نشریه علمی «پدافند الکترونیکی و سایبری»
سال هشتم، شماره ۲، تابستان ۱۳۹۹؛ ص ۵۹ - ۵۱

علمی–پژوهشی

ارائه یک الگوریتم زمانبندی جدید برای کاهش زمان محاسبات در محیط هادوپ

سید رضا پاکیزه[1*]، سیدمجید عارفی‌نژاد[2]

۱ – مربی گروه کامپیوتر، دانشگاه علمی– کاربردی، دهدشت، ایران

۲– دانشجوی کارشناسی‌ارشد، دانشگاه افسری و تربیت پاسداری امام حسین(ع)

(دریافت: ۹۸/۳/۱۱، پذیرش: ۹۸/۷/۱۰)

چکیده

امروزه پروژه متن‌باز هادوپ به‌همراه چهارچوب نگاشت-کاهش در بین مؤسسات، سازمان‌ها و محققین محبوبیت زیادی دارد که برای پردازش حجم انبوهی از داده‌ها به‌صورت موازی بر روی خوشه‌ای از کامپیوترها بسیار مناسب است. نگاشت-کاهش برای حل مشکلات محاسبات داده‌های حجیم معرفی شده است که از قاعده تقسیم-غلبه پیروی می‌کند. مانند هر جای دیگر، مبحث زمان و زمان‌بندی در نگاشت-کاهش از اهمیت بسیار بالایی برخوردار است. به‌همین دلیل در دهه اخیر الگوریتم‌های زمانبندی متعددی در این زمینه تدارک یافته است. ایده اصلی این الگوریتم‌ها افزایش نرخ محلی‌سازی داده، هم‌زمان‌سازی، کاهش زمان پاسخ و زمان اتمام وظایف می‌باشد. اکثر این الگوریتم‌ها تک هدفه می‌باشند و فقط یکی از موارد ذکر شده را مورد هدف قرار می‌دهند. الگوریتم‌های چند هدفه موجود فقط بر روی یکی از فازهای اول یا دوم نگاشت-کاهش تمرکز دارند. در این مقاله، یک الگوریتم زمان‌بندی ترکیبی مبتنی بر اولویت‌بندی پویا کارها و محلی‌سازی داده در محیط نگاشت-کاهش به نام "HSMRPL" ارائه می‌شود که هدف اصلی آن افزایش نرخ محلی‌سازی داده و کاهش زمان محاسبات می‌باشد. در این الگوریتم از دو روش اولویت‌بندی پویا و شناسه محلی‌سازی استفاده می‌شود. برای ارزیابی الگوریتم پیشنهادی، آن را با الگوریتم‌های پیش‌فرض هادوپ و به کمک محک‌های استاندارد مقایسه کردیم. نتایج حاصله نشان می‌دهد که الگوریتم پیشنهادی ما نرخ محلی‌سازی را نسبت به الگوریتم FIFO، ۱۸/۵ درصد و نسبت به الگوریتم Fair، ۱۰/۴ درصد افزایش داده است. همچنین، الگوریتم پیشنهادی ما نسبت به الگوریتم FIFO، ۳/۸ درصد و نسبت به Fair، ۱۳/۴ درصد سریعتر است.

کلید واژه‌ها: زمانبندی نگاشت-کاهش، الگوریتم ترکیبی، محلی‌سازی داده، اولویت‌بندی پویا، زمانبندی هادوپ

۱- مقدمه

روزانه حجم انبوهی از داده‌ها توسط شرکت‌ها، مؤسسات و مراکز دانشگاهی در حال تولید و توسعه می‌باشد. این داده‌ها نیاز به مکان‌هایی جهت ذخیره شدن و ابزارهایی جهت پردازش دارند. تاکنون ابزارها و روش‌های متعددی برای ذخیره‌سازی و پردازش داده‌های حجیم تدارک یافته است. یکی از محبوب‌ترین آنها "هادوپ" می‌باشد. هادوپ از پروژه‌های متن باز آپاچی است که عملیات مربوط به ذخیره‌سازی و پردازش مجموعه‌های داده عظیم را امکان‌پذیر می‌سازد. این عملیات (ذخیره‌سازی و پردازش داده) را به ترتیب با کمک دو زیر پروژه به نام‌های سامانه پرونده توزیع شده هادوپ(HDFS)[1] و نگاشت-کاهش[2] انجام می‌دهد. سامانه پرونده توزیع شده هادوپ، برای ذخیره‌سازی حجم بسیار وسیعی از داده‌ها مناسب است که بر روی هر سخت افزاری قابل اجرا می‌باشد و مانند خیلی از سامانه پرونده‌های موجود دیگر از ساختار پرونده‌های سلسله مراتبی پشتیبانی می‌کند [۱].

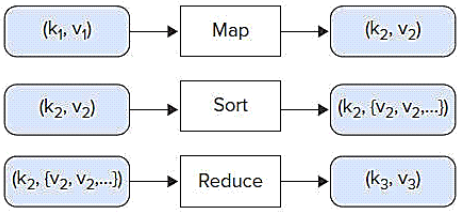
چارچوب نگاشت-کاهش، برای اجرای الگوریتم‌های توزیع‌پذیر و موازی‌پذیر در مجموعه‌های داده عظیم می‌باشد. این چارچوب نخستین بار در سال ۲۰۰۴ توسط گوگل برای پشتیبانی از پردازش‌های توزیع شده در سراسر خوشه‌هایی از کامپیوترها معرفی و به‌کار گرفته شد [۲]. چارچوب نگاشت-کاهش از قاعده تقسیم و غلبه پیروی می‌کند؛ بدین صورت که مجموعه داده‌های ورودی را به تکه‌هایی تقسیم می‌کند که در فاز نگاشت به‌صورت موازی پردازش می‌شوند. به‌طور کلی عملیات مربوط به نگاشت‌کاهش در سه فاز انجام می‌گیرد: فاز نگاشت، فاز مرتب‌سازی و فاز کاهش. فاز نگاشت که پردازش‌های مقدماتی را بر روی داده‌های ورودی انجام می‌دهد.

*رایانامه نویسنده پاسخگو: s.rezapakize@gmail.com

[1] Hadoop Distributed File System
[2] MapReduce

فاز مرتب‌سازی، خروجی حاصل از فاز نگاشت را که شامل زوج‌های کلید/مقدار می‌باشند به‌عنوان ورودی می‌گیرد. در این فاز زوج‌های کلید/مقدار توسط تابعی بخش‌بندی شده و بر اساس کلیدهایشان گروه‌بندی و مرتب می‌شوند. هر کلید یکتا با تمام مقادیرش ترکیب می‌شود که در اینجا می‌توان از یک تابع ترکیب اختیاری استفاده کرد تا اندازه داده‌های میانی را کاهش دهد. فاز کاهش که آخرین فاز می‌باشد، کلید/مقدارهای ترکیب شده را توسط تابع کاهش‌دهنده، کاهش می‌دهد و زوج کلید/مقدار نهایی را تولید می‌کند.

محلی‌سازی داده در نگاشت-کاهش به این معنی است که، یک وظیفه نگاشت بر روی گره‌ای اجرا شود که شامل داده ورودی آن باشد. در چهارچوب نگاشت-کاهش، کدهای نگاشت و کاهش را می‌توان در طول گره‌های خوشه انتقال داد، و داده‌ها را می‌توان از یک گره به گره دیگر منتقل کرد. در صورتی که کد و داده بر روی یک گره باشند، محلی‌سازی داده به‌وجود می‌آید. در اینجا منظور از وظیفه، کاری است که برای اجرا به یک یا چند گره ارسال می‌شود.



شکل (۱): عملیات مربوط به نگاشت-کاهش [۳].

وظیفه چارچوب نگاشت-کاهش فراهم کردن هماهنگی کلی و اجرای عملیات مربوطه می‌باشد. این عملیات شامل: انتخاب ماشین‌های (گره‌ها) مناسب برای اجرای نگاشت کننده، شروع کردن و مدیریت اجرای نگاشت کننده، انتخاب مکان مناسب برای اجرای کاهش دهنده می‌باشند. لازمه اجرای موفقیت‌آمیز این وظایف، مدیریت صحیح زمان و زمانبندی می‌باشد. پارامتر زمان همواره در هر زمینه‌ای جایگاه خاص خود را دارد. در بحث نگاشت- کاهش نیز زمان اجرای عملیات مربوطه یک مسأله مهم به‌شمار می‌آید. در دهه اخیر الگوریتم‌های زمانبندی متعددی تدارک دیده شده است که تقریباً تمامی آنها بر روی بهبود زمان پاسخ و افزایش نرخ محلی‌سازی داده تمرکز دارند. بیشتر آن‌ها فقط یک هدفه می‌باشند و فقط روی یک روش اصلی (مثلاً بهبود محلی‌سازی) و یک روش فرعی تمرکز دارند. همچنین برخی از الگوریتم‌های چند هدفه فقط بر روی یک فاز از فازهای مربوط به نگاشت-کاهش روش‌های خود را پیاده‌سازی کرده‌اند [۴]. به نظر می‌رسد با اعمال اولویت‌بندی پویای کارها و محلی نمودن داده‌ها می‌توان زمان اجرا را تا حد قابل توجهی بهبود بخشید و نرخ محلی‌سازی را افزایش داد. در این تحقیق سعی خواهد شد که با به‌کارگیری ترکیبی دو روش اولویت‌بندی پویا و شاخص محلی‌سازی زمان اجرای محاسبات را کاهش داد و نرخ محلی‌سازی را افزایش داد.

هدف اصلی این تحقیق کاهش زمان اتمام کارها و افزایش نرخ محلی‌سازی داده‌ها در محیط نگاشت- کاهش می‌باشد؛ که سعی خواهد شد با ترکیب دو روش: اولویت‌بندی پویای کارها و اعمال شناسه محلی‌سازی به گره‌های برده این عمل را تحقق داد. این اولویت‌بندی براساس سه پارامتر اندازه کار، زمان اجرا و زمان انتظار کار اعمال می‌شود. اعمال شناسه محلی‌سازی به گره‌های برده به‌نحوی صورت می‌پذیرد که هر گره وضعیت خود را مشخص نماید. با این کار تمام گره‌ها از یک شانس مساوی در گرفتن وظایف محلی برخوردار می‌شوند. بیشتر کارهای انجام گرفته، بر روی یکی از پارامترهای مهم در خصوص زمان‌بندی تمرکز دارند و پارامتر دیگری را در ضمیمه کار خود قرار می‌دهند، که به نسبت توانسته‌اند که زمان اجرا و محلی‌سازی را بهبود بخشند. بنابراین، در این تحقیق یک الگوریتم زمان‌بندی ترکیبی ارائه خواهد شد که تا ضمن افزایش کارایی و سرعت اجرای کارها، زمان اتمام و محلی بودن داده‌ها را نیز بهبود بخشد.

در ادامه این مقاله: در بخش دوم، مروری بر کارهای پیشین در زمینه زمانبندی وظایف در نگاشت- کاهش صورت می‌گیرد. در بخش سوم، معماری روش پیشنهادی ارائه خواهد شد. جزییات مربوط به پیاده‌سازی و نتایج حاصل از آزمایشات در بخش چهارم بیان خواهد شد. بخش پنجم به نتیجه‌گیری و توسعه‌های آتی برای بهبود کارایی روش پیشنهادی می‌پردازد.

۲- مروری بر الگوریتم های زمانبندی محیط هادوپ

هادوپ در ابتدا با سه زمان‌بند FIFO، Fair و Capacity معرفی شد. زمانبند FIFO به‌عنوان زمان‌بند پیش‌فرض هادوپ است. این زمانبند دارای محدودیت‌ها و مشکلاتی همچون: کارایی پایین در هنگام اجرای چندین نوع کار از کارها و زمان اجرای بالا می‌باشد. به منظور پوشش این محدودیت‌ها از زمانبند Fair نیز استفاده می‌شود [۶-۵]. این زمانبند در ابتدا توسط فیس‌بوک مورد استفاده قرار گرفت. ایده اصلی آن انجام عادلانه توزیع منابع محاسباتی در سامانه است. مشکلی که این زمانبند دارد این است که به وزن کارهای گره‌ها توجهی نمی‌کند. برای پوشش دادن این مشکل از زمانبند Capacity نیز استفاده می‌شود. این زمانبند به چندین مستأجر اجازه می‌دهد که یک خوشه بزرگ را به اشتراک بگذارند. در این زمانبند زمان اجرای بالا و پیچیدگی از مهم‌ترین مشکلات آن به شمار می‌آیند [۸-۷].

زاهاریا و همکاران [۹] زمانبند LATE را ارائه دادند. ایده اصلی آن تحمل‌پذیری خطا است. این الگوریتم در محیط ناهمگن نگاشت-کاهش کار می‌کند که با محاسبه زمان باقی مانده از تمام وظایف، وظایف کُند[1] را شناسایی می‌کند. LATE دارای مشکلاتی از قبیل: کارایی پایین و کند بودن (زمان اجرای طولانی) اجرای عملیات می‌باشد. ایده اصلی این زمان‌بند پوشش دادن مشکل مربوط به برخورد محلی بودن و عادلانگی می‌باشد. در الگوریتم تأخیر، هنگام تخصیص وظیفه به یک گره ترتیب کارها شکسته می‌شود. در اینجا در صورتی که در کار اول هیچ وظیفه محلی برای تخصیص به گره پیدا نشد، زمان‌بند به اندازه D زمان تأخیر ایجاد می‌کند و به وظایف محلی کار بعدی مراجعه می‌کند. زمان D به عنوان حداکثر زمان تأخیر است که از قبل می‌توان مقدار آن را مشخص نمود. در اینجا تا زمانی که به اندازه D صبر کرد و وظیفه محلی برای تخصیص پیدا نشد، یک وظیفه غیر محلی تخصیص داده می‌شود. پارامتر D یک بحران به شمار می‌آید چراکه در پیکربندی‌های مختلف سخت‌افزاری و بارهای کاری مختلف نیاز به تغییر آن می‌باشد.

چن و همکاران [۱۰]، الگوریتم SAMR را پیشنهاد دادند، که یک الگوریتم زمان‌بندی انطباقی است. این زمان‌بند با حفظ زمان اجرا و منابع سامانه موجب بهبود عملیات در نگاشت-کاهش گردید. مهم‌ترین مشکل این الگوریتم عدم توجه به محلی بودن داده است.

لی و همکاران [۱۱]، الگوریتم CREST را پیشنهاد دادند که زمان اجرای بهینه برای وظایف نگاشت و کاهش زمان پاسخ را برای چارچوب نگاشت- کاهش را در پی داشت. این الگوریتم برای کاهش دادن وظایف سرگردان[2]، معمولاً به اجرای یک کپی انحصاری از آن وظیفه می‌پردازد. ایده اصلی CREST، اجرای دوباره یک ترکیبی از وظایف بر روی یک گروه از گره‌های خوشه که ممکن است پیشرفت سریع‌تری نسبت به اجرای بر روی گره اصلی داشته باشند؛ این کار برای افزایش نرخ محلی‌سازی داده می‌باشد.

هی و همکاران [۱۲]، یک الگوریتم زمان‌بندی Match ارائه دادند. در این الگوریتم از روش شناسه محلی برای محلی‌سازی داده‌ها استفاده کرده‌اند. این الگوریتم زمان زیادی نیاز دارد تا بتواند وظایف و داده‌ها را در گره‌های محلی قرار دهد. همچنین فقط از روش‌های تصادفی برای اولویت‌بندی کارها استفاده می کند. بنابراین، علیرغم افزایش نرخ محلی سازی داده، باعث کند شدن زمان اجرای وظایف شده است.

نیون و همکاران [۱۳]، الگوریتم زمان‌بند Hybs را ارائه کرده‌اند. در این الگوریتم از روشی برای اولویت‌بندی پویای کارها استفاده می‌شود. اشکال اصلی این الگوریتم عدم استفاده از روش محلی‌سازی داده است. به همین دلیل، مدت زمان تخصیص داده طولانی می‌شود که در نتیجه باعث می‌شود تا زمان اجرا افزایش یابد.

ابراهیم و همکاران [۱۴]، الگوریتم Maestro را ارائه دادند. هدف اصلی این الگوریتم بهبود کارایی کلی در محاسبات نگاشت- کاهش است. این الگوریتم محلی بودن داده‌ها را نیز فراهم می‌کند.

آهمید و همکاران [۱۵]، الگوریتم MaRCO را ارائه دادند. این الگوریتم برای مدیریت سربار فاز میانی مورد استفاده قرار می‌گیرد. مهم‌ترین مزیت آن کاهش زمان اجرا می‌باشد.

رسولی و همکاران [۱۶]، زمانبند جدیدی به نام COSHH پیشنهاد دادند که یک روش زمان‌بندی اشتراکی برای هادوپ است. این زمانبند مناسب کارهایی است که زمان ورود آنها به سامانه، متفاوت باشد و پردازش آنها به محض امکان انجام می‌پذیرد که به‌طور چشم‌گیری کارایی سامانه را افزایش می‌دهد.

الگوریتم پیشنهادی ما از ترکیب روش‌های دو الگوریتم Hybs و Match به یک الگوریتم جامع تبدیل شده است. چراکه در بحث زمانبندی هادوپ دو پارامتر محلی‌سازی داده و اولویت‌بندی بسیار حائز اهمیت است. در الگوریتم Hybs هیچ روشی برای بحث محلی‌سازی استفاده نشده است. بنابراین، هم از نظر هزینه محلی‌سازی و هم از نظر مدت زمان انتظار کارها الگوریتم ما بهتر است. در مقایسه با الگوریتم Matc، این الگوریتم از روش اولویت‌بندی پیشفرض استفاده می‌کند که این اولویت‌بندی به‌صورت تصادفی صورت می‌گیرد. در اینجا ممکن است یک کار مدت زمان زیادی در صف منتظر باشد تا به گره تخصیص داده شود. اما در الگوریتم ما اولویت‌بندی با در نظرگرفتن تمام جوانب صورت می‌پذیرد.

۳- الگوریتم پیشنهادی

روش پیشنهادی در این بخش یک الگوریتم زمانبندی ترکیبی نگاشت-کاهش است که از دو روش شناسه محلی‌سازی و اولویت‌بندی پویای کار استفاده می‌کند. هدف اصلی این الگوریتم، افزایش نرخ محلی‌سازی داده و کاهش زمان محاسبه و اتمام کار می‌باشد که به"HSMRPL"[3] نام‌گذاری شده است.

[1] slow task
[2] straggler tasks
[3] Hybrid Scheduling MapReduce algorithm based on dynamic Priority and data Locality

در این بخش ابتدا روش اولویت‌بندی پویای مورد استفاده شده و سپس روش محلی‌سازی را به تفصیل مورد بررسی قرار می‌دهیم. در پایان مراحل این الگوریتم را به همراه شبه کد آن به صورت گام به گام توضیح می‌دهیم.

۳-۱- اولویت‌بندی پویا در HSMRPL

در الگوریتم پیشنهادی ما از اولویت‌بندی پویای کارها برای مشخص نمودن اینکه کدام وظایف از کدام کارها باید به گره منبع تخصیص داده شوند، استفاده می‌شود. اولویت‌بندی یک کار بر انتخاب وظایف تأثیر می‌گذارد. در اینجا سیاست‌های مختلفی توسط یک مدیر می‌تواند به بارهای کاری تخصیص داده شود. یک سیاست ممکن است بر اساس زمان انتظار کار در صف باشد. در این حالت، اولویت‌بندی شامل یک فاکتور وزن براساس زمان انتظار است که از رابطه زیر محاسبه می‌شود:

$$\text{Weight}_{\text{wait-time}} = \left(\frac{td}{avgWaitTime}\right)^{\alpha} \qquad (۱)$$

td در این رابطه، زمان انتظار کار در صف است. *avgWaitTime*، میانگین زمان انتظار کار در صف است و α هم فاکتور وزن می‌باشد.

سیاست دیگر ممکن است زمان اجرای کار باشد. در این حالت کارهای با زمان اجرای کوتاه‌تر اولویت بالاتری دارند که از رابطه زیر محاسبه می‌شود:

$$\text{Weight}_{\text{run-time}} = \left(\frac{t}{avgRunTime}\right)^{\beta} \qquad (۲)$$

r در این رابطه، میانگین زمان اجرا برای وظایف نگاشت از یک کار می‌باشد و *avgRunTime*، میانگینی از میانگین زمان اجرا برای وظایف نگاشت از تمام کارها می‌باشد. β هم یک فاکتور وزن است.

یک سیاست دیگر ممکن است بر اساس وظایف زمانبندی نشده، باقی مانده از یک کار باشد، که از رابطه زیر محاسبه می‌شود:

$$\text{Weight}_{\text{job-task}} = \left(\frac{n}{avgNumTask}\right)^{\gamma} \qquad (۳)$$

در این رابطه، n و *avgNumTasks* به ترتیب، تعداد وظایف باقی مانده برای یک کار و میانگین تعداد وظایف باقی مانده از کار در کل صف می‌باشند. γ هم یک فاکتور وزن است.

از ترکیب سه رابطه فوق، رابطه اولویت‌بندی کلی برای سامانه به دست می‌آید:

$$\text{Priority} = \left(\frac{td}{avgWaitTime}\right)^{\alpha} * \left(\frac{t}{avgRunTime}\right)^{\beta} * \left(\frac{n}{avgNumTask}\right)^{\gamma} \qquad (۴)$$

همان‌طور که در بالا ذکر شد، α، β و γ فاکتورهای وزن هستند که به‌عنوان پارامترهای اولویت‌بندی می‌باشند. سیاست‌های مختلفی برای این پارامترها وجود دارد. به‌عنوان مثال، اگر α=۱ و β=γ=۰ باشند، مشابه الگوریتم FIFO عمل خواهد کرد. اگر α=۰، β=-۱ و γ=۰ باشد، همانند SJF عمل می‌کند. در الگوریتم پیشنهادی ما عملگر وزن α را همواره برابر صفر قرار دادیم.

این الگوریتم به محاسبه زمان اجرا، اندازه و زمان انتظار یک کار در صف توجه دارد که، برای کاهش تأخیر ناشی از طول متغیر کارها استفاده می‌شود.

۳-۲- روش محلی‌سازی داده در HSMRPL

در الگوریتم پیشنهادی ما برای افزایش نرخ محلی‌سازی از یک روش شناسه‌گذاری استفاده کردیم. ایده اصلی ما این است که به هر گره برده برای گرفتن وظایف محلی از کار، یک شانس عادلانه بدهیم. برای تحقق این عمل، به هرکدام از گره‌های برده یک شناسه محلی‌سازی می‌دهیم. با کمک این شناسه وضعیت هر گره برده مشخص می‌شود. درصورتی که این شناسه مقدار یک داشته باشد به این معنی است که یک وظیفه غیر محلی به آن تخصیص داده شده است. در صورتی که شناسه محلی تمامی گره ها یک شود، باید منتظر بلوک داده وظایف باقی مانده شوند. در الگوریتم ما بر خلاف الگوریتم‌های پیش‌فرض هادوپ، هنگامی که به کار اول برای گرفتن وظیفه محلی مراجعه شد و با شکست مواجه شد (وظیفه محلی پیدا نشود)، یک وظیفه غیر محلی تخصیص داده نمی‌شود بلکه به جست‌وجو در کار بعدی می‌پردازد. در اینجا ابتدا به صف کارها مراجعه می‌شود، اگر یک وظیفه نگاشت محلی پیدا نشد به کار بعدی موجود در صف مراجعه می‌کند. در صورتی که در کار بعدی هم نتواند یک وظیفه محلی پیدا کند، یک وظیفه غیر محلی تخصیص می‌دهد. هدف از این عمل جلوگیری از اتلاف زمان و منابع می‌باشد. به‌طور کلی در طول یک ضربان قلب[1]، حداکثر یک وظیفه غیرمحلی می‌تواند تخصیص داده شود.

۳-۳- تشریح مراحل الگوریتم HSMRPL

جدول (۱) شبه کد الگوریتم پیشنهادی ما را نشان می‌دهد. مراحل این الگوریتم به شرح زیر می‌باشند:

- اولویت‌بندی پویای کارهای موجود در صف
- تخصیص یک شناسه محلی‌سازی به تمام گره‌های برده

[1] heart beat

- محاسبه تعداد نگاشت‌ها و کاهش‌ها برای ردیابه‌کار جاری
- تخصیص وظایف
- محاسبه تعداد وظایف نگاشت و کاهش در حال اجرا و شکست‌خورده
- تلاش برای تخصیص یک وظیفه محلی به گره برده
- عدم تخصیص وظیفه غیر محلی و تلاش مجدد برای تخصیص وظیفه محلی
- تخصیص حداکثر یک وظیفه غیر محلی بعد از دو شکست در یافتن وظیفه محلی برای جلوگیری از اتلاف منابع
- پاک شدن شناسه محلی‌سازی گره‌های برده بعد از ورود یک کار جدید
- وضعیت تمامی گره‌ها بازنشانی[1] می‌شود.

جدول (۱): شبه کد الگوریتم پیشنهادی

HSMRPl Scheduling Algorithm

```
for each heartbeat, when there are free slots on i nod
update priority for all jobs // priority is based on equation (4)
  for each node i of the N slave nodes do
      assign  Localization ID [i]=null  to all slave nodes
      end for
          set previous status= localization ID[i]
    end for
  for each job in job queue do
    assign local task to node i
    set s=s-1  // s is the count of free slots
   if Localization ID [i]==null then
     Localization ID [i]=1
   else Localization ID [i]+=1
   end if
   break for
  else continue
                end if
end for
if Localization ID [i]==0 then
assign a non-local task t' node i from the first job in the JobQueue
set s=s-1
end if

When a new job j is added into the JobQueue:
for each node i of the N slave nodes do
  set Localization ID [i]=null
end for
```

در مراحل فوق، ابتدا بر اساس رابطه (۴) تمامی کارهای موجود در صف اولویت‌بندی می‌شوند. سپس به تمامی گره‌های برده یک شناسه محلی‌سازی داده می‌شود. سپس وضعیت قبلی هر گره را برابر با آن قرار می‌دهد. در مرحله بعد، از کارهای موجود در صف، وظایف محلی را به گره برده تخصیص می‌دهد و یک واحد از شکاف‌های خالی موجود در آن کم می‌کند. سپس شناسه محلی‌سازی را بررسی می‌کند درصورتی که برابر با *Null* باشد، آن را برابر با ۱ قرار می‌دهد. در صورتی که مقدار شناسه برابر با صفر باشد، یک وظیفه غیر محلی به گره تخصیص می‌دهد و یک واحد از شکاف‌های خالی کم می‌کند. در واقع با استفاده از این شناسه، وضعیت گره‌ها برای تخصیص عادلانه وظایف مشخص می‌شود. در پایان با ورود کار جدید به صف، وضعیت تمامی گره‌ها بازنشانی می‌شود.

۴- ارزیابی و مقایسه الگوریتم پیشنهادی با الگوریتم‌های پیش‌فرض

در این بخش به بررسی پیاده سازی و ارزیابی مسئله پرداخته خواهد شد؛ که با طراحی آزمایش‌های متنوع و با استفاده از محک‌های استاندارد سعی خواهد شد کارایی الگوریتم پیشنهادی را در مقایسه با دو الگوریتم پیش‌فرض هادوپ سنجیده شود. دلیل انتخاب دو الگوریتم FIFO و Fair این است که تمامی الگوریتم‌هایی که در این زمینه ارائه شدند، با این دو الگوریتم مقایسه شده‌اند. در این بخش، ابتدا محیط پیاده‌سازی و خصوصیات استفاده شده بیان می‌شود. سپس به بررسی نتایج به‌دست آمده و تحلیل آنها پرداخته خواهد شد.

۴-۱- محیط شبیه‌سازی

برای پیاده‌سازی الگوریتم‌های زمان‌بندی نگاشت-کاهش هادوپ از روش‌های مختلفی استفاده می‌شود. این روش‌ها اغلب در محیط‌های مبتنی بر لینوکس استفاده می‌کنند. با توجه به این‌که امروزه محیط‌های گرافیکی ضمن کارایی راحت، خروجی‌های ملموس‌تری ارائه می‌دهند، بنابراین، سعی کردیم از بسترهایی که شرکت‌های فعال در زمینه کلان داده‌ها و هادوپ ارائه می‌دهند استفاده کنیم. از جمله این سکوها می‌توان به هارتن ورک و کلودرا اشاره کرد. با توجه به فیلتر بودن سایت هارتن ورک برای کاربران ایران (در حال حاضر) و همچنین امکانات بیشتر کلودرا (پشتیبانی از الگوریتم‌های پیش‌فرض)، بستر مدیریتی کلودارا را به‌عنوان محیط شبیه‌سازی خود انتخاب کردیم. برای این کار، هادوپ نسخه ۲،۳،۰ را با استفاده از نرم‌افزار Virtual box بر روی کلودارا نصب نموده و یک خوشه با ۱ گره ارباب و ۳ گره برده پیکربندی کردیم. جداول (۲) و (۳) به ترتیب، نرم‌افزارهای استفاده شده به همراه نسخه‌های آنها و خصوصیات پیکربندی

[1] Reset

گره‌های استفاده شده در محیط شبیه‌سازی را نشان می‌دهد.

جدول(۲): نرم‌افزارهای استفاده شده برای شبیه‌سازی.

نام ابزار	نسخه
Oracle VM Virtual Box	۴.۳.۳۰
Cent Os	۶.۴
Hadoop	۲.۳.۰
Cloudera Distributed Hadoop (CDH)	۵.۴.۰

جدول (۳): مشخصات محیط شبیه‌سازی.

گره‌ها	تعداد	مشخصات پیکربندی
ارباب	۱	single-core 2.2GHz intel-64 [۲] CPUs, 8GB RAM
برده	۳	single-core 2.2GHz Intel-64 [۲] CPUs, 4GB RAM

۴-۲- محک‌های استاندارد مورد استفاده

برای ارزیابی تمامی الگوریتم‌های زمانبندی هادوپ از محک‌های استانداردی همچون: WordCount، Terasort، Pi، MRbench، Gridmix استفاده می‌شود. برای ارزیابی الگوریتم پیشنهادی خود از دو محک WordCount و Terasort استفاده کردیم.

WordCount، برنامه‌ای است که تعداد دفعات تکرار کلمات را در متن محاسبه می‌کند. این برنامه توزیع منظمی از طول نگاشت و توزیع تصادفی از زوج‌های کلید/مقدار را نشان می‌دهد. باتوجه به محدودیت‌های سخت افزاری و پهنای باند، این برنامه را بر روی یک پرونده متنی ساده انجام دادیم. Terasort، یک مرتب‌ساز استاندارد نگاشت و کاهش است، که زمان مرتب‌سازی یک مجموعه‌داده را محاسبه می‌کند. همچنین میزان کارایی نگاشت‌کاهش را در هادوپ ارزیابی می‌کند. در این مقاله از یک مجموعه داده ۵ گیگابایتی[۱] برای ارزیابی الگوریتم خود استفاده کردیم.

۴-۳- معیارهای ارزیابی

در این تحقیق برای ارزیابی الگوریتم پیشنهادی با الگوریتم‌های FIFO و Fair از معیارهای نرخ محلی‌سازی و زمان اتمام کلی کار استفاده شده است. این معیارها مهم‌ترین و پر استفاده‌ترین معیارهای ارزیابی الگوریتم‌های زمانبندی در هادوپ هستند. نرخ محلی‌سازی به‌صورت زیر محاسبه می‌شود:

$$\text{Data Locality Rate} = \frac{l}{n} \qquad (۵)$$

در این رابطه l، تعداد وظایف نگاشت محلی و n تعداد کل وظایف نگاشت می‌باشد.

۴-۴- نتایج ارزیابی کلی ۴.۳.۳۰

در این بخش نتایج حاصل از چهار آزمایش را مورد بررسی قرار می‌دهیم. جدول (۴) پارامترهای محیط پیکربندی را برای این آزمایش‌ها نشان می‌دهد. این آزمایش‌ها به شرح زیر می‌باشند که در ادامه به بررسی نتایج هرکدام می‌پردازیم:

- ارزیابی الگوریتم پیشنهادی با دو الگوریتم FIFO و Fair با استفاده از محک WordCount بر اساس معیار نرخ محلی‌سازی
- ارزیابی الگوریتم پیشنهادی با دو الگوریتم FIFO و Fair با استفاده از محک Terasort بر اساس معیار نرخ محلی‌سازی
- ارزیابی الگوریتم پیشنهادی با دو الگوریتم FIFO و Fair با استفاده از محک WordCount بر اساس معیار زمان اتمام وظایف کار
- ارزیابی الگوریتم پیشنهادی با دو الگوریتم FIFO و Fair با استفاده از محک Terasort بر اساس معیار زمان اتمام وظایف کار

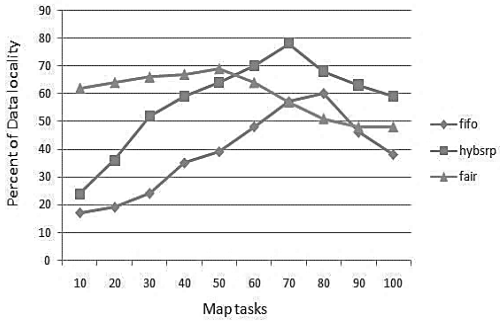
جدول (۴): پارامترهای مهم پیکربندی در کلودارا برای آزمایش‌های به‌عمل آمده

پارامترها	مقدار
mapreduce.reduce.memory.mb	۳۰۷۲
mapreduce.map.memory.mb	۱۰۲۴
mapred.maxthreads.generate.mapoutput	۲
mapreduce.tasktracker.reserved.physicalmemory.mb.low	۰/۹۵
mapred.maxthreads.partition.closer	۲
mapreduce.map.sort.spill.percent	۰/۹۹
mapreduce.reduce.merge.inmem.threshold	۰
mapreduce.job.reduce.slowstart.completedmaps	۱
mapreduce.reduce.shuffle.parallelcopies	۴۰
mapreduce.map.speculative	False
mapreduce.reduce.speculative	False
mapreduce.map.output.compress	False
mapreduce.job.reduces	۱۶۰
mapreduce.task.io.sort.mb	۴۸۰
mapreduce.task.io.sort.factor	۴۰۰
mfs.heapsize	۳۵

[۱] مجموعه داده بدون ساختار استفاده شده در اکثر مقالات مشابه

۴-۴-۱- ارزیابی الگوریتم پیشنهادی با دو الگوریتم FIFO و Fair با استفاده از محک WordCount بر اساس معیار نرخ محلی‌سازی

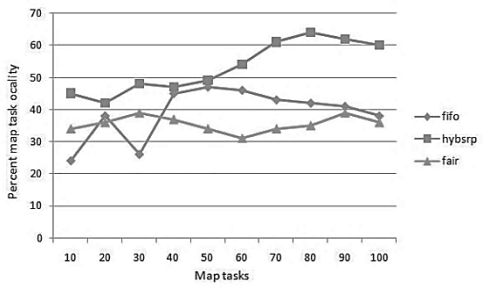
در این آزمایش برنامه شمارش کلمات با هر سه الگوریتم HSMRPL، FIFO و Fair را در شرایط یکسانی بر روی چارچوپ YARN اجرا کردیم. شکل (۲) نتیجه این آزمایش را نشان می‌دهد. در اینجا از میانگین درصد محلی‌سازی استفاده کردیم. میانگین درصد محلی‌سازی سه الگوریتم HSMRPL، FIFO و Fair به ترتیب برابر با ۶۱/۶٪، ۳۸/۸٪ و ۵۱/۲٪ می‌باشد. همان‌طور که مشاهده می‌شود، الگوریتم پیشنهادی ما نرخ محلی‌سازی را نسبت به الگوریتم FIFO، ۱۸/۵ درصد و نسبت به الگوریتم Fair، ۱۰/۴ درصد افزایش داده است.



شکل (۲): مقایسه الگوریتم پیشنهادی با الگوریتم‌های FIFO و Fair بر اساس نرخ محلی‌سازی.

۴-۴-۲- ارزیابی الگوریتم پیشنهادی با دو الگوریتم FIFO و Fair با استفاده از محک Terasort بر اساس معیار نرخ محلی‌سازی

در این آزمایش هر سه الگوریتم HSMRPL، FIFO و Fair را با محک Terasort در شرایط یکسانی بر روی چارچوپ YARN برای مرتب‌سازی یک مجموعه‌داده ۵ گیگابایتی اجرا کردیم. شکل (۳) نتیجه این آزمایش را نشان می‌دهد. در اینجا نیز از میانگین درصد محلی‌سازی استفاده کردیم. همان‌طور که مشاهده می‌شود، الگوریتم پیشنهادی ما نرخ محلی‌سازی را نسبت به الگوریتم FIFO، ۱۴ درصد و نسبت به الگوریتم Fair، ۱۷/۷ درصد افزایش داده است.



شکل (۳): مقایسه الگوریتم پیشنهادی با الگوریتم‌های FIFO و Fair بر اساس نرخ محلی‌سازی با Terasort.

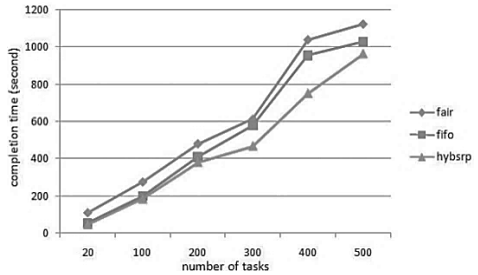
۴-۴-۳- ارزیابی الگوریتم پیشنهادی با دو الگوریتم FIFO و Fair با استفاده از محک WordCount بر اساس معیار زمان اتمام وظایف کار

در این آزمایش ما از شش بارکاری با تعداد وظایف نگاشت مختلف استفاده کردیم. سپس با استفاده از برنامه شمارش کلمات آن‌ها را اجرا نمودیم. از نتایج به‌دست‌آمده میانگین گرفته و سپس مقدار به حاصله را با یکدیگر مقایسه کردیم. جدول (۵) نتایج این آزمایش را نشان می‌دهند.

جدول(۵): نتیجه حاصل از آزمایش با محک WordCount

الگوریتم	تعداد وظایف	مدت زمان اتمام (برحسب ثانیه)
FIFO	۲۰	۵۷
	۱۰۰	۲۰۱
	۲۰۰	۴۱۱
	۳۰۰	۵۸۱
	۴۰۰	۹۵۶
	۵۰۰	۱۰۳۰
Fair	۲۰	۱۱۰
	۱۰۰	۲۷۶
	۲۰۰	۸۴۱
	۳۰۰	۹۱۴
	۴۰۰	۱۰۴۲
	۵۰۰	۱۱۲۵
HSMRPL	۲۰	۴۶
	۱۰۰	۱۸۴
	۲۰۰	۳۸۱
	۳۰۰	۴۶۸
	۴۰۰	۷۵۱
	۵۰۰	۹۶۴

میانگین زمان اتمام وظایف برای FIFO، Fair و HSMRPL به ترتیب برابر با ۵۳/۹ ، ۶۰/۷ و ۴۶/۵ درصد می‌باشند. نمودار شکل (۴-۵) نتیجه این آزمایش را نشان می‌دهد. همان‌طور که مشاهده می‌شود، الگوریتم Fair بیشترین زمان و الگوریتم HSMRPL کمترین زمان را دارد. در این آزمایش، الگوریتم پیشنهادی ما نسبت به الگوریتم FIFO، ۳/۸ درصد و نسبت به Fair، ۱۳/۴ درصد سریع‌تر می‌باشد.



شکل (۴): مقایسه سه الگوریتم بر اساس زمان اتمام وظایف با محک WordCount.

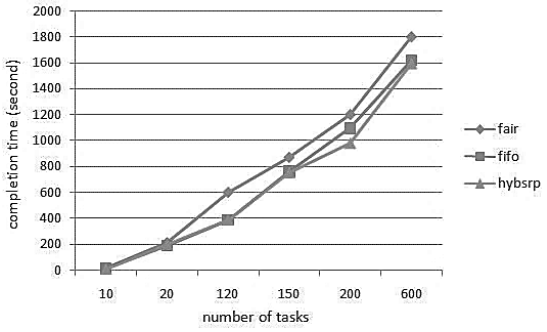
۴-۴-۴- ارزیابی الگوریتم پیشنهادی با دو الگوریتم FIFO و Fair با استفاده از محک Terasort بر اساس معیار زمان اتمام وظایف کار

در این آزمایش همانند آزمایش قبل، از شش بارکاری با تعداد وظایف نگاشت مختلف استفاده کردیم. سپس مثال Terasort را با هرکدام از الگوریتم‌ها اجرا نمودیم. از نتایج به‌دست‌آمده میانگین گرفته و سپس مقدار به‌دست‌آمده را با یکدیگر مقایسه کردیم. جدول (۶) نتایج این آزمایش را نشان می‌دهد.

جدول (۶): نتیجه حاصل از آزمایش سه الگوریتم با محک Terasort.

الگوریتم	تعداد وظایف	مدت زمان اتمام (برحسب ثانیه)
FIFO	۱۰	۱۳
	۲۰	۱۸۵
	۱۲۰	۳۸۷
	۱۵۰	۷۶۱
	۲۰۰	۱۰۴۹
	۶۰۰	۱۶۲۰
Fair	۱۰	۱۵
	۲۰	۲۱۰
	۱۲۰	۶۰۰
	۱۵۰	۸۷۱
	۲۰۰	۱۲۱۶
	۶۰۰	۱۶۲۰
HSMRPL	۱۰	۱۰
	۲۰	۲۰۰
	۱۲۰	۲۹۱
	۱۵۰	۷۵۴
	۲۰۰	۹۸۱
	۶۰۰	۱۵۹۶

میانگین زمان اتمام وظایف برای FIFO، Fair و HSMRPL به ترتیب برابر با ۶۷/۶، ۷۸/۲ و ۶۵/۵ درصد می‌باشند. در این آزمایش الگوریتم FIFO بسیار نزدیک به الگوریتم پیشنهادی ما است. نمودار شکل (۵) نتیجه این آزمایش را نشان می‌دهد. همان‌طور که مشاهده می‌شود، الگوریتم پیشنهادی ما نسبت به الگوریتم FIFO، ۲/۱ درصد و نسبت به الگوریتم Fair، ۱۲/۷ درصد سریع‌تر می‌باشد.



شکل (۵): مقایسه سه الگوریتم بر اساس زمان اتمام وظایف با محک Terasort

۵- نتیجه‌گیری

در حین انجام آزمایشات نتایج قابل توجهی گرفته شده است که می‌توان پس از تجزیه و تحلیل، نتایج را به صورت زیر بیان نمود:

- اولویت‌بندی پویای کارها براساس زمان انتظار، زمان اجرا و اندازه کار، برای جلوگیری از تأخیر ناشی از طول متغییر کارها بسیار مناسب می‌باشد که موجب کاهش زمان پاسخ کلی و زمان اتمام وظایف می‌گردد.
- محلی‌سازی داده در سامانه‌های نگاشت-کاهش یکی از موارد کلیدی است که در بهبود کارایی کلی نقش به‌سزایی ایفا می‌کند.
- با بررسی نتایج آزمایش‌ها، می‌توان دریافت که الگوریتم پیشنهادی ما هم از نظر سرعت در زمان اتمام وظایف و هم از لحاظ نرخ محلی‌سازی داده برتری چشم‌گیری نسبت به الگوریتم‌های FIFO و Fair دارد. در مقایسه زمان پاسخ، الگوریتم Fair کندترین و در مقایسه نرخ محلی‌سازی، الگوریتم FIFO کمترین نرخ را دارد.
- با روش به‌کار گرفته برای محلی‌سازی داده، نه تنها باعث کُند شدن اجرای عملیات نشده بلکه به کمک ترفند تخصیص حداکثر یک وظیفه غیرمحلی در طول هر ضربان قلب، موجب شده تا از اتلاف منابع محاسباتی موجود در سامانه نگاشت-کاهش جلوگیری شود.
- از میان الگوریتم متعدد ارائه شده در زمینه زمان‌بندی نگاشت-کاهش، بعضی از آنها فقط بر روی یک مسأله تمرکز دارند و مسأله دیگری را ضمیمه کار خود قرار می‌دهند. با توجه به اینکه در الگوریتم پیشنهادی ما به دو مسأله مهم اولویت‌بندی پویای کارها و محلی‌سازی داده‌ها، تمرکز ویژه‌ای داشتیم، این الگوریتم به‌عنوان یک الگوریتم جامع برای تحلیل مجموعه‌های داده حجیم با شرایط بهینه بسیار مناسب خواهد بود.

۶- مراجع

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing On Large Clusters," Communications of the ACM, vol. 51(1), pp. 107-113, 2008.

[2] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments," In OSDI, vol. 8, no. 4, p. 7, 2008.

[3] T. White, "Hadoop: The definitive guide," O'Reilly Media, Inc., 2012.

[4] S. Perera, "Hadoop MapReduce Cookbook," Packt Publishing Ltd, 2013.

[5] S. R. Pakize, "A Comprehensive View of Hadoop Map Reduce Scheduling Algorithms," International Journal of Computer Networks and Communications Security, ISSN, pp. 2308-9830, 2014.

[6] V. Prajapati, "Big Data Analytics with R and Hadoop," Packt Publishing Ltd, 2015.

[7] I. Hashem, T. Abaker, et al., "MapReduce Scheduling Algorithms: a review," The Journal of Supercomputing, pp. 1-31, 2018.

[8] K. Hadjar and A. Jedidi, "A New Approach for Scheduling Tasks and/or Jobs in Big Data Cluster," 4th MEC International Conference on Big Data and Smart City (ICBDSC), IEEE, 2019.

[9] M. Zaharia, D. Borthakur, J. Sen Sarma, K.Elmeleegy, S. Shenker, and I. Stoica, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling," In Proceedings of the 5th European conference on Computer systems, pp. 265-278, 2010.

[10] Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, "Samr: A Self-adaptive Mapreduce Scheduling Algorithm in Heterogeneous Environment," In Computer and Information Technology (CIT), IEEE 10th International Conference on, pp. 2736-2743, 2010.

[11] L. Lei, T. Wo, and C. Hu, "CREST: Towards Fast Speculation of Straggler Tasks in MapReduce," In e-Business Engineering (ICEBE), IEEE 8th International Conference on, pp. 311-316, 2011.

[12] C. He, Y. Lu, and D. Swanson, "Matchmaking: A new Mapreduce Scheduling Technique," In Cloud Computing Technology and Science (CloudCom), IEEE Third International Conference on, pp. 40-47, 2011.

[13] P. Nguyen, T. Simon, M. Halem, D. Chapman, and Q. Le, "A Hybrid Scheduling Algorithm for data Intensive Workloads in a Mapreduce Environment," In Proceedings of the IEEE/ACM Fifth International Conference on Utility and Cloud Computing, pp. 161-167, 2012.

[14] S. Ibrahim, H. Jin, L. Lu, B. He, G. Antoniu, and S. Wu, "Maestro: Replica-aware Map Scheduling for Mapreduce," In Cluster, Cloud and Grid Computing (CCGrid), 12th IEEE/ACM International Symposium on, pp. 435-442, 2012.

[15] F. Ahmad, S. Lee, M. Thottethodi, and T. N. Vijaykumar, "MapReduce with Communication Overlap (MaRCO)," Journal of Parallel and Distributed Computing, vol. 73(5), pp. 608-620, 2013.

[16] A.Rasooli and D. G. Down, "COSHH: A Classification and Optimization Based Scheduler for Heterogeneous Hadoop systems," Future Generation Computer Systems, vol. 36, pp. 1-15, 2014.

Journal of Electronical & Cyber Defence
Vol. 8, No. 2, 2020, Serial No. 30

A New Scheduling Algorithm to Reduce Computation Time in Hadoop Environment

S. R. Pakize*, S. M. Arefi nejad

*Department of Computer, University of Science and Aplication, Dehdasht,, Iran

(Received: 01/06/2019, Accepted: 02/10/2019)

ABSTRACT

Nowadays, the Hadoop open-source project with the MapReduce framework has become very popular as it processes vast amounts of data in parallel on large clusters of commodity hardware in a reliable and fault-tolerant manner. MapReduce was introduced to solve large-data computational problems, and is dependent on the divide and conquer principle. Time and scheduling are always the most important aspects, hence in the past decades in the MapReduce environment, many scheduling algorithms have been proposed. The main ideas of these algorithms are increasing data locality rate, and decreasing response time and completion time. In this research we have proposed a new hybrid scheduling algorithm (HSMRPL) which uses dynamic job priority and identity localization techniques, and focuses on increasing data locality rate and decreasing completion time. We have evaluated and compared our algorithm with hadoop default schedulers by running concurrent workloads consisting of the WordCount and Terasort benchmarks. The results show that our proposed algorithm has increased the localization rate by 10.4% and 18.5% and the speed by 3.14% and 3.3% compared to the FIFO algorithm and the Fair algorithm respectively.

Keywords: MapReduce Scheduling, Hybrid Algorithm, Data Locality, Dynamic Priority, Hadoop Scheduling

* Corresponding Author Email: s.rezapakize@gmail.com