پردازش کلان داده ها با استفاده از داده های HDFS و تکنیک های تکاملی خوشه

چکیده

استفاده روز افزون از سیستم‌های فعال شده اینترنت (IOT) منجر به مقادیر فراوانی از داده‌ها با ساختارهای مختلف شده است. اکثر راه حل‌های کلان داده‌ها در بالای سیستم اکو هادوپ یا از سیستم فایل توزیع شده (HDFS) استفاده می‌شوند. با این حال، بررسی‌های ناکارآمدی در این سیستم در هنگام برخورد با داده‌ها نشان داده شده‌اند. برخی از تحقیقات این مسائل را برای نوع خاصی از داده‌های گراف برطرف می‌کنند، اما امروزه اطلاعات بیشتری از داده‌ها قابل دسترس هستند. چنین مسائلی مربوط به عملکردهایی می‌شوند که منجر به مسائل بزرگی از جمله فضای بزرگ‌تر مورد نیاز در مراکز داده، و از بین رفتن منابع (مانند مصرف انرژی)، و مشکلات زیست محیطی (مانند انتشار بیشتر کربن) نیز می‌شود. ما یک ماژول اطلاعاتی را برای سیستم اکو هادوپ ارائه می‌دهیم. ما همچنین یک روش رمزگذاری توزیع شده را برای الگوریتم‌های ژنتیک ارائه می‌دهیم. چارچوب ما این امکان را می‌دهد تا هادوپ بتواند توزیع داده‌ها و قرار دادن آنها براساس تجزیه و تحلیل داده‌های خوشه‌ای را مدیریت کند. ما قادر هستیم طیف گسترده‌ای از انواع داده‌ها را مدیریت کنیم و همچنین قادر هستیم زمان استفاده از منابع پرس‌وجو را بهینه کنیم. ما آزمایش‌هایی را که در مجموعه داده‌های متعدد انجام شده‌اند را از طریق LUBM ایجاد می‌کنیم.

فهرست شرایط- روش‌های خوشه‌بندی، محاسبات توزیع شده، مدیریت اطلاعات، بهینه‌سازی، مقیاس‌پذیری

1. مقدمه

ایجاد علومی از داده‌ها با چالش‌های بسیاری مواجه بوده است. یک مسئله اصلی وجود دارد و آن این است که امروزه کلان داده‌ها، پویا و ناهمگن هستند، و چندین منابع را که اغلب ساختار استانداردی ندارند را جمع‌آوری می‌کنند.

اکثر تجزیه و تحلیل داده‌های مدرن، ابزارهای مدیریت و سرویس را طراحی می‌کنند و در سیستم فایل توزیع شده هادوپ (HDFS) به عنوان یک انبار داده‌ای مورد استفاده قرار می‌دهند؛ گاهی اوقات هم این ابزار تحلیلی از سرویس‌هایی که توسط اکوسیستم هادوپ پردازش شده‌اند استفاده می‌کند. هادوپ از لحاظ هزینه و عملکرد بسیار خوب است.

قابلیت انعطاف هادوپ، مقیاسی را روی مسائل مدیریتی داده‌ها ارائه می‌دهند و در آن کاربران به صورت ناکارآمد کار انجام می‌دهند [1]. هونگ و همکارانش (a) راهکارهایی که کاربران به منظور اضافه کردن ماشین‌ها برای غلبه بر محاسبات انجام می‌دهند نحوه تمرکز و استفاده از کدهای منابع کاربران را به ما نشان می‌دهد، و (b) بسیاری از کاربران HDFS معتقدند که این کدها برای پردازش دسته‌ای طراحی شده‌اند. از این رو، خوب است که کدها برای مدت طولانی در پس‌زمینه بکار برده شوند و حتی تصور هم نمی‌کنند که منابع مورد استفاده این فرآیندها هستند.

در باجدا پاولیکوفسکی و همکارانش روی هاداپ کار کردند [2]، و نویسندگان نمونه‌ای از چنین ناکارآمدی را ارائه دادند و داده‌های ساخت یافته را 50 بار برآورد کردند. با این حال، انفجار اقدام مهم داده‌های نیمه ساخت یافته، به صورت چندگانه و ساختارنیافته بر طبق بررسی‌های مایکل واکر است که در وبلاگ به آن اشاره شده است [3]. شرکت داده‌های بین‌المللی (IDC) برآورد کرده است که حجم داده‌های دیجیتال از 40 تا 50 درصد در سال رشد خواهد کرد [4]. تا سال 2020، IDC [4] پیش‌بینی کرده است که این تعداد به 40 زتابایت خواهد رسید. تا سال 2020، جهان 50 بار مقادیر داده‌ای و 75 بار محتوای داده‌ها را تولید خواهد کرد. نیازهای سختی برای ابزارهای تجزیه و تحلیل داده فعلی به منظور اندازه‌گیری کلان داده‌ها وجود دارد و آنها به طور موثر از منابع پردازش استفاده می‌کنند.

روهلوف و همکارانش در سال 2011 [5] نحوه ذخیره داده‌های گراف در هادوپ را با استفاده از نمای سه گانه توضیح دادند. آنها همچنین نشان دادند که چطور تطبیق الگوی زیر نمودار در یک نمودار مقیاس‌پذیر انجام می‌شود. اگرچه تمرکز بر روی گراف‌های وب معنایی بود، اما تکنیک‌هایی که در این مقاله ارائه شده بود انواع نمودارها را تعمیم می‌داد. سیستم SHARD نتیجه این مقاله بود. از این رو، تکنیک‌های هادوپ پشتیبانی می‌شوند تا ظرفیت تطبیق الگوریتم زیر نمودار ترسیم شود.

در سال 2011، هوانگ و همکارانش مقیاس پرس‌وجو SPARQL و گراف‌های RDF [1] را که به طور کارآمد در تکنیک‌ها توسط روهلوف و همکارانش بیان شده بودند را نشان دادند [5]. هوانگ و همکارانش [1] 1340 عامل را که کمتر از تکنیک‌های کارآمد است را معرفی کرده‌اند [5]. و از سایر تکنیک‌های جایگزین برای پردازش الگوریتم‌های زیر که براساس هادوپ هستند را استفاده کردند.

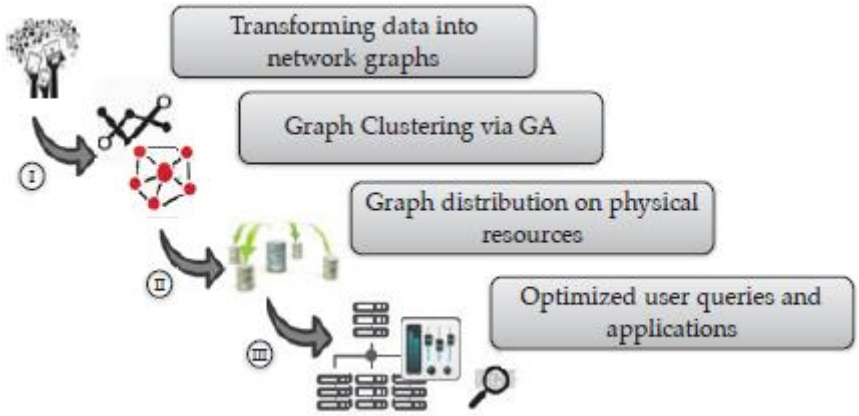
در برخی موارد، راه حل‌های کلان داده از HDFS به عنوان یک ابزار ذخیره‌ساز استفاده نمی‌کنند. با این حال، آنها از همان روش مقیاس‌پذیری افقی استفاده می‌کنند. ما راه حل‌هایی را که بر روی HDFS کار انجام می‌دهند را آزمایش و ارائه دادیم. نمونه‌هایی از ابزارهای ذخیره‌ساز HDFS به عنوان آپاچی اسپارک [6] و مسوس عمل می‌کنند [7]. سیستم هادوپ از طریق منابع YARN و منبع داده HDFS، و HAMR پشتیبانی می‌شود [8]. از این رو، بهینه‌سازی HDFS با چارچوب HDFS ارائه می‌شود و منجر به بهینه‌سازی تعداد زیادی از راه حل‌های کلان داده‌ها می‌شود.

اسپارک [6] و استورم [9] جدیدترین ابزار کلان داده‌ها را ارائه کردند. آپاچی استورم [9] یکی دیگر از راه حل‌های داده‌ای است که از YARN برای تجزیه و تحلیل داده‌های نامحدود استفاده می‌کرده‌اند. استورم هادوپ را ایجاد می‌کند و به صورت دسته‌ای آن را پردازش می‌کند. برخی از تلاش‌ها در بهینه‌سازی Storm ایجاد می‌شوند. یک مثال از زمانبندی بهینه‌سازی در [10] یافت شده است، و با افزوده شدن Storm به آن در [11] ارائه شده است. تمرکز ما در این بررسی بیشتر روی بهینه‌سازی انبار داده‌ای HDFS است (در حال حاضر داده‌ها در HDFS قرار دارند). با این حال، در برخی از موارد، Strom [9] جریان داده‌ها را پردازش می‌کند و نتایج خروجی را به منظور پردازش بیشتر در HDFS ذخیره می‌کند، تا جایی که رویکرد ما امکانات تحلیلی داده را در بسته‌های هادوپ انجام دهد.

رشد داده‌های نیمه/چند/غیر ساختاری و توانایی آنها در پردازش داده‌ها به طور موثر، یک مسئله اصلی بود، و هدف اصلی این داده‌ها بهبود کارایی سیستم فایل توزیع شده هادوپ (HDFS) به منظور بررسی داده‌های مدرن و بهبود منابع HW بود. با وجود اینکه تکنیک‌های SHARD [5] و SPALQL [1] مقیاس‌پذیر و قابل تعمیم هستند، اما محدودیت‌هایی در مورد نحوه مقابله با داده‌های مدرن نیز وجود دارد. علاوه بر این، محدودیت‌های دیگری در مورد نحوه مدیریت جریان تغییرات پویای داده نیز وجود دارند. همانطور که هاجر و

همکارانش بحث کرده‌اند یکی از این محدودیت‌ها ممکن است به کاربرد مقیاس‌پذیری و ذخیره‌سازی الگوریتم خوشه‌بندی که در چنین کارهایی مورد استفاده قرار گرفته‌اند اضافه شوند [12].

ما داده‌ها را تغییر دادیم و آنها را در گراف براساس مقیاس گرافیک ذخیره‌سازی کردیم تا ساختار آن ایجاد شود و بتواند تغییرات ضروری را انجام دهد، و سپس ساختار سه‌گانه را برای نقاط داده، به منظور ایجاد خوشه سه‌گانه که در بخش معماری به طور کامل شرح داده شده است اضافه می‌کند. این تکنیک‌ها به ما این امکان را می‌دهند که (1) داده‌ها را از منابع مختلف جمع‌آوری و سپس آنها را با وجود ساختار داده‌های مختلف به چهارگوش تبدیل می‌کنیم، (2) تغییرات جریان پایگاه داده گراف به صورت پویا می‌باشند، و (3) هاجر و همکارانش یک نسخه جدید از کاربرد داده‌ها را جمع‌آوری کرده‌اند [12]. رمزگذاری جدید کروموزوم‌ها به منظور بررسی مسائل خوشه‌بندی داده‌های مدرن به همراه تکنیک‌های نوین متقاطع، جهش و ارزیابی برای ارائه نیازمندی‌های تکنولوژی جدید رمزگذاری توزیع شده مورد استفاده قرار گرفته‌اند. بعدها، ما زیرنمودارهای موجود در HDFS را براساس وابستگی خوشه به منظور تولید پردازش داده‌های بهینه‌سازی پرس‌وجو توزیع کردیم.



شکل 1: مراحل چهارچوب ارائه شده محاسبات

شکل 1 مفاهیم و ماژول‌ها را در یک چارچوب پیشنهادی نشان می‌دهد و توضیح آن به شرح زیر است: (1) پس از جمع‌آوری داده‌ها یا جمع‌آوری مجموعه داده‌های قدیمی، این ماژول داده‌ها را به نمودارهای شبکه مورد نظر تبدیل می‌کند؛ (2) با یافتن الگوها در نمودار، ماژول داده‌ها را به بلوک‌های مناسب توزیع می‌کند؛ (3) بلوک‌ها در داخل ماشین سمت راست توزیع می‌شوند؛ و (4) HDFS بهینه‌سازی شده به عنوان یک منبع داده پرس-وجوها را اجرا می‌کند و یک سیستم‌عامل را برای اعمال الگوریتم‌های گراف به صورت کارآمد به منظور کاهش

منابع مورد استفاده ارائه می‌کند.

برای خلاصه کردن چارچوب پیشنهادی، HDFS قادر است داده‌های مدرن را با ایجاد ساختار اطلاع‌رسانی داده-ها، شناسایی، توزیع و مدیریت داده‌ها در سیستم فایل مقیاس‌پذیر را بهبود ببخشد. بدین ترتیب، چارچوب نتایج در بهینه‌سازی و در منابع کارآمد سیستم اکو هادوپ و دیگر ابزارها از HDFS به عنوان یک منبع ذخیره‌سازی توزیع شده استفاده می‌شود.

در تحقیقات اخیر راه حل‌های متقاعد کننده‌ای در تحلیل بعدی و معماری لامبدا ارائه شده است. سانگ و همکارانش [13] تحقیقات اخیر را در نوع داه‌ها، مدل‌های ذخیره‌سازی، روش‌های تجزیه و تحلیل کاربرد شبکه کلان داده‌ها مورد بررسی قرار دادند. آنها همچنین چالش‌ها و توسعه کلان داده‌ها را برای پیش‌بینی روند فعلی و آینده خلاصه کرده‌اند. سانگ و همکارانش [13] نحوه جریان داده‌ها در زمان واقعی را با افزایش سرویس‌های جریان آنلاین نشان دادند، و همچنین نحوه بررسی یک سیستم را که براساس SQL که جریان DB نام دارد به منظور تجزیه و تحلیل اطلاعات پایدار نشان دادند.

بررسی‌های [15]، [16]، [17] و [18] بر تحلیل زمان واقعی تمرکز دارند، و شامل تلاش‌های معماری لامبدا هستند، و نویسندگان [18] در آن معماری سیستم‌های کلان داده‌های بحرانی را معرفی می‌کنند. آنها نحوه زیرساخت‌های موجود در کلان داده‌ها را که زمانبر هستند نشان دادند و فقط بر روی برنامه‌های کاربردی تمرکز کردند. باسانتا و همکارانش [18] این مسئله را از دیدگاه جامعه سیستم‌های زمان واقعی مورد بررسی قرار دادند. آنها معماری تحلیلگر زمانبندی (TC) را به عنوان یک گروه TC در پردازش دستی و جریان پردازش آنلاین TC را در نظر گرفتند. باسانتا و همکارانش [18] به طور کلی تبدیل پشته کلان داده‌ها را در پشته کلان داده TC به همراه اعمال نیازمندی‌ها و چالش‌های برنامه‌های پشته TC ارائه دادند.

T-Hoarder [19] چارچوبی است که توییت‌ها را همراه با داده‌های مربوط به زمان فضا را به دست می‌آورد؛ همچنین خلاصه‌ای از اطلاعات و تحلیل مربوط به فعالیت توییتر را به صورت یک رویداد در صفحه وب ارائه می-کند. تجزیه و تحلیل بررسی‌ها و معماری لامبدا همراه با آپاچی کُدو [20] و مجموعه‌ای از بررسی‌ها را که عملکرد سریع‌تری در پردازش OLAP و عملکرد قوی در زمان بارز حجم کاری را دارد نشان می‌دهد. با این حال، این یک نتیجه است، تا جایگاه برخورد داده‌های هوشمند، و تکنولوژی‌ها بتوانند حجم کاری پردازش داده‌ها

را کاهش دهند، و اولین نسخه آزمایشی اینتل را در سفیر 2017 ارائه دادند [22]. اینتل نوع جدید حافظه پایدار را با یک فضای خاص نسبت به DRAM و تاخیر کمتری را نسبت به SSD ارائه می‌دهد.

2 محدوده کاری

حجم کاری فعلی که در سیستم در حال اجرا می‌باشد (در آن ناکارآمدی وجود دارد) منجر به فضای بیشتر نیازمندی در مراکز داده و برخی از پیامدهای محیطی و باعث افزایش انتشار کربن در مصرف انرژی می‌شود [1]. این می‌تواند به دلیل مصرف انرژی اضافی و عملکرد پایین منابع سخت‌افزاری که شرکت‌ها را تحت تاثیر قرار می‌دهد باشد. ما به یک مقیاس کارآمد نیاز داریم.

نظریه گراف یک اصول را به خوبی مورد بررسی قرار می‌دهد. دکتر روی مارستن در وبلاگش [23] خاطرنشان کرد که نظریه گراف یکی از روش‌های کلیدی در فهم و درک استفاده از کلان داده‌ها است. دکتر مارستن رو ی گوگل در طی فرآیند مدرن‌سازی گراف با استفاده از لینک‌های بین اسناد و زمینه‌های وب معنایی، تمرکز دقیقی داشت. در نتیجه، ‘‘گوگل یک موتور جستجوی وب را تولید کرد و از رقبای ثابت خود پیشی گرفت’’ [23]. بسیاری از داده‌ها را می‌توان به نمودار تبدیل کرد. برعکس، بسیاری از مسائل را می‌توان به مسائل گراف تبدیل کرد. با استفاده از نظریه‌ها و الگوریتم‌های گراف، بسیاری از این مسائل را می‌توان به طور موثر حل کرد.

ما کار SHARD روهلوف و همکارانش را در نظر گرفتیم [5] و آن را همانطور که در بخش معماری نشان داده شده است به عنوان داده‌های امروزی به منظور تبدیل انواع داده به نمودار چهارگانه ارائه کردیم. همانطور که در بخش‌های پایگاه داده گرافیکی بحث کردیم روش‌هایی برای مقیاس داده‌های گراف با استفاده از سیستم‌های اکو توزیع شده مانند هادوپ وجود دارد. علاوه بر این، موفقیت بزرگ هوانگ و همکارانش زمانی اتفاق می‌افتد [1] که بورس تحصیلی روهلوف و همکارانش مورد پذیرش قرار گرفته باشد [5] و داده‌های RDF برای غلبه بر محدودیت‌های مربوط به تکنیک روهلوف و همکارانش بهینه‌سازی می‌شوند.

قبلا تلاش‌هایی برای بهینه‌سازی گراف‌ها صورت گرفته است؛ به عنوان مثال SHARD [5]، عمل درهم‌سازی داده‌ها را انجام می‌دهد. با این حال، عمل درهم‌سازی منجر به جابه‌جایی محدودیت‌ها در گراف‌های RDF و حرکت داده‌های شبکه‌ای نیز می‌شود. مثال دیگری از هوانگ و همکارانش بدین صورت است [1] اشیاء متصل به یک موضوع پردازش می‌شوند تا یک بلوک مشابه برای یک یا دو تقاطع بین موضوع و جسم ایجاد شود (یک یا دو

لبه فاصله بین موضوع و اشیاء). با این حال، محدودیت‌های فضا به علت افزایش اندازه داده‌ها وجود دارد. همچنین، محدودیت استفاده از چنین الگوریتمی که به یک گراف متصل است نیز وجود دارد. سایر کارهایی مانند سمپالا [24]، یا Hive، PigSPARQL [25] & [26]، MapMerge [27] و MAPSIN [28] تا حدی مقیاس‌پذیری را برطرف می‌کنند. با این وجود، چنین کاری از ذخیره‌سازی‌های مختلف که سه برابر هستند و از مزایای Hive، MR و Impala استفاده می‌کند و با استفاده از چارچوب ما می‌توانیم پارتیشن‌بندی را بهینه‌سازی کنیم.

2.1 پایگاه داده‌های گراف

داده‌های جاری و برنامه‌های مدرن به محدودیت‌های ذخیره و پردازش با استفاده از پایگاه داده‌های سنتی، بویژه مدل ارتباطی منجر می‌شوند. دقت پایگاه داده‌های گراف افزایش پیدا کرده است، و این موضوع تقریبا در اوایل دهه 90 [29] مجددا مورد بررسی قرار گرفت. اهمیت پایگاه‌های اطلاعاتی معمولا براساس روابط بین داده‌ها است، و به طور مساوی یا حتی بیشتر از اطلاعات موجودیت‌ها می‌باشند [30]. پروژه‌ها در زمینه‌های مختلف به پایگاه‌های زیر دقت ویژه‌ای دارند (مانند زیست‌شناسی [31]، وب معنایی [32]، وب کاوی [33] و شیمی [34]).

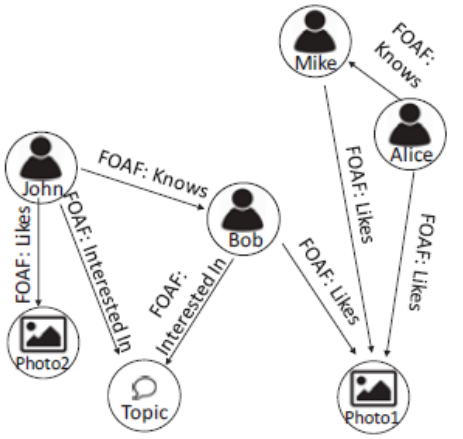
سیلبرشاتس و همکارانش [35] به طور کلی یک مدل داده (مدل پایگاه داده) را که مجموعه‌ای از ابزارهای مفهومی است را برای مدل‌سازی روابط بین شناسه‌ها مورد استفاده قرار دادند. سه مولفه مدل پایگاه داده عبارتند از [35]: (1) مجموعه‌ای از انواع ساختار داده‌ای، (2) مجموعه‌ای از قوانین یکپارچه و (3) رابط قوانین و عملگرها.

رنزو در هر مورد، "مدل‌های پایگاه داده گراف را می‌تواند به عنوان یک ساختار داده برای طرح و نمونه‌ها را به عنوان یک نمودار تعمیم یافته، و دستکاری داده‌ها را به عنوان عملیات گراف بیان کند" و آنها را مشخص می‌کنند [30].

بسیاری از پایگاه داده‌های گراف در زمینه‌های تحقیقاتی و صنعتی توسعه پیدا کرده‌اند. برخی از تفاوت‌ها مانند DEX، AllegroGraph، Neo4J، Infinite Graph، HypergraphDB و Sones در [3] یافت شدند. همانطور که هوانگ و همکارانش توضیح داده بودند بعضی از آنها حتی برای فرمت داده‌های سه‌گانه نیز طراحی

شده بودند [1].

ذخیره‌سازی سه‌گانه یا RDF یک پایگاه داده براساس گرافیک و بازیابی سه‌گانه از طریق پرس‌وجوهای معنایی می‌باشد. موجودیت سه‌گانه داده از شیء subject-predicate-subject تشکیل شده است. ذخیره‌های سه‌گانه داده‌ها به صورت سه‌گانه نشان داده می‌شوند و از طریق زبان پرس‌وجو بازیابی می‌شوند؛ با این حال، برخی از تفاوت‌های کلیدی پایگاه‌های داده رابطه‌ای نیز وجود دارند، و به طور عمده ذخیره‌های سه‌گانه را بهینه‌سازی می‌کنند.



Subject	Predicate	Object
Alice	knows	Mike
Alice	Likes	Photo1
Mike	Likes	Photo1
Bob	Likes	Photo1
Bob	Knows	John
Bob	Interested In	Topic
John	Interested In	Topic
John	Likes	Photo2

شکل 2: ذخیره RDF سه‌گانه و نشان دادن آن به عنوان گراف

ذخیره سه‌گانه در سه‌گانه در شکل 2 نشان داده شده است. در برخی از تحقیقات توسعه‌هایی در رابطه با سیستم پایگاه داده خوشه RDF صورت گرفته است. در حال حاضر پایگاه داده‌های خوشه‌ای RDF قابل دسترس هستند، و شامل YARS [36]، SHARD [5] و Jena [37]، Elephas و Virtuoso [38] هستند، و معمولا درهم‌سازی سه‌گانه را در گره‌های محاسباتی متعدد و گره‌های پرس‌وجو را در زمان به صورت موازی پارتیشن‌بندی می‌کنند.

2.2 تشخیص جامعه و الگوریتم تکاملی چند هدفه

در نظریه گراف، خوشه‌ها اغلب زمانی که معیارهای خاصی از تراکم و ضخامت توسط یک الگوریتم بهینه‌سازی می‌شوند به صورت الگوریتم تعریف می‌شوند، و نتیجه حاصل از آن تقسیم شبکه به جوامع می‌شود [39]. در بسیاری از موارد، بهینه‌سازی اندازه‌های مسائل NP-hard را پیدا می‌کند. معمولا، الگوریتم به صورت تقریبی از حل مسائل NP-hard استفاده می‌کند. یکی از الگوریتم‌های فراابتکاری برای حل تقریبی مسائل NP-hard

الگوریتم تکاملی (EA) است که در [40] توضیح داده شده است. برنامه استفاده از GA (الگوریتم ژنتیک) و تشخیص جامعه در [41]، [42]، [43]، [44]، [45] و [46] و کاربرد EA در [47] و [48] توضیح داده شده است. چانگ و همکارانش [49] برنامه کاربردی بهینه‌سازی کلونی را در تشخیص جامعه شرح داده‌اند.

مدولاسیون [50] یکی از محبوب‌ترین معیارهایی است که برای نتیجه‌گیری کیفیت خوشه‌بندی غیرتداخلی گراف، بویژه در جامعه تجزیه و تحلیل شبکه مورد استفاده قرار می‌گیرد [42]، [43]، [44]، [45]. مسئله کشف خوشه‌بندی با حداقل مدولاسیون NP-Complete نام دارد [51]. در نتیجه، بسیاری از الگوریتم‌های چندجمله‌ای اکتشافی زمانه توسعه پیدا کرده‌اند [52]، [53]، [54]، [55].

یکی از الگوریتم‌های تشخیص جامعه، الگوریتم Girvan-newman است [56]، و دارای لبه‌هایی با حداکثر مرکزیت است و به طور پیاپی از شبکه حذف می‌شود تا جایی که هیچ لبه‌ای باقی نماند. حالت مدولاسیون می‌تواند به صورت معادله (1) تعریف شود [57]، n_c تعداد کل جوامع، m تعداد لبه‌ها در گراف، l_i تعداد کل لبه‌ها در داخل جامعه i، d_i برابر مجموع تمام گره‌های i است.

$$(1) Q = \sum_{i=1}^{n_c} [\frac{l_i}{m} - (\frac{d_i}{2m})^2]$$

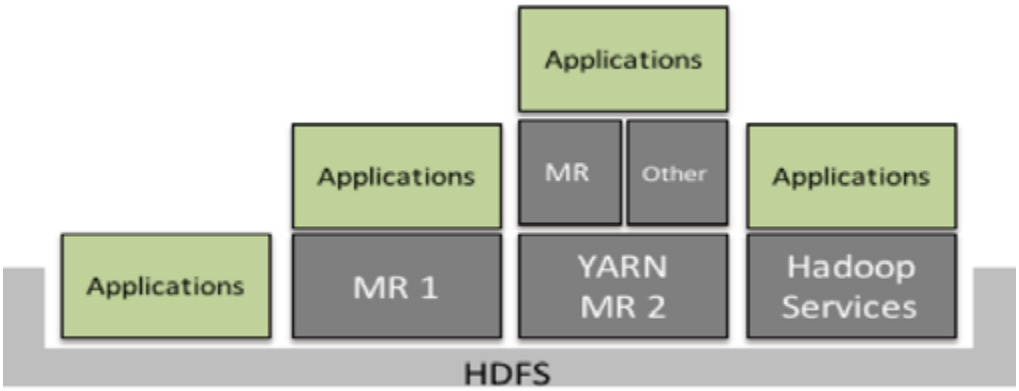
یی و همکارانش [58] "در الگوریتم ژنتیک (GA)، جمعیت کروموزوم، کدگذاری راه حل‌های انتخابی/مسائل بهینه‌سازی فردی، و راه حل‌های بهتر را تکامل داده‌اند". سپس، راه حل ژنتیک در فرمت کروموزوم و در فرمول‌های کروموزومی ارائه می‌دهد، و GA به صورت تصادفی/قطعی راه‌اندازی می‌شود. سپس، GA قصد دارد آن را از طریق برنامه‌های تکراری و چندین عمل ژنتیک مانند انتخاب، تقاطع و جهش بهبود ببخشد. سرانجام، جستجوهای محلی و عملگرهای جستجوی مرزی برای اعمال نتایج به خوبی مورد استفاده قرار می‌گیرند."

لی و سانگ [46] یک الگوریتم ژنتیک فشرده را توسعه دادند. ما کارهای قبلی خود را با استفاده از GA از [12]، [59] و [60] به دلایل مختلف انتخاب کردیم: (1) فضای جستجو بزرگ، (2) NP-Completeness حداکثر مدولاسیون [51]، (3) قادر است مقیاس نمودارها را بزرگ کند، و همچنین (4) قادر است تغییرات منابع مختلف داده‌ها را به صورت پویا منعکس کند. با این حال، برخی از تغییرات در GA [12] برای سازگاری چارچوب پیشنهادی HDFS لازم و ضروری بود. در بخش معماری، ما الگوریتم ژنتیک توزیع شده خود را با استفاده از منابع باز بهینه‌سازی چند هدفه و الگوریتم‌های ژنتیک، Jmetal [61]، و Apache Jena

Elephas [37] به منظور ذخیره و مدیریت داده‌های RDF توسعه دادیم. بعدها، ما مجموعه‌ای از داده‌های شناخته شده و به همراه آن گراف RDF بزرگ را که توسط آزمایشات دانشگاه تولید شده بود را تایید کردیم (LUBM).

2.3 سرویس توسعه HDFS

همانطور که قبلا ذکر شد، HDFS به عنوان یک منبع داده توزیع شده راه‌حل‌های داده‌های مدرن بزرگ مانند HAMR [8]، Mesos [7]، آپاچی اسپارک [6] و صدها نفر دیگر را توسعه می‌دهد. چنین راه حل‌هایی دارای بسیاری از برنامه‌های توسعه یافته HDFS یا دارای بیش از یک سرویس هستند که در HDFS در حال اجرا می‌باشند (به شکل 3 نگاه کنید).



شکل 3: توسعه سرویس‌ها و برنامه‌های کاربردی HDFS.

3 عملکرد و کارآیی HDFS

از سیستم اکو هادوپ برای پردارش سازمان داده‌ها و ایجاد برنامه‌های کاربردی بر روی آن، که بستگی به موارد و سازمان داده‌ها دارد استفاده می‌شود. گروهی از IT BI (کسب و کار هوشمند) در کسب و کارها و موسسات چنین سیستم‌هایی را به منظور رسیدن به اهداف خود پیکربندی می‌کنند، و سپس روی داده‌ها و موارد مورد استفاده در آنها تمرکز می‌کنند.

اکثر سازمان داده‌ها برای موارد خاص جمع‌آوری و استفاده می‌شوند. بعدها، این داده‌ها که در انتظار ذخیره تیم BI به منظور ایجاد و استفاده از آنها هستند، نتایج موجود در داده‌ها جمع‌آوری می‌کنند و در منابع مختلف با ساختار چندگانه مورد استفاده قرار می‌گیرند.

هوانگ و همکارانش [1] و روهلوف و همکارانش [5]، اجرای هادوپ و سرویس‌هایی که برای اجرای HDFS که دارای عدم بهینه‌سازی در گراف بودند را طراحی کردند. برخی از علل ناکارآمدی HDFS عبارتند از [1]: (1) عمل پارتیشن درهم‌سازی به طور پیش فرض توسط هادوپ ممکن است داده‌های مرتبط را هدایت کند تا محاسبات آنها به صورت فیزیکی بیش از مجموعه‌ها به پایان برسد، و این عمل به طور موثر موجب انتقال گسترده داده‌ها بین منابع می‌شود تا عملیات گراف به پایان برسد. بدین ترتیب، ترکیب داده‌های مربوطه در هر [1] حاصل می‌شود [1]؛ (2) هادوپ اهمیت تمام بلوک‌های داده و پارتیشن‌ها، و همچنین حفظ محدوده همسایگان را بین خوشه و از لحاظ فیزیکی بهبود بهره‌وری را در نظر می‌گیرد، و (3) HDFS داده‌های گراف را بهینه‌سازی نمی‌کند.

هوانگ و همکارانش [1] یک مسئله کارایی با تکنیک گفته شده روهلوف و همکارانش در سیستم مبتنی بر هادوپ نشان دادند [5]. با این حال، شیوه‌ای که هوانگ و همکارانش ارائه داده بودند [1] روهلوف و همکارانش [5] روی آن شیوه کار انجام دادند تا بتوانند سراسر مسئله را تعمیم ببخشند. آنها روی یک نوع فایل خاص و یک الگوریتم خوشه‌بندی ساده تمرکز می‌کنند، و ما معتقدیم که این روش دارای برخی مشکلات زمانی در کلان داده‌های پویا ساخت یافته/ نیمه ساخت یافته/ساخت یافته چندگانه است و با این مشکلات نیز مقابله می‌کند. در بررسی قبلی، هاجر و همکارانش [12] ما محدودیت‌هایی را تایید کردیم. این بررسی نشان داد که چگونه الگوریتم‌های ژنتیک برای خوشه‌بندی چنین داده‌هایی استفاده می‌شوند. ما از این روش در تکنیک هاجر و همکارانش [12]، Pizzuti [43] و [42] همراه با لیستی از کارهای دیگر مانند [48]، [46]، [62] و [63] پس از ساخت یک روش تبدیل به منظور تبدیل اطلاعات مورد نظر گراف داده‌ها استفاده کردیم. نتایج به دست آمده از توسعه کار در [12] به منظور تعمیم هوانگ [1] و همکارانش [5] مورد استفاده قرار گرفتند.

در هادوپ، ایده اصلی این است که محاسبات را به داده‌ها برسانیم؛ به عنوان مثال، نگاشت کاهش، قسمت نگاشت می‌تواند به سرعت محاسبات را در یک ظرف که بخشی از منبع داده می‌باشد انجام دهد. از سوی دیگر، مرحله کاهش اطلاعات داده‌ها را از خروجی‌های نگاشت‌ها جمع‌آوری می‌کند و در بیشتر موارد، بخش‌هایی از داده‌ها باید از طریق شبکه به ظروف مناسب که برخی از گیرنده‌های خاص را اجرا می‌کنند انتقال داده شوند. خروجی چنین نگاشت‌هایی معمولا از چندین نگاشت جمع‌آوری شده به دست می‌آید. برنامه‌نویسان باید این گونه مسائل را تا

حدی کنترل کنند. به همین دلیل است که حتی در آموزش کلودرا توسعه‌دهندگان تمایل دارند دستورالعمل-هایی را برای برنامه‌نویسان به منظور استفاده از نگاشت به جای استفاده از ردیف‌ها ارائه دهند. آموزش کلودرا همچنین توسعه‌دهندگان را تشویق می‌کند تا از زمانبندی پرس‌وجو [64] استفاده کنند تا زمانی که بتوانند از چنین اتصالاتی محافظت کنند. از این رو، ما به شدت بر این باوریم که خوشه هادوپ باید یک حس آگاهی از داده‌ها را داشته باشد، در حالی که داده‌های خروجی نگاشت برای همان کلید باید حداقل تا آنجا که امکان دارد یک دستگاه در مجاورت آن باشد.

از آنجا که برنامه‌های کاربردی و مشاغل متفاوت از یک usecase دیگر هستند، تمام موارد آینده را پوشش نمی‌دهند. یک راه حل برای بهینه‌سازی چنین مشاغلی، جمع‌آوری داده‌ها به عنوان یک چارچوب پیشنهادی است. تمام الگوریتم‌ها در چنین مواردی پوشش داده نمی‌شوند، اما اکثر الگوریتم‌های گراف به طور خاص به داده‌هایی که مرتبط هستند متصل می‌شوند (به عنوان مثال پیمایش گراف).

4 شاخص‌های آگاه داده‌های HDFS

4.1 تبدیل گراف

ما در مقدمه منابع و مسائل مدرن داده‌ها را مورد بحث قرار دادیم. و بیان کردیم که داده‌های ابر از منابع مختلف $S=(S_1,S_2,\ldots,S_Z)$ ناشی می‌شود که S_N منبع Z و $Z \geq 1$ است. این منابع مختلف تولید یا حاوی داده‌ها با ساختارهای مختلف می‌باشد،اما گاهی اوقات برای اشخاص مشابه، و برای داده‌ها و ساختارها مختلف می‌باشد. $D=(DS_1,DS_2,\ldots,DS_Z)$، که در آن DS_Z یک ساختار داده که از منبع Z ناشی می‌شود و شامل مجموعه بی‌پایان DS (ساختار) $DS_Z \subset D, |DS| = \infty$ است.

داده‌ها با ساختار $D \in$ در یک گراف $G(V,E)$ به عنوان یک گراف غیرثباتی و با تعداد راس‌های $|V|=m$ و تعداد لبه‌های $|E|=n$ تبدیل می‌شوند. این تغییر و تحول در بخش معماری به طور کلی توضیح داده شده است.

4.2 خوشه‌بندی گراف

ما به یک گراف از رئوس V و یال‌های E به عنوان $G(V,E)$ اشاره کردیم. همچنین، تعداد رئوس $|V|=m$، و تعداد یال‌های $|E|=n$ و خوشه $C=(C_1,C_2,C_3,\ldots,CJ)$ به عنوان پارتیشن V مجموعه‌هایی را بهم متصل می-کنند. ما C را که خوشه G است و شامل خوشه‌های J نیز می‌شود را فراخوانی می‌کنیم. تعدادی از خوشه‌های j

زمانی که C شامل فقط زیرمجموعه $C_1=V$ دارای حداقل $j=1$ است و زمانی که هر خوشه C_j شامل فقط یک رئوس است دارای حداکثر $j=m$ است. ما خوشه C_1 را به عنوان گراف G تشخیص دادیم. گراف $G[C_j]=(C_j,E(C_j))$، که بدین صورت است $E(C_j)=\{\{WW\}=\{E:WW\epsilon C_j\}\}$. سپس، $E(C)=U_{J=1}E(C_j)$ مجموعه داخلی یال‌های خوشه و E/E(C) مجموعه خارجی یال‌های خوشه است. تعدادی از یال‌های خوشه داخلی دلالت بر m(C) دارد و $\overline{m}(C)$ به تعدادی از یال‌های خوشه خارجی دلالت دارد.

در الگوریتم خوشه‌بندی، ما از یک ماژولار به عنوان یک اندازه شایستگی در روش هاجر و همکارانش استفاده کردیم [12]. سپس ماژول Q به صورت کسری از لبه‌هایی که در گروه 1 یا 2 قرار می‌گیرند، تعریف می‌شود و منهای تعداد لبه‌های مورد انتظار در گروه‌های 1 و 2 برای یک گراف تصادفی با همان توزیع درجه گره به صورت شبکه می‌شود. از این رو، تعداد واقعی یال‌ها بین v و w منهای تعداد مورد انتظار بین آنها برابر $A_{vw}-(K_vK_w)/2m$ است. ماژولار می‌تواند به صورت معادله زیر بدست آید (2) [57]:

$$(2)Q = \frac{1}{2m}\sum_{vw}[A_{vw} - \frac{k_v*k_w}{2m}]\frac{S_vS_w+1}{2}$$

به این نکته توجه داشته باشید که معادله (2) فقط دو گروه شبکه را پارتیشن‌بندی می‌کند. برای شناسایی جوامع متعدد در یک نمودار؛ فرمول باید به صورت زیر تعریف شود (3) [57]:

$$(3)Q = \sum_{vw}\left[\frac{A_{vw}}{2m} - \frac{k_v*k_w}{(2m)(2m)}\right]\delta(C_v, C_w) = \sum_{i=1}^{c}(e_{ij} - a_i^2)$$

در این فرمول e_{ij} کسری از لبه‌ها با آخرین راس‌های جامعه i و دیگری در جامعه j است (4) [57]:

$$(4)e_{ij} = \sum_{vw}\frac{A_{vw}}{2m}1_{v\in c_i}1_{w\in c_j}$$

و a_i کسری از یال‌های پایانی است که رئوس را در جامعه iام به عنوان یک معادله به هم پیوند می‌دهد (5) [57]:

$$(5)a_i = \frac{k_i}{2m}\sum_i e_{ij}$$

4.3 فرضیات و توزیع گراف

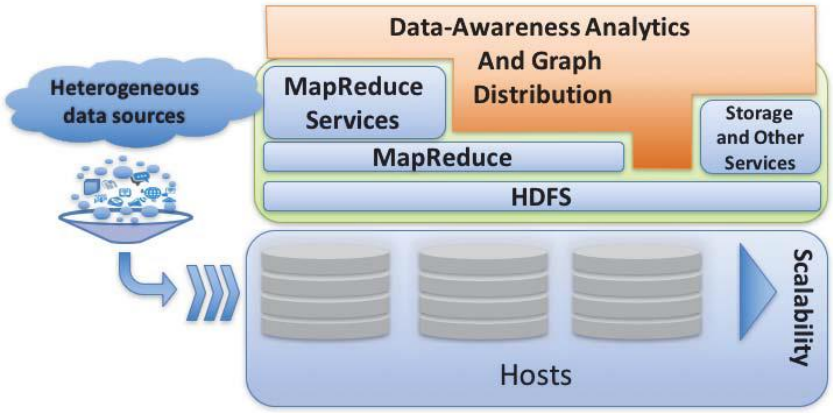
همانطور که قبلا توضیح داده شد، سه چالش عمده با بهینه‌سازی HDFS مواجه شدند؛ و دو نفر از آنها در مورد نحوه هشدار هادوپ و توزیع داده‌ها توضیح دادند. فرضیات و آزمایشاتی را که انجام دادیم عبارتند از (1) ذخیره داده‌های درون خوشه‌ای در یک ماشین واحد و (2) ذخیره داده‌های میان خوشه‌ای که نزدیک ماشین‌ها بودند و

گامی بزرگ در جهت بهینه‌سازی HDFS برداشتند. $M=(M_1,M_2,...,M_I)$ یک مجموعه از ماشین‌هایی است که منابع محاسباتی هادوپ را نشان می‌دهد، و $I\epsilon[0,\infty)$ متعلق به تعداد مجموعه شایستگی نیز است. و M_j، M_{j+1}، M_{i+n} ماشین‌هایی هستند که به صورت فیزیکی فاصله شبکه آن بین M_i و M_{i+1} نزدیکتر از M_i و M_{i+n} است. خوشه C_j با تمام رئوس‌ها و یال‌ها باید در همان گراف پارتیشن یا حداقل باید در همان ماشین M_i جایگزین شود. زمانی که در سمت چپ M_i وجود ندارد باید کمترین مقدار M_i+1 و غیره جایگزین شود، و نزدیک‌ترین ماشین، بهترین نتیجه را ارائه می‌دهد. C_j و C_{j+1} بیشترین یال‌های خوشه میانی نسبت به C_j و C_{j+n} را دارد سپس، C_j و C_{j+1} باید در کوچکترین مقدار احتمالی m جایگزین شود، و بازه M، و M_{i+m} (نزدیک‌ترین ماشین فیزیکی) بین $0\leq m\leq 1$ است.

5 کل معماری

چارچوب خوشه‌بندی ما (DEGA-Gen) بخشی از داده آگاهی ارائه شده است که در ماژول روی بالاترین داده ذخیره‌سازی توزیع شده در حال اجرا می‌باشد و در شکل 4 نشان داده شده است.

این چارچوب با HDFS و سرویس‌هایی که قابل دسترس هستند ارتباط برقرار می‌کند تا بتواند خوشه‌های به‌روز شده را به عنوان جریانی از داده‌ها در HDFS ارائه کند. هدف ما دستیابی به بهینه‌سازی به همراه قرار دادن داده‌های مرتبط و کاهش سربار در حرکت داده‌ها بین میزبان است. انتقال داده‌ها عمدتا در فرآیندهای تجمعی یا پیوند رخ می‌دهند.



شکل 4. ماژول آگاهی داده‌ها و الگوریتم خوشه‌بندی تکاملی توزیع شده که بخشی از هادوپ است.

5.1 ساختار توزیع شده گراف‌های RDF

گام اول چارچوب داده‌ای HDFS، به یک گراف RDF توزیع شده تبدیل می‌شود. این روند با استفاده از منابع باز Apache Jena و Apache Jena Elephas انجام می‌شود. داده‌های ارائه شده HDFS مجموعه‌ای از داده‌ها را چهار برابر افزایش می‌دهد؛ و دلایل این فرآیند در خوشه‌بندی براساس ژنتیک توضیح داده می‌شوند. برخلاف استفاده گسترده از عمل چهارگانه، ما از فیلد اضافی به صورت چهارگوش استفاده کردیم و وابستگی خوشه سه‌گانه را در گراف چهارگانه نشان دادیم؛ و آن را فیلد نامیدیم (کروموزوم ID). این فرآیند استفاده از ذخیره‌های چهارگانه را برای بهبود فرآیند فرعی (رمزگذاری و نمایش خوشه‌بندی توزیع شده) به کار می‌رود.

تمام انواع داده‌ها را نمی‌توان به گراف تبدیل کرد. با این حال، کلان داده‌ها در مورد use-caseها و برخی داده-های use-case هستند. این یک تغییر مستقیم (SHARD)، و اسناد پایگاه داده ارتباطی است (به عنوان مثال) و می‌توان آن را به عنوان یک گراف RDF نشان داد که هر ویژگی از آن نشان دهنده رکورد هر رابطه بین دو گره است، و یک گره نشان دهنده مقدار فیلد کلیدی و گره دیگر مقدار نشان دهنده فیلد ویژگی است. در موارد دیگر، (مانند داده‌های متنی)، روابط بین نقاط داده می‌توانند براساس use case تعریف شوند. به عنوان مثال، پردازش زبان طبیعی را می‌توان برای هر رابطه (کلمه پیش از آن) و (کلمه پس از آن) برای هر کلمه به منظور ایجاد مدل پیش‌بینی تعریف کرد.

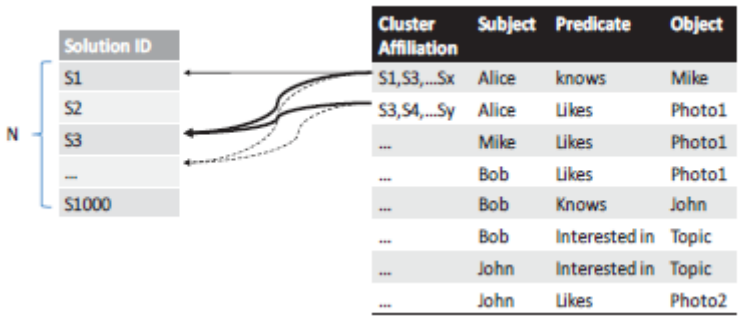
5.2 خوشه‌بندی مبتنی بر ژنتیک

5.2.1 بیان رمزگذاری

ما از کدگذاری [65] برای غلبه بر مسائل کدگذاری بزرگ در مطالعات قبلی که در [65] ذکر شده بود استفاده کردیم. چنین رمزگذاری از تعریف خوشه‌ها حاصل می‌شود.

با این حال، حتی با چنین کدگذاری در [65]، راه حل‌ها هنوز هم می‌توانند به صورت طولانی‌تر به عنوان یک مقیاس داده‌ای بیان شوند. در نتیجه، کلاینت GA در خارج از حافظه به صورت دستی راه حل‌ها را به عنوان یک مقیاس داده اجرا می‌کند. از سوی دیگر، ما از تکنیک ذخیره‌سازی به عنوان اطلاعات اضافی به همراه گراف سه-گانه HDFS، و به همراه پوشش نقاط داده‌ای triples<Node><Predicate><Node>، روهلوف و همکارانش [5]، quadruples<Node><predicate><Node>into<chromosome_part> استفاده می‌کنیم. ما به <chromosome_part> به عنوان فهرست solution_Ids که متعلق به جمعیت گره خاص

می‌باشد اشاره کردیم. این کدگذاری به جمعیت فهرستی از اعداد صحیح با اندازه ثابت در کلاینت GA که solution_IDs نام دارد منجر می‌شود. این تکنیک این امکان را فراهم می‌کند که کلاینت بتواند خوشه GA را روی مجموعه داده‌ها با اندازه بزرگ مقیاس‌بندی کند و بتواند آن را در HDFS حفظ کند.



شکل 5. انتقال RDF سه گانه به RDF چهارگانه (راه حل کدگذاری)

این ایده ایده‌ای بود که راه حل‌ها را به عنوان داده‌ها و به عنوان ارث‌بری تمام ویژگی‌های مقیاس‌پذیری که در نمودار قابل اعمال می‌باشد مورد ارزیابی قرار می‌داد. بنابراین، اندازه جمعیت X در کلاینت دارای یک اندازه ثابت است و از اندازه داده (X) صرف نظر نیز می‌کند. ما به این تکنیک به عنوان کروموزوم‌های توزیع شده، و به عنوان یک مفهوم در مورد توزیع ژن‌ها که با راه حل‌هایی همراه است اشاره کردیم. شکل 5 توضیح می‌دهد که چگونه داده‌های گراف در فرمت RDF ذخیره می‌شوند و توضیح می‌دهد که چگونه کدگذاری را حل‌ها در داده‌های RDF انجام می‌شود.

ما از Apache Jena و Jena Alephas استفاده کردیم و منابع باز را برای مطابقت نیازمان تغییر دادیم. کلاس Convert_to_quads_Chromo برای تبدیل گراف RDF به Quads در شکل 5 توسعه داده شده است. این کلاس حاوی ترکیب کننده، کاهنده، نگاهنده و قابلیت نوشتن مناسب و همچنین کلاس‌های ورودی و خروجی فرمت شده برای پردازش داده‌های RDF نیز است. هر بلوک سه‌گانه داده تبدیل به یک چهارم ژن تصادفی (بخشی از راه حل‌ها) که متعلق به آن است می‌شود و سپس آن را در HDFS ذخیره می‌کند.

5.2.2 توابع هدف

در الگوریتم خوشه‌بندی، ما مدولاسیون را به عنوان یک هدف در تکنیک هاجر و همکارانش به حداکثر رساندیم [65]. در [57]، مدولاسیون Q تعریف می‌شود و سپس به عنوان کسری از لبه‌ها در گروه 1 یا 2 قرار می‌گیرد، و

منهای تعداد لبه‌های مورد انتظار در گروه‌های 1 و 2 در یک گراف تصادفی با همان توزیع درجه گره به عنوان شبکه می‌شود. از این رو، تعداد واقعی لبه‌ها بین v و w و بدون حد انتظار از لبه‌های بین آنها برابر $A_{vw} - (k_v k_w)/2m$ است. لذا به معادلات 5-1 در بخش داده‌های HDFS مراجعه کنید. به این نکته توجه داشته باشید که حداکثرسازی ماژلار تنها هدف نیست. هدف دیگر این است که طول راه حل را به حداقل برسانیم. با توجه به لبه‌های درون خوشه‌ای به عنوان لبه‌های بین خوشه‌ای، برخی از راه حل‌ها بدون تغییر در مدولاسیون به دست می‌آیند. از این رو، این راه حل‌ها نیاز به داشتن شایستگی کمتر دارند اما به طور کامل نادیده گرفته می‌شوند (ترکیبی از راه حل‌های دیگر که ممکن است به خوشه‌بندی بهتری منجر شود).

از آنجایی که ارزیابی ما در مورد مجموعه داده‌ها، مفاهیم و روابط بین آنها در هر دو جهت در نظر گرفته می‌شود (یک گراف بی‌جهت)، ما از معادله (3) استفاده کردیم. Use caseها که در روابط تعریف شده به یک گراف بی-جهت منجر می‌شوند، و دارای یک مدولاسیون گسترده‌ای برای یک گراف جهت‌دار ارائه می‌کند و می‌تواند از آن استفاده کند. از سوی دیگر، هدف خوشه‌بندی یافتن جابجایی بهتر برای داده‌های گراف است. از این رو، گراف-های جهت‌دار به عنوان گراف‌های بدون جهت در نظر گرفته می‌شوند در حالی که خوشه‌بندی لزوما کاربر را وادار نمی‌کند که از همان فرضیه در هنگام پرس‌وجو داده‌ها استفاده کند.

5.2.3 جزئیات مراحل خوشه‌بندی ژنتیک

جمعیت اولیه

جمعیت اولیه فرآیند ایجاد مجموعه‌ای از راه حل‌های متنوع است. همانطور که در بخش کدگذاری و زیربخش توضیح داده شده است، ما داده‌های RDF سه‌گانه را به چهارگانه تبدیل کردیم و قادر بودیم یک ژن (بخشی از راه حل) را برای هر راس ذخیره کنیم، تا جایی که ژن به فهرستی از IDهای راه حل به صورت تصادفی متعلق به یک راس خاص باشد. به عنوان مثال، اگر یک D چهارگانه دارای S1 و S5 به عنوان ژن باشد، آن را به عنوان راه حل S1 و S5 ترجمه می‌کند؛ و هر دو لبه D را به عنوان یک لایه بین خوشه‌ای که دو خوشه جداگانه را متصل می‌کنند در نظر می‌گیرند.

در $\forall T, \in T$ مجموعه‌ای از راه حل‌های $\cup S_n \subseteq S$ وجود دارد، که در آن T، $\cup$ از یک t سه گانه گراف G ایجاد می‌شود و S مجموعه‌ای از راه حل‌های S در جمعیت است. این بسیار مهم است که حداکثر اندازه این

فهرست، را که اندازه عدد صحیح جمعیت است در نظر داشته باشید. فرآیند جمعیت اولیه در شکل 5 نشان داده شده است. کلاینت GA فقط یک آرایه دو بعدی از IDهای راه حل و شایستگی ماژول (اعداد صحیح و اعداد شناور) را ذخیره می‌کند، و به کلاینت این امکان را می‌دهد که فرآیند انتخاب و عملگرهای GA توزیع شده را آغاز کنند. کار بر روی یک آرایه دو بعدی ثابت کوچک، که در آن ژن‌های واقعی در بلوک‌های داده‌ای در یک روش توزیع شده با استفاده از HDFS ذخیره می‌شوند مقیاس‌پذیری بیشتری را ارائه می‌کند.

ارزیابی راه حل‌ها

ارزیابی با استفاده از توابع هدف که در بخش هدف توصیف شده بودند انجام می‌شود. هر راه حل توسط محاسبات مدولاسیون نظیر گراف ارزیابی می‌شود، و یال‌های موجود در گراف به عنوان یال‌های خوشه میانی مشخص می‌شوند. ما خوشه‌هایی را که با عمل حذف یال‌ها را مشخص می‌کند و مولفه‌های گراف غیرمتصل را در نظر می‌گیرند تشخیص دادیم. سپس، ما مدولاسیون را با در نظر گرفتن تشخیص دوباره یال‌ها به عنوان یال‌های خوشه میانی محاسبه کردیم.

فرآیند محاسبه مدولاسیون در یک گراف بزرگ هم در منابع و هم در زمان وقت‌گیر است، بنابراین ما تصمیم گرفتیم آن را با استفاده از وظایف توزیع شده به صورت چهارگانه به همراه داده‌های اضافی بهبود ببخشیم. با استفاده از HDFS و مجموعه داده‌های چندگانه در ماشین، ما قادر هستیم هر بار مجموعه‌ای از راه حل‌ها (تولید) را به صورت همزمان پردازش کنیم.

Given a Population_ID list S that contain ID's of solutions to be evaluated

```
Map (Key index, Value Quad):
  ForEach solutionID in S:
      If Quad.GetGenes in solution:
          Quad.marked = True
          Emit (solutionID , Quad)
      Else:
          Quad.marked = False
```

شکل 6. ارزیابی راه حل‌های توزیع شده (وظیفه نگاشت).

کلاینت الگوریتم‌های راه حل جاری را به صورت چهارگانه در HDFS ذخیره می‌کند، و فهرستی از شناسه‌های راه حل (فهرست اعداد صحیح) را برای ارزیابی می‌فرستد. شکل 6 کارهای ارزیابی نگاشت را نشان می‌دهد.

تابع نگاشت برای هر یک از گراف‌ها به صورت چهارگانه بخشی از گراف چانک را در نظر می‌گیرد. Jena Elephas با کلاس ورودی و خروجی اصلاح شده چهارچوب کروموزوم به جای چهارچوب پیش‌فرض مورد استفاده قرار می‌گیرد. هر ظرفی که در خوشه HDFS کار انجام می‌دهد عملیات نگاشت را بر روی تکه‌های گراف تخصیص می‌دهد. پس از ترسیم تمام چانک‌ها، در جفت <Keys,Values> شناسه‌های راه حل و چارچوب‌هایی که بخشی از راه حل‌های مربوطه هستند، کار ترکیب کردن را بدین صورت انجام می‌دهند. تمام مقادیر یک کلید همانند <Key,List of Values> که هر یک دارای راه حل‌ها و فهرستی از چارچوب‌های مشخص شده و علامت‌گذاری شده نیز هستند (گراف‌هایی که حاشیه‌های بینشان به صورت خوشه‌ای علامت‌گذار شده‌اند) نشان داده می‌شوند. مرحله نهایی شامل کاهش وظایف است که در شکل 7 توضیح داده شده است.

Given a solution S and a set of Quads marked based on S, as mappers' outputs and reducers' input for a graph G with N Quads

```
Reduce (Solution S, EdgesQuads [E1,E2,E3,....EN]):
  ForEach Quad E in QuadssList:
    If E.marked = True:
      MarkedQuads.append(E)
    Else:
      UnMarkedQuads.append(E)
  Endfor
  Communities = FindComponents(MarkedQuads, UnMarked-
Quads)
  Modularity = 0
  ForEach Community C in Communities:
    DegreeFraction = (C.InnerEdges *2+ C.OutterEdges) / (2*N)
    Modularity += (C. InnerEdges /N)-(DegreeFraction)^2
  Endfor
  Emit (S, Modularity)
```

شکل 7. ارزیابی راه حل‌های توزیع شده (وظیفه کاهش).

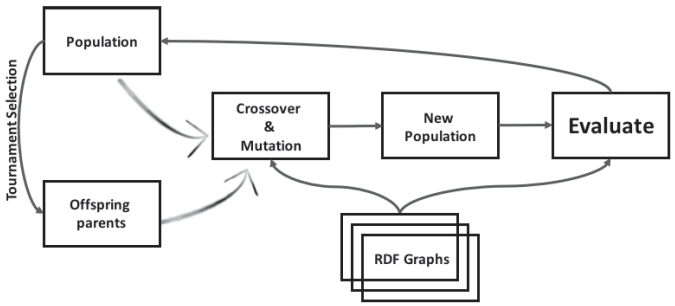
تابع FindComponents با استفاده از یک الگوریتم خطی اصلاح شده تعداد لبه‌های دورن خوشه‌ای و لبه‌های بین خوشه‌ای را برای هر جامعه ذخیره و سپس اجرا می‌کند. زمانی که وظایف کاهش پایان می‌یابند، نتایج کاهنده‌ها در HDFS نوشته می‌شوند و نتایج شامل راه حل به همراه ماژولار می‌شود. نتایج حاصل از یک آرایه دو بعدی ثابت از شناسه‌های راه حل اعداد صحیح به صورت مناسب برای هر راه حل ایجاد می‌شود. الگوریتم تکاملی این فایل را می‌خواند و همچنان ارزیابی فرآیندهای انتخاب، تقاطع و جهش ادامه دارد. در نسل گذشته، یک قطعه اضافی از اطلاعات کنترل شده توسط یک متغیر بولی پیکربندی و سپس در HDFS نوشته می‌شود؛ و این قطعه شامل وابستگی خوشه‌ای در هر گره است. دلیل اینکه آنها فقط در نسل گذشته نوشته می‌شدند، این

است که هزینه نوشتن در HDFS قبل از اینکه در هر زمان مورد نیاز باشد تعیین و سپس کاهش پیدا می‌کرد.

انتخاب، جهش و Crossover

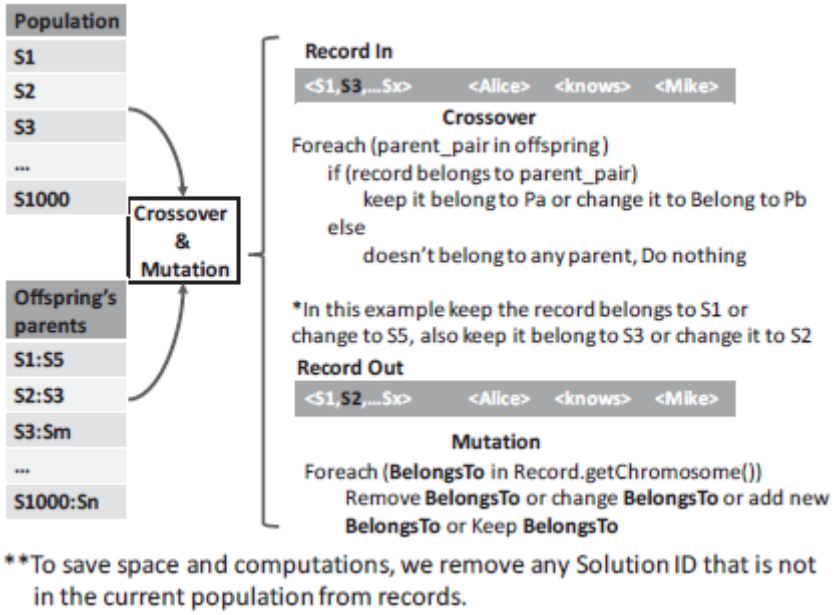
ما کروموزوم‌ها را به صورت توزیع شده ذخیره کردیم، ما نیاز داشتیم که عملگرهای GA مورد استفاده را در منبع باز Jmetal تغییر دهیم تا بتوانیم آنها را بر روی چارچوب مربوط به گراف اجرا کنیم. این روش بوسیله ماژول‌های تقاطع توزیع شده و ماژول‌های crossover انجام می‌شود، و در عوض وظایف تقاطع و جهش در جمعیت مربوطه ایجاد می‌شوند.

بعد از ارزیابی جمعیت، فرآیند انتخاب که مبتنی بر هر راه حل ID و شایستگی است شروع می‌شود. انتخاب تورنمنتن یک انتخاب مورد استفاده است، و دلیل آن این است از همگرایی راه حل‌های بهینه محلی، که مبتنی بر روش کدگذاری است جلوگیری می‌کند. با درجه‌بندی جمعیت و انتخاب راه حل هر کلاس، مجموعه‌ای از والدین همراه با IDهای فرزند ایجاد می‌شوند. شکل 8 یک نمودار سطح بالا است که مراحل ایجاد عملگر الگوریتم GA را نشان می‌دهد.



شکل 8. الگوریتم ژنتیک توزیع شده برای خوشه‌بندی RDF

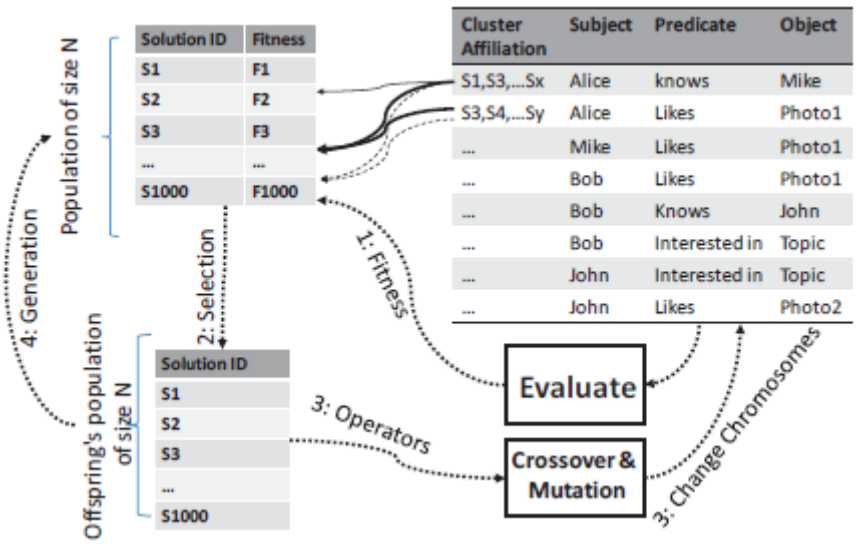
راه حل‌های ذخیره‌سازی و کدگذاری در روش توزیع شده مزایای جمعیت‌های کوچک و ثابت را در یک طرف کلاینت به ارمغان می‌آورد. با این حال، عملگرهای GA در منبع باز Jmetal نیاز به اصلاح و همچنین NSGALL دارند، تا ما بتوانیم از آنها استفاده کنیم. NSGALL راه حل‌های اصلی جمعیت و فرزندان را ایجاد می‌کند و در یک زمان ممکن این راه حل‌ها را ارزیابی می‌کند؛ چنین وضعیتی سربار کارهای HDFS است. در عوض، ما NSGALL را به DNSGALL (توزیع NSGALL) با ایجاد مجموعه‌ای از راه حل‌ها و سپس ارزیابی عملگرهای GA را در MR2 تغییر دادیم. شکل 9 وظیفه جهش توزیع شده تقاطع و توزیع را نشان می‌دهد.



شکل 9 عملگر جهش و تقاطع در جستجوی ژنتیکی

وظیفه جهش و تقاطع توزیع شده گرفتن جمعیت به عنوان ورودی در طول نتایج انتخاب است، برای هر یک از فواصل داده‌ای، کروموزوم‌ها به ترتیب تغییر می‌کنند. وظیفه هر راه حل ID (ژن) حذف جمعیت فعلی که هیچ تعلقی ندارند تا فضای محاسباتی را بتواند ذخیره کند. سپس، همانطور که در شکل 8 نشان داده شده است، جمعیت جدید فرزندان به ارزیابی ارسال می‌شوند. در اینجا ما باید به این نکته توجه داشته باشیم که راه حل-هایی که متعلق به نسل قبلی می‌باشند، ارزیابی نخواهند شد به این دلیل که آنها از قبل آمادگی داشتند. این یک تکنیک کپی شایستگی است که حجم زیادی از محاسبات را زمانی که کلان داده‌ها مجموعه‌ای طولانی از نسل‌ها را به کار می‌برند ذخیره می‌کند.

فرآیندهای بازنمایی، جمعیت اولیه، ارزیابی، انتخاب و ارزیابی جمعیت فرزندان در شکل 10 نشان داده شده است. اعداد نشان دهنده دستورات فرآیندها و وظایف هستند. ما داده‌های پویا را به عنوان یکی از 5 محدودیت-های کلان داده‌ها شناختیم (سرعت، تنوع، وراثت، مقدار و حجم)، و الگوریتم را زمانی که یک راه حل برای سلسله‌ای از نسل‌ها همگرا می‌شود، به حالت تعلیق درمی‌آید و سپس کار را ادامه می‌دهد تا زمانی که داده‌های جدید به منظور رسیدن به آخرین نسل به دست آیند.



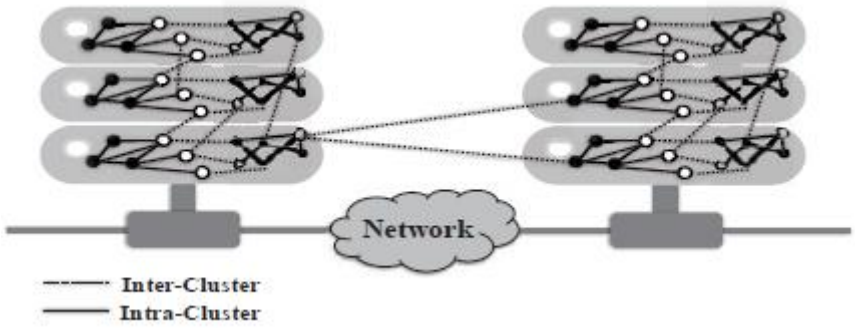
شکل 10. جریان فرآیند خوشه‌بندی الگوریتم ژنتیک توزیع شده.

تعلیق الگوریتم تضمین می‌کند که خوشه‌بندی به داده‌های جدید اعمال می‌شود در حالی که داده‌های قدیمی دارای بهترین وابستگی نیز هستند، بنابراین نیازی نیست که فرآیند خوشه‌بندی از ابتدا آغاز شود.

5.3 پارتیشن‌بندی و جاگذاری کردن

پس از خوشه‌بندی داده‌های RDF، آخرین مرحله این بود که داده‌ها را مجددا تقسیم کنیم و گراف را به ترتیب چهارگانه کنیم. در این مرحله هدف این بود که به جای عمل چهارگانه که متعلق به یک خوشه بود با درجه بالایی از اتصال برای اطمینان خوشه چهارگانه را پارتشین‌بندی می‌کردند. هدف دیگر این بود که خوشه‌های میانی را به صورت فیزیکی به خوشه پارتیشن‌بندی شده متصل کند، تا فاصله نگاشت خوشه میانی نشان داده شود. شکل 11 تخصیص مورد نظر را به صورت چهارگانه نشان می‌دهد، و فرض بر این است که فاصله افقی در شکل نشان دهنده فاصله فیزیکی بین گره‌های محاسباتی است (فاصله مسیریابی شبکه).

ما با توجه به تعداد مسیرهای بین آنها (شبکه، مسیریاب‌ها، سوئیچ‌ها و ...) فاصله گره‌های HDFS را تعیین کردیم. ما HDFS در بیشتر دستگاه‌ها بهم متصل کردیم سپس با استفاده از شبکه‌های متعدد به منظور ایجاد فاصله در مسیریابی راه اندازی کردیم. همانطور که در شکل 11 نشان داده شده است اسکریپت جایگزین شبکه چهارگانه نیز شده است.



شکل 11. قرار دادن شبکه‌های چهارگانه در گره‌های HDFS مختلف.

پارتیشن‌ها براساس تعداد ماشین‌ها ایجاد می‌شوند، هر ماشین دارای پارتیشن مخصوص به خود است. وظیفه نگاشت کاهش اسکن کردن کوآدراپل و مکان کوآدراپل که مربوط به خوشه تصادفی در یک پارتیشن بود است، سپس کوآدراپل جایگزین شده را ، که منجر به تمام اتصال‌های ID خوشه شده بود ارسال کرده بود و در مرحله بعدی ذخیره کرده بود. دومین اسکن به صورت چهارگانه نزدیک به خوشه‌های درونی است و در همان پارتیشن قرار می‌گیرد و در مجموعه داده‌های اصلی منتشر می‌شود. علاوه بر این، دومی در نزدیکترین خوشه‌های میانی قرار می‌گیرد. زمانی که خوشه‌های متصل به هم وجود نداشته باشد، یک خوشه به صورت تصادفی از مجموعه داده‌ها انتخاب می‌شود تا زمانی که اطلاعات بیشتری در دسترس نباشد. به عنوان مثال، شکل 11، نشان می‌دهد که خوشه‌ها چگونه با سه اتصال بین خوشه‌ای در همان گره HDFS قرار می‌گیرند و چگونه با دو اتصال بین خوشه‌ای در گره بعدی HDFS قرار می‌گیرند، و چگونه خوشه‌های بیشتری در گره‌های بعدی قرار می‌گیرند.

6 آزمایشات و نتایج

ما این آزمایش را به دو بخش تقسیم کردیم: بخش اول طراحی و آزمایش الگوریتم خوشه‌بندی در ذخیره‌سازی گراف به منظور تست نتایج خوشه‌بندی و بخش دوم در مورد تست‌ها و مقایسه تاثیر چارچوب بهینه‌سازی در HDFS است. تمام نمودارها و مدل‌های فرآیند با استفاده از جدول پردازش می‌شوند [66].

6.1 تبدیل گراف و خوشه‌بندی

ما صحت الگوریتم خوشه‌بندی را تایید کردیم و اطمینان حاصل کردیم که نتایج قابل اعتماد و قابل مقایسه نیز هستند. ما برخی از مجموعه داده‌های شناخته شده را با دقت انتخاب کردیم و اطمینان دادیم که آنها مجموعه داده‌هایی هستند که در بررسی‌های قبلی برای مقایسه مورد استفاده قرار گرفتند. این مجموعه‌ها عبارتند از:

انجمن کاراته زاخاری: گراف شامل 34 رئوس و 78 لبه است. گره‌ها نشان دهنده اعضای انجمن دانشگاه کاراته می‌باشند و ارتباطات بین آنها نشان دهنده الگوهای ارتباطی هستند. این انجمن در سال 1997 جمع‌آوری شد [67].

دلفین پوزه بطری معمولی: شبکه از 62 دلفین به همراه تعاملات بین آنها ایجاد شده و در طول 7 سال، یعنی سال 1994 جمع‌آوری شده است.

کتاب‌های سیاسی ایالات متحده: شبکه از 105 گره و 441 لبه ایجاد شده است. شبکه نشان می‌دهد که کتاب-های مربوط به سیاست ایالات متحده، اغلب باهم خریداری می‌شدند.

فوتبال دانشکده آمریکایی: شبکه از 115 گره که نشان دهنده تیم هستند، و 613 لبه اتصال ایجاد شده است.

جدول 1 نتایج اعتبارسنجی الگوریتم را نشان می‌دهد و آن را با بعضی از الگوریتم‌های محبوب مقایسه می‌کند. الگوریتم ما در بعضی موارد مدولاسیون را به حداکثر رساندند و حالت مدولاسیون را در بقیه بدست آورده‌اند. بعضی از الگوریتم‌ها به دلیل مدولاسیون بالا حذف شدند؛ همانگونه که دانیل آلوئیز، سونیا کافییری و همکارانش نیز وجود دارند چنین نتایجی برای خوشه سخت غیر ممکن است [68]. آنها هر یک از این مجموعه داده‌های مدولاسیون را به صورت بهینه پیدا و ثابت کرده‌اند.

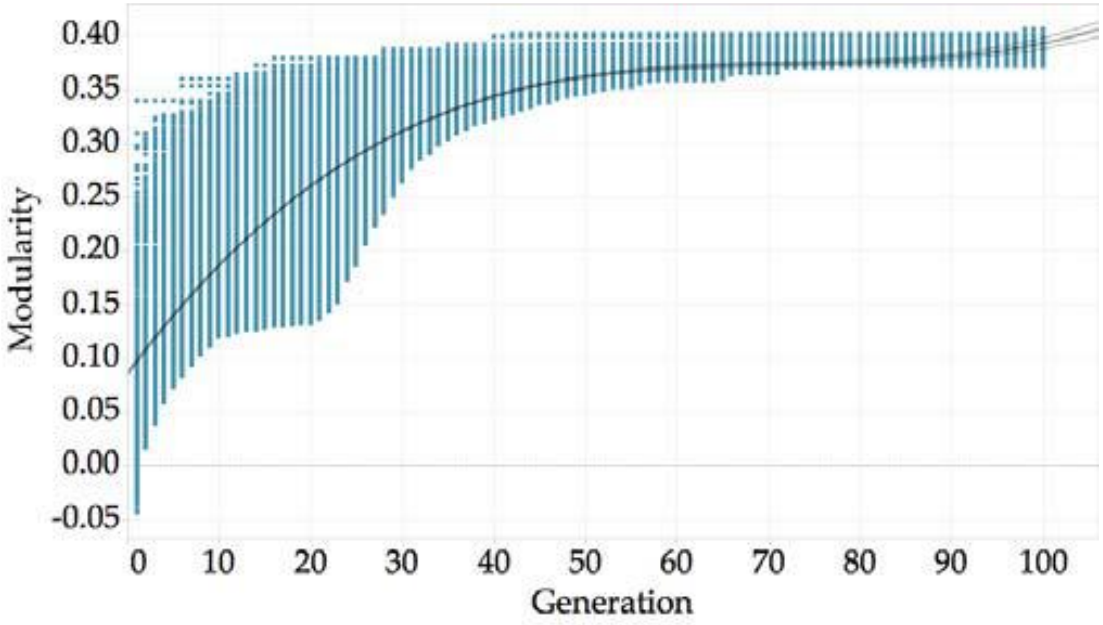
جدول 1 مقایسه به حداکثر رساندن مدولاسیون

Dataset	GN	CNM	L Max	GATHB	MOG A-Net	**Our Method**
Karate	0.4	0.380	0.419	0.4	0.416	**0.416**
Dolphins	0.52	0.495	0.523	0.52	0.505	**0.528**
Football	0.6	0.577	0.61	0.55	0.515	**0.539**
Books	0.51	0.502	0.526	0.52	0.518	**0.523**

نتایج در جدول 1 و [68] ثابت می‌کند که رویکرد ما نتایجی را که به راه حل‌های بهینه نزدیک می‌باشند را ارائه می‌دهد (در مقایسه با سایر الگوریتم‌های محبوب) و کیفیت این نتایج در بیشتر موارد بهتر است. بعضی از موارد با مدولاسیون نسبتا پایینی نشان داده شده‌اند.

ما دریافتیم کهانتخاب در نقش راه حل‌ها بسیار اهمیت دارد. علاوه بر این، برای مجموعه داده‌های خاص، انتخاب باینری به مدولاسیون بالا را در نسل‌های کوچکتر پوشش می‌دهد، در حالی که، انتخاب تصادفی می‌تواند نرخ

بالاتری از جهش‌ها را از شایستگی بهینه محلی فراهم کند.



شکل 12. شایستگی جمعیت در مقابل ترسیم توزیعی نسل (شبکه انجمن کاراته).

ارزیابی‌ها پس از هر جمعیت گزارش می‌شود و همگرایی راه حل‌ها را برای نسل‌ها تجزیه و تحلیل می‌کند. ما گراف‌ها و مدل‌های فرآیند محاسبه شده را با از بین بردن آرایه‌های جمعیت و با استفاده از ترسیم توزیعی به منظور نشان دادن نتایج بصری هر نسل را تولید کردیم. ما بین توزیع مدولاسیون و تعداد نسل‌ها به منظور استخراج مدولاسیون‌ها ارتباط برقرار کردیم. شکل 12 توزیع مدولاسیون را در مقابل نسل نشان می‌دهد.

برای مقیاس رویکرد کلان داده، ما از LUBM برای تولید داده‌های گراف RDF استفاده کردیم و همانطور که در جدول 2 نشان داده شده است از خوشه‌ای با ویژگی زیر، استفاده کردیم. هر کانتینری دارای 48 دیسک قابل دسترس، و دو هسته پردازنده و 4 گیگابایت حافظه می‌باشد.

ما از 6 گره محاسباتی در مقایسه با 10 و 20 گره در بررسی‌های مشابه استفاده کردیم. ما فقط تعداد گره‌ها را به منظور تایید اثر ارتباطات شبکه و فاصله در شبکه مشاهده کردیم. با این حال، YARN مفهومی منطقی است که منابع را نسبت به تعداد گره‌ها مقایسه می‌کند. خوشه محاسباتی و پیکربندی ما به 86 کانتینر، که هر کدام 4 گیگابایت حافظه و 2 بند (یک هسته) CPU (حداکثر 2.93 گیگاهرتز و حداقل 2.00 گیگاهرتز)، 344 گیگابایت حافظه و 172 بند CPU (86 هسته)، و در مقایسه با کل 80 گیگابایت حافظه و 40 هسته CPU و 40 دیسک مساوی بین 20 گره محاسباتی هانگ و همکارانش تقسیم می‌شود [1].

جدول 2 پیکربندی خوشه هادوپ

Machine		Threads	Memory	Disks
Master	Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz	72	64	10
Node1	Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz	56	64	10
Node2	Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz	40	64	10
Node3	Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz	40	64	10
Node4	Intel(R) Xeon(R) CPU X5570 @ 2.93GHz	16	96	2
Node5	Intel(R) Xeon(R) CPU E5520 @ 2.27GHz	16	48	6

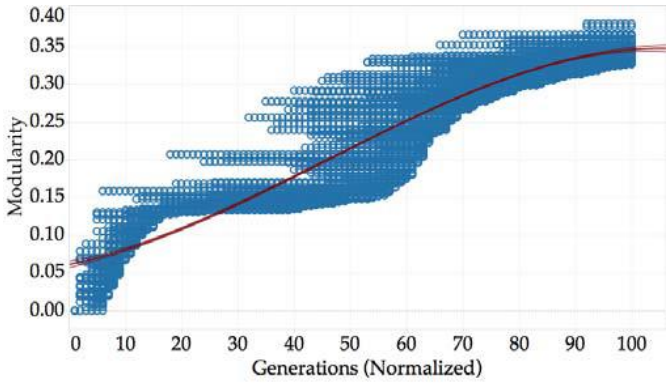
دانشگاه لهایف بنچمارک (LUBM) یک دانشگاهی است که در حوزه آنتولوژی داده‌های مصنوعی OWL و RDF با مقیاس‌پذیر بالا و به چهارده پرس‌وجو که نشان دهنده ویژگی آن دانشگاه است. LUBM یک معیار گسترده است که از جامعه وب معنایی استفاده می‌کند.

ما مجموعه‌ای از چندین داده با اندازه‌های مختلف به منظور مقایسه رفتار الگوریتم تولید کردیم. ما قبل از خوشه‌بندی به طور پیش‌فرض نوع تایپ را حذف کردیم و یک روش برای افزایش کیفیت خوشه‌ها و پیچیدگی گراف ارائه دادیم [1]. زمان اجرای راه حل‌های جمعیت اولیه را تجزیه و تحلیل کردیم؛ و حجم جمعیت موجود در آن 1000 راه حل در هر نسل است. جدول 3 زمان اجرارا به منظور تبدیل گراف به یک گراف چهارگانه و جمعیت اولیه را نشان می‌دهد. خطای مقادیر زیادی از داده‌ها، به همراه مقدار پرس‌وجوی آنها روی داده‌ها پردازش می‌شوند و قادر هستند تکنولوژی‌های جدید را بهینه‌سازی کنند [22]. به این نکته توجه کنید که چارچوب ما، زمانی که داده‌ها به شدت پردازش می‌شوند یا به طور مداوم مورد سوال قرار می‌گیرند به بهترین نحو مورد استفاده قرار می‌گیرد و باعث پاسخ سریع و کاهش هزینه‌های سیستم سخت‌افزاری نیز می‌شود.

جدول 3 زمان اجرای الگوریتم & جمعیت اولیه (مجموعه داده‌های LUBM)

Number of triples	Initialize Popula-tion (S)	Algorithm Run Time (Minutes)
8,970,048	13.556	21.7
20,637,270	19.621	31.6
30,285,222	28.611	47
221,140,408	207.314	261(~4.3 hours)

ما داده‌های LUBM را بیشتر تجزیه و تحلیل کردیم. مقدار زیادی مقادیر سه‌گانه وجود دارد، و در آن جمعیت اولیه با یال‌های تقاطع خوشه‌ای به صورت تصادفی در طول زمان اجرا نمی‌شوند. شکل 13 همگرایی جمعیت را با به حداکثر رساندن مدولاسیون یک مجموعه داده که 30M چهارچوب دارد در طول زمان نشان می‌دهد.



شکل 13. به حداکثر رساندن مدولاسیون همگرایی (30 میلیون LIBM سه‌گانه)

در شکل 14 فرآیند مدل 30 میلیون مجموعه داده سه‌گانه LUBM نشان داده شده است.

Trend Lines Model

A polynomial trend model of degree 3 is computed for Modularity given Generation. The model may be significant at p <= 0.05.

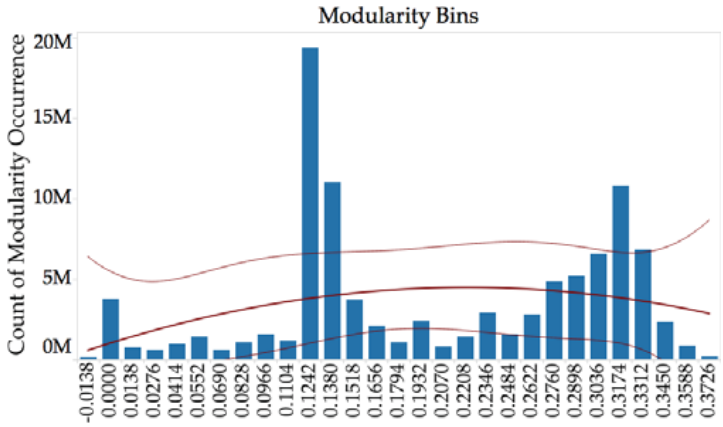
Equation:	Modularity = -2.13096e-07*Generation^3 + 2.51096e-05*Generation^2 + 0.00253695*Generation + 0.0567858
Model degrees of freedom:	4
R-Squared:	0.820795
p-value (significance):	< 0.0001

شکل 14. توصیف مدل فرآیند (30 میلیون مدل سه‌گانه)

تکنیک کدگذاری که ما معرفی کردیم فضای راه حل به همراه راه حل‌های بهینه محلی را ایجاد می‌کند. از این رو، با استفاده از نرخ جهش 100 درصدی و تقاطع چندین نقطه از سقوط راه حل‌های محلی بهینه جلوگیری می‌کنیم. با وجود اینکه نرخ جهش روی 100 درصد تنظیم شده بود، یال‌های درونی خوشه به عنوان یال‌های بین خوشه‌ای هیچگونه تفاوتی را در نوع مدولاسیون راه حل مشاهده نکرده بودند. از سوی دیگر، کدگذاری راه حل‌هایی را تحت تاثیر جهش کمتر از 100 درصد بودند را ارائه داد (72~ درصد از راه حل‌ها تحت تاثیر داده-های استفاده شده هستند). کدگذاری ارائه شده نسبت به کدگذاری سنتی کمتر تحت تاثیر جهش است. از این

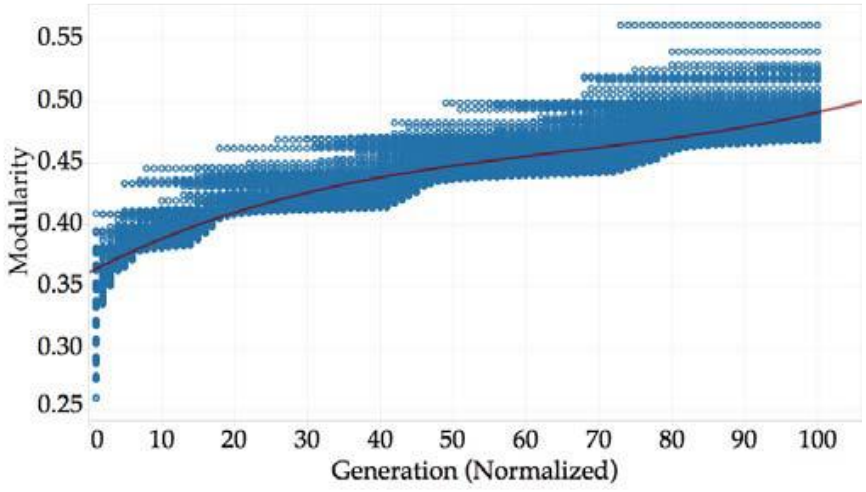
رو، نرخ جهش بالا برای تاثیر راه حل‌ها لازم و ضروری است.

شکل 15 مدولاسیون و تعداد آنها را در تمام نسل‌ها توصیف می‌کند (هر بین برابر محدوده‌ای از مقدار بین x تا مقدار بین x بعدی است). از آنجایی که بیشتر یال‌های درون خوشه تحت تاثیر تعدادی از جوامع نیستند، با این حال با داشتن چنین یال‌هایی راه حل‌ها تحت تاثیر شایستگی و تعدادی از توضیحات مدولاسیون بالا با غیر حداکثر ماژول نیز نیستند. بنابراین، راه حل‌های چهارگانه بر توانایی الگوریتم هیچ تاثیری نمی‌گذارند.



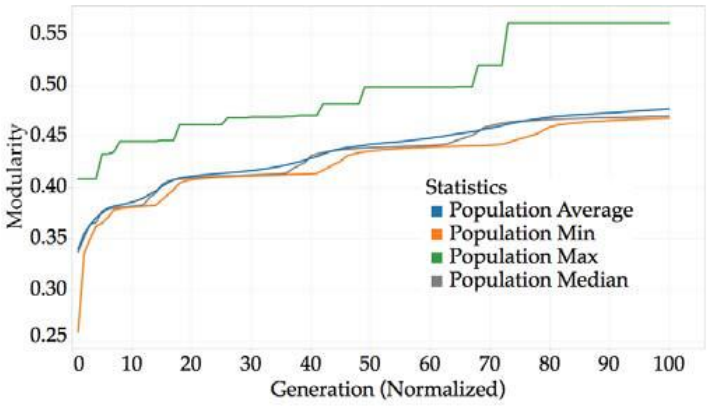
شکل 15. شمارش مدولاسیون در تمام نسل‌ها به عنوان مدولاسیون بین (30 میلیون LUBM سه‌گانه).

در مقیاس بالا، ما 221M فاصله اولیه را توسط LUBM سه‌گانه پوشش دادیم. شکل 16 توزیع و همبستگی بین اندازه شایستگی و فرکانس را در تمام نسل‌ها نشان می‌دهد.



شکل 16. همگرایی به منظور به حداکثر رساندن مدولاسیون بالای نسل‌ها (221 میلیون LUBM سه‌گانه).

شکل 17. حداقل، حداکثر، میانگین و میانی را برای هر مدولاسیون سازگاری نشان می‌دهد. نرخ جهش بالا فضایی را برای برخی از نسل‌های مختلف نگهداری می‌کند.



شکل 17. همگرایی به منظور به حداکثر رساندن مدولاسیون بالا نسل‌ها را نشان می‌دهد (221 میلیون LUBM سه‌گانه).

6.2 آزمایشات عملکرد سیستم

در این بخش، ما عملکرد سیستم را برخلاف ذخیره‌های RDF چندگانه که شامل SHARD بعد از خوشه‌بندی است اندازه‌گیری می‌کنیم. ما کلودرا ایمپالا را برای ایجاد جدول روی داده استفاده کردیم. ما عمدتا روی استفاده از زمان و منابع تمرکز می‌کنیم. همانطور که در جدول 2 دیده می‌شود تنظیمات سیستم مورد استفاده قرار گرفته‌اند. ما به چارچوب به عنوان پارتیشن‌بندی خوشه اشاره می‌کنیم.

در آزمایشات، ما یک مجموعه داده را با استفاده از LUBM تولید کردیم. اندازه مجموعه داده‌ها از 37 تا 142 گیگابایت در قالب Ntriples و از 200 تا 600 میلیون به صورت سه‌گانه است (جدول 5 را مشاهده کنید). جدول 6 هر پیچیدگی پرس‌وجو را به عنوان تعدادی از پیوست‌ها در هر پرس‌وجو را نشان می‌دهد.

6.2.1 زمان بارگذاری و خوشه‌بندی

هوانگ و همکارانش نتایج بارگذاری 270 میلیون RDF سه‌گانه در 20 ماشین خوشه HDFS را در جدول 4 نشان دادند. الکساندر و همکارانش [24]، 10 ماشین خوشه را در 40 دقیقه بارگذاری کردند. نتایج ما در جدول 5 نشان داده شده است.

به دلیل تفاوت در منابع، معیارهای پرس‌وجو در بخش مقایسه عملکرد پرس‌وجو، ما نتایج را نرمالسازی کردیم تا بتوانیم زمان پاسخ پرس‌وجو را مقایسه کنیم. به طور قابل توجهی یک کار در مورد پارتیشن RDF براساس تجزیه و تحلیل خوشه‌های داده‌ای وجود دارد. با این حال، تاثیر آن در بخش مقایسه عملکرد پرس‌وجو توضیح داده شده است، و در زمان پرس‌وجو در بخش بهینه‌سازی ذخیره شده است. این نکته بسیار مهم است زمانی که

مقادیر نسبت به حجم ذخیره‌سازی GB سه برابر بزرگتر می‌شود؛ عمل فشرده‌سازی می‌تواند اندازه مسئله ذخیره‌سازی را حل کند، اما نمی‌تواند میزان داده‌های مورد نیاز را پردازش کند. همچنین این نکته هم بسیار مهم است زمانی که نتایج خوشه‌بندی توسط معماری ارائه شده ذخیره می‌شود، داده‌ها و تغییرات جدید قابل دسترس هستند و نیازی به شروع مجدد ندارند. به عبارت دیگر، اضافه کردن تغییرات به داده‌ها، تا حدودی به زمان واقعی نزدیک است و با اضافه کردن داده‌های سه‌گانه جدید به سرور فیزیکی خوشه نیز به آن متصل می‌شود.

جدول 4 زمان بارگذاری 270 میلیون سیستم توسط هوانگ و همکارانش [1]

System	Load Time
RDF-3X	2.5 H
SHARD	6.5 H
Hash Partitioning	0.5 H
Graph Partitioning	4.2 H

جدول 5 زمان اجرای پارتیشن خوشه

Number of Triples	Size (GB)	Load Time (~)
221,278,374	37.1	3.8 Min Conversion 4.3 H clustering and placement
427,016,108	82.7	6.5 Min Conversion 7.2 H clustering and placement
613,190,853	142.3	8.1 Min Conversion 9.4 H clustering and placement

6.2.2 مقایسه عملکرد پرس‌وجو

برای مقایسه بقیه بررسی‌ها، ما ابتدا به درک پیچیدگی هر پرس‌وجو از 14 پرس‌وجو نیاز داریم. همانطور که در جدول 6 نشان داده شده است [1]، هر پرس‌وجو دارای تعدادی پیوست‌های subject-to-subject (S-S) و subject-to-object (S-O) است.

پارتیشن‌بندی عمل نگاشت درهم‌سازی یک کلید (Subject) را برای بلوک مشابه نشان می‌دهد. بنابراین، مسائل بهینه‌سازی در subject-to-subject کمتر از subject-to-subject رخ می‌دهد. پیوست‌های subject-to-object باعث می‌شود که بسیاری از مسائل بهینه‌سازی در پراکندگی داده‌های درهم‌سازی ایجاد شود، به این دلیل که مقادیر کلید مشابه (نقاط مربوط به داده) در همان بلوک‌های داده‌ای تایید نمی‌شوند.

جدول 6 پیچیدگی پرس‌وجو LUBM [1]

Query	S-S Joins	S-O Joins	Total
Q1	1	0	1
Q2	3	3	6
Q3	1	0	1
Q4	4	0	4
Q5	1	0	1
Q6	0	0	0
Q7	1	2	3
Q8	3	1	4
Q9	3	3	6
Q10	1	0	1
Q11	1	1	2
Q12	2	1	3
Q13	1	0	1
Q14	0	0	0

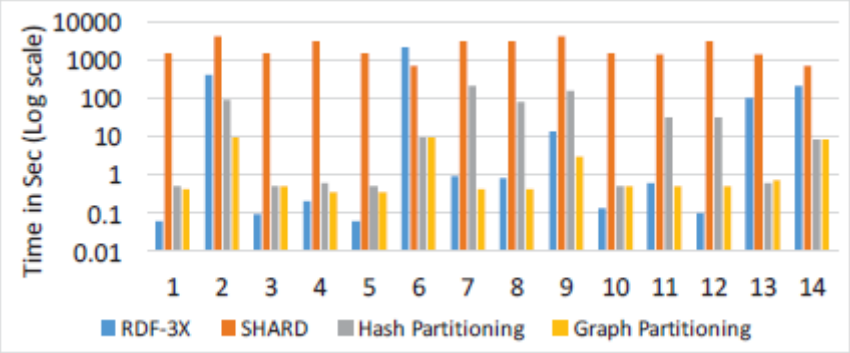
هوانگ و همکارانش [1] پرسشنامه‌ای را به دو گروه تقسیم کردند. پرس‌وجوهای 1، 3، 4، 5، 7، 8، 10، 11 و 12 به سرعت در RDF به صورت تک ماشین ذخیره می‌شدند و در RDF-3X اجرا می‌شدند؛ و پرس‌وجوهای 2، 6، 9، 13 و 14 در ذخیره‌های RDF کندتر می‌شدند. برای پرس‌وجوهای سریع، اندازه داده‌ها زیاد اهمیتی ندارند به این دلیل قبل از اینکه اسکن شوند، کاهش پیدا می‌کردند [1]. از این رو، از کار انداختن چنین ذخیره-هایی در چندین ماشین مزایای بسیاری ندارد و تنها تاخیر شبکه‌ای را برای شروع نشان می‌دهد [1]. از سوی دیگر، پرس‌وجوهای آهسته در ذخیره‌های RDF تک ماشین فاقد مقیاس‌پذیری هستند.

برخلاف چارچوب داده‌های آگاه HDFS، برخی از تلاش‌های قبلی برای پرس‌وجوهای آهسته به دست آمده است، به عنوان مثال پارتیشن‌بندی داده‌ها در SHARD. با این حال، عمل درهم‌سازی منجر به محدودیت‌های subject-to-subject به دلیل نیاز به حرکت داده‌های متوسط در شبکه می‌شود. مثال دیگری از هوانگ و همکارانش ارائه شده است و بدین صورت است اشیاء متصل به یک موضوع مورد پردازش قرار می‌گیرند تا به یک بلوک مشابه به منظور تقسیم بین subject و object تبدیل شوند. با ای حال، محدودیت‌های فضا به علت افزایش اندازه داده‌ها وجود خواهد داشت؛ همچنین محدودیت استفاده از چنین الگوریتمی در یک گراف متصل نیز وجود دارد [1]. ما از خوشه‌بندی در چارچوبمان استفاده کردیم و از جهش استفاده نکردیم. بنابراین، داده-های سه‌گانه نیازی به تکرار در بسیاری از موارد مختلف پراتشین‌ها ندارند، و هیچ افزایشی در اندازه داده‌ها رخ نخواهد داد (به غیر از تکرار قابلیت اطمینان HDFS از چیزی استفاده نخواهند کرد). سایر کارهایی مانند

سمپالا [24]، یا با استفاده از HIVE، PigSPARQL [25] & [26]، ادغام نگاشت [27] و MAPSIN [28] مقیاس‌پذیری را برطرف می‌کنند و از ذخیره‌های مختلف سه‌گانه نیز استفاده می‌کنند.

همانطور که در شکل 18 قابل مشاهده است هوانگ و همکارانش [1] چارچوب مختلف 270 میلیون LUBM سه‌گانه را پوشش دادند.

همانطور که هوانگ و همکارانش مشاهده کردند [1] نتایج تغییر داده‌ها در هر پارتیشن زیرگروهی شامل راس-هایی است که دو جهش هر subject را در subject سه‌گانه به طور موثر در زمان اجرا افزایش می‌دهد. سپس، به طور قابل توجهی افزایش پیدا می‌کند و همچنین یک گام از پردازش تکراری را پس از پایان پرس‌وجو اضافه می‌کند. هوانگ و همکارانش [1] آن را به عنوان یک توافق میان اندازه و زمان پاسخ سریع توضیح دادند.

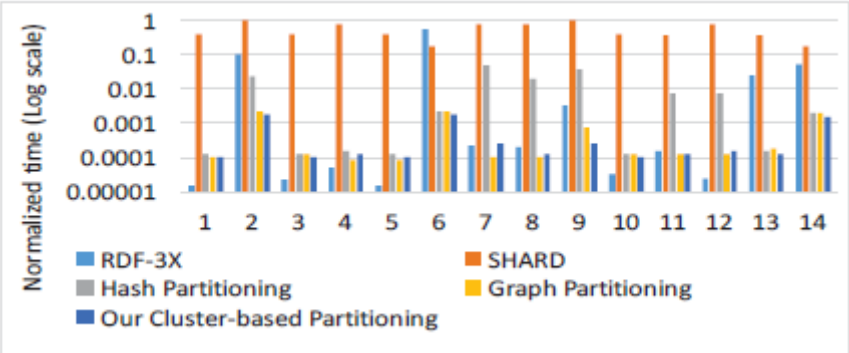


شکل 18. زمان پرس‌وجو هوانگ (270 میلیون LUBM سه‌گانه)

دو جهش تضمین می‌کنند که تکنیک بهینه‌سازی پرس‌وجوهای 2، 7، 8، 9، 11 و 12 فعال شده هستند. از سوی دیگر، بقیه پرس‌وجوها در مقایسه با داده‌های پارتشین‌بندی شده درهم به آهستگی متوقف می‌شوند، به این دلیل که یک گام مورد نیاز برای حذف تکرارهایی که توسط الگوریتم تکراری ایجاد شده‌اند نیز وجود دارد. تضمین جهش‌ها هیچ پرس‌وجویی را بهینه نمی‌کند. پرس‌وجوهای 1، 3، 4، 5، 6، 10، 13 و 14 در نمودار گرافیکی درهم‌سازی تقسیم‌بندی می‌شوند یا فقط پیوست‌های تکراری subject-to-subject را حذف می‌کنند. مقایسه پارتیشن‌بندی براساس خوشه‌بندی و SHARD به شما این امکان را می‌دهند که به طور مستقیم از مزایای تکنیک پارتیشن‌بندی پیشنهادی در سه مکان سه‌بعدی ساده HDFS استفاده کنید. برای مقایسه نتایج ما ابتدا تفاوت منابع HDFS را مورد بررسی قرار می‌دهیم. دو اکوسیستم مختلف هادوپ بسیار سخت هستند به این دلیل که ما ایده‌ای دقیق برای تنظیمات و یا آنچه که در ماشین در حال اجرا باشد را مورد پرس‌وجو قرار

دادیم. با این حال، یک راه حل این است که SHARD را در اکوسیستم هادوپ به صورت سه‌گانه انجام دهیم، سپس نسبت تفاوت را در آن پیدا کنیم. به عبارت دیگر، اگر پرس‌وجو Q1 در HDFS زمان X را در خوشه (A) و 2X را در خوشه (B) با همان HDFS به صورت بومی داشته باشد، خوشه (A) برای Q1 سریعتر عمل می‌کند. بنابراین، بهینه‌سازی Q1 با سرعت Y با استفاده از یک الگوریتم خاص Q1 در زمان X/Y در خوشه A و 2X/Y در خوشه B در الگوریتم خاص شروع می‌شود. به این نکته توجه داشته باشید که بهینه‌سازی یک الگوریتم خاص از لحاظ Y دارای عملکرد برابری است و نسبت به HDFS بومی از خوشه‌ای که صرف نظر شده است استفاده می‌کند. شکل 19 نتایج مقایسه‌ها (میانگین) را برای چارچوب HDFS آگاه (پارتیشن‌بندی مبتنی بر خوشه) پس از اجرای 20 پرس‌وجو نشان می‌دهد و میانگین واریانس آن نیز 0.06 می‌باشد.

نتایج افزایش عملکرد زمان اجرا برای پرس‌وجوهای 2، 3، 6، 9، 13 و 14 گزارش شده‌اند. عملکرد بسیار نزدیکی در پرس‌وجو 1 که تاثیر آن کمتر است در پرس‌وجو 4، 5، 7، 8، 11 و 12 رخ داده است. دلیل بهبود subject-to-subject گام اضافی است که تکرار در پراکندگی گراف را از بین می‌برد و توسط هوانگ و همکارانش مورد استفاده قرار گرفته است [1]. از سوی دیگر، افزایش عملکرد مربوط به اتصال subject می‌شود که باعث کمتر شدن مقادیر داده‌های اسکن شده می‌شود و توسط عمل پارتیشن‌بندی مبتنی بر خوشه منتقل می‌شود، به این دلیل که پارتیشن‌بندی مبتنی بر خوشه حضور اکثر اشیاء مرتبط با افراد را در یک پارتیشن‌بندی بدون تکرار تضمین می‌کند (اسکن کمتر). چنین نتایجی فضای ذخیره‌سازی را نسبت به بخش گرافیکی کمتر گزارش می‌کنند [1].

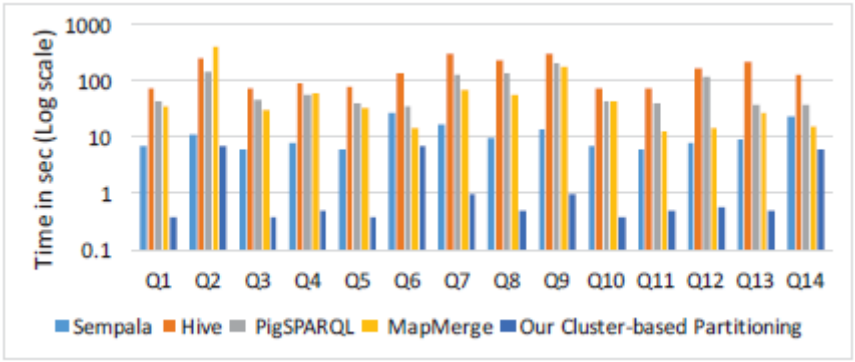


شکل 19. زمان اجرای پرس‌وجو، داده HDFS آگاه (270 میلیون LUBM سه‌گانه).

ما بیشتر ترافیک شبکه را در هر گره محاسباتی HDFS با استفاده از ابزارهای مانیتورینگ عملکرد لینوکس را

تحلیل کردیم. دستورات SAR برای نظارت بر آداپتورهای شبکه بر روی هر ماشین در طی پرس‌وجو استفاده می‌شوند. نتایج حاصل از گزارش هر گره HDFS به صورت یک فایل جمع‌آوری می‌شوند. تاثیر چارچوب ما در بهینه‌سازی ترافیک شبکه به منظور نشان دادن LUBM در 221 میلیون گراف سه‌گانه RDF به شرح زیر است: یک عامل بهینه‌سازی بزرگ در اغلب موارد به S-O متصل می‌شود، و باعث کاهش یک عامل 200X به منظور تقسیم‌بندی SHARD درهم Q7 می‌شود و سپس در یک خوشه سه‌گانه قرار می‌گیرد. علاوه بر این، 160X~ برای Q8 به منظور بهینه‌سازی Q9 در ترافیک شبکه گزارش شده است.

یکی از اهداف ما این است که با داده‌های پویا برخورد کنیم، و براساس چارچوب‌هایی مانند سمپالا، ادغام نگاشت، PigSPARQL و سمپالا HIVE، بدون در نظر گرفتن محدودیت‌های آنها در برخورد با به روزرسانی-های پویا، بتوانیم جستجو کنیم. شکل 20 نتایج را برای 221 میلیون رکورد RDF نشان می‌دهد.



شکل 20. زمان اجرای پرس‌وجو، داده HDFS آگاه (221 میلیون LUBM سه‌گانه).

7 نتیجه‌گیری

در این مقاله، ما داده‌های آگاه HDFS و سرویس‌های بالای HDFS را که به صورت بهینه‌سازی در state-of-the-srt در حال اجرا می‌باشد را ارائه کردیم. ما یک پارتیشن‌بندی داده‌ها را براساس خوشه به منظور جابجایی مکان فیزیکی داده‌ها و منطبق شدن با منطقه گراف و در فرآیندهای HDFS ارائه کردیم. این امکان را می‌دهد که پردازش موازیِ پرس‌وجوهای داده HDFS را برای منابع کمتر استفاده کنیم. چارچوب ما قادر بود برخی از تلاش‌ها را سریع‌تر انجام دهد و قادر بود به آرامی ذخیره‌سازی داده‌های RDF مقیاس‌پذیر را انجام دهد. با این حال، از منابع کمتر استفاده می‌کردند. بررسی‌هایی که در تجزیه و تحلیل نسل بعدی و معماری لامبدا [15]، [16]، [17] و [18] همراه با آپاچی کودو و مجموعه‌ای از بررسی‌ها در [21] ثابت شدند عملکرد سریع‌تری در

پردازش حجم کاری OLAP و عملکرد قوی در حجم کاری زمان دارند و بسیار هم مهم می‌باشند. با این حال، تلاش می‌کند که داده‌های هوشمند را در چنین روش‌هایی تحت تاثیر قرار دهد. برای کارهای آتی، ما قصد داریم برای بهبود بیشتر از کدگذاری توزیع و عملگرهای ژنتیک به منظور کاهش هزینه‌های محاسباتی پشتیبانی کنیم. همچنین ما قصد داریم آزمایشات را به صورت پویا به منظور سرعت بخشیدن بیشتر به جریان داده‌ها به روز رسانی کنیم و به منظور استفاده از ابزارها و چارچوب معماری لامبدا و تجزیه و تحلیل نسل بعدی که در بررسی-های اخیر ارائه شده آزمایش کنیم.

REFERENCES

[1] J. Huang, D. J. Abadi and K. Ren, "Scalable SPARQL querying of large RDF graphs," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 1123-113, 2011.

[2] K. Bajda-Pawlikowski, D. J. Abadi, A. Silberschatz and E. Paulson, "Efficient processing of data warehousing queries in a split execution environment," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011.

[3] M. Walker, "Data Science Central," 19 Dec 2012. [Online]. Available: http://www.datasciencecentral.com/profiles/blogs/structured-vs-unstructured-data-the-rise-of-data-anarchy. [Accessed 16 Oct 2015].

[4] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the future*, vol. 2007, no. 2012, pp. 1--16.

[5] K. Rohloff and R. E. Schantz, "Clause-iteration with MapReduce to scalably query datagraphs in the SHARD graph-store," in *Proceedings of the fourth international workshop on Data-intensive distributed computing*, 2011.

[6] T. A. S. Foundation, "Apache Spark," The Apache Software Foundation, [Online]. Available: http://spark.apache.org. [Accessed Jan 2016].

[7] T. A. S. Foundation, "Apache Mesos," The Apache Software Foundation., [Online]. Available: http://mesos.apache.org. [Accessed 19 Jan 2016].

[8] E. I. Inc., "HAMR - Faster than the speed of data," ET International, Inc., [Online]. Available: http://www.hamrtech.com/index.html. [Accessed 19 Jan 2016].

[9] Apache Storm, "Apache STORM," Apache Software Foundation , [Online]. Available: http://storm.apache.org. [Accessed 16 9 2016].

[10] L. Aniello, R. Baldoni and L. Querzoni, "Adaptive online scheduling in storm," in *Proceedings of the 7th ACM international conference on Distributed event-based systems*, 2013.

[11] P. Basanta-Val, N. Fernandez-Garcia, A. Wellings and N. Audsley, "Improving the predictability of distributed stream processors," *Future Generation Computer Systems*, vol. 52, pp. 22--36, 2015.

[12] M. Hajeer, D. Dasgupta, A. Semenov and J. Veijalainen, "Distributed evolutionary approach to data clustering and modeling," in *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium*, 2014.

[13] H. Song, P. Basanta-Val, A. Steed, M. Jo and Z. Lv, "Next-generation big data analytics: State of the art, challenges, and future research topics," *IEEE Transactions on Industrial Informatics*, p. In Press, 2017.

[14] N. Agnihotri and A. K. Sharma, "Proposed algorithms for effective real time stream analysis in big data," in *Image Information Processing (ICIIP), 2015 Third International Conference on*, 2015.

[15] L. Aniello, R. Baldoni and L. Querzoni, "Adaptive online scheduling in storm," in *Proceedings of the 7th ACM international conference on Distributed event-based systems*, 2013.

[16] P. Basanta-Val, N. Fernandez-Garcia, A. J. Wellings and N. C. Audsley, "Improving the predictability of distributed stream processors," *Future Generation Computer Systems*, vol. 52, pp. 22--36, 2015.

[17] P. Basanta-Val and M. Garcia-Valls, "A distributed real-time java-centric architecture for industrial systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 27--34, 2014.

the International Workshop on Semantic Web Information Management, 2011.

[27] M. Przyjaciel-Zablocki, A. Schatzle, E. Skaley, T. Hornung and G. Lausen, "Map-side merge joins for scalable SPARQL BGP processing," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, 2013.

[28] A. Schatzle, M. Przyjaciel-Zablocki, C. Dorner, T. Hornung and G. Lausen, "Cascading map-side joins over HBase for scalable join processing," *SSWS+ HPCSW*, p. 59, 2012.

[29] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, p. 1, 2008.

[30] R. Angles, "A comparison of current graph database models," in *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*, 2012.

[31] B. A. Eckman and P. G. Brown, "Graph data management for molecular and cell biology," *IBM journal of research and development*, vol. 50, no. 6, pp. 545-560, 2006.

[32] J. Hayes and C. Gutierrez, "Bipartite graphs as intermediate model for RDF," in *The Semantic Web--ISWC 2004*, Springer, 2004, pp. 47-61.

[33] A. Schenker, Graph-theoretic techniques for web content mining, World Scientific, 2005, p. 62.

[34] A. Nayak and I. Stojmenovic, Handbook of applied algorithms: Solving scientific, engineering, and practical problems, John Wiley & Sons, 2007.

[18] P. Basanta-Val, N. C. Audsley, A. J. a. G. I. Wellings and N. Fernandez-Garcia, "Architecting Time-Critical Big-Data Systems," *IEEE Transactions on Big Data*, vol. 2, no. 4, pp. 310--324, 2016.

[19] M. Congosto, P. Basanta-Val and L. Sanchez-Fernandez, "T-Hoarder: A framework to process Twitter data streams," *Journal of Network and Computer Applications*, vol. 83, pp. 28--39, 2017.

[20] T. A. S. Foundation, "Introducing Apache Kudu," The Apache Software Foundation, 2017. [Online]. Available: https://kudu.apache.org/docs/. [Accessed 5 April 2017].

[21] N. Marz and J. Warren, Big Data: Principles and best practices of scalable realtime data systems, Manning Publications Co., 2015.

[22] M. Ferron-Jones, "It Peer Network," 16 May 2017. [Online]. Available: https://itpeernetwork.intel.com/new-breakthrough-persistent-memory-first-public-demo/. [Accessed 1 June 2017].

[23] E. Roy Marsten, "Is graph theory key to Understand Big Data," March 2014. [Online]. Available: http://www.wired.com/insights/2014/03/graph-theory-key-understanding-big-data/. [Accessed Oct 2015].

[24] A. Schatzle, M. Przyjaciel-Zablocki, A. Neu and G. Lausen, "Sempala: Interactive SPARQL query processing on hadoop," *The Semantic Web--ISWC 2014*, pp. 164--179, 2014.

[25] A. Schatzle, M. Przyjaciel-Zablocki, T. Hornung and G. Lausen, "PigSPARQL: a SPARQL query processing baseline for big data," in *Proceedings of the 2013th International Conference on Posters Demonstrations Track-Volume 1035*, 2013.

[26] A. Schatzle, M. Przyjaciel-Zablocki and G. Lausen, "PigSPARQL: Mapping SPARQL to Pig Latin," in *Proceedings of*

[35] A. Silberschatz, H. F. Korth and S. Sudarshan, "Data models," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 105-108, 1996.

[36] A. Harth, J. Umbrich, A. Hogan and S. Decker, "Yars2: A federated repository for querying graph structured data from the web," Springer, 2007, pp. 211--224.

[37] Apache, "Apache Jena Elephas," Apache, [Online]. Available: https://jena.apache.org/documentation/hadoop/. [Accessed 10 Feb 2016].

[38] O. Erling and I. Mikhailov, "Towards web scale RDF," *Proc. SSWS*, 2008.

[39] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75-174, 2010.

[40] Goldberg, D. Edward and others, Genetic algorithms in search, optimization and machine learning, vol. 412, Addison-wesley Reading Menlo Park, 1989.

[41] M. Tasgin, A. Herdagdelen and H. Bingol, "Community detection in complex networks using genetic algorithms," *arXiv preprint arXiv:0711.0491*, 2007.

[42] C. Pizzuti, GA-Net: A genetic algorithm for community detection in social networks, Springer, 2008, pp. 1081-1090.

[43] C. Pizzuti, A multi-objective genetic algorithm for community detection in networks, IEEE, International Conference on Tools With Artificial Intelligence. ICTAI, 2009, pp. 379-386.

[44] M. Gong, L. Ma, Q. Zhang and L. Jiao, "Community detection in networks by using multiobjective evolutionary algorithm with decomposition," *Physica A: Statistical Mechanics and its Applications*, 2012.

[45] R. Shang, J. Bai, L. Jiao and C. Jin, "Community detection based on modularity and an improved genetic algorithm," *Physica A:*

Statistical Mechanics and its Applications, vol. 392, no. 5, pp. 1215-1231, 2013.

[46] J. Li and Y. Song, "Community detection in complex networks using extended compact genetic algorithm," *Soft Computing*, vol. 17, no. 6, pp. 925-937, 2013.

[47] M. Gong, X. Chen, L. Ma, Q. Zhang and L. Jiao, "Identification of multi-resolution network structures with multi-objective immune algorithm," *Applied Soft Computing*, vol. 13, no. 4, pp. 1705-1717, 2013.

[48] G. K. Kumar and V. K. Jayaraman, "Clustering of Complex Networks and Community Detection Using Group Search Optimization," *arXiv preprint arXiv:1307.1372*, 2013.

[49] C. Honghao, F. Zuren and R. Zhigang, Community detection using Ant Colony Optimization, IEEE CEC, 2013, pp. 3072-3078.

[50] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[51] U. Brandes, D. Delling, M. Gaertler, R. G{\"o}rke, M. Hoefer, Z. Nikoloski and D. Wagner, "On modularity clustering," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 2, pp. 172--188, 2008.

[52] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.

[53] A. Clauset, M. E. Newman and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[61] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760--771, 2011.

[62] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[63] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95-142, 2013.

[64] A. Thusoo, J. S. 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," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626-1629, 2009.

[65] M. Hajeer, D. Dasgupta, A. Semenov and J. Veijalainen, "Distributed evolutionary approach to data clustering and modeling," in *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*, Orlando, 2014.

[66] T. SOFTWARE, "Tableau," TABLEAU, [Online]. Available: https://www.tableau.com. [Accessed 1 Oct 2017].

[67] W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452-473, 1977.

[68] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron and L. Liberti, "Column generation algorithms for exact modularity maximization in networks," *Physical Review E*, vol. 82, p. 046112, 2010.

[54] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577--8582, 2006.

[55] Y. Zhang, J. Wang, Y. Wang and L. Zhou, "Parallel community detection on large networks with propinquity dynamics," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[56] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *ConferenceProceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821-7826, 2002.

[57] Wikipedia, "Modularity (Networks)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Modularity_(networks). [Accessed 10 March 2016].

[58] Y. Gu, S.-L. Shenq, Q. Wu and D. Dasgupta, "On a multi-objective evolutionary algorithm for optimizing end-to-end performance of scientific workflows in distributed environments," in *Proceedings of the 45th Annual Simulation Symposium*, 2012.

[59] M. H. Hajeer, D. Dasgupta and K.-I. Lin, "Distributed Evolutionary Algorithm for Clustering Multi-Characteristic Social Networks," in *Proceedings of the International Conference on Data Mining (DMIN)*, 2014.

[60] A. Semenov, J. Veijalainen, M. Hajeer and D. Dasgupta, "Political Communities in Russian Portion of LiveJournal," in *In the Proceedings of International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, USA, 2014.