ارزیابی عملکرد زمان بندهای کار بر روی Hadoop YARN

چکیده

به منظور حل محدودیتHadoop در زمینه مقیاس پذیری، تسهیم منابع و پشتیبانی برنامه، انجمن متن باز، نسل جدید پلتفرم محاسبه هادوپ موسوم به YARN[1] با تفکیک دستورات مدیریت منابع از مدل برنامه نویسی را پیشنهاد کرده است.

این تفکیک، امکان اجرای موازی انواع برنامه‌های مختلف را بر روی YARN می‌دهد. برای دست یابی به تسهیم برابر و بهره گیری بالا از منابع، YARN، زمان بند ظرفیت و زمان بند عادلانه را در اختیار گذاشته است. با این حال، اثرات عملکرد دو زمان بند در زمانی که برنامه‌های ترکیبی بر روی یک کلاستر YARN اجرا می‌شود، مشخص نیست. از این روی در این مقاله ما چهار مجموعه سیاست زمان بندی را مطالعه می‌کنیم (مخففSPC) که از دو زمان بند مشتق شده و سپس به ارزیابی چهار SPC در سناریوهای گسترده می‌پردازیم که نه تنها چهار نوع برنامه، بلکه سه ساختار صف بندی متفاوت را برای سازمان دهی برنامه‌ها در نظر می‌گیرد. نتایج آزمایشی، مدیران YARN را قادر به درک اثرات SPC های مختلف و ساختارهای صف بندی مختلف بر روی برنامه‌های ترکیبی می‌کند. هم چنین این نتایج به آن‌ها در انتخاب یک SPC و ساختار صف بندی مطلوب جهت دست یابی به عملکرد اجرای برنامه بهتر کمک می‌کند.

1- مقدمه

هادوپ (1) یک چارچوب نرم افزاری متن باز می‌باشد که توسط آپاچی برای پردازش حجم بالای مجموعه داده‌ها بر روی یک کلاستر (خوشه) شامل تعداد زیادی از دستگاه‌ها پشتیبانی می‌شود. به دلیل سهولت، مقرون به صرفگی، مقیاس پذیری و تحمل پذیری خطا، طیف وسیعی از سازمان‌ها و شرکت‌ها نظیر گوگل، یاهو، فیسبوک و آمازون از هادوپ برای هم برای تحقیق و هم برای تولید استفاده کرده‌اند (2). با این حال هادوپ اصلی دارای چندین محدودیت می‌باشد (3). یک نمونه این است که تخصیص منابع مبتنی بر اسلات برای دستورات نقشه و دستورات کاهش موجب محدودیت منابع کلاستر هادوپ شده و منجر به بهره گیری کم‌تر از

[1] Yet Another Resource Negotiator

منابع می‌شود (3). یک مثال دیگر این اسن که هادوپ اصلی تنها یک نوع مدل برنامه نویسی را پشتیبانی می‌کند یعنی MapReduce(4) که برای پردازش همه نوع محاسبات بزرگ مقیاس (3-5-6) مناسب نیست.

به منظور حل این محدودیت‌ها، انجمن متن باز نسل جدیدی از پلتفرم محاسبه هادوپ موسوم به YARN(مخفف Yet Another Resource Negotiator)(3) را معرفی کرده است. نام‌های دیگر شامل MapReduce 2.0 وMRv2 می‌باشند.YARN به برنامه‌ها و نرم افزارها امکان بهره گیری از منابع یک کلاستر را به شکلی مشترک و چند مستأجری می‌دهد. YARN بر خلاف هادوپ اصلی (یعنی همه نسخه‌های قبل از MRv2) دستورات مدیریت منابع را از مدل برنامه نویسی جدا کرده و از این روی می‌تواند نه تنها از MapReduce بلکه از سایر مدل‌های برنامه نویسی نظیر Spark [5], Storm [7], Tez [8] وREEF [9] پشتیبانی کند. به عبارت دیگر، این تفکیک امکان اجرای انواع مختلفی از نرمافزار ها را به طور موازی بر روی YARN می‌دهد.

به منظور پشتیبانی از یک محیط محاسباتی مشترک،YARN دو زمان بند را برای زمان بندی منابع با نرم افزارها ارائه می‌کند. یکی زمان بند ظرفیت (زمان بند پیش فرض بر روی YARN)(10) و دیگری زمان بند عادلانه (11). هر دوی آن‌ها قادر به سازمان دهی ارسال برنامه به یک سلسله مراتب صف بندی می‌باشند. با این حال، اولی مقدار حداقل منابع را برای هر صف تضمین کرده و از FIFO(که مخفف به ترتیب ورود است) برای زمان بندی نرم افزارها در صف برگ استفاده می‌کند. مورد دوم، منابع را به طور عادلانه در میان همه صف‌ها تسهیم کرده و سه سیاست از جمله FIFO، Fair و عدالت منابع غالب (DRF) (12) را برای تسهیم منابع برای همه برنامه‌های در حال اجرا در یک صف ارائه می‌کند. همه رویکردهای زمان بندی فوق الذکر تشکیل چهار مجموعه سیاست زمان بندی زیر (SPC) را داده و انعطاف پذیری بالایی را برای مدیران YARN برای دست یابی به اهدافشان نظیر تسهیم منابع عادلانه و بهره گیری بالا از منابع در اختیار می‌گذارد.

1- Cap-FIFO که زمان بند ظرفیت با سیاست زمان بندی FIFO است

2- Fair-FIFO که زمان بند عادلانه با سیاست زمان بندی FIFO است

3- Fair-Fair که زمان بند عادلانه با سیاست زمان بندی عادلانه است

4- Fair-DRF که زمان بند عادلانه با سیاست زمان بندی DRF است

اگرچه YARN از چهار SPC و انواع متنوعی از برنامه‌ها پشتیبانی می‌کند با این حال هنوز مشخص نیست که این SPC ها وقتی که به طور جداگانه برای زمان بندی نرم افزارهای ترکیبی استفاده می‌شوند، چگونه کار می‌کنند. به علاوه، عملکرد آن‌ها زمانی نامشخص است که ساختارهای مختلف صف بندی استفاده می‌شود. از این روی، در این مقاله، ما به بررسی چهار SPC و همه مدل‌های برنامه نویسی پشتیبانی شده توسط YARN پرداخته همه برنامه‌ها را به چندین نوع طبقه بندی می‌کنیم. با این همه، ما آزمایشات گسترده‌ای را برای ارزیابی و مقایسه اثرات عملکرد چهار SPC بر روی متریک‌های متنوع با در نظر گرفتن نه تنها بارکار متشکل از انواع برنامه‌های مختلف، بلکه متشکل از سه سناریوی زیر انجام دادیم. هدف این مطالعه، بررسی این است که آیا ساختارهای صف بندی بر عملکرد چهار SPC تأثیر دارند یا خیر.

1- سناریوی تک صف: در این سناریو تنها یک صف در کلاستر YARN ما وجود دارد. از این روی همه نرم افزارها بایستی قبل از اجرا در این صف منتظر بمانند

2- سناریوی صف مجزا: در این سناریو، هر نوع از برنامه‌ها به طور جداگانه در یک صف مجزا قرارمی گیرند.

3- سناریوی صف ادغام شده: در این سناریوف دو نوع صف وجود دارند: یکی برای برنامه‌هایی که در نهایت توسط خودشان متوقف می‌شود. صف دیگر برای بقیه برنامه‌هاست. نتایج آزمایش نشان می‌دهد که 1- همه SPC ها از مسئله تقسیم منابع که در ادامه توضیح داده شده است رنج می‌برند. این مسئله موجب می‌شود تا هیچ یک از SPC ها نتوانند به طور موفق، بارکار متشکل از برنامه‌های ترکیبی را کامل کنند 2- هیچ یک از چهار SPC معمولاً، بهترین عملکرد اجرایی برنامه در همه سناریوها نبوده است و 3- در میان سه سناریو، استفاده از سناریوی صف ادغام شده، مناسب‌ترین سناریو برای همه SPC ها می‌باشد زیرا آن‌ها می‌توانند به نرخ تکمیل بارکار بالاتر و زمان بارکار کوتاه‌تر از دو سناریوی دیگر برسند. این مقاله دو اهمیت اصلی را دارد. 1- این مقاله یک مرور جامعی را بر روی زمان بندهای فعلی،SPC ها، مدل‌های برنامه نویسی و انواع برنامه‌های پشتیبانی شده توسط YARN دارد 2- ما به طور گسترده اقدام به ارزیابی و مقایسه چهار SPC با در نظر گرفتن نه تنها انواع برنامه‌های ترکیبی، بلکه سناریوهای متنوع ساختار صف می‌کنیم و 3- بر اساس نتایج ازمایشی، مدیران YARN می‌توانند یک ساختار صف و SPC مناسب را برای دست یابی به عملکرد برنامه بهتر برای کلاسترهای YARN انتخاب کنند.

ادامه این مقاله به صورت زیر سازمان دهی شده است: بخش دوم به توصیف کارهای مرتبط می‌پردازد. بخش سوم به بررسی منشأ YARN خواهد پرداخت. بخش چهارم دو زمان بند پشتیبانی شده توسط YARN و چهار SPC مشتق شده از دو زمان بند را معرفی می‌کند. بخش پنجم مدل‌های برنامه نویسی پشتیبانی شده توسط YARN و برنامه‌هایی که هر مدل برنامه نویسی می‌تواند بیان و پردازش کند را توصیف می‌کند. در بخش ششم، آزمایشات گسترده‌ای انجام شده و نتایج آزمایش بحث می‌شوند. بخش هفتم بخش نتیجه گیری این مقاله بوده و کارهای آینده ما را شامل می‌شود.

2-کارهای مرتبط

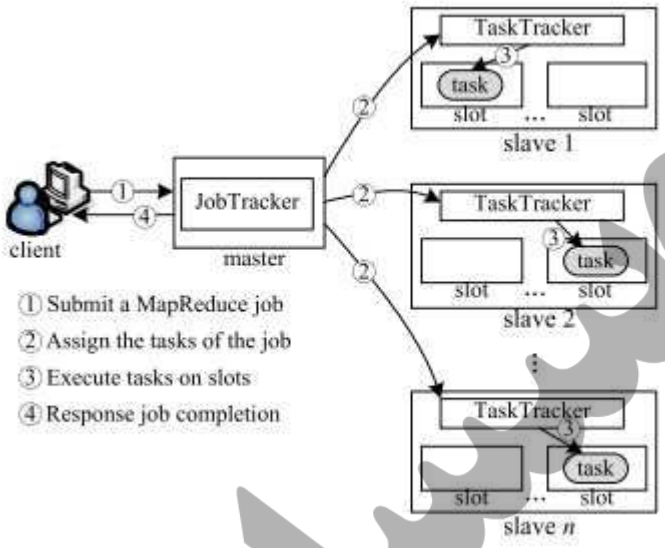
چندین مقاله پیمایشی در زمینه زمان بندی کار در هادوپ وجود دارد. رائو و ردی (13) به مطالعه زمان بندهای مختلف هادوپ از جمله زمان بند FIFO پیشفرض (4)، زمان بند عادلانه، زمان بند ظرفیت و زمان بندی تأخیر (14) با خلاصه سازی مزایا و معایب این زمان بندها پرداخته‌اند. با این حال، محققان تنها زمان بندها را بدون انجام ازمایشاتی برای ارزیابی و مقایسه عملکرد آن‌ها پیشنهاد داده‌اند. کالکارنی و خانداول (15) نیز اقدام به بررسی الگوریتم‌های زمان بند کار در هادوپ کردند. با این حال مشابه با (13)، هیچ ارزیابی عملکردی در (15) ارائه نشده است. مسئله دیگر را می‌توان در (16) یافت. به منظور بهبود زمان بندی هادوپ از حیث زمان تکمیل کار، موقعیت داده‌ها و یا سایر متریک‌های عملکرد، بسیاری از محققان اقدام به معرفی الگوریتم‌های زمان بندی برای هادوپ کرده و ازمایشاتی را برای مقایسه الگوریتم‌های آن‌ها با الگوریتم‌های مورد استفاده توسط هادوپ نظیر FIFO، زمان بند ظرفیت و زمان بند عادلانه انجام دادند. برای مثال، زاهاریا و همکاران (14) الگوریتم زمان بندی تأخیر را برای بهبود موقعیت داده‌ها ضمن حفظ برابری پیشنهاد کردند. محققان الگوریتم خود را با زمان بند FIFO و عادلانه ارزیابی کرده و اثبات کرده‌اند که الگوریتم آن‌ها عملکرد بهتری از سایر الگوریتم‌ها از حیث موقعیت داده‌ها و زمان پاسخ کار دارد. زمان بند زمینه آگاه پیشنهاد شده توسط کومار و همکاران (17) و زمان بند توان عملیاتی پیشنهادی توسط گوپتا و همکاران (18) دو نمونه برای بهبود عملکرد بر روی کلاسترهای ناهمگن هادوپ می‌باشند. هر دوی آن‌ها برای تخصیص وظایف به توانمندترین گره‌ها طراحی شده‌اند به طوری که نیاز این دستورات و وظایف به منابع براورده شود. آن هم چنین اقدام به ارزیابی زمان بندهای خود با زمان بندهای مورد استفاده توسط هادوپ کرده‌اند. با این حال، ارزیابی در (17) بر اساس یک شبیه سازی است تا یک

ازمایش واقعی انجام شده در (18). لی و همکاران (19) محل داده‌ها را برای کارهای نقشه بندی و کاهش، اجتناب از قحطی کار و بهبود عملکرد اجرای کار با معرفی JoSS بهبود بخشیدند (که کوتاه شده عبارت طرح زمان بندی کار محور هیبریدی می‌باشد). دو نسخه JoSS برای دست یابی مجزا به محل داده‌ها و تسریع تخصیص وظایف معرفی شده‌اند. محققان آزمایشات گسترده‌ای را برای ارزیابی و مقایسه دو نسخه با الگوریتم‌های زمان بندی فعلی پشتیبانی شده توسط هادوپ انجام داده‌اند. بر خلاف همه مطالعات فوق الذکر، در این مقاله ما بر مطالعه اثرات عملکردی مجموعه سیاست‌های زمان بندی مختلف پشتیبانی شده توسط YARN بر روی برنامه‌های ترکیبی تاکید می‌کنیم.

مطالعات دیگر برای مطالعه عملکرد هادوپ از دیدگاه‌های مختلف ارائه شده‌اند. گو و لی (20) اقدام به ارزیابی عملکرد هادوپ و اسپارک از حیث هزینه حافظه و زمان در هنگام اجرای عملیات تکراری کردند. نتایج نشان داد که اسپارک سریع‌تر از هادوپ است با این حال حافظه بیشتری از هادوپ را مصرف می‌کند. از این روی، اگر حافظه در یک لحظه ناکافی باشد، مزیت سرعت اسپارک کاهش می‌یابد. خاویر و همکاران (21) مقایسه عملکردی بین سیستم‌های مبتنی بر حافظه فعلی ازجمله Linux VServer, OpenVZ, Linux و Containers (LXC), را برای کلاسترهای MapReduce انجام دادند. لین و همکاران (22) اقدام به مطالعه اثر سیاست‌های مختلف MapReduce بر روی اطمینان پذیری تکمیل کار و مصرف انرژی کار کردند. تا آنجا که ما می دانیم، مطالعه ارائه شده در این مقاله، اولین مطالعه‌ای است که به طور جامع به بررسی اثر مجموعه سیاست‌های زمان بندی فعلی پشتیبانی شده توسط YARN بر روی انواع مختلف برنامه‌ها پرداخته و در آن ساختارهای مختلف صف بندی نیز در نظر گرفته می‌شود.

3-منشأ YARN

در این بخش، ما به طور مختصر به توصیف هادوپ اصلی و محدودیت‌های آن پرداخته و سپس نشان می‌دهیم که چگونه YARN می‌تواند این محدودیت‌ها را حل کند



شکل1

3-1 هادوپ

هادوپ (1) عمدتاً متشکل از دو مؤلفه کلیدی است: سیستم فایل توزیعی هادوپ (HDFS) و MapReduce. مورد اول برای ذخیره فایل‌های بزرگ در دستگاه‌ها در یک خوشه بزرگ با تقسیم هر فایل به چندین بلوک طراحی شده و هر بلوک را به ماشین‌های مختلف تکرار می‌کند. مورد دوم، یک مدل برنامه نویسی توزیع شده برای کاربران برای تعیین کارهای آن‌ها به صورت دو دستور اولیه یعنی MapوReduce بدون نیاز به مدیریت منابع، زمان بندی کار و تحمل پذیری خطا است (4). شکل 1 جریان اجرایی کار MapReduce را بر روی هادوپ نشان می‌دهد. اولاً، کلاینت یک کار را به JobTracker ارسال می‌کند که یک سرور اصلی مسئول هماهنگ سازی و زمان بندی اجرای همه کارهاست. سپس، JobTracker هر یک از وظایف مربوط به کار را برای گره فرعی موجود موسوم به TaskTracker زمان بندی می‌کند. هر TaskTracker تعداد ثابتی از اسلات های نقشه اسلات های ردیوس را به ترتیب برای اجرای وظایف مربوط به نقشه و ردیوس تعیین شده توسط JobTracker ارائه می‌کند. در طی اجرای شغل،JobTracker نه تنها پیشرفت کار را پایش می‌کند، بلکه تحمل پذیری خطرا را برای هر کار ناقص در اختیار می‌گذارد. وقتی که همه وظایف مربوط به کار کامل شد، JobTracker به کلاینت در مورد تکمیل کار اطلاع رسانی می‌کند. طراحی هادوپ منجر به چندین محدودیت در زمینه دسترسی، مقیاس پذیری، استفاده از منابع و پشتیبانی از برنامه می‌شود (3). اولاً، JobTracker یک تک نقطه خرابی اسن. در صورتی که خراب شود، همه کارها متوقف شده و بایستی مجدداً آغاز شوند. دوم،

هادوپ تنها از یک نوع مدل برنامه نویسی یعنی MapReduce پشتیبانی می‌کند. اگرچه MapReduce قادر به بیان و پردازش برنامه‌های بسیاری است، با این حال برای برنامه‌های تکراری، برنامه‌های استریمینگ، برنامه‌های داده کاوی تعاملی و برنامه‌های گراف نامناسب می‌باشد (3). سوما، محدودیت یک اسلات برای اجرای یک نوع کار (یعنی کار Map یا کار Reduce) موجب می‌شود تا بهره گیری از خوشه پایین باشد زیرا اسلات نقشه می‌تواند به طور کامل استفاده شود ضمن این که اسلات های کاهش خالی باشند (و برعکس)

YARN 2-3

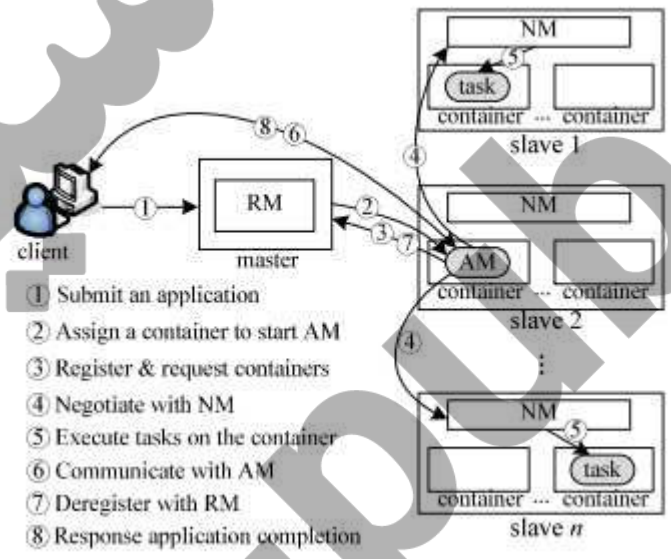
برای حل محدودیت‌های فوق، YARN دستورات مدیریت منبع را از مدل برنامه نویسی جدا کرده و مؤلفه‌ها و نقش‌های زیر را معرفی می‌کند:

- مدیر منابع جهانی (مخفف RM): این به عنوان یک مرجع مرکزی در کلاستر YARN عمل کرده و بر مسیر یابی استفاده از منابع و تخصیص منابع موجود به برنامه‌ها بر اساس نیازهای برنامه‌ای منابع تاکید می‌کند. بر خلاف JobTracker، RM قادر به پایش وضعیت برنامه‌ها نبوده و کارهای ناقص را دو باره اجرا می‌کند. این تقسیم مسئولیت، RM را قادر به بهبود مقیاس پذیری YARN می‌کند.
- مدیر گره فرعی (مخفف NM): این یک عامل در گره فرعی برای گزارش سلامت گره به RM، گرفتن دستور العمل از RM، مدیریت حافظه بر روی گره، اجرای کانتینرها برای برنامه‌ها و پایش استفاده از منابع تک تک کانتینرها می‌باشد.
- برنامه مستر (AM): این هد یک کار بوده و کانتینرهایی را از RM درخواست کرده و با NM برای اجرا و مدیریت جریان اجرای کار، کار می‌کند.
- کانتینر برنامه: این شامل یک مجموعه منطقی از برنامه‌ها (برای مثال 1 گیگ حافظه و 1 سی پی یو) بر روی گره برده (3) است. بر خلاف اسلات های نقشه و ردیوس مورد استفاده در هادوپ اصلی، یک کانتینر می‌تواند هر کار دیگری را اجرا کند. این خود به YARN امکان تخصیص منابع منابع را به برنامه‌ها و بهبود استفاده از منابع می‌دهد.

با پیشرفت‌های فوق الذکر، YARN از مدل‌های برنامه نویسی متنوع پشتیبانی کرده و امکان استفاده از برنامه‌های مختلف را بر روی YARN می‌دهد. شکل 2 جریان اجرای یک برنامه را بر روی YARN نشان می‌دهد.

در مرحله 1، یک کلاینت، برنامه را به RM ارسال می‌کند. سپس RM در مرحله 2 با یک کانتینر مشخص برای اجرای AM برنامه بر روی کانتینر مذاکره می‌کند. پس از شروع AM، این با Rm ثبت شده و درخواست کانتینر از RM(به مرحله 3 مراجعه کنید) می‌کند. شکل 1 جریان اجرای کار MapReduce را بر روی هادوپ نشان می‌دهد.

AM با دریافت یک کانتینر از RM در مرحله 4 و 5 امکان اجرای NM و گره برنامه را بر روی کانتینر می‌دهد. در طی اجرای برنامه، کلاینت می‌تواند مستقیماً با AM برای آگاهی از میزان پیشرفت فعلی و وضعیت فعلی ارتباط برقرار کند (به مرحله 6 مراجعه شود). وقتی که برنامه کامل شد،AM، با RM از حالت ثبت خارج شده و همه کانتینرهای مورد استفاده در مرحله 7 را آزاد می‌کند. در نهایت، AM به مشتری در مورد تکمیل کار اطلاع رسانی می‌کند.



شکل2

4-زمان بندهای کار پشتیبانی شده با YARN

در این بخش، ما به توصیف مفاهیم اصلی زمان بند ظرفیت و زمان بند عادلانه پرداخته و سپس چهار SPC مشتق شده از دو زمان بند را معرفی می‌کنیم.

4-1 زمان بند ظرفیت

زمان بند ظرفیت (10) برای مستاجران مختلف برای تسهیم یک خوشه بزرگ طراحی شده است که برنامه‌های آن‌ها می‌توانند منابع تخصیص داده شده باشند. زمان بند ظرفیت از صف‌های سلسله مراتبی برای منعکس کردن

ساختار سازمان‌ها و گروه‌هایی که از منابع خوشه‌ای استفاده می‌کنند پشتیبانی می‌کند. به طور کلی، یک سلسله مراتب صف بندی شامل سه نوع صف است: ریشه، مادر، برگ. تنها صف‌های برگ درخواست را می‌پذیرند. در حقیقت، صف ریشه بیانگر خود خوشه است در حالی که صف مادر بیانگر هر سازمان یا گروه و یا زیر سازمان یا زیر گروه می‌باشد.

زمان بند ظرفیت، ضمانت ظرفیت را با تخصیص کسری از منابع خوشه برای هر صف ارائه می‌کند. مدیران YARN، نیز قادر به محدودیت ظرفیت حداکثر برای هر صف می‌باشند. برای مثال، اگر تخصیص ظرفیت حداقل یا حداکثر یک صف به ترتیب 40 و 60 درصد باشد، این بدین معنی است که صف می‌تواند حداقل 40 درصد و حداکثر 60 درصد منابع را استفاده کند. برای ایجاد انعطاف پذیری، زمان بند ظرفیت می‌تواند امکان استفاده از منابع بیشتر را نسبت به تخصیص ظرفیت بدهد به خصوص اگر ظرفیت به سایر صف‌هایی که هنوز استفاده نشده‌اند تخصیص داده شده باشد.

وقتی که منبع کلاستر موجود باشد، زمان بند ظرفیت به صورت زیر کار می‌کند:

1- مرحله 1: ابتدا ظرفیت فعلی مورد استفاده توسط هر صف برگ را محاسبه می‌کند یعنی مقدار کل منابع مورد استفاده توسط همه برنامه‌ها در هر صف برگ. سپس زمان بند محروم‌ترین صف را انتخاب می‌کند یعنی صف با کم‌ترین ظرفیت مورد استفاده در میان همه صف‌های برگ.

2- زمان بند یک برنامه را از محروم‌ترین صف به ترتیب FIFO(اولین ورودی و اولین خروجی) انتخاب می‌کند یعنی برنامه‌ای که اولین بار داده می‌شود، اولین منابع را نیز شامل می‌شود

3- با انتخاب برنامه، منابع به کار برنامه بر اساس اولویت‌های درخواست منابع تعیین شده توسط برنامه زمان بندی می‌شوند.

4- ش کل 2: جریان اجرای برنامه بر روی YARN

در این مقاله، ما از زمان بندی بین صف برای نشان دادن فرایند انتخاب صف برگ از همه صف‌های برگ و از اصطلاح زمان بندی درون صف برای نشان دادن فرایند انتخاب یک برنامه از یک صف برگ استفاده می‌کنیم. از این روی می‌توان دید که زمان بند صف تنها یک SPC یعنی Cap-FIFO را ارائه می‌کند.

2-4 زمان بند عادلانه

هدف زمان بند عادلانه (11) تخصیص منابع به برنامه‌ها می‌باشد به طوری که این برنامه‌ها به مرور زمان منابع عادلانه‌ای را کسب کنند. مشابه با زمان بند ظرفیت، زمان بند عادلانه از صف‌های سلسله مراتبی برای منعکس کردن ساختار یک گروه یا سازمان پشتیبانی کرده و به هر صف امکان دریافت حداقل ظرفیت تضمین شده را داده و ظرفیت حداکثر را برای هر صف محدود می‌کند. با این حال بر خلاف زمان بند ظرفیت، زمان بند عادلانه سه سیاست را برای مدیران YARN برای تسهیم انعطاف پذیر منابع به برنامه‌های درون صف در نظر می‌گیرد:

1- FIFO: وقتی که این به صف برگ اعمال می شودف منابع موجود به برنامه‌ای که ابتدا به صف وارد می‌شود تخصیص داده می‌شود

2- FAIR: وقتی که این به صف برگ اعمال می‌شود، منابع موجود به برنامه‌ای تخصیص داده می‌شوند که اکنون از حداقل مقدار حافظه در میان همه برنامه‌های موجود در صف استفاده می‌کند.

3- برابری منابع غالب: DRF یک مدل تسهیم منبع عادلانه می‌باشد که توسط قدوسی و همکاران (12) برای تعمیم برابری و عدالت حداقل- حداکثر به انواع منابع چندگانه معرفی شده است. برای هر کاربر، DRF قادر به محاسبه تسهیم هر منبع تخصیص داده شده به کاربر بوده و حداقل مقدار را در میان سهام‌های مختلف به صورت سهم غالب کاربر در نظر می‌گیرد. منابع متناظر با هر منبع غالب، منبع غالب کاربر است. برای مثال، کاربر U، کم‌تر از 1 سی پی یو، ب 5000 مگابایت و ظرفیت کل خوشه <4 CPUs, 8000MB>, را به خود اختصاص داده است. این بدین معنی است که به ترتیب سهم CPU فعلی و سهم حافظه یک چهارم و پنج هشتم است. از این روی سهم غالب U برابر با پنج هشتم و منبع غالب U حافظه است. هدف DRF برابر سازی سهام غالب همه کاربران است. وقتی که DRF به یک صف برگ اعمال شودف منابع موجود به ترجیحاً به برنامه‌ای تخصیص داده می‌شوند که دارای کم‌ترین سهم غالب در صف است.

با توجه به سیاست‌های زمان بندی فوق، زمان بند عادلانه سه SPC را ارائه می‌کند یعنی Fair-FIFO, Fair-Fair, Fair-DRF و . وقتی که منبع خوشه موجود باشد، زمان بند عادلانه به صورت زیر کار می‌کند.

مرحله 1: زمان بند عادلانه یک صف برگ را بر اساس مجموعه سیاست‌های زمان بندی برای هر سطح از سلسله مراتب صف انتخاب می‌کند. در ابتدا، یک صف فرعی را از صف ریشه موسوم به صف X بر اساس سیاست زمان بندی طراحی شده انتخاب می‌کند. سپس، همین روش را تا زمان رسیدن به صف برگ تکرار می‌کند

مرحله 2: زمان بند یک برنامه را از صف برگ انتخاب شده بر اساس مجموعه سیاست زمان بندی برای هر صف برگ انتخاب می‌کند

مرحله 3: با انتخاب برنامه، منبع به برنامه بر اساس اولویت‌های درخواست منابع تعیین شده با برنامه زمان بندی می‌شود.

5-مدل‌های برنامه نویسی و انواع برنامه‌ها

در این بخش، ما مدل‌های برنامه نویسی پشتیبانی شده توسط YARN را معرفی کرده و برنامه‌های اجرا شده بر روی YARN را بر اساس ویژگی‌های آن‌ها طبقه بندی می‌کنیم.

5-1 مدل‌های برنامه نویسی

5-1-1 MapReduce. MapReduce [4] به برنامه نویس امکان بیان محاسبه خود را به صورت یک دستور نقشه و یا دستور ردیوس می‌دهد. اولی از یک جفت کلید/مقدار استفاده کرده و تولید جفت کلید/مقدار متوسط می‌کند. دومی، همه جفت کلید مقدارها را با کلید مشابه ترکیب کرده و تولید نتایج نهایی می‌کند. به دلیل دو دستور، اجرای یک کار MapReduce شامل مرحله نقشه و مرحله ردیوس است. در مرحله نقشه، هر نقشه دستور تعریف شده توسط کاربر را برای پردازش بلوک داده‌های ورودی اجرا کرده و تولید داده‌های کلید/ مقدار متوسط می‌کند. در مرحله ردیوس هر کار ردیوس، دستور ردیوس تعریف شده توسط کاربر را برای پردازش داده‌های مقدار کلید میانی اجرا کرده و نتیجه نهایی را تولید می‌کند. بدیهی است که MapReduce برای برنامه‌های بچ (23) نظیر تحلیل لگاریتمی و پردازش متن مناسب است.

5-1-2 Apache Tez: Apache Tez برای تعمیم پارادایم MapReduce طراحی شده است. با مدل سازی پردازش داده‌ها به صورت یک گراف جهت دار غیر مدور (DAG) و رئوسی که نشان دهنده منطق برنامه است و یال‌هایی که نشان دهنده حرکت داده‌هاست، Apache Tez به کاربران امکان بیان کارهای پیچیده پردازش داده‌ها را می‌دهد. وقتی که کار Tez اجرا می‌شود، در رئوس ریشه‌ای DAG شروع شده و به سمت یال‌های همبند تا زمان رسیدن به رئوس برگ ادامه می‌یابد. تنها وقتی که همه رئوس در DAG کامل شدند، کار کامل شده است.

5-1-3 اسپارک: اسپارک یک چارچوب محاسباتی متن باز می‌باشد که برای پشتیبانی از برنامه‌هایی توسعه یافته است که به طور کارامد توسط MapReduce قابل پردازش نیست یعنی برنامه‌هایی که از یک مجموعه از داده‌ها در برنامه‌های موازی مختلف استفاده می‌کنند. مثال‌ها شامل برنامه‌های یادگیری ماشینی تکراری و برنامه‌های تحلیل داده تعاملی می‌باشند. اسپارک از یک انتزاع موسوم به مجموعه داده‌های توزیعی انعطاف پذیر استفاده می‌کند که یک مجموعه از اشیای تفکیک شده در میان گره‌ها و ماشین‌های مختلف است. کاربران می‌توانند RDD را در حافظه در کارکنان مختلف اسپارک ذخیره کرده و با استفاده از عملیات موازی به جای بازیابی آن از HDFS مجدداً استفاده کنند. با موتور پیشرفته DAG اسپارک، یک برنامه اسپارک می‌تواند مراحل زیادی داشته باشد. به علاوه، اسپارک، استریمینگ اسپارک (24) و GraphX(25) را ارائه می‌کند. اولی به کاربر امکان پردازش دیتا استریم های زنده را در یک حالت با توان عملیاتی بالا و متحمل به خطا می‌دهد، در حالی که دومی، کاربر را قادر به محاسبه موازی گراف بزرگ مقیاس می‌کند.

5-1-4 استورم: استورم (7) یک سیستم رایانشی توزیعی متن باز برای پردازش استریم های بزرگ داده‌ها در زمان واقعی است. در استورم، استریم یک توالی نامحدود از تاپل هاست. هر تاپل یک لیست منظم از عناصر است. برای مثال (3-2-5) سه تاپل است. هر برنامه استورم به صورت توپولوژی برای پردازش استریم های داده‌های ورودی تعریف می‌شود. یک توپولوژی یک گراف جهت دار با مجموعه‌ای از رئوس و یال‌هاست. رئوس می‌تواند به صورت دهانه‌ای یا لکه‌ای باشد. دهانه تنها تاپل را از منبع خارجی خوانده و آن را به توپولوژی وارد می‌کند. یک بلات، استریم های ورودی را پردازش کرده و تولید استریم های خروجی می‌کند. یک توپولوژی استورم در نهایت به خودی خود به پایان نمی‌رسد. در عوض، پردازش استریم های ورودی را تا زمان کشته شدن ادامه می‌دهد.

5-2 انواع برنامه‌ها

بر اساس بخش 5-1، ما مشاهده کردیم که YARN از برنامه‌های مختلف پشتیبانی کرده و این برنامه‌ها را می‌توان به چهار نوع طبقه بندی کرد:

1- دو مرحله‌ای: این نوع اشاره به همه برنامه‌های بیان شده توسط MapReduce دارد

2- DAG: این نوع اشاره به همه برنامه‌هایی دارد که به صورت DAG صرف نظر از ساختار و تعداد مراحل، رئوس و یال‌های آن بیان می‌شود

3- گراف مدور جهت دار (DCG): این نوع اشاره به همه محاسبات موازی گراف به جز برنامه‌های مربوط به نوع DAG دارد.

4- استریمینگ: این نوع شامل همه برنامه‌ها برای پردازش استریم داده‌هاست.

6-ارزیابی عملکرد و مقایسه

در این بخش ما اقدام به ارزیابی و تحلیل عملکردهای چهار SPC فوق الذکر (یعنی Cap FIFO, Fair-Fair, Fair-FIFOوFair-DRF) می‌پردازیم. برای این منظور، یک کلاستر YARN واقعی با اجاره 31 سرور خصوصی مجازی (VPS) از لینود (26) ایجاد کردیم که یک عرضه کننده سرور خصوصی مجازی است که در نیوجرسی واقع می‌باشد. همه VPS ها در یک مرکز داده در دالاس امریکا واقع شده‌اند. یک VPS به عنوان RM عمل کرده و سایر Vps ها به عنوان گره‌های فرعی عمل می‌کند. هر Ubuntu 12.04.3 LTS VPS را با 2 CPU Cores, 2GB RAM، حافظه 48GB SSD، 3TB Transfer, 40 Gbps Network In و 250 Mbps Network Out اجرا می‌کند. از این روی، کل ظرفیت CPU و ظرفیت حافظه کل خوشه YARN به ترتیب 60 هسته سی پی یو و 60 گیگ بود. همه ازمایشات بر روی هادوپ 2.2(28) با اسپارک 1.0.2(29) انجام شدند. جدول 1 فهرستی از همه شرایط پارامتر را در ازمایشات نشان می‌دهد. سایر پارامترهای ذکر نشده از تنظیمات پیش فرض YARN تبعیت می‌کنند (28).

بدون از دست دادن تعمیم، ما یک بارکار ترکیبی را برای ارزیابی چهار SPC ایجاد کردیم. جدول 2 خلاصه‌ای از جزییات بارکار را نشان می‌دهد. توجه کنید که تعداد هر نوع از برنامه (به جز نوع استریمینگ) و ترتیب ارسال همه برنامه‌ها در بارکار به طور تصادفی تعیین می‌شود. تعداد کل برنامه‌ها 94 مورد است که شامل 37 برنامه دو مرحله‌ای، 28 برنامه DAG، 28 برنامه DCG و یک برنامه استریمینگ است. معیارهای برنامه‌های دو مرحله‌ای مربوط به (30) بوده و معیارهای سایر انواع برنامه‌های اجرا شده در حالت کلاینت YARN را می‌توان در (31) دید. اگرچه تنها یک برنامه استریمینگ در بارکار وجود دارد، ویژگی اجرایی پیوسته، مقدار خاصی از منابع را مصرف می‌کند که در ادامه نشان داده شده است. برنامه استریمینگ، اولین کار در بارکار بوده و استریم های پردازش شده آن تقریباً هر 5 ثانیه تولید می‌شود. رسیدن سایر درخواست‌ها از یک توزیع پواسون با بازه متوسط 31.11 ثانیه و انحراف معیار 27 و 63 ثانیه پیروی می‌کند. صرف نظر از هر نوع درخواست، هریک از این‌ها

نیازمند یک کانتینر برای اجرای AM می‌باشد. هر برنامه دو مرحله‌ای نیازمند کانتینرهای با اندازه 128 مگابایت برای اجرای وظایف آن بوده با این حال هر یک از سایر انواع برنامه‌ها نیازمند دو کانتینر می‌باشند زیرا هر یک از آن‌ها به دو بخش تقسیم می‌شوند. جدول 3 ملزومات منابع کانتینر را برای هر نوع درخواست بارکار ترکیبی نشان می‌دهد. همان طور که در مقدمه گفته شد، ما سه سناریوی زیر را برای ارزیابی هر SPC در نظر می‌گیریم. هدف اصلی تعیین مناسب‌ترین SPC برای هر ساختار صف و یافتن مناسب‌ترین ساختار صف برای برنامه‌های ترکیبی است.

1- سناریوی تک صف: در این سناریو، خوشه YARN دارای تنها یک صف برگ است و این نشان می‌دهد که همه برنامه‌ها در بارکار ترکیبی وارد این صف شده و منتظر اجرا می‌شوند. این هم چنین به این معنی است که این صف می‌تواند از منابع کل خوشه استفاده کند.

Parameter	Value
yarn.scheduler.minimum-allocation-mb (i.e., the minimum memory allocation for every container request at RM.)	1024 MB
yarn.scheduler.maximum-allocation-mb (i.e., the maximum memory allocation for every container request at RM.)	2048 MB
yarn.scheduler.minimum-allocation-vcores (i.e., the minimum virtual-CPU-core allocation for every container request.)	1
yarn.scheduler.maximum-allocation-vcores (i.e., the maximum virtual-CPU-core allocation for every container request.)	2
yarn.nodemanager.resource.cpu-vcores (i.e., number of vCores that can be allocated by a node for containers.)	2
yarn.nodemanager.resource.memory-mb (i.e., amount of memory that can be allocated by a node for containers.)	2048 MB

جدول 1: پارامتر گذاری خوشه YARN

2-سناریوی صف مجزا: در این سناریو، خوشه YARN ما دارای چهار صف برگ است. هر صف دارای یک نوع متفاوتی از برنامه می‌باشد. از این روی برنامه‌های مربوط به یک گروه در صف برنامه قرار می‌گیرد. حداقل و حداکثر ظرفیت هر صف به ترتیب 25 و 30 درصد منابع خوشه‌ای است.

3- سناریوی صف ادغام شده: دو صف برگ در این سناریو قرار دارد. یک صف، برای برنامه‌های استریمینگ با حداقل ظرفیت 20 درصد و حداکثر ظرفیت 30 درصد می‌باشد. صف دیگر برای قرار دادن سایر برنامه‌ها استفاده می‌شود. حداقل و حداکثر ظرفیت آن به ترتیب 80 و 90 درصد می‌باشد. به علاوه، برای ارزیابی و مقایسه جامع چهار SPC، شش متریک زیر استفاده می‌شود:

1- نرخ تکمیل بار کار: این درصد بارکاری را نشان می‌دهد که می‌تواند به طور موفق تکمیل شود. توجه کنید که در این مقاله اگر یک برنامه به طور موفق به پایان برسد، این برنامه را می‌توان به صورت کامل در نظر گرفت. در غیر این صورت به صورت ناموفق در نظر گرفته می‌شود. به علاوهف اگر برنامه استریمینگ بتواند در طی اجرای بارکار کامل به طور پیوسته باشد، آن را می‌توان کامل در نظر گرفت.

2- نرخ تکمیل بارکار تجمعی: این متریک نرخ تکمیل بارکار تجمعی را در طی اجرای بارکار نشان می‌دهد.

3- زمان فعال سازی بارکار: این دوره زمانی است که از اولین درخواست کاربار به خوشه YARN شروع شده و زمانی که اجرای بارکار کامل به پایان می‌رسد خاتمه می‌یابد (به جز برنامه استریمینگ).

4- بار سیستم میانگین: این تعداد متوسط کانتینرهای اجرا شده توسط کلاستر YARN را در طی اجرای بارکار نشان می‌دهد.

5- توان عملیاتی استریمینگ: این مقدار استریم داده‌هایی است که خوشه YARN می‌تواند در هر دقیقه آن‌ها را پردازش کند.

6- تأخیر کل: این مقدار زمان مورد نیاز توسط خوشه YARN برای زمان بندی و پردازش یک استریم از داده‌ها می‌باشد. برای دست یابی به یک مقایسه عملکرد منصفانه، هر یک از چهار SPC به طور دقیق تست شده و به مدت 5 بار صرف نظر از نوع سناریوی مورد استفاده ارزیابی می‌شود.

Application type	Number	Benchmark description	Note
Two-stage	37	5 wordcount applications 3 sort applications 8 grep applications 6 wordmean applications 15 wordstandarddeviation applications	Data size: 1 GB: 64.86%; 5 GB: 29.73%; 10 GB: 5.40%
DAG	28	9 JavaHdfsLR applications 9 JavaKMeans applications 10 JavaPageRank applications	192.5 KB of input size 17.31 MB of input size 14.83 MB of input size
DCG	28	28 LiveJournalPageRank applications	32 bytes of input size
Streaming	1	1 JavaQueueStream application	Data stream interval: 5 s

جدول2

6-1 سناریوی تک صفی

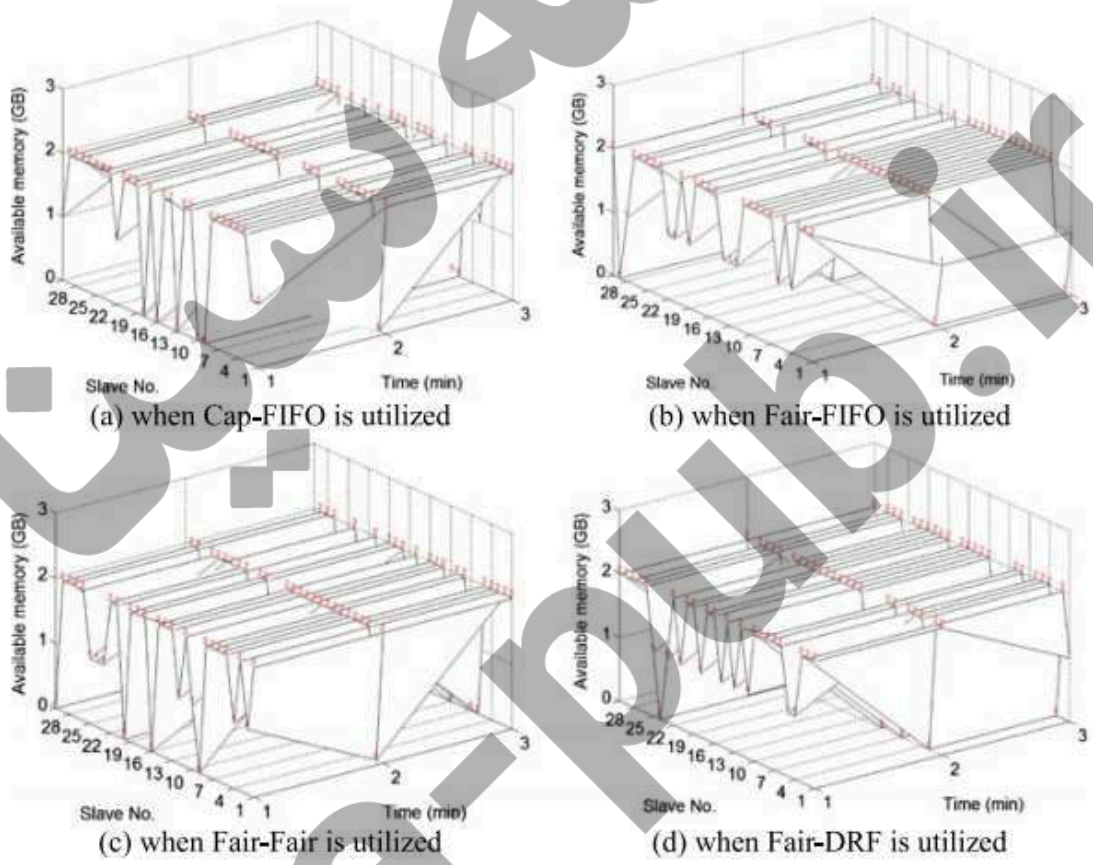
در این زیر بخش، ما عملکردهای اجرایی چهار SPC را در سناریوی تک صفی نشان می‌دهیم. شکل 3 نشان می‌دهد که وقتی چهار SPC به طور جدا و انفرادی برای اجرای بار کار استفاده می‌شوند برخی از درخواست‌های بارکار به دلیل عدم دست یابی به کانتینرها، نمی‌توانند کامل شوند. هیچ یک از آن‌ها به نرخ تکمیل 100 درصد نمی‌رسند. دلیل این است که تخصیص منابع مبتنی بر کانتینر مورد استفاده توسط YARN موجب می‌شود تا هیچ یک از گره‌های فرعی در یک لحظه دارای منابع کافی برای اجرای یک کانتینر مطلوب برای هر برنامه نباشند. این موسوم به مسئله تقسیم منابع است. در این ازمایش، برخی از کانتینرها درخواست حافظه 1024 مگابایت می‌کنند و برخی دیگر درخواست حافظه 2048 مگابایت می‌کنند. از این روی، اگر برنامه نیازمند یک کانتینر با 2048 مگابایت حافظه باشد ولی هیچ گره فرعی نتواند آن را براورده کند، این برنامه را نمی‌توان اجرا کرد.

اگرچه همه SPC ها از مسئله تقسیم منابع رنج می‌برند، Cap-FIFO بالاترین نرخ تکمیل بارکار (حدود 98.09 درصد) را ارائه می‌کند. دلیل این است که Cap-FIFO کانتینر جدید را از یک گره فرعی مادامی که منابع باقی مانده گره برای ایجاد کانتینر کافی باشد اجرا می‌کند. این ویژگی را می‌توان با مقایسه شکل 4 الف با شکل‌های 4 ب-ت مشاهده کرد. شکل 4 الف نشان می‌دهد که وقتی Cap-FIFO تست شود، پنج گره فرعی هیچ حافظه‌ای را نداشته و چهار گره دارای حافظه یک گیگ در اولین دقیقه از اجرای بارکار می‌باشد. با این حال، شکل 4 ب نشان می‌دهد که وقتی سه SPC دیگر تست شوند، بیش از چهار گره دارای حافظه یک گیگ در اولین دقیقه است به این معنی که این گره‌های فرعی قادر به ایجاد یک جافظه برای سایر برنامه‌هایی که نیازمند حافظه 2 گیگ می‌باشند نمی‌باشد. بر اساس نتایج فوق، می‌توان دید که شیوه اجرای کانتینر توسط Cap-FIFO ارام می‌باشد و این موجب حل مسئله تقسیم منابع می‌شود.

Application type	Container resource requirement for AM	Container resource requirement for each task
Two-stage	vCore: 1, Memory: 2048 MB	vCore: 1, memory: 1024 MB
DAG	vCore: 1, Memory: 1024 MB	vCore: 1, memory: 2048 MB
DCG	vCore: 1, Memory: 1024 MB	vCore: 1, memory: 2048 MB
Streaming	vCore: 1, Memory: 1024 MB	vCore: 1, memory: 2048 MB

جدول 3: نیاز منبع کانتینر برای هر نوع برنامه بارکار ترکیبی

شکل 3: نرخ تکمیل بارکار متوسط و زمان فعال سازی بارکار میانگین چهار SPC در سناریوی تک صفی



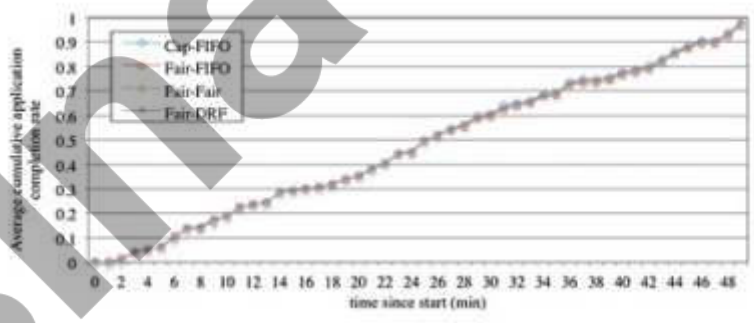
شکل 4: حافظه موجود همه گره‌های فرعی در سه دقیقه اول اجرای بارکار

از این روی نرخ تکمیل بارکار بهبود می‌یابد. به همین دلیل و با توجه به مسئله تقسیم منابع، هر دو Fair-FIFO و Fair-Fair دارای سرعت تکمیل کم‌تری از Cap-FIFO می‌باشند. با این حال، نتایج نشان داد که سرعت تکمیل کاربار FAIR-DRF به طور معنی داری تحت تأثیر قرار نگرفته است و این نشان می‌دهد که سیاست DRF مورد استفاده توسط Fair-DRF قادر به کاهش مسائل فوق می‌باشد.

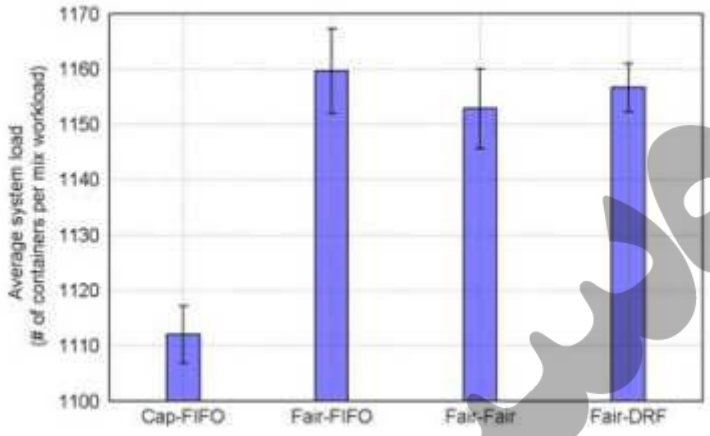
شکل 3 نیز زمان چرخش بارکار میانگین چهار SPC را نشان می‌دهد. اگرچه Fair-Fair منجر به کوتاه‌ترین زمان چرخش بارکار شده است، با این حال spc خوبی برای سناریوی تک صفی نمی‌باشد زیرا سرع تکمیل آن کم‌تر از سرعت تکمیل Cap-FIFO و Fair-DRF می‌باشد. بر اساس نرخ تکمیل بارکار متوسط، زمان چرخش بارکار متوسط و انحراف معیار نشان داده شده در شکل 3 می‌توان دید کهFair-DRF بهترین عملکرد را دارد در حالی که Fair-FIFO بدترین عملکرد را دارد.

شکل 5 نرخ تکمیل بارکار تجمعی متوسط چهار SPC را در طی اجرای بارکار نشان می‌دهد. می‌توان دید که چهار منحنی تقریباً هم پوشانی دارند و این بدان معنی است که همه SPC ها دارای سرعت اجرای کاربار مشابهی می‌باشند. شکل 6 سیستم میانگین بار چهار SPC را نشان می‌دهد. وقتی که Fair-CAP-FIFO تست شد، کلاستر به طور متوسط 1112 کانتینر را برای اجرای بارکار اجرا می‌کند. این مقدار کم‌تر از مقدار سه SPC دیگر می‌باشد به این معنی که CAP-FIFO چندین کانتینر را نسبت به سایر SPC ها ذخیره می‌کند.

شکل 7 توان عملیاتی میانگین استریمینگ چهار SPC را نشان می‌دهد. در شروع اجرای بارکار، همه SPC ها قادر به پردازش بیش از 12 استریم داده در هر دقیقه می‌باشند. با این حال، چون درخواست‌های بیشتری از بارکار به کلاستر داده می‌شود، همه توان عملیاتی SPC کاهش می‌یابد. با این وجود، می‌توان دید که Fair-DRF توان عملیاتی بالاتری را از سایرین ارائه می‌کند. شکل 8 میانگین تأخیر کل چهار SPC را نشان می‌دهد. صرف نظر از نوع SPC مورد استفاده، تفاوت‌های میان تأخیر کل آن‌ها در چارک اول، میانه و چارک سوم غیر معنی دار است و از این روی انحراف معیار آن‌ها مشابه با یک دیگر می‌باشد و این بدین معنی است که این چهار SPC دارای عملکرد غیر قابل تمایز از حیث تأخیر کل می‌باشد.



شکل 5: میانگین نرخ تکمیل بارکار چهار SPC در سناریوی تک صف



شکل 6: بار سیستم میانگین در زمانی که چهار SPC به طور انفرادی در سناریوی تک صف استفاده می‌شود.

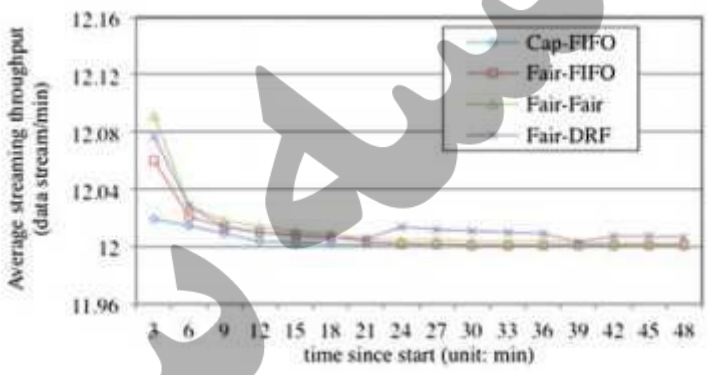
بر اساس نتایج نشان داده شده در شکل 3-8، ما نتایج زیر را بدست اوردیم: اگر عملکرد اجرای برنامه یک مسئله مهم باشد، Fair-DRF رایج‌ترین SPC برای سناریوی تک صف به دلیل عملکرد خوب آن از حیث سرعت تکمیل بارکار، زمان چرخش بارکار و توان عملیاتی استریمینگ است. با این حال در صورتی که تنها از کارایی مصرف منابع آن را در نظر بگیریم، CAP-FIFO پیشنهاد می‌شود زیرا از کانتینرهای کم‌تری از سایر Spc ها استفاده می‌کند.

2-6 سناریوی صف مجزا

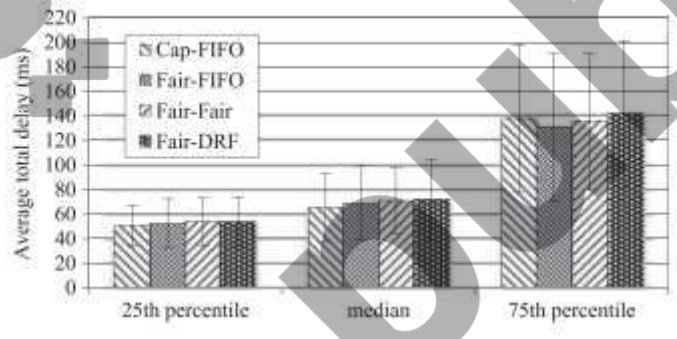
در این زیر بخش، ما اقدام به ارزیابی چهار SPC در سناریوی صف مجزا می‌کنیم. توجه کنید که همه SPC ها دارای سرعت تکمیل بارکار تجمعی در طی اجرای کاربار می‌باشد و از این روی نتایج در این جا به دلیل کمبود فضا تشریح نشده‌اند.

شکل 9 نشان می‌دهد که CAP-FIFO بالاترین نرخ و سرعت تکمیل بارکار و کم‌ترین انحراف معیار را در میان همه SPC ها دارد. با این حال، CAP-FIFO در این سناریو نمی‌تواند کامل شود زیرا برنامه‌های بسیاری در یک سناریوی تک صف وجود دارند (شکل 9 و 3). دلیل اصلی دو مورد است. اولاً، هر صف در سناریوی صف مجزا می‌تواند حداکثر از 30 درصد منابع خوشه‌ای استفاده کند. دوم، برنامه استریمینگ 5 vCores و 5120 MB را اشغال می‌کند یعنی 8.3 درصد منابع خوشه‌ای. از این روی، سه صف دیگر برای برنامه‌های دو مرحله‌ای، DAG و DCG تنها از 30 درصد منابع خوشه‌ای استفاده می‌کند. در مقایسه با سناریوی تک صفی، منابع موجود برای برنامه‌های فوق در سناریوی صف مجزا کاهش یافته و منجر به خطای برنامه بیشتری می‌شود.

پدیده فوق نه تنها وقتی رخ می‌دهد که Cap-FIFO استفاده می‌شود، بلکه زمانی به وقوع می‌پیوندد که سه SPC دیگر تست شوند. با مقایسه شکل 9 و شکل 3، می‌توان دید که سرعت تکمیل بارکار Fair-FIFO, Fair-Fair, Fair-DRF در سناریوی صف مجزا به اندازه سناریوهای تک صفی نیست. این وضعیت حتی برای FAIR-SDRF بدتر است زیرا سرعت تکمیل متوسط آن به 95.96 درصد با انحراف معیار بسیار بزرگ کاهش می‌یابد و این بدین معنی است که FAIR-SDRF برای سناریوی صف مجزا نامناسب است.



شکل 7 بازده عملیاتی استریمینگ متوسط چهار SPC در سناریوی تک صف



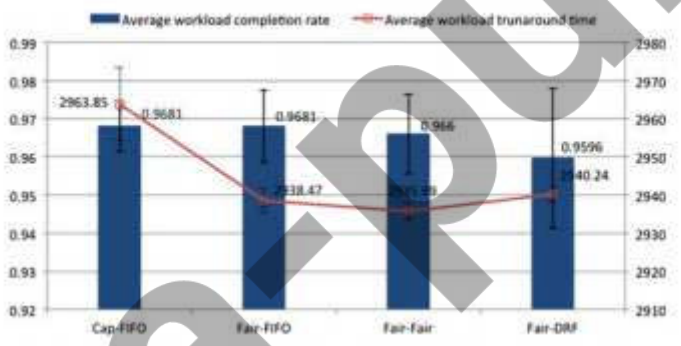
شکل 8: میانگین کل تاخیرات چهار SPC در سناریوی تک صف

شکل 9 نیز نشان می‌دهد که زمان چرخش بارکار Cap-FIFO طولانی‌تر از زمان تأخیر سه SPC می‌باشد و انحراف معیار متناظر آن علی رغم نرخ تکمیل بار کار آن بیشترین مقدار است. از سوی دیگر، اگرچه نرخ اکمیل بارکار Cap-FIFO در رتبه دوم قرار دارد، زمان چرخش کاربار آن کوتاه‌تر از Cap-FIFO است. از این روی بر اساس هر دو نرخ تکمیل بارکار و زمان چرخش، Cap-FIFO برای سناریوی صف مجزا مناسب‌تر است. با مقایسه شکل 9 و شکل 3، می‌توان دید که چهار SPC در سناریوی صف مجزا منجر به زمان چرخش کاربار طولانی‌تری

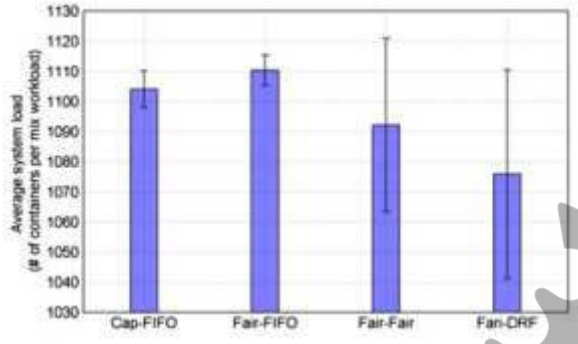
از سناریوی تک صف می‌شود و این بدین معنی است که استفاده از چهار صف برگ بهتر از کاربرد یک صف برگ نیست.

شکل 10 بار سیستم میانگین ناشی از چهار SPC را نشان می‌دهد. با مقایسه شکل 10 با شکل 6، نتایج نشان داد که همه SPC ها در سناریوی صف مجزا منجر به بار سیستم میانگین کم‌تری نسبت به سناریوی تک صف می‌شود. دلیل این است که مسئله تقسیم بندی منبع و محدودیت ظرفیت برای هر صف، امکان استفاده از کانتینرهای بیشتر توسط SPC ها برای اجرای بارکار را نمی‌دهد و از این روی بر نرخ تکمیل بارکار تأثیر می‌گذارد.

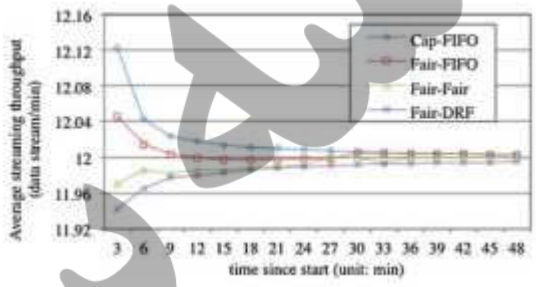
شکل 11 توان عملیاتی استریمینگ متوسط چهار SPC را در سناریوی صف مجزا نشان می‌دهد. می‌توان دید که توان‌های عملیاتی استریمینگ Fair-Fair وFair-DRF کم‌تر از 12 استریم در هر دقیقه می‌باشند. دلیل کلیدی این است که وقتی این دو SPC در سناریوی صف مجزا استفاده شد، منابع تخصیص داده شده به صف استریمینگ با سایر برنامه‌ها اشغال شد. از این روی هر صف برای برنامه‌های فوق از 30 درصد منابع خوشه‌ای استفاده کرده است و صف استریمینگ تنها از 10 درصد استفاده کرده است. به دلیل این رقابت منابع در Fair-Fair DRF, Fair-Fair، برنامه‌های استریمینگ قادر به ارائه توان عملیاتی خوبی نبودند.



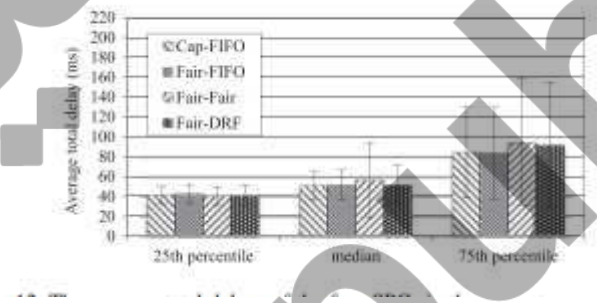
شکل 9: نرخ تکمیل بارکار متوسط و زمان چرخش بارکار متوسط چهار SPC در سناریوی صف مجزا.



شکل 10: بار سیستم میانگین در زمانی که چهار SPC به طور مجزا در سناریوی صف مجزا استفاده می‌شود



شکل 11



شکل 12 تأخیر کل متوسط چهار SPC را نشان می‌دهد چون Fair-Fair وFair-DRF دارای توان عملیاتی استریمینگ پایین است می‌توان دید که تأخیر متوسط کلان‌ها در چارک سوم و میانه طولانی‌تر از Cap-FIFO و Fair-FIFO می‌باشد. بر اساس نتایج نشان داده شده در شکل 9 تا 12، نتایج نشان داد که Fair-FIFO در سناریوهای صف مجزا از نظر کارایی اجرای برنامه بهترین عملکرد را دارد. به علاوه اگر عملکرد چهار SPC را در سناریوی صف مجزا با یک صف مقایسه کنیم، نتایج نشان می‌دهد که همه SPC ها دارای عملکرد بهتری در سناریوی تک صف می‌باشند زیرا آن‌ها مسئله کمبود منابع ناشی از محدودیت ظرفیت هر صف برگ در سناریوی صف مجزا را ندارند. از این روی استفاده از یک صف برای سازمان دهی برنامه‌های ترکیبی بهتر از استفاده از چهار صف است.

3-6 سناریوی صف ادغام شده

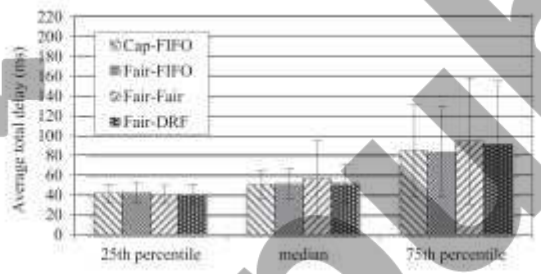
به منظور مطالعه این که آیا این SPC ها عملکرد بهتری از سناریوی های قبلی دارند، در این زیر بخش ما آن‌ها را در سناریوی صف ادغام شده ارزیابی می‌کنیم. شکل 13 نشان می‌دهد که هر دو CAP-FIF0 و FAIR-DRF به بالاترین زمان تکمیل رسیده‌اند و هر دوی Fair-FIFOوFair-Fair بهه دومین نرخ تکمیل رسیده‌اند (97.52 درصد). با مقایسه شکل 13 و 3، می‌توان دید که سرعت تکمیل بار کار همه SPC ها در سناریوی صف ادغام شده افزایش می‌یابد و این نشان می‌دهد که برای این SPC ها، تفکیک برنامه استریمینگ و سایر برنامه‌ها به دو صف امکان تکمیل موفق برنامه‌ها را می‌دهد. دلیل کلیدی این است که منبع مورد استفاده توسط برنامه استریمینگ حدود 8.3 درصد منابع بوده است. از این روی، بقیه منابع به صف استریمینگ تخصیص داده شده است. اگرچه Cap-FIFO عملکرد خوبی مانند Fair-DRF از حیث نرخ تکمیل کاربار دارد، زمان چرخش بارکاران اندکی طولانی‌تر از Fair-DRF می‌باشد (شکل 13 را ببینید). به طور مشابه، اگرچه FIFO- FAIR دارای سرعت تکمیل مشابه با Fair-Fair می‌باشد، زمان فعال سازی بار کار آن طولانی‌تر از Fair-Fair می‌باشد. با مقایسه شکل 13 با شکل 3، بدیهی است که همه spc ها منجر به زمان بار کاری کوتاه‌تری از سناریوی صف ادغام شده می‌شوند. از این روی می‌توان نتیجه گرفت که سناریوی صف ادغام شده نه تنها موجب بهبود سرعت تکمیل بارکار برای همه SPC ها می‌شود بلکه موجب کوتاه‌تر شدن زمان چرخش بارکار می‌شود.

شکل 11: توان عملیاتی استریمینگ متوسط چهار SPC در سناریوی صف مجزا

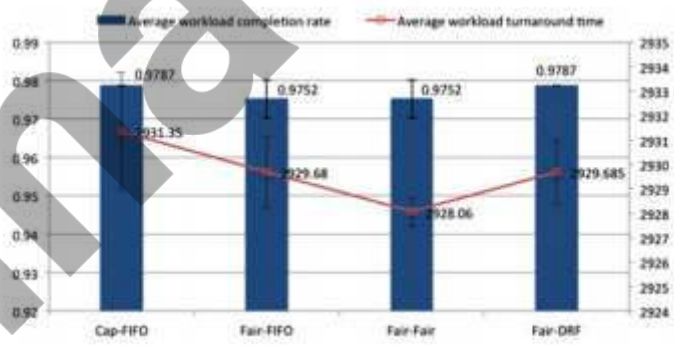
شکل 12: میانگین کل تأخیر چهار SPC در سناریوی صف مجزا

شکل 14 نشان می‌دهد که Cap-FIFO دارای کم‌ترین بار سیستم متوسط در میان همه SPC ها بوده و تحت تأثیر همه سه سناریو قرار نمی‌گیرد و از این روی می‌توان گفت که Cap-FIFO کارامد ترین از حیث مصرف کانتینر است. با این حال وقتی که SPC های دیگری تست شوند این وضعیت رخ نمی‌دهد. می‌توان دید که بار متوسط سیستم Fair-Fairو Fiar-DRF به طور خفیفی در سناریوی صف ادغام شده افزایش یافته است. دلیل این است که در سناریوی صف ادغام شده، این SPC ها قادر به تکمیل درخواست‌های بارکار می‌باشند و از این روی تعداد کل کانتینرهای مورد استفاده برای اجرای بار کار افزایش می‌یابد.

شکل 15 توان عملیاتی استریمینگ همه SPC ها را نشان می‌دهد. می‌توان دید که تنها Fair-DRF دارای توان عملیاتی کم‌تر از 12 استریم در هر دقیقه می‌باشد زیرا منابع تخصیص داده شده به صف استریمینگ تحت این SPC توسط سایر رنامه ها استفاده نمی‌شود. با این حال در صورت استفاده از FAIR-FAIR این مسئله حل می‌شود. با مقایسه شکل 15 با شکل 11، بدیهی است که توان عملیاتی استریمینگ FAIR-FAIR زمانی بهبود می‌یابد که سناریوی صف ادغام شده استفاده شود. شکل 16 تأخیر کل متوسط چهار SPC را نشان می‌دهد. چون توان عملیاتی استریمینگ Fiar-DRF در اجرای بارکار کافی نبود، تأخیر کل متوسط آن و انحراف معیار بالاتر از سایر SPC ها بود. بر اساس نتایج ازمایشی نشان داده شده در شکل‌های 13 تا 16، می‌توان گفت که Fiar-DRF و Cap-FIFO برای سناریوی صف ادغام شده توصیه می‌شود که به تکمیل بارکار بالا می‌انجامد. با این حال، اگر زمان چرخش بارکار در نظر گرفته شود، Fiar-DRF عملکرد بهتری از Cap-FIFO خواهد داشت. از سوی دیگر، اگر توان عملیاتی استریمینگ در نظر گرفته شود، Cap-FIFO بهتر از Fiar-DRF می‌باشد. با این حال از نظر کارایی مصرف منابع، Cap-FIFO بهترین است.



شکل 14: بار سیستم میانگین در زمانی که چهار SPC به طور مجزا در سناریوی صف ادغام شده استفاده می‌شوند



شکل 13: سرعت تکمیل بارکار متوسط و زمان چرخش بارکار متوسط چهار SPC در سناریوی صف ادغام شده

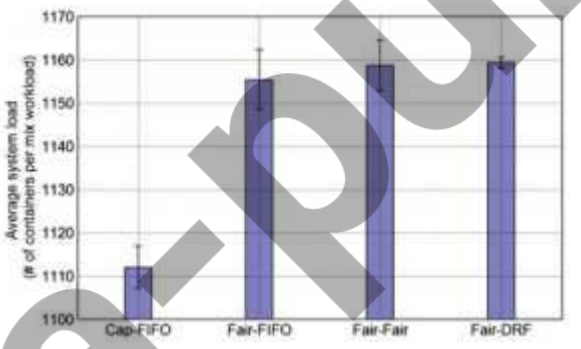
7-نتیجه گیری و کارهای آینده

در این مقاله، ما به بررسی چهار SPC و چهار نوع برنامه پشتیبانی شده توسط YARN پرداخته‌ایم. برای درک کامل اثرات عملکرد چهار SPC بر روی انواع برنامه‌های ترکیبی، ما ازمایشات گسترده‌ای را با در نظر گرفتن نه تنها انواع برنامه‌های ترکیبی، بلکه سه سناریوی با ساختار صف متفاوت (سناریوی تک صف، سناریوی صف مجزا و سناریوی صف ادغام شده) انجام دادیم. بر اساس نتایج ازمایشی، نتایج زیر بدست آمده و به طور خلاصه در جدول 4 نشان داده شده‌اند.

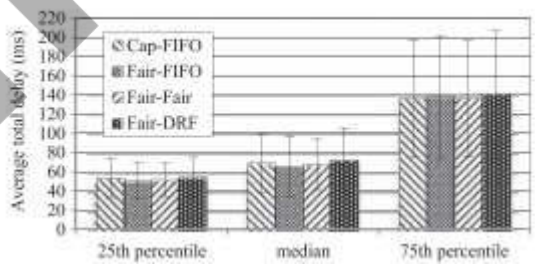
1- Fair-DRF بهترین گزینه برای سناریوی تک صف است زیرا منجر به سرعت تکمیل بارکار بالاتر و زمان چرخش بارکار کوتاه‌تر در مقایسه با سایر سه SPC می‌شود

2- Fair-FIFO رایج‌ترین SPC توصیه شده برای سناریوی صف مجزا به دلیل عملکرد خوب آن از حیث هر دو زمان تکمیل بارکار و زمان چرخش بارکار است

3- Cap-FIFOو Fair-DRF هر دو برای سناریوی صف ادغام شده مناسب هستند. از این روی Cap-FIFO اندکی بهتر از Fair-DRF در توان عملیاتی استریمینک و کارایی استفاده از منابع است، در حالی که Fair-DRF اندکی بهتر از Cap-FIFO در زمان چرخش بارکار است.



شکل 15: میانگین توان عملیاتی چهار SPC در سناریوی صف ادغام شده



شکل 16: میانگین تأخیر کل چهار SPC در سناریوی صف ادغام شده

Metric / Scenario	Applications execution performance	Resource-usage efficiency
One-queue scenario	Fair-DRF	Cap-FIFO
Separate-queue scenario	Fair-FIFO	Fair-DRF
Merged-queue scenario	Cap-FIFO and Fair-DRF	Cap-FIFO

جدول 4: رایج‌ترین SPC در زمانی که متریک‌ها و سناریوهای صف بندی مختلفی در نظر گرفته می‌شوند.

4-از دیدگاه کارایی مصرف منابع، Cap-FIFO بهترین عملکرد را در سناریوی صف ادغام شده و تک صفی دارد زیرا تعداد کل کانتینرهای اجرا شده توسط Cap-FIFO برای اجرای کاربار کم‌تر از کانتینرهای اجرا شده توسط سه SPC می‌باشد.

در صورتی که ما نتایج آزمایشی همه سناریوها را در نظر بگیریم، بدیهی است که استفاده از سناریوی صف ادغام شده بهترین گزینه برای همه SPC ها است زیرا امکان دست یابی به همه SPC ها با نرخ تکمیل کاربار بالا و زمان چرخش کاربار کوتاه را می‌دهد. بر عکس، استفاده از سناریوی صف مجزا توصیه نمی‌شود زیرا موجب بدتر شدن نرخ تکمیل بارکار و طولانی شدن زمان چرخش کاربار برای تقریباً همه SPC می‌شود. کار آینده ما، مطالعه شیوه تأثیر گذاری ترکیبی از برنامه‌های بارکار بر SPC های فوق و ارائه یک زمان بند جدید برای YARN می‌باشد به طوری که مسئله تقسیم بندی منابع را می‌توان حل کرد و سرعت تکمیل بارکار را می‌توان بهبود بخشید.