زمان‌بندی بی‌درنگ چندپردازنده‌ای شبه‌افرازی در سیستم‌های مدیریت جریان داده

مهدی عالمی و مصطفی حق‌جو

چکیده: در سیستم‌های مدیریت جریان داده، داده‌های جریانی وارد سیستم می‌شوند و پرس و جوهای ذخیره‌شده بر روی این داده‌ها اجرا می‌شوند. با توجه به بار کاری بالا نیاز به ظرفیت پردازشی بالا است و استفاده از چندپردازنده باید در نظر گرفته شود. همچنین در سیستم‌های بی‌درنگ پرس و جوها تحت مهلت مشخصی باید کار خود را به اتمام برساند. از رویکردهای موجود در زمان‌بندی چندپردازنده‌ای بی‌درنگ رویکرد افرازی است که هر پرس و جو با توجه به بهره‌وری که نسبت زمان اجرا به دوره است به پردازنده‌ها انتساب داده می‌شود و فقط در آن اجرا می‌شود. برای نزدیک‌شدن به جواب بهینه در اینجا پرس و جوهایی که در یک پردازنده جا نمی‌گیرند بر اساس بهره‌وری شکسته می‌شوند و در بین پردازنده‌ها پخش می‌شوند. این سیستم با داده‌های واقعی شبکه تست شده است. مقایسه‌ها نشان می‌دهد که رویکرد مورد نظر توانسته است نسبت به رویکرد افرازی ساده میزان از دست رفتن مهلت‌ها را کاهش دهد و میزان بهره‌وری سیستم را بالا ببرد.

کلید واژه: **توزیع بار، چندپردازنده، زمان‌بندی بی‌درنگ، افرازی، بهره‌وری.**

۱- مقدمه

یک جریان داده دنباله‌ای پیوسته، نامتناهی، سریع، متغیر با زمان و شاید غیر قابل پیش‌بینی از عنصرهای داده است که همواره به انتهای آن افزوده می‌شود [۱]. جریان داده در کاربردهایی از جمله برنامه‌های مالی و بورس، برنامه‌های اندازه‌گیری کارایی در مانیتورینگ شبکه و مدیریت ترافیک، ثبت وقایع و جریان‌های کلیک[1] در پیگردی وب[2]، شبکه‌های حسگر بی‌سیم، ماهواره‌ها و رکوردهای تماس در مخابرات مطرح می‌شود [۲] و [۳].

در سیستم مدیریت پایگاه داده سنتی همه داده‌ها به صورت پایدار ذخیره می‌شوند و پرس و جوهای مختلف بر روی این داده‌ها اجرا می‌شوند اما در جریان داده، داده‌ها نامحدود هستند و قابل ذخیره‌شدن نیستند. این پرس و جوها هستند که ذخیره می‌شوند و به صورت پیوسته بر روی داده‌ها اجرا می‌شوند [۱]. بنابراین برای کاربردهای مرتبط با جریان‌های داده، سیستم‌های مدیریت جریان داده ایجاد شده‌اند.

افزایش نرخ جریان‌های داده موجب بالارفتن بار کاری سیستم می‌شود و ممکن است سبب شود که پردازش بعضی از پرس و جوها به موقع انجام نگیرد [۴] و در چنین شرایطی ممکن است کیفیت خدمات تضمین نشود. بنابراین نیاز است توان پردازشی سیستم بالا باشد و برای دستیابی به توان پردازشی بالا نیاز به در نظر گرفتن تکنولوژی‌های چندپردازنده‌ای در این سیستم‌ها است.

برای سیستم‌های تک‌پردازنده‌ای در صورتی که وظیفه‌ها قابل قبضه کردن باشند، الگوریتم زمان‌بندی بی‌درنگ بهینه EDF وجود دارد که در زمان اجرا زمان‌بندی انجام می‌دهد [۵]. برای سیستم‌های چندپردازنده‌ای در حالت کلی الگوریتم زمان‌بندی بی‌درنگ بهینه‌ای وجود ندارد ولی با ایجاد محدودیت‌هایی می‌توان به الگوریتم بهینه دست یافت [۶]. دو رویکرد کلی در زمان‌بندی چندپردازنده‌ای وجود دارد: سراسری و افرازی [۷]. رویکرد افرازی ساده است ولی بهینه نیست در حالی که با رویکرد سراسری با اعمال محدودیت‌هایی می‌توان به زمان‌بندی بهینه دست یافت.

در رویکرد افرازی ابتدا یک مجموعه وظیفه به پردازنده‌ها انتساب داده می‌شود. سپس هر وظیفه فقط در پردازنده انتساب داده شده بر اساس الگوریتم زمان‌بندی بی‌درنگ تک‌پردازنده‌ای EDF برای اجرا انتخاب می‌شود. بنابراین در این رویکرد مسئله زمان‌بندی بی‌درنگ چندپردازنده‌ای به چند مسئله زمان‌بندی تک‌پردازنده‌ای تبدیل می‌شود و هر وظیفه تنها در یک پردازنده اجرا می‌شود. در رویکرد سراسری یک زمان‌بند بی‌درنگ سراسری وجود دارد و با توجه به الگوریتم تعبیه‌شده هنگامی که نمونه‌ای از وظیفه آماده اجرا شد آن را در یکی از پردازنده‌های خالی اجرا می‌کند. بنابراین در این رویکرد نمونه‌های مختلف هر وظیفه در طول زمان ممکن است در پردازنده‌های مختلفی اجرا شود. در [۸] رویکردی ترکیبی ارائه شده است که در آن ابتدا وظیفه‌ها با رویکرد افرازی به پردازنده‌ها انتساب داده می‌شوند. سپس وظیفه‌هایی که نتوانستند در هیچ یک از پردازنده‌ها توسط رویکرد افرازی قرار بگیرند با رویکرد سراسری زمان‌بندی می‌شوند.

در این مقاله یک سیستم مدیریت جریان داده بی‌درنگ نرم توسعه داده شده است و پرس و جوها به صورت دوره‌ای اجرا می‌شوند. در این سیستم به تعداد پردازنده‌ها موتور اجرای پرس و جو وجود دارد و هر موتور پرس و جو از یک پردازنده مخصوص خود استفاده می‌کند. با رویکرد افرازی هر وظیفه به یکی از این موتورها انتساب داده می‌شود و انتساب با توجه به بهره‌وری پرس و جو و موتور پرس و جو صورت می‌گیرد.

در ادامه در بخش دوم کارهای مرتبط در زمینه سیستم‌های مدیریت جریان داده و سیستم‌های بی‌درنگ ارائه شده است. در بخش سوم مدل پرس و جو و داده و مشخصه‌های بی‌درنگ سیستم ارائه شده است. رویکرد ارائه‌شده همراه با معماری سیستم پیشنهادی در بخش چهارم تشریح می‌شود. نتایج آزمایش‌ها و ارزیابی‌ها در بخش پنجم ارائه شده و در نهایت در بخش ششم نتیجه‌گیری به عمل آمده است.

۲- کارهای مرتبط

اولین سیستم مدیریت جریان داده همه‌منظوره توسط دانشگاه استنفورد با نام STREAM ارائه شده است و تلاش می‌کند که عملکرد سیستم

این مقاله در تاریخ ۲۸ خرداد ماه ۱۳۹۰ دریافت و در تاریخ ۱۲ اسفند ماه ۱۳۹۱ بازنگری شد.

مهدی عالمی، دانشکده مهندسی برق و کامپیوتر، دانشگاه شهید بهشتی، تهران، (email: m_alemi@sbu.ac.ir).

مصطفی حق‌جو، بین‌المللی پیام نور کیش، کیش، (email: haghjoom@iust.ac.ir).

1. Click Stream
2. Web Tracking

مدیریت پایگاه داده را همراه با پردازش پرس و جوهای پیوسته فراهم آورد. STREAM زبان پرس و جوی CQL که مشابه SQL است را تعریف کرده و از عملگرهای تبدیل جریان به رابطه، رابطه به رابطه، و رابطه به جریان که می‌تواند سربار زیادی در هنگام بار سنگین ایجاد کند استفاده می‌کند [۱]. هرچند این سیستم دارای قابلیت بی‌درنگ نیست اما در [۳] قابلیت بی‌درنگ بودن پرس و جوها به STREAM افزوده شده است که نام این سیستم RTSTREAM است. در RTSTREAM مدل PQuery ارائه شده است که هر پرس و جو در هر بار اجرا حداکثر بر روی m تاپل می‌تواند اجرا شود. همچنین هر پرس و جو دارای مهلت و دوره مشخصی است. در RTSTREAM با استفاده از EDF زمان‌بندی بی‌درنگ صورت می‌گیرد که این یک زمان‌بند تک‌پردازنده‌ای است. وجود این الگوریتم سبب می‌شود که تنها در سیستم‌های تک‌پردازنده‌ای بتوان از RTSTREAM استفاده کرد. در روش پیشنهادی در این مقاله استفاده از چندین پردازنده در نظر گرفته شده است که سبب می‌شود بتوان به ظرفیت پردازشی بالا و کاهش نسبت از دست رفتن مهلت‌ها دست یافت.

Aurora نیز سعی در ارائه یک سیستم همه‌جانبه دارد که بتواند در بسیاری از کاربردها مورد استفاده قرار گیرد. عملگرها در Aurora می‌توانند به صورت توزیع‌شده اجرا گردند. هدف اصلی آن یک پردازشگر جریان همه‌جانبه است که بتواند تعداد بسیار زیادی از جریان‌ها را پردازش کند [۹]. همچنین این سیستم محدودیت حافظه را در نظر گرفته است و سعی می‌کند ترتیب اجرای عملگرها را بهینه کند تا میزان حافظه و زمان پاسخ کاهش یابد. در Aurora اولویت پرس و جوها برای اجرا بر اساس پارامترهای کیفیت خدمات است. در اینجا ورود هر تاپل به سیستم به صورت مفهومی همانند یک وظیفه محسوب می‌شود که باید زمان‌بندی و اجرا شود. اما برای کاهش سربار ناشی از زمان‌بندی سعی می‌کند که پردازش تاپل‌ها را به صورت دسته‌ای انجام دهد. هر چند تأخیر تاپل در Aurora به عنوان یکی از پارامترهای کیفیت خدمات در محاسبه اولویت کار است اما با این وجود Aurora در مورد میزان تأخیر تاپل‌ها هیچ تضمینی ندارد و امیدوار است که پارامترهای ورودی پرس و جو طوری تنظیم شده باشد که پرس و جو در زمان اجرا دچار تاخیر نشود. در سیستم‌های بی‌درنگ میزان از دست رفتن مهلت‌ها عامل بسیار مهمی در ارزیابی کارایی سیستم است در حالی که در Aurora این پارامتر چندان اهمیتی ندارد. بنابراین در سیستم ارائه‌شده در این مقاله پارامتر اصلی در محاسبه کارایی سیستم میزان از دست رفتن مهلت‌ها است که کاهش آن افزایش کارایی را نشان می‌دهد.

در [۱۰] یک سیستم مدیریت جریان داده بی‌درنگ به نام QStream ارائه شده است. در این سیستم فرض بر این است که DSMS بر روی یک سیستم عامل بی‌درنگ سوار است. سیستم‌های بی‌درنگ خود باید قادر باشند تا کیفیت خدمات را تضمین کنند و بنابراین نیاز است که قابلیت بی‌درنگ بودن در خود سیستم گنجانده شود. در سیستم ارائه‌شده قابلیت بی‌درنگ بودن در هسته سیستم گنجانده شده است. با توجه به ثابت‌بودن مشخصات جریان داده از جمله نرخ ورود ثابت در [۱۰] و کاربردی‌نبودن این فرضیات در [۱۱] قابلیت وفق‌پذیری به QStream افزوده شده است.

در [۱۲] یک سیستم پردازش پرس و جوی پیوسته و دوره‌ای ارائه شده است. برای پرس و جوهای پیوسته از TB Server استفاده شده که هدف آن این است که مهلت پرس و جوهای دوره‌ای تضمین شود و تعداد از دست رفتن مهلت پرس و جوهای غیر دوره‌ای[1] را کاهش دهد. بر اساس میزان از دست رفتن مهلت پرس و جوهای aperiodic کاهش بار انجام می‌شود.

یک سیستم جریان داده باید بتواند تاپل‌ها را با سرعت پردازش کند. در [۱۳] یک موتور پردازش پرس و جوی PQP ارائه شده است که شامل k پردازنده (ماشین منطقی) است. این ماشین‌های موازی با یکدیگر در ارتباط هستند که هر یک شامل یک کپی از پرس و جوی در حال اجرا است. هر عملگر در هر ماشین می‌تواند خروجی خود را به کپی پرس و جو در ماشین دیگر بفرستد و ادامه‌ی پردازش در ماشین دیگر انجام گیرد. این پرس و جوهای کپی‌شده با هم تشکیل یک گراف جهت‌دار می‌دهند و هر تاپل کوتاه‌ترین مسیر در این گراف را طی می‌کند تا به خروجی برسد. کوتاه‌ترین مسیر به صورت دوره‌ای و توسط الگوریتم دایجسترا مشخص می‌شود. در [۱۴] موتور پردازشی Disp ارائه شده که PQP را بهبود داده است. در Disp زمان‌بندی عملگرها به صورت پیوسته انجام می‌گیرد اما پارامترهای سیستم بی‌درنگ در آن گنجانده نشده است.

۳- مشخصات سیستم

برای روشن‌شدن مسئله در این قسمت مشخصات سیستم مد نظر از جمله مدل داده، مدل پرس و جو و انواع پرس و جوهایی که می‌توان تعریف کرد و خصوصیات آنها ارائه شده است.

۳-۱ مدل داده

یک جریان داده دنباله‌ای پیوسته، نامتناهی، سریع، انفجاری، غیر قابل پیش‌بینی و متغیر با زمان از عناصر داده است که می‌توان آن را به صورت $S=\langle s_1,s_2,\ldots\rangle$ مشخص کرد. هر s_i یک مؤلفه داده است که به صورت زوج مرتب $\langle t_i,tuple_i\rangle$ مشخص می‌شود. t_i برچسب زمانی s_i است که مشخص‌کننده زمان ورود $tuple_i$ است.

۳-۲ مدل پرس و جوی بی‌درنگ

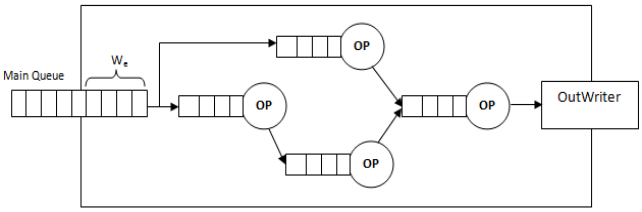
مدل پرس و جوی تعریف‌شده در این سیستم به این صورت است که برای هر پرس و جو یک دوره زمانی تعریف می‌شود و هر پرس و جو مطابق با این دوره فعال می‌شود. هر پرس و جو دارای یک صف است و تاپل‌هایی که باید توسط پرس و جو پردازش شوند در صف پرس و جو کپی می‌شوند. هر پرس و جو پس از فعال‌شدن حداکثر به اندازه M تاپل از صف خود را برای پردازش انتخاب و شروع به کار می‌کند. در اینجا به این M تاپل پنجره اجرایی تاپل W_e گفته می‌شود. پرس و جو باید تا قبل از به پایان رسیدن مهلتش، پردازش این M تاپل را به پایان برساند. در صورتی که پرس و جویی در حال اجرا باشد و تاپل جدیدی به آن پرس و جو رسیده باشد آن تاپل در اجراهای بعدی آن پرس و جو برای پردازش انتخاب می‌شود. بنابراین در این مدل بهره‌وری پرس و جوی i مطابق با (۱) تعریف می‌شود

$$U_i=\frac{C_i\times M}{P_i} \qquad (۱)$$

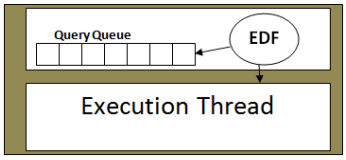
که C_i برابر با زمان لازم برای پردازش هر تاپل توسط پرس و جوی i، M اندازه پنجره اجرایی و P_i دوره پرس و جوی i است.

مؤلفه‌های تشکیل‌دهنده یک پرس و جو در شکل ۱ نشان داده شده است. هر پرس و جو شامل یک سری عملگر است که هر یک دارای

1. Aperiodic



شکل ۱: مؤلفه‌های تشکیل‌دهنده یک پرس و جو.



شکل ۳: ساختار داخلی موتور پرس و جو.

صف مخصوص به خود هستند و ورودی خود را ممکن است از صف اصلی یا از عملگر قبلی خود دریافت کنند. هر عملگر وقتی برای اجرا انتخاب شد یک تاپل را پردازش می‌کند و نتیجه را در صف ورودی عملگر بعدی خود قرار می‌دهد. هر پرس و جو پس از فعال‌شدن حداکثر به اندازه پنجره اجرایی خود $(W_e = M)$ تاپل‌ها را برای پردازش انتخاب می‌کند و هر پرس و جو خروجی خود را توسط مؤلفه OutWriter می‌نویسد.

هر پرس و جو حداکثر به اندازه مهلت خود فرصت اجرا دارد. اگر پرس و جو زودتر از مهلت تعریف شده، اجرای خود را به پایان برساند و هیچ تاپل دیگری برای پردازش باقی نمانده باشد آنگاه نسبت از دست رفتن مهلت پرس و جو (*Miss-Ratio*) در آن نمونه‌ی اجرا برابر با صفر و در غیر این صورت برابر با $(W_e - N)/W_e$ است که N برابر با تعداد تاپل‌های پردازش‌شده می‌باشد.

۳-۳ مشخصات سیستم پیشنهادی

نوع بی‌درنگ بودن: در سیستم‌های مدیریت جریان داده با توجه به پویابودن محیط، پیش‌بینی دقیق زمان اجرای پرس و جوها امکان‌پذیر نیست. بنابراین سیستم مد نظر در دسته سیستم‌های بی‌درنگ سخت قرار نمی‌گیرد. در سیستم پیشنهادی نسبت از دست رفتن مهلت یک پرس و جو به صورت $0 \le Miss \le 1$ است. اگر نسبت از دست رفتن مهلت یک پرس و جو برابر با یک باشد آن گاه آن پرس و جو هیچ ارزشی را به سیستم باز نگردانده است. همین طور بالعکس، اگر نسبت از دست رفتن مهلت یک پرس و جو برابر با صفر باشد آن گاه آن پرس و جو ارزش کاملی را به سیستم باز گردانده است. بنابراین سیستم پیشنهادی یک سیستم بی‌درنگ نرم است که میزان ارزشی که نمونه پرس و جوی i به سیستم باز می‌گرداند مطابق با (۲) محاسبه می‌شود

$$V_i = 1 - Miss_i \qquad (2)$$

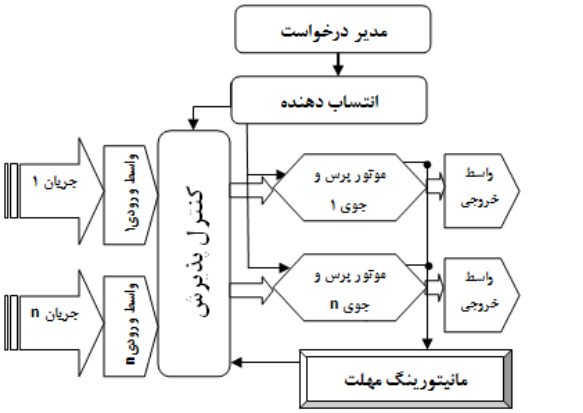
که V_i میزان ارزش باز گردانده شده توسط نمونه پرس و جوی i و $Miss_i$ نیز میزان از دست رفتن مهلت پس از اجرای نمونه پرس و جوی i است.

رهاشدن پرس و جوها: مطابق با مدل پرس و جوی تعریف‌شده در قسمت قبلی، رهاشدن پرس و جوها به صورت دوره‌ای است.

وابستگی بین پرس و جوها: بین پرس و جوها هیچ وابستگی وجود ندارد و هر پرس و جو به صورت مستقل از دیگری اجرا می‌شود.

اولویت پرس و جو: اولویت اجرای پرس و جوها در مؤلفه زمان‌بند به صورت پویا و در زمان اجرا محاسبه می‌شود.

قبضه‌ای بودن پرس و جوها: پرس و جوی با اولویت بالاتر پس از



شکل ۲: معماری سیستم پیشنهادی.

فعال‌شدن می‌تواند پرس و جوی با اولویت پایین‌تر و در حال اجرا را قبضه کند.

زمان‌بندی ایستا یا پویا: با توجه به پویابودن محیط، زمان‌بندی به صورت پویا است.

حلقه بسته یا حلقه باز: در این سیستم مکانیزم بازخورد به منظور وفق‌پذیری سیستم با محیط وجود دارد. بنابراین این سیستم یک سیستم حلقه بسته است.

تک‌پردازنده‌ای یا چندپردازنده‌ای: هر چند این سیستم با یک پردازنده می‌تواند به کار خود ادامه بدهد، قابلیت استفاده از چندپردازنده در این سیستم وجود دارد.

مهاجرت پرس و جوها: با توجه به استفاده از رویکرد افرازی، پس از انتساب یک پرس و جو به یک پردازنده دیگر آن پرس و جو نمی‌تواند به پردازنده دیگری مهاجرت کند.

٤- سیستم مدیریت جریان داده پیشنهادی

در این سیستم استفاده از چندپردازنده مستقیماً مد نظر قرار گرفته است. استفاده از چندپردازنده سبب افزایش ظرفیت پردازشی سیستم می‌شود. قسمت‌های مختلف سیستم پیشنهادی در معماری ارائه‌شده در شکل ۲ نشان داده شده است. پرس و جوهای ورودی از طریق مدیر درخواست بررسی می‌شود و پس از تولید گراف پرس و جو آنها را به انتساب‌دهنده برای انتساب به یکی از موتورهای پرس و جو می‌دهد. جریان‌های ورودی توسط واسط ورودی دریافت می‌شود و تاپل‌های ورودی تحویل کنترل پذیرش داده می‌شوند. هر واسط ورودی از قالب جریان داده مربوط به خود آشنا است، بنابراین جریان‌های داده مختلف پس از عبور از واسط ورودی به صورت ترتیبی از تاپل‌ها وارد سیستم می‌شوند. کنترل پذیرش تاپل‌های ورودی را در صف پرس و جوهای منتظر آن تاپل‌ها کپی می‌کند و ممکن است بعضی تاپل‌ها را بر حسب پارامترهای دریافت‌شده از متوازن‌کننده بار دور بریزد. هر پرس و جو در یکی از موتورهای پرس و جو قرار دارد و پس از اجرا نتیجه خود را به واسط خروجی می‌دهند تا خروجی تولید شود. موتورهای پرس و جو به صورت موازی اجرا می‌شوند.

٤-۱ موتور پرس و جو

پرس و جوها باید در یکی از موتورهای پرس و جو اجرا شوند. هر موتور پرس و جو از یک پردازنده برای اجرای پرس و جوهای بی‌درنگ استفاده می‌کند. در موتورهای پرس و جو از الگوریتم زمانبندی بی‌درنگ EDF استفاده شده است.

شکل ۳ نشان‌دهنده ساختار داخلی موتور پرس و جو است. پرس و جوها در صف Query Queue قرار می‌گیرند و الگوریتم زمان‌بندی از بین پرس و جوهای فعال یکی از آنها را که مهلت نزدیک‌تری دارد برای اجرا انتخاب می‌کند. پرس و جوی انتخاب‌شده به Execution Thread سپرده می‌شود و این مؤلفه نیز پرس و جوی مد نظر را اجرا می‌کند. واحد زمان‌بند EDF بر اساس سه شرط زیر اجرای پرس و جوی جاری را متوقف می‌سازد: ۱) پرس و جو همه تاپل‌های موجود در پنجره اجرایی خود را پردازش کرده باشد، ۲) مهلت پرس و جو به اتمام رسیده باشد و ۳) یک پرس و جویی که مهلت نزدیک‌تری نسبت به پرس و جوی در حال اجرا دارد فعال شود و به علت قابل قبضه‌ای بودن پرس و جوها، پرس و جوی جاری قبضه می‌شود و پرس و جوی با مهلت نزدیک‌تر شروع به اجرا می‌کند.

۴-۲ بهبود زمان‌بندی بی‌درنگ چندپردازه‌ای با رویکرد افرازی

در این سیستم از زمان‌بندی چندپردازنده‌ای بی‌درنگ با رویکرد افرازی به همراه الگوریتم تک‌پردازنده‌ای EDF استفاده می‌شود. رویکرد افرازی ساده است و به راحتی قابل پیاده‌سازی است. همان طور که قبلاً ذکر شد این روش زمان‌بندی دارای ۲ قسمت است: انتساب‌دهنده و زمان‌بند بی‌درنگ تک‌پردازنده‌ای. انتساب‌دهنده با یکی از الگوریتم‌های انتساب یک پرس و جو را به یکی از موتورهای اجرای پرس و جو انتساب می‌دهد. با توجه به NP-Hardبودن مسئله روش انتساب در حالت بهینه از روش‌های مکاشفه‌ای اولین جایدهی، بهترین جایدهی یا بدترین جایدهی استفاده می‌شود. با توجه به سادگی و کارایی، اولین جایدهی در این سیستم در قسمت انتساب‌دهنده گنجانده شده است. هر موتور اجرایی پرس و جو از یک پردازنده مجزا برای اجرای پرس و جوها استفاده می‌کند. به دلیل این که از الگوریتم زمان‌بندی بی‌درنگ EDF استفاده می‌شود مجموع بهره‌وری وظیفه‌هایی که به یک موتور پرس و جو انتساب داده می‌شوند باید کمتر یا مساوی با یک شود.

روش ارائه‌شده در بالا حالت ساده روش زمان‌بندی بی‌درنگ چندپردازنده افرازی است. برای بهبود کارایی این الگوریتم در این سیستم به این صورت عمل می‌شود که اگر پرس و جویی با توجه به میزان بهره‌وری در هیچ یک از موتورهای پرس و جو جای نگیرد به این صورت عمل می‌شود که آن پرس و جو به پرس و جوهای کوچک‌تری شکسته می‌شود به طوری که بتواند در چندین موتور پرس و جو پخش شود. منظور از شکستن پرس و جو تولید پرس و جوهای یکسان از پرس و جوی اولیه اما با بهره‌وری کمتر است به طوری که مجموع بهره‌وری این پرس و جوها برابر با بهره‌وری پرس و جوی اولیه شود. در ادامه جزئیات این الگوریتم شرح داده شده است.

تعریف ۱) شکستن پرس و جو: پرس و جوی q_i می‌تواند به دو یا چند پرس و جوی یکسان $q_{i1},q_{i2},\ldots,q_{in}$ شکسته شود به طوری که مجموع بهره‌وری پرس و جوها برابر با پرس و جوی اولیه است، یعنی $\sum_{j=1}^{n} u_{ij} = u_i$ و نقشه پرس و جوی همه $q_{i1},q_{i2},\ldots,q_{in}$ برابر با نقشه پرس و جوی q_i است. همچنین پنجره اجرایی پرس و جوهای $q_{i1},q_{i2},\ldots,q_{in}$ به ترتیب برابر با $m_{i1},m_{i2},\ldots,m_{in}$ است به طوری که $\sum_{j=1}^{n} m_{ij} = M$ برقرار می‌باشد.

همان طور که قبلاً ذکر شد بهره‌وری هر پرس و جو (U_i) برابر با $C_i \times M / P_i$ است. در این رابطه بهره‌وری نسبت مستقیم با M یعنی تعداد تاپل‌های مورد پردازش توسط پرس و جو دارد. با توجه به M می‌توان بهره‌وری یک پرس و جو را به چند قسمت شکست. به عنوان مثال اگر برای پرس و جوی i مقادیر C_i و P_i به ترتیب برابر با ۱ و ۱۰۰ و $M = 20$ باشد، آن گاه بهره‌وری این پرس و جو برابر با $(1\times20)/100 = 0.2$ است. حال اگر بخواهیم بهره‌وری این پرس و جو را به ۲ پرس و جوی q_{i1} و q_{i2} بشکنیم می‌توان ۲ پرس و جوی i_1 و i_2 از این پرس و جو تولید کرد به طوری که گراف یا نقشه پرس و جوی آنها دقیقاً مانند پرس و جوی i باشد. مجموع پنجره اجرایی (W_e) این دو پرس و جو (M) باید برابر با ۲۰ باشد. فرض کنیم برای این دو پرس و جو مقادیر W_e به ترتیب برابر با $m_1 = 5$ و $m_2 = 15$ است. بنابراین بهره‌وری هر یک به ترتیب برابر با $(1\times5)/100 = 0.05$ و $(1\times15)/100 = 0.15$ است و در نتیجه مجموع بهره‌وری این دو پرس و جو برابر با پرس و جوی اولیه است یعنی $u_{i1} + u_{i2} = 0.05 + 0.15 = u_i$.

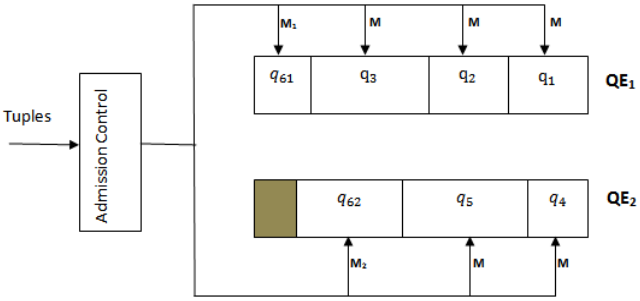
تعریف ۲) بهره‌وری QE: بهره‌وری هر موتور پرس و جوی j (QE_j) برابر با مجموع بهره‌وری پرس و جوهای انتساب داده شده به آن موتور پرس و جو است.

تعریف ۳) فضای خالی QE: با توجه به این که در هر موتور پرس و جو از الگوریتم زمان‌بندی بی‌درنگ EDF استفاده شده است به منظور تضمین مهلت‌ها بهره‌وری موتور پرس و جو نباید بیش از یک باشد. بنابراین فضای خالی هر موتور پرس و جو به صورت $E_{QEj} = 1 - U_{QEj}$ تعریف می‌شود. موتور پرس و جو پر محسوب می‌شود اگر و فقط اگر $E_{QEj} = 1$.

تعریف ٤) جایگیری پرس و جو در QE: فرض کنیم بهره‌وری پرس و جوی q_i برابر با u_i و فضای خالی موتور پرس و جوی j برابر با E_{QEj} باشد. اگر $u_i < E_{QEj}$ باشد آن گاه q_i در QEj جای می‌گیرد، در غیر این صورت q_i نمی‌تواند در QEj قرار گیرد و توسط آن اجرا شود.

انتساب‌دهنده پس از دریافت هر پرس و جو ابتدا طبق الگوریتم اولین جایدهی آن پرس و جو را به یکی از موتورهای پرس و جو انتساب می‌دهد. اگر انتساب پرس و جو در حالت عادی با شکست مواجه شد پرس و جو به دو یا چند پرس و جوی با بهره‌وری کوچک‌تر با شرایط ارائه‌شده در بالا شکسته می‌شود و هر یک از پرس و جوهای جدید به یکی از موتورهای پرس و جو انتساب داده می‌شود. در ادامه نحوه‌ی شکستن پرس و جو و محاسبه پنجره اجرایی پرس و جوهای جدید تولیدشده شرح داده می‌شود.

محاسبه اندازه پنجره اجرایی در پرس و جوهای شکسته‌شده به این صورت انجام می‌شود که ابتدا موتورهای پرس و جو بر حسب E_{QEj} به صورت نزولی مرتب شده و سپس اولین موتور پرس و جو برای انتساب کپی اول یعنی q_{i1} انتخاب می‌شود. میزان بهره‌وری q_{i1} برابر با $E_{QE_{first}}$ است. بنابراین پنجره اجرایی q_{i1} (m_{i1}) برابر با $[P_i \times E_{QE_{first}} / C_i]$ می‌باشد و q_{i1} به QE_{first} با پنجره اجرایی m_{i1} انتساب داده می‌شود. پنجره اجرایی باقیمانده برابر با $m_x = M - m_{i1}$ است. سپس موتور پرس و جوی دوم از این لیست مرتب‌شده نزولی برای انتساب انتخاب می‌شود. اگر $(C_i \times m_x / P_i) < E_{QE_{sec}}$ بود آن گاه q_{i2} با پنجره اجرایی $m_{i2} = m_x$ به $E_{QE_{sec}}$ انتساب داده می‌شود و کار تمام است، در غیر این صورت q_{i2} با پنجره اجرایی $m_{i2} = [P_i \times E_{QE_{sec}} / C_i]$ ایجاد می‌شود و باقیمانده پنجره اجرایی یعنی $m_x = M - m_{i1} - m_{i2}$ باید در ادامه توسط کپی‌های بعدی پوشش داده شود. این کار ادامه پیدا می‌کند تا زمانی که $m_x = 0$ شود. اگر موتور پرس و جوی دیگری باقی نماند و $m_x > 0$ باقی ماند، آن گاه پرس و جوی مد نظر نمی‌تواند به هیچ یک از موتورهای پرس و جو



شکل ۴: نحوه ارسال تاپل‌ها از کنترل پذیرش به پرس و جوها.

انتساب داده شود. شبه‌کد الگوریتم انتساب در زیر نشان داده شده است

```
Assignment (Query qi)
{
  Assign qi to a query based on First Fit
  If it is assigned then return success
  QEs = Sort QEs based on EQE in descending order
  mx = M, u = (Ci*M)/Pi, k = 1
  foreach QEj in QEs {
    if (EQEj < u) {
      m = roof ((Pi*EQEj)/Ci)
      mx = mx – m
      qik = Copy new query from qi with We = m
      Assign qik to QEj
      u = u – (Ci*m)/Pi
      k = k + 1
    }//end if
    else{
      qik = Copy new query from qi with We = mx
      Assign qik to QEj
      Return success
    }//end else
  }//end for
  return fail
}
```

برای این که این روش را بتوان به طور کامل پیاده‌سازی کرد باید ایجاد کپی‌های پرس و جو و شکستن بار در آن را به اطلاع Admission Control رساند زیرا Admission Control وظیفه توزیع بار را بر عهده دارد. در Admission Control وقتی یک تاپل جدید رسید باید آن را در صف پرس و جوهای مربوطه کپی کند. برای پرس و جوهایی که بار آنها شکسته می‌شود، کپی‌های ایجاد شده‌ی آن پرس و جو در موتورهای پرس و جوی متفاوتی اجرا می‌گردند. حال اگر تاپل مربوط به این پرس و جو رسیده باشد آن گاه این تاپل باید تنها به صف یکی از این کپی‌ها منتقل شود.

به عنوان مثال در شکل ۴ اگر دو موتور پرس و جو داشته باشیم و اگر از $q_{۶}$ در هیچ یک از موتورهای پرس و جو جای نگیرد، آن گاه دو کپی $q_{۶۱}$ و $q_{۶۲}$ با پنجره اجرایی به ترتیب $M_{۶۱}$ و $M_{۶۲}$ ایجاد می‌شود به طوری که $M = M_{۶۱} + M_{۶۲}$ است. آن گاه از M تاپلی که به Admission Control می‌رسد $M_{۶۱}$ تای آن به $q_{۶۱}$ و $M_{۶۲}$ تای آن به $q_{۶۲}$ منتقل می‌شود.

۴-۳ کنترل پذیرش داده

کنترل پذیرش از طریق واسط ورودی تاپل‌های ورودی را دریافت می‌کند و دو وظیفه مهم بر عهده دارد: کاهش بار و توزیع بار. پارامترهای کاهش بار توسط مانیتورینگ مهلت و پارامترهای توزیع بار توسط انتساب‌دهنده دریافت می‌شود.

در شرایطی که بار کاری سیستم بالا باشد ممکن است سیستم توانایی پردازش همه تاپل‌ها را نداشته باشد و میزان از دست رفتن مهلت‌ها افزایش یابد. در چنین شرایطی نیاز است که که بار کاری سیستم در هنگام ورود داده در قسمت کنترل پذیرش داده کاهش یابد که میزان کاهش بار توسط پارامتر $۰ \le \alpha \le ۱$ مشخص می‌شود. مقدار $\alpha = ۱$ به این مفهوم است که همه تاپل‌های ورودی به پرس و جوها تحویل داده می‌شوند و به ازای $\alpha = ۰$ همه تاپل‌های ورودی دور انداخته می‌شوند. هر چه مقدار α نزدیک‌تر به ۱ باشد تاپل‌های کمتری دور انداخته می‌شوند و بالعکس هر چه مقدار α نزدیک‌تر به صفر باشد تاپل‌های بیشتری دور انداخته می‌شوند.

وظیفه دیگری که کنترل پذیرش بر عهده دارد ارسال تاپل‌ها به پرس و جوهای منتظر برای آن تاپل‌ها است. با توجه به بخش قبلی هر پرس و جو می‌تواند در چند ماشین به طور موازی اجرا شود و این کار با ایجاد چند کپی از یک پرس و جو ایجاد می‌شود. البته اگر پرس و جو در حالت عادی m تاپل دریافت کند آن گاه کپی اول به تعداد $m_{۱}$ تاپل، کپی دوم $m_{۲}$ تاپل و به همین ترتیب کپی xام m_x تاپل از این m تاپل ورودی دریافت می‌کند به طوری که $\sum_{i=۱}^{x} m_i = M$ برقرار است. بنابراین کنترل پذیرش باید از نسبت دریافت تاپل‌ها توسط کپی‌های هر پرس و جو آگاه باشد. نسبت دریافت تاپل‌های ورودی برای پرس و جویی که بر مبنای بهره‌وری شکسته می‌شوند و چند کپی از آن پرس و جو ایجاد می‌شود توسط انتساب‌دهنده مشخص می‌شود و کنترل پذیرش تنها این نسبت‌ها را دریافت می‌کند و با توجه به آنها تاپل‌های ورودی را به پرس و جوها می‌دهد.

به عنوان مثال اگر یک پرس و جو شکسته شده باشد و کپی‌های آن پرس و جو در موتور پرس و جوی شماره ۱ و ۲ پخش شده باشد به طوری که سهم کپی اول از تاپل‌های دریافتی پرس و جو برابر با ۰/۴ و سهم کپی دوم از تاپل‌های ورودی پرس و جو ۰/۶ باشد، آن گاه از هر ۵ تاپل ورودی ۲ تای آن وارد کپی اول در موتور پرس و جوی ۱ و ۳ تای آن وارد کپی دوم در موتور پرس و جوی ۲ می‌شود.

۴-۴ مانیتورینگ مهلت

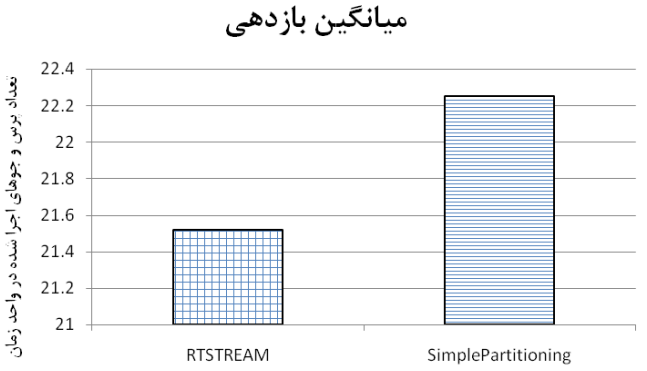
هر پرس و جو پس از اجرا میزان مهلت از دست رفته را به مانیتورینگ مهلت گزارش می‌دهد. میزان مهلت از دست رفته عددی بزگ‌تر- مساوی صفر یا کوچک‌تر- مساوی یک است. مانیتورینگ مهلت بر اساس دو میانگین میزان افزایش یا کاهش بار و مقدار α نهایی را محاسبه می‌کند. میانگین اول مربوط به میزان مهلت‌های از دست رفته اخیر در بازه کوچک و میانگین دوم مربوط به میزان مهلت‌های از دست رفته اخیر در بازه طولانی‌تر است. محاسبه α در (۳) نشان داده شده است

$$\Delta = -۱ \times (\beta_{۱}(Avg_{miss}(r_{۲}) - \max_{miss}) + \beta_{۲}(Avg_{miss}(r_{۱}) - \max_{miss})) \quad (۳)$$
$$\alpha = \alpha + \Delta$$

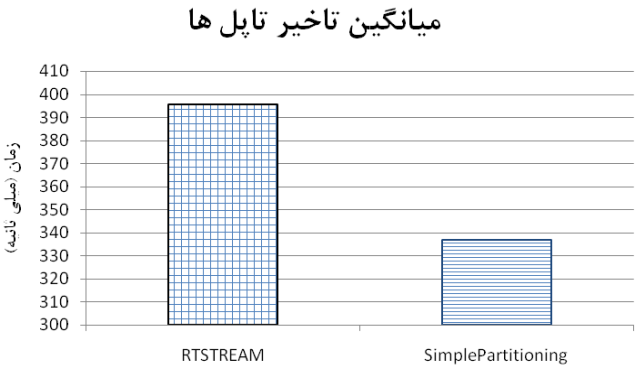
در (۳) $Avg_{miss}(r_{۲})$ مشخص‌کننده میانگین از دست رفتن مهلت در $r_{۲}$ تکرار قبلی، $Avg_{miss}(r_{۱})$ میانگین از دست رفتن مهلت در $r_{۱}$ تکرار قبلی، $\max_{miss}$ حداکثر میزان از دست رفتن مهلت قابل قبول و $\beta_{۱}$ و $\beta_{۲}$ مشخص‌کننده نسبت تأثیر میانگین‌ها است. همچنین شرط $\beta_{۱} + \beta_{۲} = ۱$ و $r_{۱} < r_{۲}$ باید برقرار باشد. مطابق با فرمول مقدار جدید α پس از جمع Δ با مقدار قبلی خود به روز رسانی می‌شود. در صورتی که مقدار جدید α بزرگ‌تر از یک شد آن گاه $\alpha = ۱$ و در صورتی که مقدار جدید α کوچک‌تر از صفر شد آن گاه $\alpha = ۰$ می‌شود.

۵- ارزیابی سیستم پیشنهادی

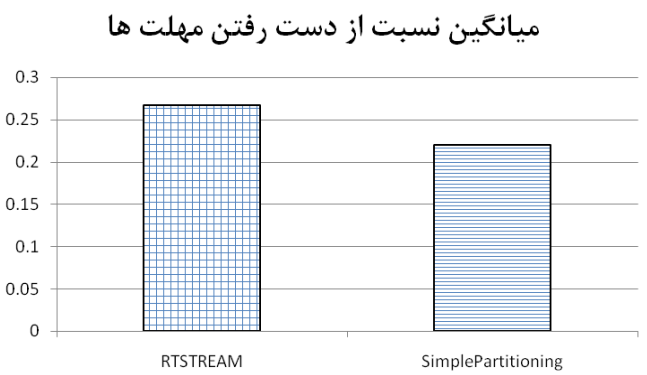
این بخش شامل دو سری آزمایش است که آزمایش اول مربوط به



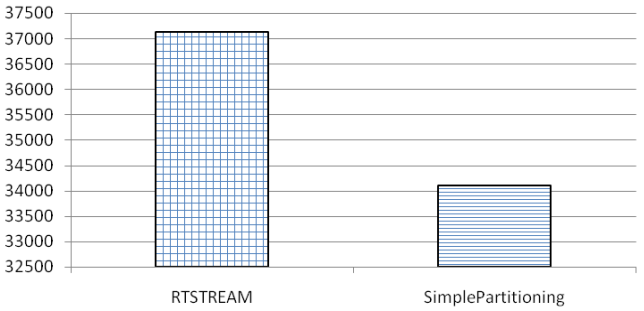
شکل ۵: مقایسه میانگین بازدهی.



شکل ۷: میانگین تأخیر تاپل‌ها در RTSTREAM و SimplePartitioning.



شکل ۶: مقایسه میانگین از دست رفتن مهلت‌ها.



شکل ۸: مقایسه میانگین از دست رفتن تاپل‌ها.

مقایسه سیستم رویکرد افرازی ساده[1] با سیستم RTSTREAM و آزمایش دوم مقایسه سیستم پیشنهادی در این مقاله با رویکرد افرازی ساده است. رویکرد افرازی برتری خود را به RTSTREAM نشان داده است [۱۵] و برتری روش شبه‌افرازی پیشنهادی این مقاله نسبت به رویکرد افرازی نشان‌دهنده کارایی بالاتر این سیستم نسبت به RTSTREAM است.

۵-۱ آزمایش اول: مقایسه SimplePartitioning با RTSTREAM

در این قسمت رویکرد افرازی ساده با RTSTREAM مقایسه شده است. در ابتدا تنظیمات آزمایش و سپس نمودارهای مربوط به نتایج نشان داده شده است.

۵-۱-۱ تنظیمات آزمایش

آزمایش‌ها بر روی یک سیستم دوهسته‌ای ۱٫۷۳ با یک گیگابایت حافظه اصلی انجام شده است. سیستم عامل آن ویندوز ۷ است و برنامه به زبان جاوا با کتابخانه‌های JDK۶ در محیط Netbeans۶.۸ نوشته شده است.

تعداد موتورهای پرس و جو برابر با ۲ است و ۱۷ پرس و جو به صورت دستی تعریف شده و همه پرس و جوها به صورت دوره‌ای هستند. مهلت و دوره‌ی تمام پرس و جوها با هم برابرند. دوره‌ی هر پرس و جو ۱۲ برابر زمان تخمینی پرس و جو درنظر گرفته شده است. از ۱۷ پرس و جو ۷ تای آن دارای ۴ عملگر، ۸ تای آن دارای ۳ عملگر و ۲ پرس و جوی دیگر نیز دارای ۲ عملگر است که از این بین ۵ پرس و جو حاوی عملگر Join است. سه عملگر Join، Filter و Project وجود دارد و Join ها دودویی هستند. زمان شبیه‌سازی ۲۰۰۰۰ میلی‌ثانیه است و برای هر سیستم ۷ بار آزمایش انجام شده و نتیجه میانگین ۷ آزمایش است.

۵-۱-۲ مقایسه نتایج

در این قسمت نتایج مقایسه بین دو سیستم ذکرشده به صورت میانگین نمایش داده شده است. این دو سیستم از نظر بازدهی (تعداد کار انجام‌شده در ثانیه)، نسبت از دست رفتن تاپل‌ها، میانگین تأخیر تاپل‌ها، میانگین از دست رفتن تاپل‌ها و میانگین تاپل‌های خروجی مقایسه شده‌اند.

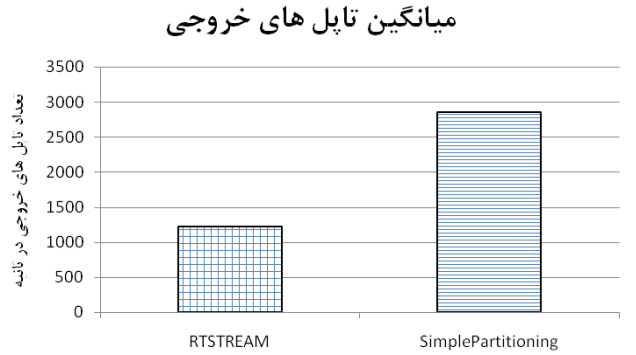
در شکل ۵ میانگین بازدهی سیستم‌ها که تعداد کارهای انجام‌شده در واحد زمان (ثانیه) است مقایسه شده است. میانگین تعداد پرس و جوهای انجام‌شده در SimplePartitioning بیشتر از RTSTREAM است. پرس و جوها به صورت دوره‌ای فعال می‌شوند و پس از فعال‌شدن تا مهلت تعریف‌شده باید کار خود را تمام کنند. بنابراین تعداد پرس و جوهای اجراشده بستگی به میزان دوره‌ها و مهلت پرس و جوها دارد. با این حال با توجه به این که بازدهی در SimplePartitioning بیشتر است انتظار می‌رود که نسبت از دست رفتن مهلت‌ها نیز در این سیستم در مقایسه با RTSTREAM نیز کمتر باشد که این در نمودار شکل ۶ نشان داده شده است.

شکل ۷ مقایسه میانگین تأخیر تاپل‌ها را نشان می‌دهد. تأخیر هر تاپل برابر با اختلاف زمان دریافت تاپل در خروجی با زمان دریافت آن تاپل توسط سیستم در ورودی است. کمتربودن میانگین تأخیر تاپل‌ها در SimplePartitioning نشان‌دهنده این است که در حالت میانگین ۱۷ پرس و جو سریع‌تر نسبت به RTSTREAM اجرا شده‌اند و نشان‌دهنده بهبود زمان پاسخ در SimplePartitioning نیز است.

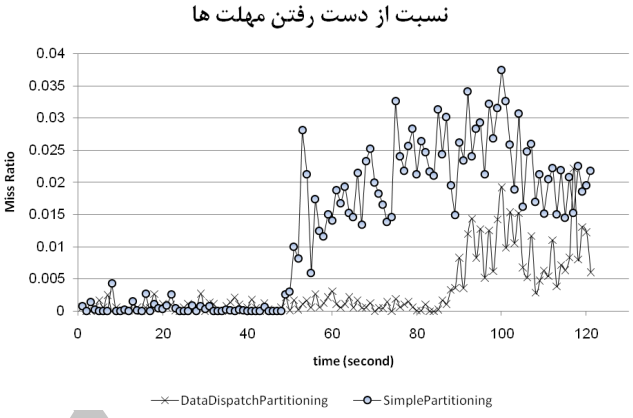
مقایسه میانگین از دست رفتن تاپل‌ها در شکل ۸ نشان داده شده است. از دست رفتن تاپل‌ها برابر با تعداد تاپل‌هایی است که پرس و جو نتوانسته است از پنجره m تایی پردازش کند. نمودار میانگین از دست رفتن تاپل‌ها نشان می‌دهد که در SimplePartitioning تعداد تاپل‌های کمتری نسبت به RTSTREAM بدون پردازش رها شده‌اند.

با توجه به این نکته که تعداد پرس و جوهای اجراشده در

1. Simple Partitioning



شکل ۹: مقایسه میانگین تاپل‌های خروجی.



شکل ۱۰: مقایسه نسبت از دست رفتن مهلت‌ها.

SimplePartitioning بیشتر و میزان از دست رفتن تاپل‌ها کمتر است انتظار می‌رود که تعداد خروجی‌های تولیدشده در این روش نیز بیشتر باشد که در شکل ۹ این مقایسه به صورت نمودار نشان داده شده است.

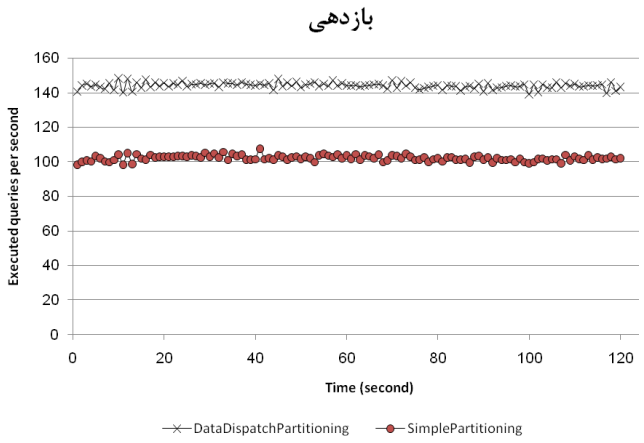
۵-۲ آزمایش دوم: مقایسه DataDispatchPartitioning با SimplePartitioning

در این بخش سیستم مدیریت جریان داده بی‌درنگ چندپردازنده‌ای ارائه‌شده در این مقاله با زمان‌بندی شبه‌افرازی با روش رویکرد افرازی ساده مقایسه شده است. در نمودارها روش شبه‌افرازی با نام DataDispatchPartitioning نمایش داده شده است و مقایسه‌ها برتری DataDispatchPartitioning به SimplePartitioning را نشان می‌دهد. در ادامه تنظیمات آزمایش ذکر شده و نتایج و نمودارهای مربوط نشان داده شده است.

۵-۲-۱ تنظیمات آزمایش

آزمایش‌ها بر روی یک سیستم با پردازنده Corei۷ با ۶ گیگابایت حافظه اصلی انجام شده است. سیستم عامل آن ویندوز ۷ است و برنامه به زبان جاوا با کتابخانه‌های JDK۶ در محیط Netbeans۶.۸ توسعه داده شده است.

تعداد موتورهای پرس و جو برابر با ۳ و تعداد پرس و جوها برابر با ۱۰ است و همه پرس و جوها به صورت دوره‌ای هستند. مهلت و دوره پرس و جوها با هم برابر و بهره‌وری همه پرس و جوها تقریباً برابر با ۰/۳ است. از ۱۰ پرس و جو ۶ تای آن دارای ۵ عملگر، ۲ پرس و جو ۶ عملگر و ۲ پرس و جو نیز دارای ۷ عملگر هستند. عملگرهای Filter، Join، Project، اجتماع، اشتراک و Count در پرس و جوها استفاده شده است. زمان آزمایش ۱۲۰۰۰۰ میلی‌ثانیه است و برای هر سیستم ۱۰ بار آزمایش انجام شده و نتیجه نهایی مقایسه میانگین این ۱۰ آزمایش است.



شکل ۱۱: مقایسه بازدهی DataDispatchPartitioning و SimplePartitioning در طول زمان.

۵-۲-۲ نتایج آزمایشات

در این قسمت با توجه به تنظیمات ذکرشده در قسمت قبلی رویکرد افرازی همراه با توزیع بار با نام DataDispatchPartitioning در مقابل رویکرد SimplePartitioning مقایسه شده است. لازم به ذکر است که SimplePartitioning ساده نسبت به RTSTREAM که تک‌پردازنده‌ای است نتایج بهتری را از خود نشان داده [۱۶] و نتایج مقایسه‌ها هم به صورت لحظه‌ای و هم به صورت میانگین نمایش داده شده است. در نمودارهای لحظه‌ای هر یک ثانیه یک بار میانگین بازه یک ثانیه‌ای نشان داده شده است.

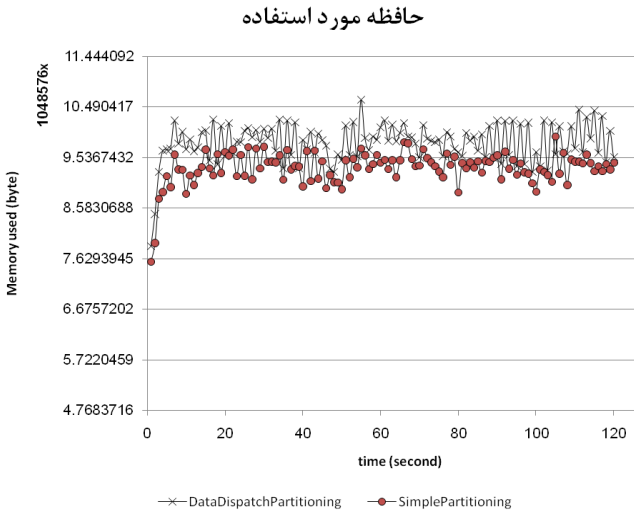
میزان نسبت از دست رفتن مهلت‌ها در نمودار شکل ۱۰ مقایسه شده‌اند. همان طور که شکل نشان می‌دهد میزان از دست رفتن مهلت‌ها تا ثانیه تقریباً ۵۰ نزدیک به هم است اما در ادامه رویکرد ارائه‌شده نتیجه بهتری از خود نشان می‌دهد.

در این نمودار پس از گذشت مدت زمانی بار کاری سیستم کمی بالا می‌رود. با افزایش بار کاری نسبت از دست رفتن مهلت‌ها در SimplePartitioning افزایش می‌یابد و در این حالت می‌ماند اما DataDispatchPartitioning نسبت به این بار کاری مقاومت بیشتری از خود نشان داده است. علاوه بر این که نسبت از دست رفتن مهلت‌ها در رویکرد ارائه‌شده کمتر است، میزان تغییرات از دست رفتن مهلت این روش نیز نسبت به افرازی ساده کمتر است.

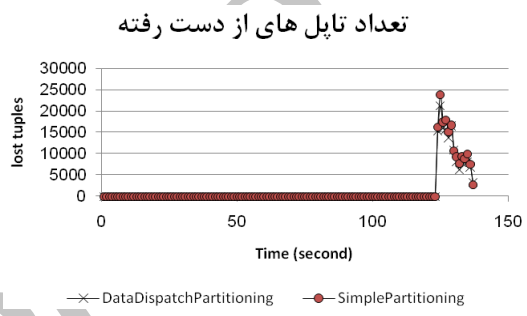
تعداد پرس و جوهای انجام‌شده در یک ثانیه در شکل ۱۱ نشان داده شده است. با توجه به شکسته‌شدن پرس و جوها و کمتر از دست رفتن مهلت‌ها در رویکرد ارائه‌شده، انتظار می‌رود میزان بازدهی در آن نسبت رویکرد افرازی ساده بالاتر باشد که نمودار شکل ۱۱ این مطلب را نشان می‌دهد. در طول زمان همیشه میزان بازدهی در رویکرد ارائه‌شده نسبت به افرازی ساده بیشتر بوده است.

شکل ۱۲ نشان‌دهنده میزان حافظه مصرفی در طول زمان است. در روش پیشنهادی چند پرس و جو ممکن است شکسته شود. بنابراین تعداد پرس و جوها افزایش می‌یابد و بنابراین میزان حافظه مصرفی نیز افزایش می‌یابد اما میزان حافظه مصرفی اختلافی زیاد با حالت افرازی ساده ندارد.

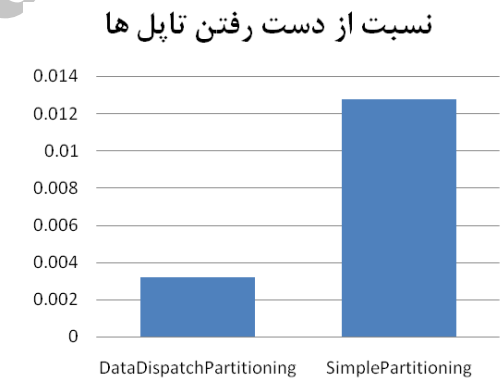
میزان تاپل‌های از دست رفته برابر با تاپل‌هایی است که به دلیل پربودن بافر پرس و جوها دور انداخته شده‌اند. از دست رفتن تاپل‌ها برای دو روش مورد مقایسه در طول زمان در شکل ۱۳ نشان داده شده است. میزان از دست رفتن تاپل‌ها تا ثانیه ۱۰۶ در هر دو رویکرد یکسان است. اما از این زمان به بعد با توجه به پرشدن بافر پرس و جوها ناگهان



شکل ۱۲: مقایسه میزان حافظه مصرفی.



شکل ۱۳: مقایسه تاپل‌های از دست رفته.

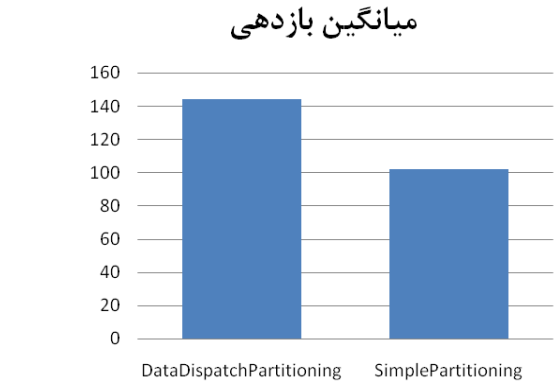


شکل ۱۴: متوسط نسبت از دست رفتن تاپل‌ها.

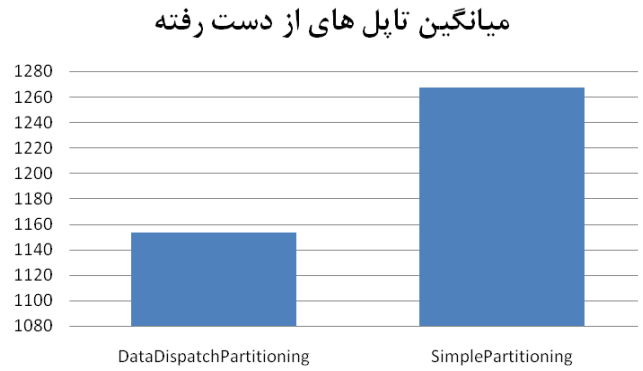
میزان از دست رفتن تاپل‌ها افزایش یافته است که در ادامه با خالی‌شدن بافر پرس و جوها از دست رفتن تاپل‌ها کاهش می‌یابد. البته همان طور که از نمودار پیداست از دست رفتن تاپل‌ها در رویکرد ارائه‌شده همیشه کمتر از رویکرد افرازی ساده بوده است.

شکل ۱۴ میانگین نسبت از دست رفتن مهلت‌ها را نشان می‌دهد. همان طور که نمودار نشان می‌دهد در DataDispatchPartitioning نسبت از دست رفتن مهلت‌ها برابر با ۰/۰۰۳ و در SimplePartitioning برابر با ۰/۰۱۲ است. این نتیجه نشان می‌دهد روش پیشنهادی به طور میانگین سه برابر کارایی بالاتری در ارتباط با نسبت از دست رفتن مهلت که مهم‌ترین پارامتر در سیستم‌های بی‌درنگ است داشته است.

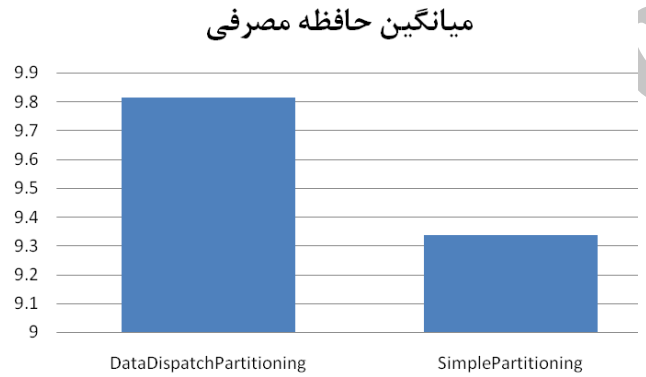
با توجه به شکل ۱۵ در رویکرد ارائه‌شده در واحد زمان به طور میانگین ۱۴۳/۹۵ کار انجام گرفته و میانگین کارهای انجام‌شده در افرازی ساده برابر با ۱۰۲/۱۳۵ کار در ثانیه است. همان طور که قبلاً ذکر شد با توجه به میزان نسبت از دست رفته و شکستن پرس و جوها بازدهی شبه‌افرازی



شکل ۱۵: میانگین بازدهی دو روش DataDispatchPartitioning و SimplePartitioning.



شکل ۱۶: میانگین تاپل‌های از دست رفته در دو روش DataDispatchPartitioning و SimplePartitioning.



شکل ۱۷: میانگین حافظه مصرفی در دو روش DataDispatchPartitioning و SimplePartitioning.

بیشتر از افرازی ساده است.

نمودار شکل ۱۶ مربوط به تعداد تاپل‌های از دست رفته در طول زمان آزمایش است. در روش DataDispatchPartitioning جمع تاپل‌های از دست رفته برابر با ۱۱۵۳ تاپل است و در برابر SimplePartitioning که ۱۲۶۷ تاپل است نتیجه بهتری را نشان می‌دهد و میزان کمتری تاپل از دست داده است.

در نهایت شکل ۱۷ مربوط به میانگین میزان حافظه مصرفی در ثانیه برای دو روش مورد مقایسه است. با توجه به شکسته‌شدن پرس و جو و افزایش تعداد پرس و جوها در رویکرد افرازی با توزیع بار انتظار می‌رود که میزان استفاده از حافظه مصرفی بیشتر باشد. میزان حافظه مصرفی در رویکرد ارائه‌شده برابر با ۹/۸۱ مگابایت و در رویکرد افرازی ساده برابر با ۹/۳۳ مگابایت است.

همان طور که نتایج نشان می‌دهد این روش به خوبی توانسته است

میزان از دست رفتن مهلت‌ها را کاهش، میزان بازدهی را افزایش و میزان نسبت از دست رفتن تاپل‌ها را کاهش دهد. برای بهبود این پارامترها مقداری حافظه مصرفی نسبت به رویکرد افرازی ساده افزایش یافته است. با توجه به افزایش ظرفیت حافظه این مقدار افزایش حافظه می‌تواند قابل قبول باشد.

٦- نتیجه‌گیری

در سیستم‌های مدیریت جریان داده بی‌درنگ پرس و جوها ذخیره شده و این پرس و جوها بر روی جریان‌های داده ورودی اجرا می‌شوند. هر پرس و جو دارای مهلتی است و تحت مهلت تعیین‌شده باید نتیجه در خروجی تولید کند. سیستم مد نظر یک سیستم بی‌درنگ نرم است و هر چه نسبت از دست رفتن مهلت‌ها کمتر باشد این سیستم ارزش بالاتری را به سیستم برمی‌گرداند.

با توجه به بالابودن میزان بار شبکه چنین سیستم‌هایی نیاز به قدرت محاسباتی بالایی دارند و یکی از راه‌های افزایش قدرت محاسباتی استفاده از چندپردازنده در چنین سیستم‌هایی است. در نمونه‌های پیاده‌سازی شده دیگر استفاده از چندپردازنده در نظر گرفته نشده است. بنابراین این سیستم با افزایش تعداد پردازنده‌ها توانسته است عملکرد سیستم را بهبود بخشد.

الگوریتم‌های زمان‌بندی چندپردازنده‌ای بی‌درنگ از الگوریتم‌های زمان‌بندی تک‌پردازنده‌ای پیچیده‌تر هستند و الگوریتم چندپردازنده‌ای بهینه در حالت کلی وجود ندارد. برای الگوریتم‌های چندپردازنده‌ای دو رویکرد کلی وجود دارد: رویکرد افرازی و سراسری. رویکرد افرازی نسبت به سراسری ساده‌تر است ولی قابل بهینه‌سازی با اعمال محدودیت‌ها نیست. در اینجا سعی شده است که با توجه به ماهیت سیستم، بتوان با تغییر الگوریتم زمان‌بندی افرازی، کارایی را افزایش داد. این هدف از طریق ایجاد کپی‌های پرس و جو و توزیع بار ورودی حاصل شده است که سبب کاهش از دست رفتن مهلت‌ها نسبت به زمان‌بندی افرازی ساده گردیده است.علاوه بر کاهش میزان از دست رفتن مهلت‌ها، میزان بازدهی افزایش یافته و میزان از دست رفتن تاپل‌ها کاهش یافته است.

مراجع

[1] A. Arasu, et al., "STREAM: the stanford stream data manager," in *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, *SIGMOD'03*, p. 265, New York, USA, Jun. 2004.

[2] S. Babu and J. Widom, "Continuous queries over data streams," *ACM SIGMOD Record*, vol. 30, no. 3, pp. 109-120, 2001.

[3] Y. Wei, S. H. Son, and J. A. Stankovic, "RTSTREAM: real-time query processing for data streams," in *Proc. 9th IEEE Int. Symp. On Object/Component/Service-Oriented Real-Time Distributed Computing*, pp. 141-150, 2006.

[4] S. Sven, L. Thomas, S. Daniel, and L. Wolfgang, "Real-time scheduling for data stream management systems," in *Proc. of the 17th Euromicro Conf. on Real-Time Systems*, pp. 167-176, 6-8 Jul. 2005.

[5] L. Hennadiy, C. Samarjit, and H. A. James, "Multiprocessor extensions to real-time calculus," in *Proc. 30th IEEE Real-Time Systems Symp., RTSS*, pp. 410-421, 1-4 Dec. 2009.

[6] P. L. Holman, On the Implementation of Pfair - Scheduled Multiprocessor Systems, Ph. D Thesis, Chapel Hill, 2004.

[7] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah, "A categorization of real - time multiprocessor scheduling problems and algorithms," *Handbook on Scheduling Algorithms, Methods, and Models*, vol. 30, pp. 1-19, 2004.

[8] A. A. Safaei, M. Alemi, M. Haghjoo, and S. Mohammadi, "Hybrid multiprocessor real-time scheduling approach," *International Journal of Computer Science Issues*, vol. 8, no. 2, pp. 171-178, Mar. 2011.

[9] D. Abadi, et al., "Aurora: a data stream management system," *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, *SIGMOD'03*, p. 666, New York, USA, Jun. 2004.

[10] S. Schmidt, H. Berthold, and W. Lehner, "Qstream: deterministic querying of data streams," in *Proc. of Int. Conf. on Very Large Data Bases*, pp. 1365-1368, 2004.

[11] S. Schmidt, T. Legler, and W. Lehner, "Robust real-time query processing with QStream," in *Proc. of the 31st Very Large Data Bases Conf.*, pp. 1299-1301, Trondheim, Norway, 2005.

[12] X. Li and H. Wang, "Adaptive real-time query scheduling over data streams," in *Proc. Int. Conf. on Very Large Databases, PhD Workshop*, Sep. 2007.

[13] A. A. Safaei and M. S. Haghjoo, "Parallel processing of continuous queries over data streams," *Distributed and Parallel Databases*, vol. 28, no. 2-3, pp. 93-118, Dec. 2010.

[14] A. A. Safaei and M. S. Haghjoo, "Dispatching of stream operators in parallel execution of continuous queries," *J. of Scheduling*, vol. 61, no. 3, pp. 619-641, 2010

[۱۵] م. عالمی، م. حق‌جو و ع. صفایی، "زمان‌بندی بی‌درنگ چندپردازنده‌ای در سیستم‌های مدیریت جریان داده بی‌درنگ،" *سومین همایش ملی مهندسی کامپیوتر و فناوری اطلاعات*، همدان، ۱۳۸۹.

مهدی عالمی تحصیلات خود را در مقطع کارشناسی علوم کامپیوتر در سال ۱۳۸۷ از دانشگاه تبریز و در مقطع کارشناسی ارشد مهندسی کامپیوتر در سال ۱۳۹۰ از دانشگاه علم و صنعت ایران ادامه داد و از سال ۱۳۹۲ دانشجوی دکتری مهندسی کامپیوتر دانشگاه شهید بهشتی می‌باشد. نام‌برده از سال ۱۳۸۹ تا کنون در پروژه موتور جستجوی ملی مشغول به فعالیت بوده است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: الگوریتم‌های موازی، سیستم‌های جریان داده، داده‌کاوی، بازیابی اطلاعات، و سیستم‌های توزیع‌شده.

مصطفی حق جو در سال ۱۳۵۶ مدرک کارشناسی علوم کامپیوتر خود را از دانشگاه شیراز و در سال ۱۳۵۸و ۱۳۵۹ مدرک کارشناسی ارشد و دانشوری را در رشته علوم کامپیوتر خود را از دانشگاه جورج واشنگتن در آمریکا دریافت کرد و مدرک دکتری علوم کامپیوتر را در سال ۱۳۷۴ از دانشگاه ملی استرالیا اخذ نمود. دکتر حق جو از سال ۱۳۶۰ تا سال ۱۳۶۹ در سازمان سنجش آموزش کشور فعالیت داشت و همچنین از سال ۱۳۶۳ تا سال ۱۳۹۲ عضو هیأت علمی دانشکده مهندسی کامپیوتر دانشگاه علم و صنعت ایران بود و اینک در دانشگاه بین‌المللی پیام نور کیش عضو هیأت علمی است. زمینه‌های علمی مورد علاقه ایشان شامل پایگاه داده توزیع شده، پایگاه داده‌های نوین، و سیستم‌های جریان داده می‌باشد.