دوفصلنامه محاسبات و سامانه‌های توزیع شده
www.jdcs.ir

توسعه یک الگوریتم جانمایی بلاک‌ها در سیستم هادوپ ناهمگن با توجه به مشخصات نودها و نوع کارها

قاسم الهی

هیئت علمی، بخش کامپیوتر، دانشکده فنی و حرفه‌ای شهید چمران، کرمان[1]

چکیده

پیشرفت در فن‌آوری شبکه‌های رایانه‌ای و منابع محاسباتی در مقیاس بزرگ منجر به ایجاد خوشه‌های محاسباتی[2] است که می‌توانند حجم زیادی از اطلاعات را در خود ذخیره و پردازش کنند. سیستم هادوپ، اجرا و تسهیم بسیاری از وظایف در یک مرکز داده مشترک را فراهم می‌کند. هادوپ[3] با فرض این مهم که تمام منابع پردازشی یکسان و وظایف داده‌های محلی دارند، سیاست خاصی در مورد انتخاب گره اعمال نمی‌کند. در صورتی که اگر گره‌ها ناهمگن باشند، متوسط زمان اجرا از یک گره به گره دیگر متفاوت خواهد شد؛ به ویژه وقتی که داده محلی نباشد. در این تحقیق سعی شده است، تا هنگام تقسیم فایل ورودی به بلاک‌های داده‌ای یکسان توسط هادوپ، بلاک‌ها براساس میانگین وزنی گره‌ها، توزیع شود. و همچنین با دسته بندی گره‌ها و انتقال یک رونوشت به هر یک از این دسته‌ها علاوه بر متوسط زمان اجرا و محلی بودن، قابلیت اعتماد و توازن فضای خالی ذخیره‌سازی خوشه را در سیستم افزایش داد.

کلمات کلیدی: پردازش داده، نگاشت کاهش، خوشه ناهمگن، هادوپ، جانمایی بلاک

تاریخچه مقاله:

تاریخ ارسال:۹۷/۳/۱۵

تاریخ اصلاحات:۹۷/۵/۲

تاریخ پذیرش:۹۷/۵/۵

تاریخ انتشار:**۹۷/۵/۱۵**

Keywords:

Data processing
MapReduce
Heterogeneous cluster
Hadoop
Block placement

Developing a blocks placement algorithm according to the specifications of node and jobs types in heterogeneous Hadoop Systems

Ghasem Elahi

Faculty member, shahid Chamran technical and vocational university, Kerman,

Abstract

Advances in network computer technology and large-scale computing resources have led to computingclusters that can store and process large volumes of data. Apache Hadoop systems provide the possibility of running and sharing many common tasks in the data center. Assuming that all local data processing resources andtasks are the same, Hadoop doesn't adopt a particular policy regarding the selection of nodes. If the nodes are heterogeneous, average run time will vary from one node to another, especially if the data are not local. Inthis study, we tried to distribute blocks according to the average weight of nodes when an input file was divided into blocks ofsame size by Hadoop. Moreover, by putting nodes into categories and delivering a copy to each of these categories, we increased the reliability and balance of the gaps in the cluster's storage system as wellas the locality and average execution time.

روش ارجاع به مقاله: ق.الهی، توسعه یک الگوریتم جانمایی بلاک‌ها در سیستم هادوپ ناهمگن با توجه به مشخصات نودها و نوع کارها، دوفصلنامه محاسبات و سامانه‌های توزیع شده، سال اول، شماره اول، ص ۹-۲۰، سال انتشار ۱۳۹۷.

[1] Corresponding author : نویسنده ی عهده دار مکاتبات

[2] Computing clusters

[3] Hadoop

۱ – مقدمه

در سال‌های اخیر رشد بسیار زیاد داده‌ها، ذخیره، تجزیه و تحلیل، استخراج اطلاعات معنی‌دار از آن را به یک چالش تبدیل کرده است. با در نظر گرفتن این مهم که مقدار داده تولید و ذخیره شده توسط شرکت‌های اینترنتی، شرکت‌ها و دولت‌ها رو به افزایش است، و با توجه به پیش‌بینی‌ها در سال‌های آینده، بایستی رشد افزایش آن به صورت‌نمایی حفظ شود.

سیستم‌های ذخیره سازی سنتی مانند پایگاه داده‌های رابطه‌ای و سیستم‌های فایل، به اندازه کافی مقیاس‌پذیر برای مدیریت مجموعه داده رو به افزایش نیستند. این افزایش به نوع جدیدی از سیستم‌ها نیاز دارد که علاوه بر این که بتواند با قابلیت اعتماد مناسب، حجم انبوهی از داده‌ها را روی بیش از یک خوشه ذخیره کنند، بایستی روشی برای تجزیه و تحلیل داده‌های به صورت موازی نیز ارائه کنند.

امروزه یکی از سیستم‌هایی که علاقه‌مندان زیادی در دولت‌ها، شرکت‌های بزرگ و موسسات تحقیقاتی پیدا کرده است، هادوپ است. هادوپ؛ سیستم متن باز و شناخته شده‌ای است که توسط شرکت آپاچی[۴] برای ذخیره سازی و پردازش داده‌های بزرگ در سال ۲۰۰۶ م. (۱۳۸۵ ش.) ارائه شد.

طراح این سیستم شخصی به نامDoug Cutting ، نام هادوپ را بر اساس اسم فیل عروسکی پسرش انتخاب کرد. هادوپ شامل دو بخش اصلی است: نخست سیستم فایل توزیع شده به عنوان سیستم فایل توزیع شده هادوپ، یا HDFS که با اطمینان بالا، پهنای باند زیاد و هزینه کم چندین پتا بایت (برابر با 10^{15} بایت) داده را در فایل‌هایی روی ماشین‌های خوشه با بیش از هزار سرور به آسانی مدیریت کند. بخش دیگر، چارچوب پردازش توزیع شده به نام نگاشت کاهش (MapReduce) است، که الگوریتم‌های نگاشت کاهش را به صورت موازی و توزیع شده با عملکرد بالا پردازش می‌کند.

الهام بخش معماری HDFS ، سیستم فایل گوگل یا GFSبوده و نگاشت کاهش براساس مقاله‌ای تحت عنوان «نگاشت کاهش: پردازش ساده داده‌ها در خوشه های بزرگ» که در سال ۲۰۰۴ م. توسط آقایان جفری دین و سانجی گماوات[۵] از شرکت گوگل ارائه شد[۱]، طراحی شده است .

از آنجایی که اجرای سریع یک کار در هادوپ، متأثر از زمان‌بندی کارها و محلی بودن داده‌ها است؛ در این مقاله سعی بر آن است روشی برای محلی کردن داده‌ها ارائه گردد. با این توضیح که هادوپ برای ذخیره یک فایل، آن را به بلاک‌هایی (پیش‌فرض ۶۴ مگابایتی) تقسیم و با فاکتور تکرار سه بر روی گره‌های خوشه توزیع و ذخیره می‌کند. آن گاه فرض می‌کند تمام منابع پردازشی یکسان هستند و وظایف داده‌های محلی خود را دارند، به همین دلیل سیاست خاصی در مورد انتخاب گره اعمال نمی‌کند. در صورتی که اگر گره‌ها نا همگن باشند، متوسط زمان اجرا از یک گره به گره دیگر متفاوت خواهد بود، به ویژه وقتی که داده محلی نباشد.

حال در این تحقیق روشی به کار برده می‌شود تا بلاک‌های داده‌ای بر اساس میانگین وزنی گره‌ها و نوع

[4] Apache Software Foundation

[5] Jeffrey Dean and Sanjay Ghemawat

کارها توزیع شود؛ بدان هدف که متوسط زمان اجرا، محلی بودن و توازن بار، را در سیستم افزایش داد.

۲ -تعریف BigData

ارزش داده بزرگ در توانایی ما برای استخراج بینش و تصمیم گیری بهتر نهفته است. [۲]

۲-۱- BigData چیست و چرا اهمیت دارد؟

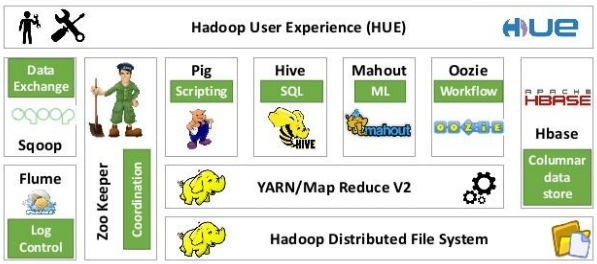
از نظر لغوی «BigData» به معنای «داده بزرگ»، و اصطلاحی است که به مجموعه‌ای از داده‌هایی اطلاق می‌شود که مدیریت، کنترل و پردازش آن‌ها فراتر از توانایی ابزارهای نرم‌افزاری در یک زمان قابل تحمل و مورد انتظار است. در تعریفی دیگر می‌توان آن را اصطلاحی نامید که رشد و در دسترس بودن داده، چه ساختارمند و چه غیرساختارمند، را توصیف می‌کند. به عبارت دیگر در داده بزرگ به طور عموم با داده‌هایی سر و کار داریم که حجم آن‌ها بیشتر از ظرفیت نرم افزارهای مدیریت پایگاه داده موجود و یا برنامه‌های کاربردی سنتی پردازش داده است. بنابراین، فرآیند تحلیل، فرآیندی مشکل و نیازمند نرم‌افزارهایی است که به صورت موازی در حال کار بر روی ده‌ها، صدها و یا حتی هزاران سرور باشند. چالش‌های مهم نیز شامل: استخراج، گزینش داده، ذخیره‌سازی، جستجو، اشتراک، انتقال، آنالیز و بصری‌سازی است.

۳ - هادوپ

هادوپ یک چارچوب، نرم‌افزار متن باز، برنامه نویسی شده بر پایه جاوا است که برای پردازش مجموعه داده‌های بزرگ، توزیع شده در محیط توزیع شده محاسباتی توسط بنیاد نرم‌افزاری آپاچی طراحی شده است.

۳-۱ -پشته تکنولوژی هادوپ

هسته هادوپ از دو بخش «HDFS» سیستم فایل توزیع شده برای ذخیره مجموعه داده‌های بزرگ، در خوشه‌ای از یک کامپیوتر تا هزاران کامپیوتر، و «MapReduce» یک مدل برنامه‌نویسی توزیع شده برای محاسبات توزیع شده سریع روی مقدار زیادی داده تشکیل شده است. در ورژن جدید هادوپ، یک بخش جدید به هسته سیستم هادوپ به نام «YRAN» ـ جهت مدیریت زمان بندی توزیع شده ـ اضافه شده است. فایل سیستم «HDFS» محدود به نگاشت کاهش نمی‌شود و بسته به نوع استفاده از هادوپ سیستم‌های دیگری در کنار هادوپ طراحی شده است که در مجموع پشته تکنولوژی هادوپ را تشکیل می‌دهند (شکل ۱).



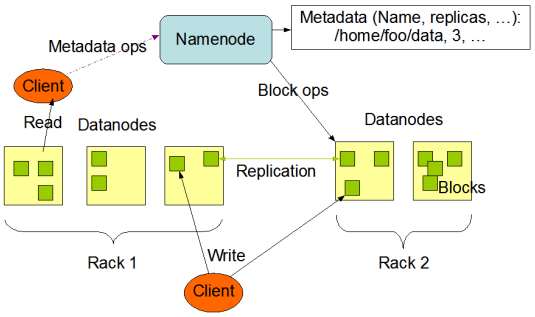
(شکل ۱): پشته تکنولوژی هادوپ [۳]

(Figure-1): Hadoop technology stack [1]

۳-۲ -معماری HDSF

وقتی که حجم یک مجموعه داده از ظرفیت ذخیره سازی یک ماشین فیزیکی بیشتر باشد، لازم است آن روی چندین ماشین مجزا تقسیم شود. فایل سیستمی که ذخیره سازی روی ماشین‌های یک شبکه را مدیریت می‌کند، سیستم فایل توزیع شده نامیده می‌شود. «HDFS» سیستم فایل توزیع شده هادوپ است و برای ذخیره سازی فایل‌های خیلی بزرگ بر اساس الگوی دسترسی جریانی به داده‌ها روی خوشه‌ای از

ماشین‌های مناسب و معمولی طراحی شده است. معماری HDFS براساس Master/Slave است، که در آن Master گره اصلی یا ارباب، به اسم گره نام (NameNode) و تعدادی گره Salve یا کارگر، که گره داده (DataNode) نامیده می‌شوند. گره نام مسئولیت نگه‌داری سیستم فایل و مدیریت گره‌های داده را به عهده دارد. از آنجایی که اگر گره نام در حال اجرا خراب شود، تمامی فایل‌ها و بلاک‌ها غیرقابل دسترس می‌شوند و هیچ راهی برای بازیابی آن‌ها از گره‌های کارگر نمی‌باشد، به همین دلیل یک گره دیگر به نام گره نام ثانویه (Secondary Namenode) وجود دارد،



شکل ۲ : معماری HDFS [۴]

(Figure-2): HDFS Architecture [4]

که مسئولیت نگه‌داری یک کپی از سیستم فایل توزیع شده را به عهده دارد. شکل (۲) معماری HDFS را نمایش می دهد.

۳-۳-سیاست پیش فرض جای گذاری داده (بلاک‌ها)

در هادوپ عملکرد برنامه‌های نگاشت کاهش تا حد زیادی وابسته به این است که چگونه داده‌ها در سیستم فایل توزیع شده هادوپ، HDFSذخیره می‌شوند. در حالت پیش فرض به این شکل است که هادوپ یک مجموعه داده با حجم زیاد را به بخش‌های مساوی ۶۴ یا ۱۲۸ مگابایتی تقسیم و این بخش‌های را روی صدها سرور یک خوشه توسط سیستم فایل توزیع شده HDFSخود ذخیره نموده و علاوه بر آن برای بالا بردن ضریب اطمینان و در دسترس بودن داده‌ها، هر کدام از بخش‌ها را روی سه ماشین ذخیره می‌کند. این روش ضمن آن که باعث می‌شود نیازی به پشتیبان‌گیری معمول مانند روش RAID از داده‌ها نباشد، میزان دسترس بودن داده‌ها را نیز افزایش می‌دهد. روش انتخاب سه ماشین برای ذخیره سه نسخه از بلاک بدین صورت است[۷-۸].

هنگام ذخیره اولین نسخه بلاک HDFS بررسی می‌کند اگر گره‌ای که می‌خواهد فایلی را در خوشه ذخیره کند، خود عضوی از خوشه هادوپ باشد، بلاک در همان گره ذخیره می‌شود. در غیر این صورت گره‌ای از خوشه به صورت تصادفی انتخاب می‌شود.

- ذخیره دومین نسخه از بلاک در گره‌ای از رک دیگری که به صورت تصادفی خارج از رک محلی گره اول انتخاب می‌شود، صورت می‌گیرد.
- برای ذخیره سومین نسخه یک گره دیگر در رک نسخه دوم انتخاب می‌شود.
- در هر گره فقط یک کپی از هر بلاک باید ذخیره شود.

هادوپ یک ابزار برای متعادل کردن فضای خوشه و گره‌ها دارد که بعد از اجرای آن هنگام اضافه کردن یک گره جدید یا حذف بلاک، فضای هر گره را بر اساس فضای خالی کل خوشه متوازن می‌کند[۵, ۶].

به گفته بسیاری از محققان این سیاست جای‌گذاری بلاک‌های داده در شرایطی دارای کارایی، به ویژه در

خوشه ناهمگن مناسب نسیت. جیونگ زی در [۹] آورده است: «در یک خوشه ناهمگن که هر گره دارای یک دیسک محلی می‌باشد، برای افزایش کارایی اجرای نگاشت کاهش بهتر این است فرآیند نگاشت یا کاهش توسط گره نام به گره‌ای واگذار شود که در آن داده‌ها محلی هستند. در صورتی که داده‌ها محلی نباشند، داده‌ها باید از طریق ارتباطات بین شبکه‌ای به گره پردازش کننده انتقال یابند. انتقال حجم زیادی از داده‌ها باعث به وجود آمدن ازدحام و ترافیک زیاد در شبکه و پایین آمدن کارآیی سیستم خواهد شد. در خوشه‌های ناهمگن، ظرفیت پردازشی هر گره متفاوت است. یک گره با سرعت پردازشی بالا تسک‌های بیشتری را روی داده‌های محلی خود نسبت به همتایش که سرعت پردازش کم‌تری را دارد به اتمام می‌رساند. بعد از این که گره سریع‌تر داده‌های محلی خود را پردازش کرد، برای پشیبانی از بار اشتراکی باید داده‌های پردازش نشده گره‌های کندتر را به دیسک محلی خود انتقال داده و پردازش نماید. در صورتی که حجم داده‌های انتقالی به دلیل بار اشتراکی زیاد شود، بار اضافه ناشی از انتقال داده‌های پردازش نشده از گره‌های کندتر به گره‌های سریع‌تر خود مشکلی برای کارآیی سیستم هادوپ به وجود می‌آورد».

چیاوی لی در [۱۰] آورده است: «در معماری هادوپ، محلی بودن داده‌ها یکی از عوامل مهم و مؤثر بر عملکرد هادوپ است. در یک محیط ناهمگن، داده‌های مورد نیاز برای اجرای وظایف اغلب غیر محلی هستند که این بر کارایی هادوپ اثر می‌گذارد. در محیط پیش فرض هادوپ، گره‌های یک خوشه همگن همه دارای قابلیت پردازش و ظرفیت هارد دیسک یکسان هستند و بلوک‌های داده برای تعادل بار در هادوپ به طور یکسان توزیع بین گره‌ها می‌شوند. زیرا انتقال داده از یک گره به گره دیگر غیر ضروری است. اما در یک محیط ناهمگن که گره‌ها ظرفیت اجرا و هارد دیسک یکسان ندارد، اعمال استراتژی پیش‌فرض هادوپ باعث می‌شود، گره‌ای که ظرفیت اجرای بالاتری دارد وظیفه را سریع‌تر از گره‌ای که ظرفیت اجرای پایین‌تر دارد به اتمام رساند، و بعد از آن که داده‌های محلی پردازش شدند، شروع به پردازش داده‌های غیر محلی کند. از آنجا که ممکن است بلوک داده‌ی پردازش نشده در گره‌ای است با ظرفیت پردازش کم‌تر باشد، یک سربار اضافی برای انتقال بلوک داده از گره کند به گره سریع‌تر به وجود می‌آید؛ بنابراین عملکرد سیستم هادوپ پائین می‌آید».

۳-۴ -کارهای انجام شده

Xie و همکارانش [۹] در مقاله‌ای تحت عنوان «بهبود عملکرد نگاشت کاهش از طریق قراردادن داده در خوشه ناهمگن هادوپ» روشی برای جانمایی بلاک‌ها ارائه داده‌اند. آن‌ها معتقدند: «برای افزایش کارآیی سیستم هادوپ در خوشه ناهمگن، باید انتقال داده از گره کندتر به گره سریع‌تر را به حداقل رساند. به نظر می‌رسد بتوان با طرح توزیع داده‌ها در سراسر گره‌های یک خوشه ناهمگن بر مبنای نرخ محاسباتی، به هدف رسید و حرکت داده‌ها را کاهش داد؛ اگر تعداد قطعات فایل تقسیم شده، روی دیسک یک گره بر اساس ظرفیت محاسباتی آن گره قرار دهیم».

در روش مطرح شده مذکور، دو الگوریتم پیاده سازی و با هم روی HDFS سیستم هادوپ اعمال می‌شود. اولین الگوریتم قطعات فایل را روی خوشه هادوپ توزیع می‌کند. دومین الگوریتم سازماندهی مجدد قطعات فایل

برای حل مشکل انحراف داده‌ها ناشی از اولاً اضافه شدن یک گره محاسباتی جدید به خوشه، دوماً اضافه شدن داده جدید به یک فایل که از قبل موجود بوده است. الگوریتم اول فایل ورودی را به تعدادی قطعه یک اندازه تقسیم می‌کند، سپس قطعات را بین گره‌های محاسباتی خوشه بر اساس سرعت پردازش توزیع می‌کند. در مقاسیه گره با کارآیی پایین، گره با کارآیی بالاتر، تعداد بیشری از قطعات فایل ورودی در آن ذخیره می‌شود. به طوری که همه گره‌های در زمان یکسان کار پردازش داده‌های محلی خود را به انجام رسانند.

مقاله دیگری با عنوان «استراتژی جای‌گذاری پویا داده‌ها در هادوپ از محیط ناهمگن» نوشته لی و همکارانش ارائه شده است[۱۰]. در روش آن‌ها، بلاک‌های داده را با توجه قدرت اجرای هر گره اختصاص داده می‌شود. در خوشه ناهمگن ظرفیت محاسباتی گره‌ها یکسان نیست. به علاوه برای هر نوع کار هم، نرخ ظرفیت محاسباتی گره‌ها هم متفاوت است. بنابراین جای‌گذاری پویا داده‌ها با در نظر گرفتن نوع کارها جای‌گذاری بلاک‌ها را تنظیم و توزیع می‌کند .

استراتژی آن‌ها در دو فاز اصلی طراحی شده است: فاز اول هنگامی اجرا می‌شود که می‌خواهد داده‌ای در HDFS نوشته شود. فاز دوم وقتی که کار پردازش می‌شود. در فاز اول هنگامی که هادوپ اجرا می‌شود، یک جدول به نام جدول نرخ در گره نام ایجاد می‌شود و از آن برای تعیین نسبت بلاک‌های اختصاصی به گره، موقع نوشتن بلاک‌های داده در HDFS استفاده می‌شود. در این جدول نوع کار و نسبت ظرفیت محاسباتی هر گره ثبت می‌شود. ظرفیت محاسباتی بر پایه میانگین مدت زمان اجرای یک وظیفه ساده در یک گره محاسبه می‌شود. در این فاز هنگام نوشتن داده در HDFS، گره نام اول جدول نسبت را چک، و تعیین می‌کند چه نوع کاری از این داده‌ها استفاده می‌کند. در صورتی که در این جدول اطلاعاتی از این نوع کار باشد، داده‌ها بر اساس ظرفیت محاسباتی مشخص شده هر گره داده‌ها را توزیع می‌کند. در غیر این صورت، داده‌ها به نسبت مساوی بین گره‌ها خوشه توزیع و یک رکورد در مورد کار جدید با ظرفیت محاسباتی ۱ برای هر گره به جدول اضافه می‌شود .

فاز دوم از زمانی که یک کار برای اجرا شروع می‌شود، آغاز و به محض شروع کار، هر گره اولین دسته وظایف خود را دریافت می‌کند. وقتی که اجرای وظیفه در گره داده خاتمه یافت، هر گره داده مدت زمان اجرا را به گره نام اعلام می‌کند. گره نام مدت زمان اجرا ارسالی را با نسبت ظرفیت محاسباتی هر گره در جدول نسبت مقایسه می‌کند، اگر برابر بود دیگر بلاکی ارسال نمی‌شود. ولی در صورتی که متفاوت بود، بلاک‌ها با توجه به نسبت محاسباتی جدید که توسط گره نام محاسبه شده، ارسال و در پایان اجرای کار جدول نسبت اصلاح می‌شود.

سومین مقاله مورد بررسی که در خصوص جای‌گذاری بلاک‌های داده بحث می‌کند، با عنوان «قراردادن پویای دادها برای هادوپ با استفاده از الگوی زمان واقعی دسترسی» نوشته پونتوتم و کمار[۶] است[۸].

آن‌ها بیان کردند، طرح هادوپ پیش فرض هادوپ برای انتخاب گره داده و ذخیره نسخه‌های بلاک داده بدون در نظر گرفتن وضعیت زمان واقعی، کمبودهایی دارد.

[6] Viju P Poonthottam & Madhu Kumar

طرح آنان روی انتخاب گره‌هایی که به کاربر نزدیک‌تر باشد بر اساس مسئله مکان یابی امکانات ، مقاله [۱۱] با این پیش فرض که فقط ۲ نسخه از بلاک داده به طور جداگانه روی ۲ تا گره داده قرار گیرد، پیاده سازی کردند. در این طرح برای پیدا کردن گره‌ای که نزدیک‌ترین فاصله به کاربر داشته باشد، از گراف G(V,E) که مشخص کننده ترافیک شبکه است استفاده شده است. گراف G به وسیله ماتریس مسافت D=d(x,y) که در آن x,y∈V و d زمان واقعی دسترسی از گره x به گره y است نمایش داده می‌شود. در این ماتریس d با استفاده از دستور ping این محاسبه می‌شود و در ماتریس مسافت D قرار می‌گیرد. البته مسئله کلاسیک مکان یابی امکانات[۷] می‌تواند به یک مسئله بهینه سازی روی ماتریس مسافت تبدیل شود. و راه حل آن، تعریف راس مرکزی C(x) است، بدین صورت:

$$C(x) = \{x \in V | e(x) = \rho(G)\}$$
$$\rho(x) = \max d(x,y), y \in V \qquad (1)$$
$$\rho(G) = \min(e(x), x \in V)$$

دو ایراد مهم به روش آنان وارد است؛ نخست این که تعداد نسخه‌های بلاک را به ۲ محدود کرده‌اند، و این امر قابلیت دسترس‌پذیری را کاهش می‌دهد.

۴- الگوریتم جانمایی بلاک‌ها (پیشنهادی)

با توجه به مطالب ارائه شده، مشاهده می‌شود، سیستم هادوپ، یک خوشه محاسباتی است. و هدف اساسی استفاده از قدرت پردازش سیستم‌های موجود و منابع ذخیره سازی در آن برای حل مسائل پیچیده و زمان‌بر است. لذا برای رسیدن به این هدف و استفاده از حداکثر منابع موجود در خوشه، نحوه توزیع بلاک داده‌ها و چگونگی زمان‌بندی وظایف از اهمیت ویژه‌ای برخوردار است. نحوه توزیع بلاک‌ها برای زمان‌بندی بهتر و افزایش قابلیت اطمینان و در دسترس‌پذیری اهمیت دارد.

۴-۱ -سیاست جدید جانمایی بلاک‌ها

همان طور که گفته شد، سیاست پیش‌فرض هادوپ برای جانمایی بلاک‌های داده، قرار دادن یک نسخه از بلاک در رک محلی و دو نسخه در رک دیگر است.هدف این تحقیق، تغییر این سیاست نیست، بلکه مد نظر، مدیریت کردن انتخاب گره برای ذخیره سازی بلاک است. با این توضیح، چون در سیستم هادوپ، فرض می‌شود محیط همگن است، گره‌ها به صورت تصادفی برای ذخیره بلاک‌ها انتخاب می‌شوند، در راستا این پژوهش قصد داریم، در دو بخش روشی ارائه دهیم تا اولاً تعداد رونوشت‌ها محدود نشود و انتخاب گره‌ها را براساس ظرفیت پردازش و تعداد رونوشت بلاک‌ها هدفمند نمائیم. ثانیاً بر اساس نوع کارها معیار ارزیابی دیگری ارائه خواهیم کرد.

۴-۲- پیش فرض‌ها

- محیط ناهمگن

در این تحقیق فرض بر این است که خوشه ناهمگن است، یعنی گره‌ها از نظر ظرفیت پردازش و حجم ذخیره سازی متفاوت هستند. در صورتی که خوشه همگن باشد، تفاوتی ندارد که بلاک داده روی کدام گره ذخیره شود. فقط باید تعداد بلاک‌های ذخیره شده روی گره‌های مختلف متناسب باشد تا برنامه‌های نگاشت و همچنین کاهش در یک زمان خاتمه یابند. دلیل دیگر

[7] Facility location problem

انتخاب این پیش فرض این است که در کشور ما شرکت یا مراکز تحقیقاتی که بتوانند خوشه‌ای با چند صد گره از یک نوع ماشین ایجاد کنند وجود ندارد و این تحقیق کمک می‌کند تا بتوان کارایی خوشه‌های ناهمگن را افزایش داد.

- تعداد نسخه‌های بلاک

در مقاله [۹] تعداد نسخه‌های بلاک را از عدد پیش فرض ۳ به ۲ کاهش داده و در مقاله [۱۰] هم هنگام پیاده‌سازی الگوریتم، روشی برای مدیریت کردن محل ذخیره‌سازی دیگر نسخه‌های ارائه نداده است. در محیط‌های ناهمگن تعداد نسخه‌ها بیشتر اهمیت دارد، برای مثال اگر در یک سازمان از سیستم‌های کارمندان برای انجام پردازش‌های داده‌های بزرگ استفاده شود و تعداد نسخه‌ها را به ۱ تغییر دهیم، خاموش بودن حتی یک سیستم به هر دلیلی مانع از انجام اجرای برنامه نگاشت کاهش خواهد شد. اما در این تحقیق با اذعان به این امر که ما قصد تغییر یا ثابت نگه داشتن عدد پیش فرض ۳ برای تعداد نسخه‌های بلاک را نداریم و مسئولیت تغییر را بر عهده مدیر سیستم می‌گذاریم، تا بر اساس محیط عملیاتی و نوع سیستم‌های موجود در آن نسبت به تغییر آن تصمیم بگیرد. ولی باید بدانیم اگر تعداد نسخه‌ها از عدد پیش فرض ۳ کم‌تر شود، قابلیت اعتماد و دسترس پذیری را کاهش می‌یابد. از طرف دیگر اگر افزایش یابد، فضای ذخیره سازی بیشتری برای ذخیره سازی نسخه‌های چهارم به بعد نیاز است.

۴-۳- پیاده سازی

۴-۳-۱- بخش اول

مدیریت هدفمند انتخاب گره هنگام ذخیره سازی بلاک داده، افزایش لوکالیتی، کاهش ترافیک شبکه ناشی از انتقال بلاک‌ها به گره‌های سریع‌تر را باعث خواهد شد، و در نهایت زمان اجرای برنامه‌های نگاشت کاهش را پائین می‌آورد. برای انجام آن ظرفیت محاسباتی، هر گره را محاسبه نموده و متناسب با ظرفیت محاسباتی هر گره، بلاک‌ها را به گره‌ها انتقال می‌دهیم. با این دید که، گره‌هایی که ظرفیت محاسباتی بیشتری دارند سهم بیشتری از بخش‌های یک فایل ورودی را دریافت کنند.

برای بدست آوردن ظرفیت پردازشی هر گره، یک کار مشخص مثلاً نگاشت کاهش «جستجو» را با یک بلاک داده یکسان به هر گره در خوشه انتقال می‌دهیم و زمان اجرای آن را به عنوان ظرفیت پردازشی گره در نظر می‌گیریم. البته بهتر است این برنامه نگاشت کاهش شامل الگوریتمی باشد که به نحوی مناسب بخش‌های مختلف گره مانند: حافظه، پردازنده و دیسک را مورد آزمایش قرار دهد.

برای پیاده سازی این روش، یک فایل به نام NodeState در گره نام سیستم هادوپ ایجاد کرده و در آن اطلاعاتی شامل نام گره و ظرفیت محاسباتی آن را ذخیره می‌کنیم.

(Table-1): Sample NodeState file

جدول ۱: نمونه فایل NodeState

Node#	Rate Compute	CPU-(Core)	RAM-(GB)
DN1	۳/۵	۴	۶
DN2	۲	۲	۳
DN3	۴	۷	۸
DN4	۱	۱	۱

برای به روز آوری اطلاعات فایل NodeState (جدول ۱) وقتی که گره به خوشه محاسباتی اضافه شد، یک بلاک با برنامه نگاشت کاهش مشخص برای اجرا به آن

گره ارسال می‌شود و سپس مدت زمان اجرای کار را به عنوان ظرفیت محاسباتی آن گره ذخیره می‌نماییم. در صورت جدا شدن یک گره از خوشه اطلاعات آن را از جدول فوق حذف می‌کنیم تا محاسبه طرفیت پردازش کل خوشه دقیق‌تر محاسبه شود. حال با استفاده از الگوریتم خوشه‌بندی k-means ، که در آن مقدار k برابر تعداد نسخه‌ها بلاک است، گره‌ها را بر اساس ظرفیت محاسباتی آن‌ها دسته بندی می‌کنیم .

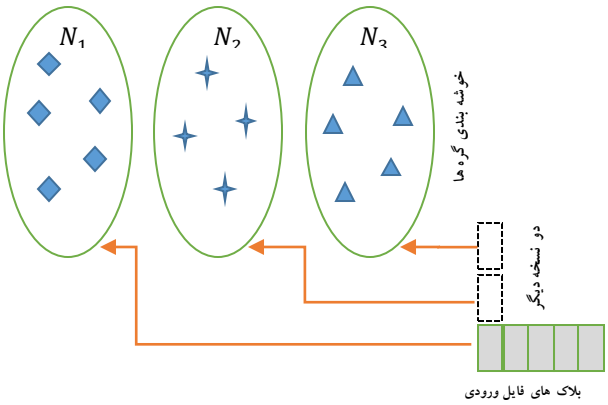
$N_{(i:1..\text{Replica})}$ نمایش هر کدام از دسته ها بعداز اجرای الگوریتم k-mean و $n_{(i,j)}$ گره j ام در دسته i ام است. و به علت اینکه الگوریتم k-mean یک الگوریتم خوشه بندی انحصاری است، هر گره فقط در یک دسته قرار دارد، همچنین داریم که:

$$n_{a,i} < n_{b,j}:$$
$$if\ a < b,\ \forall n_{a,i} \in N_a \wedge \forall n_{b,j} \in N_b \quad (2)$$

چون تعداد دسته های ایجاد شده بر اساس تعداد نسخه های بلاک است، هنگام ذخیره کردن یک فایل، بعد از تقسیم به بلاک‌ها یکسان ـ مثلاً ۶۴ مگابایتی ـ متناظر با هر کدام از نسخه‌ها، یک دسته را انتخاب و بلاک مورد نظر را به گره‌ای از این دسته ارسال می‌کنیم. در این مرحله هم تعداد بلاک‌های ذخیره شده روی یک گره بر اساس ظرفیت پردازشی آن گره در آن دسته محاسبه می‌شود.

الگوریتم (۱) مراحل مختلف این روش جانمایی بلاک را نشان می‌دهد. برای شرح این الگوریتم فرض کنیم می خواهیم یک فایل ۱۵۰ مگابایتی را روی خوشه‌ای که مشخصات گره‌های آن در جدول (۱) آمده است، با اندازه بلاک ۴ مگابایتی ذخیره کنیم. ابتدا فایل NodeState را خوانده و گره‌های خوشه را بر اساس ظرفیت پردازشی هر گره با الگوریتم k-means که در آن k برابر تعداد رونوشت‌های بلاک - برای مثال ۳- دسته بندی می‌کنیم (شکل ۳). سه دسته به شکل زیر بدست می‌آید.

$$C_1 = \{(DN_1, 4), (DN_2, 3.5)\}$$
$$C_2 = \{(DN_3, 2)\} \quad (3)$$
$$C_3 = \{(DN_4, 1)\}$$



(شکل ۳): نمای کلی روش پیشنهادی

(Figure-3): Overview of the proposed method

تعداد بلاک‌های چاک‌های مورد نیاز برای ذخیره بلاک های فایل برابر با $\left\lceil \frac{150}{16} \right\rceil = 10$ است. حال رونوشت اول بلاک‌ها را در دسته C1 ذخیره می‌شود، به این نحوه گره DN1 تعداد $10 \times \left\lceil \frac{4}{4+3.5} \right\rceil = 6$ بلاک و گره DN2 تعداد $10 \times \left\lceil \frac{3.5}{4+3.5} \right\rceil = 4$ بلاک ذخیره می‌شود. و به همین ترتیب یک رونوشت دوم و سوم، هر کدام از این ۱۰ بلاک، روی DN3 و DN4 قرار می‌گیرد.

نکته مهم در این روش این است که اگر تعداد رونوشت ها برابر یک فرض شود، در هنگام دسته بندی گره‌ها، فقط یک دسته به وجود می‌آید. و در نتیجه اجرای این

الگوریتم مشابه الگوریتم چیا وی لی در مقاله [۱۰] خواهد شد.

با استفاده از الگوریتم فوق هم می‌شود، هر الگوریتم جای‌گذاری بلاک‌ها را که در آن تعداد نسخه‌های بلاک نادیده انگاشته شده را ارتقاء بخشید.

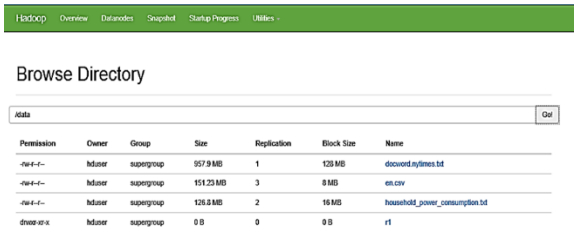
۴-۳-۲- بخش دوم

در بررسی چند مقاله مطرح شده در بخش کارهای انجام شده، به این نتیجه می‌رسیم؛ تاکید تمامی آن‌ها به نوع کار به عنوان عامل تاثیرگذار بر سرعت اجرا کار روی یک گره است. و ظرفیت محاسباتی یک گره را نیز بر اساس سرعت اجرای نوع کارها به دست می‌آورند. چنان که در مقاله [۹] حتی از میانگین زمان اجرای دو برنامه جستجو و شمارش کلمات در آزمایش‌ها استفاده شده است. اما تفاوت سرعت اجرای کارهای مختلف در یک خوشه ناشی از مرتبه اجرایی الگوریتم کارها، آرایش داده‌ی ورودی، حجم داده‌ها و سخت افزار است. و با توجه به بندهای زیر این معیار مناسبی جهت اولویت دادن به نوع کار نیست.

- در هادوپ اندازه بلاک، یک مقدار ثابت نیست (شکل ۴). زیرا هنگام ذخیره کردن یک فایل می‌توان اندازه بلاک را به دلخواه انتخاب نمود. پس یک نوع کار می‌تواند روی داده‌هایی با حجم‌های متفاوت اعمال شود.
- الگوریتم نوع کار A ممکن است پیچیدگی زمانی بیشتری نسبت به نوع کار B داشته باشد. ولی در یک بازه زمانی مشخص امکان دارد کار B بیشتر از A اجرا شود. و انتقال داده-های کار A به گره‌های سریع باعث میشود کار B همیشه روی گره‌های کند اجرا شود و این باعث کاهش زمان پاسخگویی برای کارهای سبک تر خواهد شد. در حالیکه بیشترین تعداد اجرا را دارند.
- نوع سیستم دسترسی به فایل‌ها در HDFS یک بار نوشتن و چندین بار خواندن است. یعنی در هادوپ فایل‌های ذخیره شده قابل تغییر نیستند[۱۲].
- بلاک‌های داده‌ی جدید را فقط می‌توان به انتهای یک فایل اضافه کرد[۱۳].

(شکل ۴) : متفاوت بودن اندازه بلاک و تعداد رونوشتهای

(Figure-4): Different size of block and number of replications



از بندهای فوق به این نتیجه می‌رسیم که روی یک فایل ورودی یک کار یا چندین نوع کار، ممکن است بارها اعمال شود. برای مثال الگوریتم نگاشت کاهش مربوط به پیداکردن دوستان مشترک در فیس بوک، بارها در بازه‌های زمانی مختلفی اجرا می‌شود. در نتیجه در ادامه ما می‌خواهیم با استفاده از ظرفیت پردازش گره‌ها، با ارائه مقیاسی به نام تعداد اجرای نوع کار، انتخاب یک گره برای اجرای کار را مدیریت نمائیم. پس به مرتبه‌ی اجرای نوع کارها، اولویت می‌دهیم. برای بدست آوردن معیاری برای این اولویت این نکته حائز اهمیت است، که یک نوع کار در یک بازه زمانی ممکن است تعداد مرتبه اجرای زیادی داشته باشد ولی در زمانی دیگر به ندرت یا اصلاً اجرا نشود. و این بر اولویت دهی به نوع کارها بر اساس تعداد اجرای آن اثر می‌گذارد. به همین دلیل اولویت دادن به تعداد اجرا را باید در یک بازه

زمانی مشخص بررسی نمائیم. این بازه زمانی (روزانه، هفتگی، ماهانه و غیره) در محیطهای عملیاتی مختلف می تواند متفاوت باشد.

برای پیاده سازی این روش یک فایل به نام JobRepeat در گره نام سیستم هادوپ ایجاد می‌کنیم. این فایل شامل اطلاعاتی در مورد نوع کار و تعداد اجرای نوع کار در بازه زمانی گذشته، و همچنین در بازه زمانی جاری است.

(Table-2): Sample JobRepeat file

جدول ۲: نمونه فایل JobRepeat

Job Type	Run In Last Period	Run In Current Period
Advertisement products	۱۰۰	۰
Data analytics	۱۲۰	۰
Advertisement targeting	۵۰۰	۰
Search ranking	۲۵۶	۰
Yahoo! Mail anti-spam	۲۰	۰
User interest prediction	۱۴۵	۰

برای به روزآوری داده‌های این فایل به محض ورود یک کار جدید، تعداد اجرا در بازه زمانی جاری را افزایش می‌دهیم و بعداز سپری شدن بازه زمانی داده‌های ستون "Run In Current Period" را به ستون "Run In Last Period" انتقال می‌دهیم و تعداد اجرا در بازه زمانی جاری را صفر می‌کنیم. و با استفاده از داده‌های ستون تعداد اجرا در بازه زمانی گذشته اولویت نوع کارها را بررسی می‌کنیم.

براساس الگوریتم ارائه شده در بخش قبل تعداد رونوشت بلاک‌ها را عدد پیش فرض سه قرار داده می‌شود، پس سه دسته گره با ظرفیت پردازشی بالا، متوسط و کم خواهیم داشت که هر دسته یک نسخه از بلاک‌های یک فایل را دارند. کارها را هم به سه دسته براساس اولویت محاسبه شده در جدول (۲) تقسیم می‌کنیم. حال زمانبندکار[8] هنگام انتخاب گره برای اجرا متناسب با اولویت هر کار، گره مناسب را از دسته مربوطه انتخاب خواهدکرد. بدین ترتیب گره هایی که ظرفیت پردازشی سریعتری دارند کارها با اولویت بالاتر و گره‌ها با ظرفیت پردازشی کمتر کارها با اولویت کمتر را اجرا خواهند کرد.

۵- نتیجه گیری

همان طور که بیان شد، این روش با استفاده از نسخه‌های دیگر بلاک، داده‌ها را تقریباً به صورت یکنواخت روی خوشه ذخیره می‌کند. در الگوریتم ارائه شده در صورتی که تعداد نسخه بلاک سه باشد، به دلیل خوشه‌بندی بر اساس ظرفیت پردازش، سه دسته سیستم از هم تفکیک می‌شوند:

دسته اول شامل کامپیوترهای سریع. دسته دوم دارای سیستم‌ها با قدرت پردازش متوسط و دسته سوم شامل سیستم‌های کند می‌باشند. روی هرکدام از دسته‌ها یک کپی از بلاک‌های فایل ورودی ذخیره می‌شود. در صورتی که چندین کار به صورت هم زمان هم اجرا شوند، با استفاده از الگوریتم زمان‌بندی مناسب – الگوریتم بخش دوم - می‌توان سرعت اجرای برنامه‌های نگاشت کاهش را افزایش داد.

[8] Task scheduling process.

مراجع

[1] Dean, J. and S. Ghemawat, MapReduce: simplified data processing on large clusters, in Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6. 2004, USENIX Association: San Francisco, CA. p. 10-10.
[2] Michael, R. *Big Data Commission Releases Report*. 2012 [cited 2015; Available from: http://analytics.ncsu.edu/?p=4770.
[3] *Scaling up with Hadoop and Banyan – Presentation at ITRIX-2015*. 2015 [cited 2015; Available from: http://growbanyan.com/2015/02/scaling-up-with-hadoop-and-banyan-presentation-at-itrix-2015/.
[4] Foundation, A.S. *HDFS Architecture*. 2014 2014-11-13 [cited 2015; Available from: https://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html.
[5] Shvachko, K., et al., *The Hadoop Distributed File System*, in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. 2010, IEEE Computer Society. p. 1-10.
[6] Dean, J. and S. Ghemawat, *MapReduce: simplified data processing on large clusters*. Commun. ACM, 2008. **51**(1): p. 107-113.
[7] Xianglong, Y., et al. *A Novel Blocks Placement Strategy for Hadoop*. in *Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on*. 2012.
[8] Poonthottam, V.P. and S.D. Madhu Kumar. *A Dynamic Data Placement Scheme for Hadoop Using Real-time Access Patterns*. in *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*. 2013.
[9] Jiong, X., et al. *Improving MapReduce performance through data placement in heterogeneous Hadoop clusters*. in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*. 2010.
[10] Lee, C.-W., et al., *A Dynamic Data Placement Strategy for Hadoop in Heterogeneous Environments*. Big Data Research, 2014(0).
[11] Song, L., *A Distributed Algorithm for Graph Center Problem*. Master's thesis, 2003.
[12] Foundation, A.S. *HDFS Architecture*. 2014 2014-11-13 [cited 2015; Available from: https://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html.
[13] 13. Shvachko, K., et al., *The Hadoop Distributed File System*, in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. 2010, IEEE Computer Society. p. 1-10.

قاسم الهی

عضو هیات علمی دانشکده شهید چمران کرمان

gh.elahi@gmail.com