005. IEEE, 2010.*, 2010.

[4] A. Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff and R. Murthy, "Hive: a warehousing solution over a map-reduce framework".

[5] Vlad Mihai MIREL, *Benchmarking Big Data SQL Frameworks.*, Aalto University, 2016.

[6] "Apache ORC," [Online]. Available: https://orc.apache.org/.

[7] "Hive design and architecture," [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/Design.

[8] Hortonworks, *Apache Hive Performance Tuning*, Hortonworks Docs, 2017.

[9] W. Tom, *Hadoop: The definitive guide.*, O'Reilly Media, 2012.

[10] K. Shvachko, Hairong Kuang, Sanjay Radia and Robert Chansler, *The hadoop distributed file system*, Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on (pp. 1-10), 2010.

[11] Z. Tariq, "Query optimization techniques in Apache Hive," 2015. [Online]. Available: https://www.slideshare.net/ZaraTariq/query-optimization-techniques-in-apache-hive.

میثم پاسداری‌هِریس

دانشجوی کارشناسی ارشد مهندسی نرم‌افزار، دانشگاه زنجان
پژوهشگر در حوزه علوم و مهندسی کامپیوتر

زمینه‌های پژوهشی:
تحلیل شبکه‌های اجتماعی، نسل جدید سیستم‌های توصیه‌گر، پردازش و ذخیره سازی کلان‌داده، چالش‌های سیستمی در اپلیکیشن‌های تحت‌وب در مقیاس بزرگ
نشانی رایانامه ایشان عبارت است از:
pasdari.meysam@znu.ac.ir

داود محمدپورزنجانی

عضو هیئت علمی، گروه کامپیوتر، دانشگاه زنجان، زنجان
مولف کتاب پایگاه‌داده پیشرفته

زمینه‌های پژوهشی:
تئوری زبان های برنامه نویسی و مدل های طراحی نرم افزار، پردازش داده های حجیم، بانک های اطلاعاتی غیر رابطه ای، طراحی و تولید سیستم های اطلاعاتی، سیستم های وبی و اینترانتی، سیستم های اطلاعاتی توزیع شده، مدل سازی سیستم ها، مطالعه سیستم ها و ارائه راه کارهای اطلاعاتی، بانک های اطلاعاتی ابری.
نشانی رایانامه ایشان عبارت است از:
dmp@znu.ac.ir