علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۱، شماره ۱ (الف)، ۱۳۹۲
صفحات ۱۱-۲۹
مقاله عادی

پردازش موازی جریان داده‌ها

علی اصغر صفائی[۱] مصطفی سانیچی حق‌جو[۲]

[۱]دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی خواجه نصیرالدین طوسی، تهران، ایران
[۲]دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران

چکیده

طبیعت پیوسته‌ی جریان‌های داده و پرس‌وجوها از یک سو و نیاز اغلب کاربردهای جریان داده به برخط بودن پردازش‌ها از سوی دیگر، نیاز بـه مـوازی‌سـازی را هرچـه پُررنـگ‌تـر و جـدّی‌تـر می‌نماید. موازی‌سازی اصولاً از طریق دو رهیافتِ کنترل موازی و افراز داده‌ها عینیّت می‌یابد.
در این مقاله، پردازش موازی جریان‌های داده با هر دو رهیافت به چالش کشیده شده است. معماری یک سیستم مدیریت جریان داده موازی بر بستر سیستم‌های چندپردازه‌ای کـه متشـکل از چند ماشین مُحاسب یکسان است، و نحوه‌ی تعامل و همکاری این ماشین‌ها در اجرای موازی یک پرس‌وجوی پیوسته ارائه شده است. بیان صوری و اثبات صحّت عملکـرد سیسـتم بـه صـورت صوری، و ارزیابی کارایی آن به وسیله مدل‌سازی در شبکه‌های پتری انجام شده است. به منظور سازگاری بیشتر با طبیعت پیوسته‌ی پرس‌وجوها و جریان‌های داده، الگوریتم زمان‌بندی پیوسته و پویا به عنوان جایگزین زمان‌بندی رویدادگرا ارائه شده است که منجر به بهبود کارائی و همگرایی آن به سَمتِ مقدار بهینه گردید. علاوه بر این، مبتنی بر مدل محاسباتی نگاشت-کاهش کـه برای موازی‌سازی با رهیافت افراز داده‌ها در کاربردهای داده-محور و پردازش دسته‌ای ارائه شده بوده است، روش نگاشت-کاهش پیوسته نامتوازن نیز بـرای پـردازش مـوازی جریـان‌هـای داده ارائه شده است.

کلمات کلیدی: جریان داده، موازی‌سازی، زمان‌بندی رویدادگرا، زمان‌بندی پیوسته و پویا، مدل محاسباتی نگاشت-کاهش، تاخیر تاپل.

۱- مقدمه

در بسیاری از کاربردهای نوین اطلاعاتی، داده‌ها دیگر مانند سیستم‌هـای مـدیریت پایگاه داده[۱] به صورت مانا در رسانه ذخیره نمی‌شوند بلکه به صورت جریان‌هایی از دنباله‌های نامتناهی، سریع، غیرقابل پیش‌بینی و متغییر با زمان دریافت می‌شـوند. از آنجا که سیستم‌های مدیریت پایگاه داده سنتی قادر به پردازش جریان‌هـای داده با چنین ویژگی‌هایی نیستند سیستم‌هایی به نام سیستم مدیریت جریـان داده‌هـا[۲] برای این منظور ارائه شده‌اند [۱] [۲] [۳].

با توجه به ویژگی‌های جریان‌های داده، در اغلـب کاربردهـا نظیـر نظـارت[۳] بـر شبکه‌های کامپیوتری، شبکه‌های حسگر و مالی نیاز به پردازش سریع و بـی‌درنـگ داده‌ها وجود دارد [۴] [۵]. این نیاز از یک منظـر مسـتلزم انجـام سـریع پـردازش پرس‌وجو روی جریان داده است.

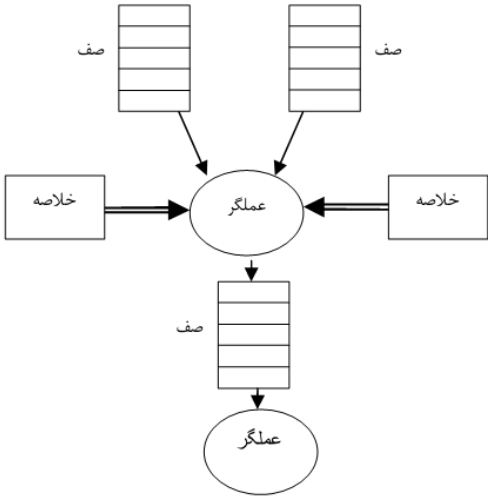
ماهیت پیوسته، سریع، انفجاری و غیرقابل پیش‌بینی جریان‌های داده‌ی ورودی از یک سو و پیوسته بودن پرس‌وجوها روی جریان‌های داده از سـوی دیگـر سـبب می‌شوند که یک پردازنده واحد غالباً قادر به پردازش این حجم از داده‌ها نباشد. در این راستا، به‌کارگیری مکانیزم‌هایی مانند کاهش بار[۴]، کنترل پذیرش[۵] و نظایر آنهـا ناگزیر است [۶]. با این همه، تک‌پردازنده‌ای بـودن سیسـتم مـدیریت جریـان داده منجر به کاهش کیفیت پاسخ‌ها از نظر تاخیر پردازش و نیز از دست دادن برخی از المان‌های داده خواهد شد.

در این مقاله، برای حل این مشکل و رفع گلوگاه تک‌پردازنده‌ای بودن سیستم، موازی‌سازی پردازش پرس‌وجوها بر بستر یک سیستم چندپردازنده‌ای ارائه می‌شود. این موازی سازی می‌تواند در جهت نیل به اهداف پردازش بی‌درنـگ پـرس‌وجوهـا بسیار مفید باشد.

موازی‌سازی در پردازش پرس‌وجو مزایایی از قبیل کارایی[۶] بهتر، دسترس‌پذیری[۷] و توسعه‌پذیری[۸] را به دنبال خواهد داشت. موازی‌سازی پردازش پرس‌وجو شامل موارد زیر است [۷]:

الف) ترجمه پرس‌وجو به نقشه پرس‌وجو

در سیستم مدیریت جریان داده، برای هر پرس‌وجوی ثبت‌شده، نقشه پرس‌وجوی آن به صورت یک DAG ایجاد می‌شود که شامل عملگرهای آن پرس‌وجو و توالی آنها و نیز صفهایی برای بافر کردن تاپل‌های پردازش شده هر عملگر می‌باشد تا عملگر بعدی از آنها به عنوان ورودی خود استفاده کند [۱] [۲] (شکل ۱).



شکل ۱- نقشه پرس‌وجو برای پرس‌وجوهای پیوسته جریان‌های داده

هر تاپل، دنباله‌ای از عملگرها را پشت سر می‌گذارد که به آن مسیر *عملگر*[۹] می‌گویند [۸].

با توجه به اینکه حجم داده‌هایی که قرار است توسط هر پرس‌وجو پردازش شوند بسیار زیاد است، بهبودی جزئی در نقشه پرس‌وجو و اجرای موازی آن، منجر به تاثیر چشمگیری در کارایی سیستم خواهد شد. برای پردازش موازی پرس‌وجو، پس از ایجاد نقشه پرس‌وجوی بهینه برای پرس‌وجوی ثبت‌شده، نحوه موازی سازی تعیین می‌شود و درخت عملگرها که به طور موازی زمانبندی شده است برای اجرا روی داده‌ها به کار گرفته می‌شود [۹]. ایجاد نقشه پرس‌وجو و بهینه سازی آن در مراجع متعددی مورد بررسی قرار گرفته‌اند [۱۰] [۱۱] [۱۲].

ب) اجرای موازی نقشه پرس‌وجو

به طور کلی، موازی‌سازی را می‌توان با یکی از دو رویکرد زیر اعمال نمود [۹]:

۱- افراز داده‌ها[۱۰]:

کل داده‌هایی که باید پردازش شوند را به تکه‌هایی افراز کرده و هر تکه را برای اجرا به یک سیستم تخصیص می‌دهیم. در این روش، همه ماشین‌ها کار یکسانی را روی داده‌های متمایز (افراز شده) انجام می‌دهند.

۲- کنترل موازی[۱۱]:

کل داده‌ها در اختیار هر یک از سیستم‌ها قرار دارند اما هر سیستم بخشی از کار را روی داده‌ها انجام می‌دهد. به عبارتی، داده‌ها بین همه سیستم‌ها مشترک هستند اما کاری که هر سیستم روی داده‌ها انجام می‌دهد متمایز از بقیه است.

با توجه به اینکه در سیستم‌های مدیریت جریان داده به دلیل نامتناهی بودن جریان، کل داده در اختیار ما نیست، اصولاً رویکرد کنترل موازی برای انجام موازی‌سازی مناسب‌تر به نظر می‌رسد. در این مقاله ابتدا پردازش موازی پرس‌وجوهای پیوسته روی جریان‌داده‌ها با رویکرد کنترل موازی ارائه می‌شود و سپس نحوه‌ی به کارگیری رویکرد افراز داده‌ها (مدل محاسباتی نگاشت-کاهش) برای ایجاد توازی در پردازش جریان داده ارائه خواهد شد.

نوآوری‌های این مقاله عبارتند از:

- ارائه پردازش موازی پرس‌وجوهای پیوسته روی جریان‌ها داده با رویکرد کنترل موازی (بخش ۲)
- معماری سیستم مدیریت جریان داده موازی بر بستر سیستم‌های چندپردازه‌ای (بخش ۲-۱-۱)
- ارائه دید منطقی از گراف پرس‌وجو روی همه ماشین‌های محاسب در معماری موازی (بخش ۲-۱-۲)
- بیان صوری سیستم و اثبات صحّتِ عملکرد آن به صورت صوری
- الگوریتم‌های زمانبندی عملگرها برای اجرای موازی آنها (بخش‌های ۲-۱-۳ و ۲-۱-۴)
- تحلیل پیچیدگی پردازش موازی پرس‌وجو (بخش ۲-۱-۵)
- مدل‌سازی سیستم ارائه شده به وسیله شبکه‌های پتری و ارزیابی کارایی آن (بخش ۲-۱-۶)
- پردازش موازی جریان داده‌ها با رویکرد افراز داده‌ها و ارائه روش نگاشت-کاهش پیوسته‌ی نامتوازن (بخش ۲-۲)

در پایان، کارهای مرتبط را در بخش ۳ معرفی نموده و نتیجه‌گیری و کارهای آتی را در بخش ۴ ارائه می‌کنیم.

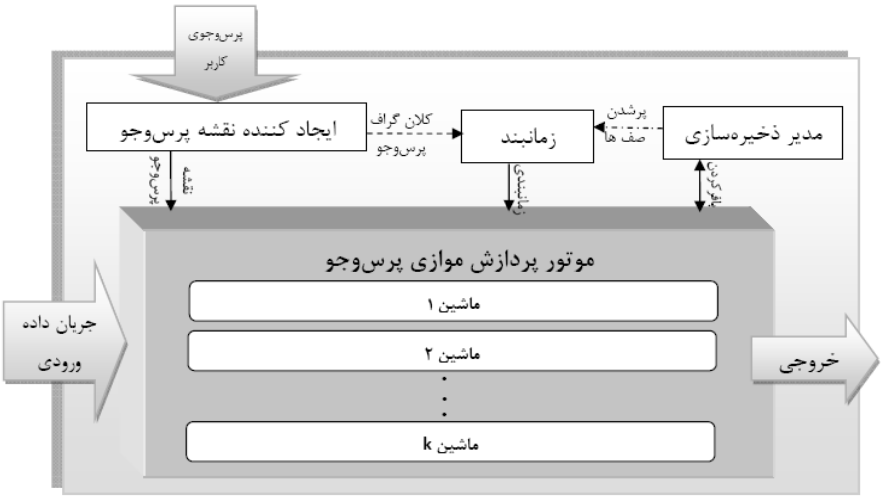
۲- پردازش موازی جریان‌داده‌ها

۲-۱- رویکرد کنترل موازی (موازی‌سازی اجرای پرس‌وجوها)

پرس‌وجوهای جریان داده‌ها عموما از نوع پیوسته هستند. برای هر پرس‌وجوی ثبت‌شده در سیستم مدیریت جریان داده، نقشه پرس‌وجوی آن ایجاد می‌شود و عملگرها پس از پردازش هر تاپل از صف ورودی، نتیجه را در صف خروجی خود قرار می‌دهند تا توسط عملگر بعدی نقشه پرس‌وجو مورد استفاده قرار گیرد.

از آنجا که با توجه به ماهیت پیوسته جریان داده و پیوسته بودن پرس‌وجو [۱۳] [۱۴]، فرآیند پردازش تاپل‌ها توسط هر عملگر باید به طور مستمر انجام شود، و از طرفی در نقشه پرس‌وجو چندین عملگر وجود دارد که باید به طور پشت سر هم روی نتایج عملگرهای قبلی خود کار کنند، لذا اجرای کل یک نقشه پرس‌وجو توسط یک پردازنده واحد چالش بزرگی در پردازش سریع پرس‌وجوها به شمار می‌رود. این بدان معنی است که یک واحد پردازنده همواره قادر نیست به طور همزمان عملگرهای مختلف را روی تاپل‌های جریان داده اجرا کند و گلوگاهی در پردازش سریع پرس‌وجوها محسوب می‌شود.

به عبارت دیگر در بسیاری از مواقع، نرخ ورود تاپل‌ها بیشتر از نرخ پردازش تاپل‌ها توسط موتور پردازش پرس‌وجو خواهد بود. در چنین شرایطی (شرایط اضافه‌بار[۱۲])، سیستم مجبور به کاهش بار یا کنترل پذیرش است (دور ریختن بخشی از داده‌ها) که در این صورت کیفیت نتایج کاهش می‌یابد [۶]. در صورتی که تعداد بیشتری از داده‌ها در صفهای ورودی بافر شوند، ارائه نتایج با تاخیر زیادی همراه خواهد بود که با نیازمندی‌های بی‌درنگی کاربردهای جریان داده در تضاد است.



شکل ۲- معماری سیستم پیشنهادی برای پردازش موازی پرس‌وجو روی جریان‌های داده

۲-۱-۱- معماری سیستم پیشنهادی

برای رفع گلوگاه تک‌پردازنده‌ای بودن سیستم در اجرای سریال عملگرهای نقشه پرس‌وجو، در این مقاله پردازش موازی پرس‌وجوها بر بستر یک سیستم چندپردازنده‌ای ارائه شده است. معماری سیستم پیشنهادی در شکل ۲ نمایش داده شده است.

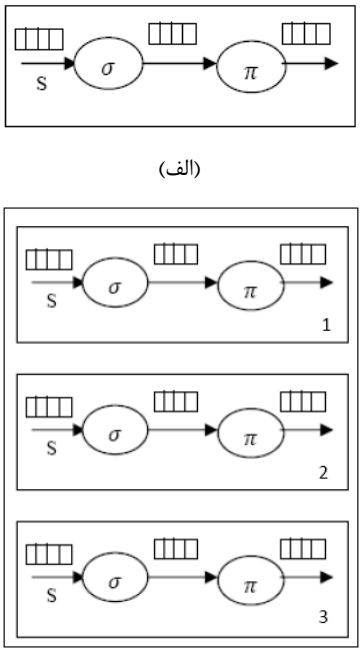
در موتور پردازش موازی پرس‌وجوی سیستم پیشنهادی K پردازنده (ماشین منطقی[۱۳]) یکسان داریم که با یکدیگر در ارتباط هستند. این ماشین‌ها می‌توانند ماشینهای فیزیکی باشند (مثلاً پردازنده‌های یک سیستم چندپردازنده‌ای یا گره‌های یک خوشه[۱۴] کامپیوتری) یا ماشین‌های مجازی (مثل ریسمان‌هایی که روی هر یک از هسته‌های[۱۵] یک پردازنده چند هسته‌ای اجرا می‌شوند). جزئیات سطح پایین بستری که این امکان را برای ما فراهم می‌کند خارج از حوزه بحث این مقاله است. فرض بر این است که در یک محیط چندپردازه‌ای[۱۶]، k ماشین منطقی داریم که می‌توانند به صورت کاملاً موازی[۱۷] با یکدیگر کار کنند (با یکدیگر ارتباط و تعامل نیز دارند). این ماشین‌ها امکان اجرای موازی عملگرهای پرس‌وجو را فراهم می‌کنند.

برای اجرای پرس‌وجو به صورت موازی روی این k ماشین منطقی، برای هر پرس‌وجوی ثبت‌شده در سیستم ارائه شده، ابتدا نقشه پرس‌وجوی آن ایجاد می‌شود. سپس k کپی دقیقا یکسان از این نقشه پرس‌وجو ایجاد و هر کدام به یکی از k ماشین منطقی ارسال می‌گردد. هدف از انجام این کار آن است که همه ماشین‌ها از نقشه پرس‌وجو (عملگرها و توالی آنها) مطلع شوند اما اینکه هر ماشین باید کدام عملگر را اجرا کند و چگونه با ماشین‌های دیگر ارتباط برقرار کند تا نقشه پرس‌وجو به طور موازی روی همه ماشین‌ها اجرا شود، توسط الگوریتم زمانبندی مشخص می‌شود (بخش ۳).

مثال ۱: اگر سه ماشین منطقی برای اجرای پرس‌وجوی Q_1 داشته باشیم، یک کپی از نقشه پرس‌وجو (شکل ۳-الف) به هر یک از این ماشین‌های منطقی تخصیص می‌یابد (شکل ۳- ب):

Q_1: SELECT x FROM s Where p;

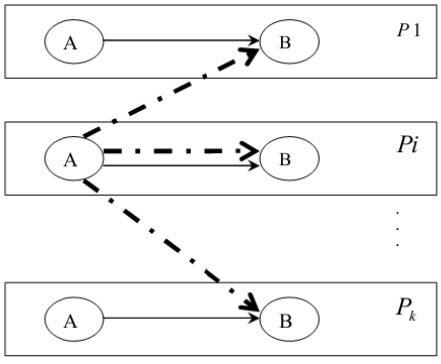
با فرض امکان ارتباط ماشین‌ها با یکدیگر، چنانچه در نقشه پرس‌وجو، عملگر A باید نتایج پردازش خود را به عملگر B بدهد، عملگر A از ماشین i، تاپل‌های خروجی خود را هم می‌تواند به عملگر B همان ماشین ارسال کند و هم به عملگر B در هر یک از ماشینهای دیگر (شکل ۴).



(الف)

(ب)

شکل ۳- (الف) ایجاد نقشه پرس‌وجو ی Q_1 و (ب) تخصیص کپی‌های نقشه پرس‌وجو به هریک از ماشین‌های منطقی در سیستم پیشنهادی



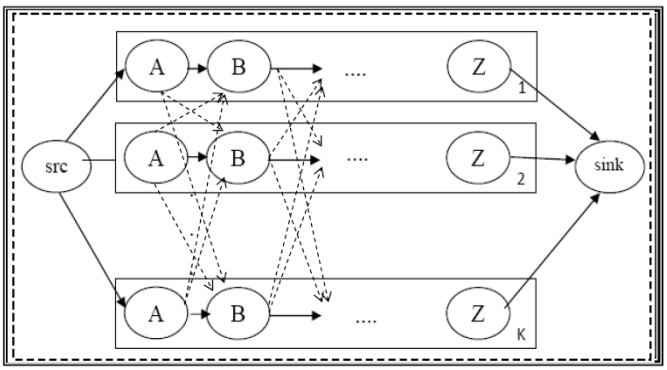
شکل ۴- ارتباط بین ماشین‌های منطقی در اجرای عملگرهای نقشه پرس‌وجو

بر این اساس، واحد ایجاد کننده نقشه پرس‌وجو، "کلان گراف پرس‌وجو" را ایجاد کرده و آنرا به واحد زمانبند می‌دهد تا زمانبندی اجرای موازی عملگرها روی k ماشین منطقی موجود انجام گیرد (زمانبندی در بخش ۳-۳ بحث می‌شود).

۲-۱-۲- کلان گراف پرس‌وجو[۱۸]

- **ایجاد کلان گراف پرس‌وجو**

چنانچه از روی نقشه پرس‌وجوی ایجاد شده برای یک پرس‌وجو، k کپی یکسان ایجاد کنیم که عملگرهای آن ترتیب منطقی (توالی بین) عملگرهای نقشه پرس‌وجوی اصلی را در ارتباطات بین کپی‌ها نیز تکرار کرده باشند، یک DAG حاصل خواهد شد. گرهی را به عنوان مبداء[۱۹] و گرهی را نیز به عنوان مقصد[۲۰] به ترتیب قبل و بعد از گره‌های عملگرهای نقشه پرس‌وجو در نظر می‌گیریم. از گره مبداء، لبه‌ای به هر یک از گره‌های سطح (مرحله[۲۱]) اول نقشه پرس‌وجوی اصلی، و از هر یک از گره‌های سطح آخر نیز لبه‌ای به گره مقصد وصل می‌کنیم. گراف حاصل را کلان گراف پرس‌وجو می‌نامیم. شکل ۵ کلان گراف پرس‌وجوی ایجاد شده برای نقشه پرس‌وجویی شامل عملگرهای A تا Z روی k ماشین منطقی را نشان می‌دهد:



شکل ۵- کلان گراف پرس‌وجوی ایجاد شده برای نقشه پرس‌وجویی شامل عملگرهای A تا Z روی k ماشین منطقی

در کلان گراف پرس‌وجو، هر یک از لبه‌های بین گره‌ها نمایانگر صف بین عملگرهای متصل به آن لبه نیز می‌باشد. اگرتعداد المان‌های داده موجود در صف مربوط به هر لبه را به عنوان وزن آن لبه در نظر بگیریم، کلان گراف پرس‌وجو را می‌توان به صورت یک DAG وزن‌دار مدل نمود که وضعیت سیستم از لحاظ توزیع بارکاری در هر لحظه را بیان می‌نماید. کلان گراف پرس‌وجوی ایجاد شده، در زمانبندی اجرای عملگرها به منظور پردازش موازی پرس‌وجو مورد استفاده قرار می‌گیرد.

به منظور ارائه دقیق‌تری از کلان گراف پرس‌وجو، در زیر به تعریف رسمی آن می‌پردازیم و از این تعاریف برای اثبات قضایا استفاده می‌کنیم.

- **تعریف رسمی کلان گراف پرس‌وجو**

تعریف ۱: نقشه پرس‌وجو عبارتست از گراف جهت داری که عملگرها و اولویت آنها در اجرای پرس‌وجو را نشان می‌دهد که می‌توان آنرا به صورت گراف G_{QP} بیان نمود:

$$G_{QP}=<V,E>$$
$$V=\{o|\ o \in \text{stream query processing operators}\} \quad (۱)$$
$$E=\{<A,B>|\ A,B \in V\}$$

نمادگذاری: اگر نماد O_j^i را به معنی عملگر O که i امین عملگر از نقشه پرس‌وجوی اصلی در ماشین j ام تعریف کنیم، و به همین ترتیب O_-^i یعنی عملگر iام از نقشه پرس‌وجوی اصلی و O_j^- یعنی عملگر O در ماشین j ام، آنگاه کلان گراف پرس‌وجو را می‌توان به صورت فرمال زیر بیان نمود:

نکته: شماره (شناسه[۲۲]) هر عملگر در نقشه پرس‌وجوی اصلی، عددی افزایشی و منحصر به فرد است که به هر عملگر تخصیص می‌یابد. همچنین تعداد عملگرهای نقشه پرس‌وجو برابر است با تعداد اعضاء مجموعه V که با |V| نمایش داده می‌شود. لذا، منظور از $O_-^{|V|}$، آخرین عملگر از نقشه پرس‌وجوی اصلی است.

تعریف ۲: بر این اساس، کلان گراف پرس‌وجو گراف وزن‌داری است که از روی گراف نقشه پرس‌وجو و به صورت زیر ایجاد می‌شود [۱۹]:

$$QMG=<V', E', W'>$$
$$V'= \bigcup_{\substack{1\le i\le |V| \\ 1\le j\le K}} O_j^i \quad U \quad \{src, sink\}$$
$$E'= E \cup \{<x,y>|\forall A,B \in V(< A,B >\in E) \Rightarrow (\forall i = 1,2,..,k, \forall j = 1,2,...,k, i \neq j \left(< A_i^-, B_j^- >\in E' \wedge \left(x = A_i^- \wedge\ y = B_j^-\right)\right)$$
$$V\ \ (\forall i = 1,2,...,k \left(\left(x = src \wedge y = O_i^1\right) V\ (x = O_i^{|V|} \wedge y = sink)\right)\right)\} \quad (۲)$$

W' تابعی است که وزن هر لبه از کلان گراف پرس‌وجو را می‌دهد:

$$W': E' \rightarrow \mathbb{Z}^+ \qquad \forall a,b \in V'(W'(a,b) = q\_count(a,b))$$

وزن لبه برابر است با تعداد المان‌های داده موجود در صف متناظر آن لبه که توسط تابع q-count() محاسبه می‌شود.

تعداد گره‌ها و تعداد لبه‌های کلان گراف پرس‌وجو طبق رابطه (۳) قابل محاسبه است:

$$|V'|=(|V|.K)+2 \quad (۳)$$
$$|E'|=(|E|.K)+(K-1).\sum_{i=1}^{|V|} fan\_out+2K$$

fan_out_i درجه خروجی (تعداد لبه‌های خارج شونده) گره i را نشان می‌دهد.

قضیه ۱: کلان گراف پرس‌وجو معادل نقشه پرس‌وجوی اصلی است (منطق پرس‌وجوی اصلی را تغییر نمی‌دهد).

اثبات: برای اثبات این قضیه کافیست ثابت کنیم که مسیر عملگری که هر تاپل در کلان گراف پرس‌وجو پیمایش می‌کند معادل همان مسیر عملگر نقشه پرس‌وجوی اصلی است. یعنی اولویت (ترتیب) هر لبه از کلان گراف پرس‌وجو، ترتیب لبه‌ای از گراف نقشه پرس‌وجوی اصلی است.

طبق تعریف فرمال کلان گراف پرس‌وجو، در مورد لبه‌هایی که به گره‌های مبداء یا مقصد متصل نیستند معادل لبه‌ای از نقشه پرس‌وجوی اصلی هستند، یعنی:

$$\forall x,y\left((< x,y >\in E') \wedge (x \neq src\ \wedge\ y \neq sink)\right) \Rightarrow$$
$$\exists i,j,k,l,A,B \left(\left(x = A_j^i \wedge\ y = B_k^l\right) \wedge < A_-^i, B_-^l >\in E'\right) \quad (۴)$$

در مورد لبه‌های متصل به مبداء یا مقصد نیز لبه‌های اضافه شده در کلان گراف پرس‌وجو صرفا تاپل‌ها را (بدون هیچ پردازش اضافی) به گره مورد نظر از نقشه پرس‌وجوی اصلی (طبق الگوریتم بخش ۴) انتقال می‌دهند:

$$\forall x,y\big((<x,y>\in E')\wedge(x=src\vee y=sink)\big)\Rightarrow \nexists X\Big(\forall O\in V\big((<src,X>\in E'\wedge<X,O^{1}_{\underline{}}>\in E')\vee(<O^{|V|}_{\underline{},X}>\in E'\wedge<X,sink>\in E')\big)\Big) \quad (۵)$$

با توجه به معادل بودن کلان‌گراف پرس‌وجو و نقشه پرس‌وجوی اصلی، واحد زمانبند می‌تواند مسیر عملگری از کلان‌گراف پرس‌وجو را برای پیمایش تاپل‌های جریان داده ورودی مورد استفاده قرار دهد.

برای اینکه تاپل‌های جریان داده بتوانند به سریعترین نحو پردازش شوند، باید مسیری از کلان گراف پرس‌وجو که برای پردازش تاپل مورد استفاده قرار می‌گیرد، کوتاهترین مسیر باشد (کمترین تاخیر را داشته باشد).

قضیه ۲: کوتاهترین مسیر در کلان گراف پرس‌وجو، کمترین تاخیر تاپل در پردازش پرس‌وجو روی تاپل را دارد.

اثبات: از آنجا که طبق تعریف، وزن هر لبه از کلان گراف پرس‌وجو برابر است با تعداد المان‌های داده موجود در صف متناظر با آن لبه، لذا تاخیری که هر تاپل در پیمایش مسیر عملگرها متحمل می‌شود برابر است با مجموع تاخیری که در صف‌های بین عملگرها تحمل می‌کند یعنی مجموع تعداد المان‌های موجود در صف‌ها (وزن لبه‌های مسیر پیمایش شده توسط تاپل).

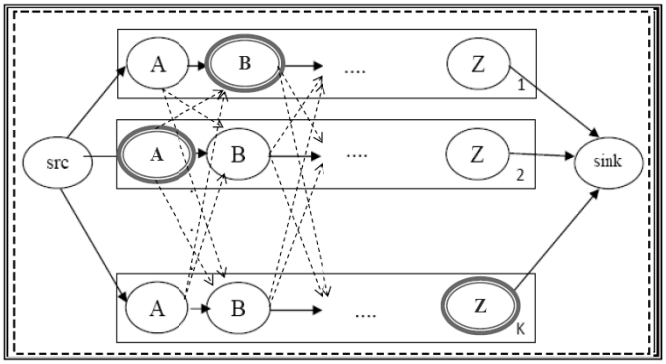
کوتاهترین مسیر کلان گراف پرس‌وجو دارای حداقل مجموع وزن لبه‌ها (کمترین تعداد المان‌های منتظر در صف ورودی عملگرها) است و لذا تاپل‌هایی از جریان داده که از این مسیر استفاده می‌کنند با کمترین تاخیر تاپل مواجه می‌شوند.

۲-۱-۳- زمانبندی

فرآیند ایجاد کلان گراف پرس‌وجو امکان پردازش موازی پرس‌وجو را فراهم می‌کند اما برای استفاده از این امکان باید بتوان زمانبندی مناسبی برای اجرا تعیین نمود. با توجه به این که از بین مسیرهای عملگر موجود در کلان گراف پرس‌وجو کمترین تاخیر در پردازش تاپل‌ها مربوط به کوتاهترین مسیر است، لذا الگوریتم زمانبندی کوتاهترین مسیر در کلان گراف پرس‌وجو را تعیین و استفاده می‌کند [۱۹].

در واقع، کلان گراف پرس‌وجو یک دید منطقی است از k ماشین منطقی موجود که همه آنها قابلیت اجرای عملگرهای نقشه پرس‌وجو را دارند. اما برای اینکه بتوانیم از توان همه این ماشین‌ها برای اجرای موازی عملگرهای نقشه پرس‌وجو استفاده کنیم نیاز به زمانبندی عملگرها و توزیع اجرای نقشه پرس‌وجو روی ماشین‌ها به طور موازی داریم. در سیستم ارائه شده، الگوریتم زمانبندی موازی عملگرها شامل یافتن کوتاهترین مسیر در کلان گراف پرس‌وجو از گره مبداء به گره مقصد می‌باشد که برای یافتن این مسیر می‌توان از الگوریتم کوتاهترین مسیر دایجسترا استفاده کرد.

مسیری که توسط الگوریتم کوتاهترین مسیر دایجسترا (الگوریتم زمانبندی) به دست می‌آید دنباله‌ای است از عملگرهای نقشه پرس‌وجو که گره‌های این مسیر (همان عملگرهای نقشه پرس‌وجو) روی k ماشین منطقی موجود توزیع شده‌اند. شکل ۶ مثالی از نتیجه الگوریتم زمانبندی (الگوریتم مسیریابی دایجسترا) روی کلان گراف پرس‌وجوی شکل ۵ را نشان می‌دهد:



شکل ۶- مثالی از نتیجه الگوریتم مسیریابی دایجسترا روی کلان گراف پرس‌وجو (الگوریتم زمانبندی موازی عملگرها)

واحد زمانبند سیستم، کوتاهترین مسیر را در کلان گراف پرس‌وجوی دریافتی از واحد ایجادکننده نقشه پرس‌وجو می‌یابد. پس از یافتن این مسیر، به هر یک از ماشین‌های منطقی، شناسه عملگری از نقشه پرس‌وجوی اصلی که آن ماشین باید اجرا کند (o) به همراه مشخصه ماشین قبلی[۲۳] (p) و بعدی[۲۴] (s) آن ماشین را در قالب سه‌تایی <p,o,s> ارسال می‌کند. به عنوان مثال در شکل ۶، پس از یافتن کوتاهترین مسیر، سه‌تایی <2,B,5> به ماشین ۱ ارسال می‌شود. بدین ترتیب، هر ماشین از این پس تاپل‌های جریان داده را از ماشین مشخص شده قبلی خود دریافت کرده، عملگر مربوطه را روی آنها اعمال و تاپل‌های نتیجه را در صف ورودی عملگر ماشین بعدی قرار می‌دهد.

قضیه ۳: نتیجه الگوریتم یافتن کوتاهترین مسیر در کلان گراف پرس‌وجو همان تعیین زمانبندی اجرای موازی عملگرهای پرس‌وجو می‌باشد.

اثبات: همانگونه که بیان شد، تخصیص کپی‌های نقشه پرس‌وجو به ماشین‌های منطقی صرفا به این خاطر بود که همه ماشین‌ها از نقشه پرس‌وجو و ماهیت هر عملگر مطلع باشند. اما آنچه که در عمل برای پردازش تاپلها اتفاق می‌افتد آنست که مسیری که توسط الگوریتم کوتاهترین مسیر انتخاب شده، دنباله‌ای از عملگرها را تعیین می‌کند که برای اجرای نقشه پرس‌وجو در زمان اجرا به کار می‌روند. یعنی این مسیر همان نقشه پرس‌وجوی اصلی در زمان اجراست که روی ماشین‌ها توزیع شده است. به عبارتی، به جای اینکه نقشه پرس‌وجو روی یک ماشین واحد اجرا شود، بدین صورت روی k ماشین منطقی موجود به طور موازی اجرا می‌شود.

از طرفی، تخصیص عملگرها به ماشین‌ها به گونه‌ای انجام می‌شود که اجرای پرس‌وجو روی تاپل‌های جریان داده کمترین تاخیر تاپل را به همراه داشته باشد (استفاده از کوتاهترین مسیر کلان گراف پرس‌وجو برای پردازش تاپل‌ها).

از این رو، انجام الگوریتم مسیریابی و یافتن کوتاهترین مسیر در کلان گراف پرس‌وجو را می‌توان معادل الگوریتم زمانبندی اجرای موازی عملگرهای پرس‌وجو با هدف به حداقل رساندن تاخیر تاپل دانست.

- **زمانبندی اولیه**

در آغاز کار سیستم که وزن همه لبه‌های کلان گراف پرس‌وجو یکسان و برابر است با صفر (زیرا بافر عملگرها خالی هستند)، مسیر حاصل از الگوریتم مسیریابی احتمالا به طور کامل روی یک ماشین قرار گیرد. می‌توان مقداردهی اولیه وزن لبه‌های کلان گراف را به گونه‌ای انجام داد که حاصل الگوریتم مسیریابی، مسیری توزیع شده روی همه ماشین‌ها باشد. اما برای اینکه از آغاز کار، بارکاری[۲۵] را به طور کاملاً متعادل[۲۶] روی همه ماشین‌های موجود توزیع کنیم و از طرفی متحمل سربار پردازشی تخصیص وزن اولیه به لبه‌ها و اجرای الگوریتم مسیریابی نشویم، در

زمانبندی اولیه (آغاز کار سیستم)، عملگرها را به صورت دستی (بدون استفاده از الگوریتم مسیریابی) و از رابطه (۲) به ماشینها تخصیص می‌دهیم:

$$j=i \bmod k \qquad (۶)$$

که در آن j شماره ماشین، i شماره عملگر نقشه پرس‌وجوی اصلی و k تعداد ماشینهای منطقی می‌باشد.

- **نیاز به زمانبندی مجدد**

از آنجا که عملگرهای مختلف نقشه پرس‌وجو (مانند گزینش[۲۷]، پرتو[۲۸] و پیوند[۲۹]) از نظر سرعت در پردازش تاپل‌ها یکسان نیستند (برخی مانند پرتو سریعتر و برخی مانند پیوند- که در حالت کلی یک عملگر انسدادی است- بسیار کندتر عمل می‌کنند)، با گذشت زمان و رسیدن تاپل‌های جریان داده، سیستم به حالتی سوق پیدا می‌کند که در یک لحظه زمانی[۳۰] از وضعیت سیستم، برخی صف‌ها پرتر و برخی خالی‌تر هستند. مثلا صف ورودی یک عملگر پیوند دارای ۸ تاپل است در حالیکه یک عملگر پرتو در صف ورودی خود ۲ تاپل را برای پردازش دارد. بنابراین، با گذشت زمان وزن لبه‌های مختلف کلان گراف پرس‌وجو با هم تفاوت خواهد داشت.

در صورتی که بافر برخی از عملگرها پر شده باشد سیستم در شرایط اضافه بار به سر می‌برد و مجبور است تاپلهای بعدی ان عملگر را دور بریزد (کاهش بار) که در این صورت کیفیت خروجی پایین می‌آید. در چنین شرایطی که با توجه به کاربردها و ویژگی‌های جریان‌های داده احتمال وقوع آن بسیار زیاد است [۶] نیاز به انجام زمانبندی مجدد (مسیریابی مجدد) داریم تا در صورت امکان شرایط اضافه بار را اداره[۳۱] کرده و کارایی سیستم را بهبود بخشیم.

مثال ۲: اگر مثلاً عملگر x در ماشین a دارای ۸ تاپل در صف ورودی خود باشد و این صف پر شده باشد، (و عملگری در ماشین دیگری وزن کمتر از ۸ داشته باشد) رویه زمانبندی را مجددا فراخوانی می‌کنیم. در این مسیریابی، قطعا کوتاهترین مسیر دیگر از گره x در ماشین a استفاده نمی‌کند بلکه از عملگر x در ماشین دیگری بهره می‌گیرد. در این زمان ماشین a می‌تواند پردازش تاپل‌های در صف مانده عملگر x را تکمیل کند. این در واقع کاهش بار[۳۲] هوشمندانه یا به عبارت بهتر نوعی توازن بار[۳۳] پویا به جای کاهش بار است. این فرآیند به جای از بین بردن تاپلهای مربوط به نتایج میانی (که مهم هستند و بخشی از زمان پردازشی سیستم نیز صرف آنها شده است)، با متوازن کردن بارکاری سعی در پردازش آنها دارد.

لذا برای تعیین وزن‌های جدید لبه‌های کلان گراف پرس‌وجو می‌توان به گونه‌ای عمل کرد که اگر به فرض عملگر a در ماشین A دارای وزن مثلاً ۸ است و عملگر b در ماشین B دارای وزن ۳ می‌باشد، در زمانبندی (مسیریابی) مجدد جای عملگرهای a و b در ماشین‌های A و B عوض شود. یعنی a در B و b در ماشین A اجرا شوند تا A بتواند تاپل‌های قبلی‌اش را پردازش کرده و سپس تاپل‌های عملگر سبک b را پردازش کند و در این حین ماشین B هم ابتدا تاپلهای اندک عملگر سبک b و سپس تاپل‌های عملگر سنگین a جدید را پردازش نماید (تعیین مقدار وزن لبه‌های کلان گراف پرس‌وجو در بخش۳-۳ تشریح می‌شود). در این حالت ممکن است ترتیب تاپل‌های جریان داده ورودی در حین پردازش حفظ نشود (تاپلهای جدید در عملگر a از ماشین B پردازش شده و به عملگر بعدی ارسال می‌شوند در حالیکه تاپل‌های قبلی در صف عملگر a از ماشین A، منتظر پردازش هستند).

با توجه به حالت فوق، تعداد عملگرهایی که یک ماشین در هر لحظه اجرا می‌کند در عمل بیشتر از یک عملگر است (هم عملگر قبلی آن که هنوز پردازش برخی از تاپل‌های آن تمام نشده است و هم عملگر جدیدی که پس از زمانبندی مجدد اجرای آن به عهده این ماشین گذاشته می‌شود) که نیاز به اجرای همروند عملگرهای درون یک ماشین نیز به وجود می‌آید.

شاید اینگونه به نظر برسد که پس از مسیریابی مجدد، به جای اجرای همروند عملگرهای جدید و قدیم در ماشین‌های جدید و قدیم، بهتر است تاپل‌های در صف مانده عملگر قدیم را به ابتدای صف عملگر جدید متناظر منتقل کنیم تا هم ترتیب ورودی تاپل‌ها بهم نخورد و هم هر ماشین در هر زمان تنها یک عملگر را برای اجرا داشته باشد.

- **تعیین مقدار وزن لبه‌های کلان گراف پرس‌وجو**

در تعیین وزن لبه‌های کلان گراف به منظور زمانبندی مجدد (بخش ۳-۲)، باید به گونه‌ای عمل کنیم که ماشین‌هایی که کار باقی‌مانده بیشتری دارند، پس از به‌روز رسانی وزن لبه‌ها و مسیریابی مجدد کمتر مورد استفاده قرار گیرند تا بتوانند تاپلهای از قبل مانده خود را پردازش نمایند (توازن بار[۳۴]). لذا برای تعیین وزن جدید لبه‌های کلان گراف پرس‌وجو، مجموع تعداد تاپل‌های درون صف‌های ورودی عملگرهایی که آن ماشین اجرا می‌کند را به وزن همه لبه‌های کلان گراف پرس‌وجو به مقصد هر یک از عملگرهای آن ماشین اضافه می‌کنیم.

$$\forall x,y,z \in V, \forall j = 1,2,\dots,k, 1 \le i \le |V| \;\; (W''(z_i^j, y_j^-) \leftarrow W'(z_i^j, y_j^-) + \sum_{i=1}^{|V|} q\_count(x_j^i, y_j^-)) \qquad (۷)$$
$$W'(x,y) \leftarrow W''(x,y)$$

با توجه به طبیعت پیوسته جریان‌های داده از یک سو و پیوسته بودن عمده پرس‌وجوهای سیستم‌های جریان داده از سوی دیگر، همخوانی و سازگاری سیستم مدیریت جریان داده با این طبیعت پیوسته می‌تواند به بهبود عملکرد آن کمک نماید. در روش موازی‌سازی ارائه شده در بخش قبل، از آنجا که زمان‌بندی مجدد در صورت وقوع شرایط اضافه‌بار انجام می‌گیرد، می‌توان این زمان‌بندی را رویدادگرا[۳۵] نامید که با وقوع رویداد پرشدن صف‌های عملگرها اجرا می‌شود. در مقابل زمان‌بندی رویدادگرا، در این بخش زمان‌بندی پیوسته و پویای اجرای موازی عملگرها ارائه می‌شود که با طبیعت پیوسته جریان‌ها و پرس‌وجوها سازگاری بیشتری نیز خواهد داشت.

۲-۱-۴- مسیریابی پیوسته و پویا در کلان‌گراف پرس‌وجو

به منظور اجرای موثرتر فرآیند زمان‌بندی اجرای موازی عملگرها در سیستم مدیریت جریان داده موازی، در این بخش روشی برای مسیریابی پیوسته و پویا در کلان‌گراف پرس‌وجو ارائه می‌شود که بهبود قابل توجهی در عملکرد سیستم مذکور به ارمغان می‌آورد. ایده اصلی در مسیریابی پیوسته و پویا به جای مسیریابی دایجسترا در کلان‌گراف پرس‌وجو از ویژگی چندمرحله‌ای بودن کلان‌گراف پرس‌وجو نشات می‌گیرد[۲۰].

تعریف ۳ (مرحله[۳۶]). مجموعه گره‌هایی یک گراف چندمرحله‌ای به چند مجموعه به نام مرحله افراز می‌شوند به گونه‌ای که هر لبه از این گراف از گرهی از مرحله i فقط به گرهی در مرحله i+1 می‌رود.

قضیه ۴: کلان‌گراف پرس‌وجو یک گراف چندمرحله‌ای است.

اثبات: نقشه پرس‌وجوی یک پرس‌وجوی پیوسته، یک گراف چندمرحله‌ای است. کلان‌گراف پرس‌وجو نیز طبق تعریف متشکل است از k کپی از نقشه پرس‌وجو بعلاوه گره‌ها و لبه‌هایی که با حفظ مراحل اضافه می‌شوند، یعنی لبه‌ای از گره عملگر a در ماشین i به عملگر b در ماشین j به شرطی وجود دارد که لبه‌ای از a به b در نقشه پرس‌وجوی اصلی وجود داشته باشد:

$$\forall a, b \in V \ \wedge i, j = 1,2, \dots, |V| \wedge l, t = 1,2, \dots, k \ \left(\left(a_l^i, b_t^j\right) \in E' \Rightarrow \left(a_{\_}^i, b_{\_}^j\right) \in E\right) \quad (۸)$$

لذا a در مرحله بلافاصله قبلی عملگر b قرار دارد.

همچنین لبه‌های گره مبداء به گره‌های مرحله اول نقشه پرس‌وجوی اصلی متصل هستند و لبه‌های متصل به گره مقصد فقط از گره‌های مرحله آخر نقشه پرس‌وجوی اصلی شروع می‌شوند. لذا هر لبه اضافه شده در کلان‌گراف پرس‌وجو نیز همواره از گرهی از مرحله i را تنها به گرهی از مرحله i+1 متصل می‌نماید.

با توجه به قضیه (۴)، در مورد کلان‌گراف پرس‌وجوی QMG=<V',E',W'> می‌توان گفت $OS_i \subseteq V'$ مرحله iام از کلان‌گراف پرس‌وجوی QMG است اگر و فقط اگر:

$$V' = \cup_{i=1}^{L} OS_i \text{ such that:} \\ (\forall i, j \therefore i \neq j\left(OS_i \cap OS_j = \emptyset\right)) \wedge (\forall e \in E', e = (a, b)(\nexists c, d \in V'((a, c) \in E' \wedge (d, b) \in E'))) \quad (۹)$$

تعریف ۴ (گام[۳۷]**).** انتقال از گرهی از مرحله i به گرهی در مرحله i+1 را یک گام می‌نامیم. براین اساس، هر یال بین دو گره در کلان‌گراف پرس‌وجو معرف یک گام است.

هر تاپل رسیده از جریان داده ورودی باید دنباله عملگرهای مسیر عملگر تعیین شده را پیمایش کند.

تعریف ۵ (مسیر عملگر). هر یک از مسیرهای عملگر موجود در کلان‌گراف پرس‌وجو دنباله‌ای است از گره‌های به ترتیب موجود در مراحل 1 تا L (به فرض L مرحله‌ای بودن کلان‌گراف پرس‌وجو) به گونه‌ای که هیچ عملگری دوبار تکرار نمی‌شود:

$$\text{Operator\_Paths(QMG)}=\{<x_1,x_2,\dots,x_l>|((x_i \in OS_i \Rightarrow x_j \in OS_{i+1}) \Leftrightarrow (j=i+1)) \wedge ((\forall i, j \therefore i \neq j\ (x_i = O_{\_}^r \wedge x_j = O_{\_}^s) \Rightarrow r \neq s))\} \quad (۱۰)$$

قضیه ۵: کوتاهترین مسیر در گراف چندمرحله‌ای کلان‌گراف پرس‌وجو برابر است با دنباله‌ای از لبه‌های مربوط به گام‌های به ترتیب بین مراحل ۱ تا L به گونه‌ای که وزن هر لبه انتخاب شده، مینیمم وزن بین لبه‌های همان گام باشد.

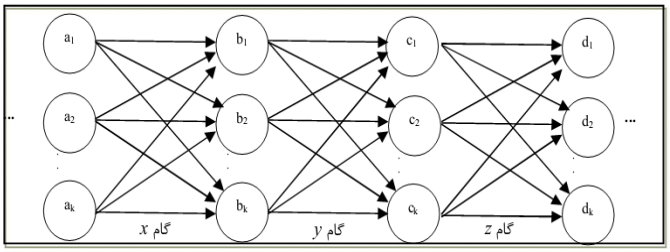
اثبات: هر مسیر عملگر در کلان‌گراف پرس‌وجو (گرافی چندمرحله‌ای) دنباله‌ای است از لبه‌های متناظر با گام‌های ۱ تا L و وزن هر مسیر برابر است با مجموع وزن لبه‌های آن مسیر. از آنجا که وزن هر لبه (که طبق تعریف برابر است با تعداد المان‌های موجود در صف متناظر آن لبه) عددی مثبت است، مجموع وزن این لبه‌ها در صورتی مقدار مینیمم خواهد داشت که هر یک از لبه‌های تشکیل دهنده مسیر مینیمم مقدار را به خود بگیرند (مجموع چند متغییر از نوع عدد مثبت مینیمم است اگر و فقط اگر هر یک از آن متغییرها مینیمم مقدار ممکن را داشته باشند). به فرض، $D_a, D_b, \dots, D_z$ به ترتیب دامنه‌های مقادیر متغییرهای $a, b, \dots, z$ باشند که $D_a, D_b, \dots, D_z \subseteq \mathbb{N}$.

$$a+b+\dots+z=\text{MIN\_VALUE} \Leftrightarrow \left(a = \min_{a_i \in D_a}\{a_i\}\right) \wedge \left(b = \min_{b_i \in D_b}\{b_i\}\right) \wedge \dots \wedge \left(z = \min_{z_i \in D_z}\{z_i\}\right) \quad (۱۱)$$

لذا برای اینکه مسیر عملگری که انتخاب می‌شود دارای حداقل هزینه باشد (مجموع وزن لبه‌ها مینیمم باشد)، در هر گام، از بین لبه‌های موجود که از گره‌های مرحله‌ای به گره‌های مرحله بعدی می‌روند، لبه‌ای انتخاب می‌شود که وزن آن مینیمم بین لبه‌های همان گام باشد.

طبق قضیه (۵)، برای تعیین کوتاهترین مسیر در کلان‌گراف پرس‌وجو به جای استفاده از الگوریتم دایجسترا کافیست در هریک از گام‌ها از بین لبه‌هایی که از عملگری از مرحله i به گرهی در مرحله $i+1$ می‌روند، لبه‌ای انتخاب شود که وزن آن مینیمم است (شکل ۷):

$$\text{Cost}(x \longrightarrow y \longrightarrow z) = \text{MIN\_VALUE} \Leftrightarrow \{(x = \min_{\substack{1 \le i \le k, \\ 1 \le j \le k}}\left\{\text{q\_count}_{(a_i,b_j)}\right\}) \wedge (y = \min_{\substack{1 \le i \le k, \\ 1 \le j \le k}}\left\{\text{q\_count}_{(b_i,c_j)}\right\}) \wedge (z = \min_{\substack{1 \le i \le k, \\ 1 \le j \le k}}\left\{\text{q\_count}_{(c_i,d_j)}\right\})\}$$



شکل ۷- انتخاب کوتاهترین لبه در هر گام برای تعیین کوتاهترین مسیر در کلان گراف پرس‌وجو

روش مسیریابی (زمانبندی) پیوسته و پویای ارائه شده در این بخش که از این پس به نام مسیریابی (زمانبندی) XYZ از آن یاد می‌کنیم در واقع جایگزینی برای مسیریابی رویدادگرای دایجسترا در روش موازی‌سازی ارائه شده در بخش (۲-۱-۳) (که در شرایط پر شدن صف فراخوانی می‌شد) است.

پس از تخصیص کپی‌های نقشه پرس‌وجو به k ماشین منطقی و ایجاد کلان‌گراف پرس‌وجو، در آغاز کار سیستم و فقط برای اولین بار، عملگرها به صورت دستی و طبق رابطه j=i *mod* k به طور متوازن به ماشین‌ها تخصیص داده می‌شوند. هر عملگر با دریافت هر تاپل از عملگر قبلی خود آنرا پردازش و از بین لبه‌هایی به یکی از گره‌های (عملگرهای) مرحله بعد (در ماشین‌های مختلف) می‌روند، لبه با وزن مینیمم را انتخاب می‌نماید.

پس از انتخاب لبه مورد نظر، تاپل نتیجه گره مبداء لبه به همراه سرآیندی[۳۸] شامل شماره عملگری از نقشه پرس‌وجوی اصلی که ماشین مقصد آن لبه باید آنرا روی تاپل دریافتی اجرا کند ارسال می‌شود.

شماره عملگری که ماشین مقصد هر لبه باید اجرا کند برابر است با یکی بیشتر از Id عملگری که ماشین مبداء اجرا کرده است. به عبارتی، عملگر مبداء تاپل نتیجه خود را به همراه سرآیند شامل یکی بیشتر از Id عملگر خود در نقشه پرس‌وجوی اصلی را به عملگر مقصد لبه ارسال می‌کند.

بدین ترتیب، در هر گام از کلان‌گراف پرس‌وجو لبه با وزن مینیمم انتخاب و مورد استفاده قرار می‌گیرد و بر اساس قضیه (۵)، کل مسیر پیمایش شده توسط تاپل‌های جریان داده کمترین هزینه را خواهد داشت (کوتاهترین مسیر). لذا به جای آنکه یافتن کوتاهترین مسیر در کلان‌گراف پرس‌وجو (زمان‌بندی اجرای موازی عملگرها) به صورت رویدادگرا انجام گیرد، به طور پیوسته و پویا با ورود هر

تاپل انجام می‌شود (زمان‌بندی پیوسته مبتنی بر تاپل[۳۹]). در اینصورت فرآیند زمان‌بندی در طول زمان پردازش جریان داده توزیع خواهد شد.

الگوریتم زمان‌بندی پیوسته و پویای XYZ بهترین مسیر عملگر در هر لحظه را برای پیمایش تاپل‌ها به کار می‌گیرد. تحلیل پیچیدگی زمانی الگوریتم XYZ و مقایسه آن با روش استفاده از زمان‌بندی رویدادگرای ارائه شده در بخش قبلی، در بخش (۲-۵) ارائه می‌شود.

قضیه ۶: هیچ مسیر عملگری وجود ندارد که هزینه (تاخیر تاپل) آن از مسیری که توسط الگوریتم مسیریابی پیوسته و پویای XYZ تعیین می‌شود کمتر باشد.

اثبات: مسیری که در زمان‌بندی رویدادگرا انتخاب می‌شود، کوتاهترین مسیر در کل کلان‌گراف پرس‌وجو (با وزن‌های به روز رسانی شده که بار کاری هر ماشین را نشان می‌دهند) در یک لحظه زمانی از وضعیت سیستم مدیریت جریان داده (که در آن صف عملگری پر شده است)، می‌باشد. مسیری که در مسیریابی پیوسته و پویای XYZ تعیین می‌شود نیز (با فرض اینکه نحوه تعیین وزن لبه‌ها یکسان باشد)، طبق قضیه (۵)، کوتاهترین مسیر در کلان‌گراف پرس‌وجو در هر لحظه زمانی از وضعیت سیستم را ارائه می‌کند.

اگر چه این مسیر ممکن است از نظر دنباله ماشین‌هایی که برای اجرای عملگرها به طور موازی مورد استفاده قرار می‌گیرند با مسیر تعیین شده توسط الگوریتم دایجسترا (مسیریابی رویدادگرا) متفاوت باشد، اما از آنجا که هر دو مسیر، کوتاهترین مسیر در کلان‌گراف پرس‌وجو هستند (مسیر با مقدار وزن مینیمم)، هزینه هر دو برابر خواهد بود. (به فرض، $OP_1 =< x_1, x_2, \dots, x_k >$ کوتاهترین مسیر تعیین شده توسط الگوریتم XYZ است، پس $cost(op_1) =$ MIN_VALUE.

برهان خلف: اگر مسیر دیگری مانند $OP_2 =< x'_1, x'_2, \dots, x'_k >$ در کلان‌گراف پرس‌وجو وجود داشته باشد که بهتر از OP_1 باشد، آنگاه:

$$\begin{cases} cost(< x'_1, x'_2, \dots, x'_k >) = \text{MIN\_VALUE} \\ \quad \text{AND} \\ cost(< x'_1, x'_2, \dots, x'_k >) < cost(< x_1, x_2, \dots, x_k >) \end{cases} \quad (۱۲)$$

رابطه (۱۲) فقط به شرطی برقرار خواهد بود که $cost(< x_1, x_2, \dots, x_k >) \neq$ MIN_VALUE که این برخلاف فرض است و لذا چنین مسیری وجود ندارد.) بنابراین مسیر یافته شده توسط الگوریتم مسیریابی پیوسته و پویای XYZ کوتاهترین و بهترین مسیر در کلان‌گراف پرس‌وجو در هر لحظه زمانی از وضعیت سیستم است و هیچ مسیری نمی‌توان یافت که از نظر هزینه (همان تاخیر تاپل طبق قضیه ۲) از آن کمتر باشد.

قضیه ۷: با استفاده از زمان‌بندی پیوسته و پویای XYZ، نیازی به زمان‌بندی مجدد وجود ندارد.

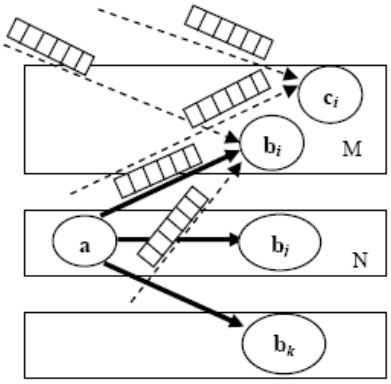
اثبات: از آنجا که هدف اصلی از ارائه سیستم مدیریت پردازش پرس‌وجوی موازی، رفع گلوگاه تک‌پردازنده‌ای بودن و پردازش سریعتر جریان داده به منظور کاهش تاخیر تاپل است، طبق قضیه (۴)، مسیری که در مسیریابی پیوسته و پویای XYZ انتخاب می‌شود کوتاهترین مسیر در هر لحظه بوده و دارای کمترین هزینه (تاخیر تاپل) است. لذا طبق قضیه (۵)، هیچ مسیر عملگری که از نظر تاخیر تاپل بهتر از مسیر تعیین شده توسط الگوریتم XYZ بهتر باشد در کلان‌گراف پرس‌وجو وجود ندارد که بخواهیم با انجام مسیریابی مجدد (زمان‌بندی مجدد) آن را بیابیم. براساس قضیه (۷)، با استفاده از زمان‌بندی پیوسته و پویای XYZ دیگر نیازی به انجام زمان‌بندی مجدد وجود ندارد و مسیریابی XYZ به طور پیوسته و پویا، همواره بهترین مسیر عملگر را مورد استفاده و پیمایش تاپل‌های جریان داده قرار می‌دهد.

قضیه ۸: استفاده از زمان‌بندی پیوسته و پویای پیشنهادی به جای زمان‌بندی رویدادگرا سبب کاهش نوسانات کارایی سیستم می‌شود.

اثبات: در سیستم موازی ارائه شده در بخش قبل، که از زمان‌بندی رویدادگرا استفاده می‌شود، کارایی سیستم (مثلاً تاخیر تاپل) در لحظه بعد از انجام زمان‌بندی بسیار خوب خواهد بود اما به مرور زمان کارایی کاهش می‌یابد. تا بدانجا که صف عملگری کاملاً پر شده (به حداقل رسیدن کارایی). در این حالت زمانبندی مجدد انجام می‌شود و کارایی سیستم مجدداً به حالت مطلوب برمی‌گردد. این شرایط بیانگر وجود نوسانات در کارایی سیستم در روش زمان‌بندی رویدادگراست. اما در روش زمان‌بندی پیوسته و پویای ارائه شده، هر عملگر در هنگام انجام زمان‌بندی بهترین ماشین (عملگر) را انتخاب می‌کند. لذا مسیر عملگر به کار رفته، بهترین مسیر در هر لحظه زمانی است (قضیه ۷). به عبارت دیگر، کارایی سیستم در هر لحظه زمانی به مقدار بهینه نزدیک و همگرا می‌شود.

- **به روزرسانی وزن لبه‌ها**

در زمان‌بندی پیوسته و پویای XYZ، عملگر مبداء هر لبه با دریافت تاپل ورودی خود، از بین لبه‌هایی که به عملگر بعدی می‌روند لبه‌ای را انتخاب و برای ارسال تاپل‌های نتیجه‌اش استفاده می‌کند که وزن آن مینیمم باشد. به منظور آنکه تاخیر تاپل در مسیر عملگرمورد پیمایش تاپل‌ها مینیمم باشد، هزینه هر لبه از مسیر عملگر باید هزینه‌ای را نشان دهد که هر تاپل برای پردازش شدن متحمل می‌شود. به عبارتی هزینه هر گام از کلان‌گراف پرس‌وجو برابر است با هزینه پردازشی ماشینی که آن گام را اجرا می‌کند. یعنی برای هر یک از لبه‌های متصل به گره a در ماشین N که به یکی از عملگرهای b در ماشین‌های مختلف می‌روند (شکل ۸)، هزینه لبه برابر است با بار کاری ماشین مقصد آن لبه، یعنی تعداد تاپل‌هایی که آن ماشین باید پردازش کند ضربدر هزینه پردازش هر یک از تاپل‌ها.



شکل ۸- مثالی از انتخاب لبه در یک گام

هزینه پردازش هر تاپل توسط یک ماشین به عملگری که آن ماشین روی تاپل اجرا می‌کند بستگی دارد. اگر زمان اجرای عملگر i روی یک تاپل e_i و بهگزینی آن s_i باشد، اگرچه این عملگر پس از پردازش تاپل‌های ورودی، تنها به اندازه نسبت s_i از تعداد تاپل‌های ورودی‌اش را برمی‌گرداند اما در واقع همه تاپل‌های ورودی خود را پردازش کرده است. پس هزینه پردازش این عملگر یعنی e_i را باید به ازای همه تاپل‌های ورودی عملگر محاسبه نمود. تعداد تاپل‌های ورودی عملگر برابر است با طول صف متناظر ورودی عملگر. با توجه به اینکه یک ماشین ممکن است بیش از

یک عملگر را (به صورت همروند) اجرا کند، هزینه پردازشی هر ماشین برابر است با مجموع هزینه پردازش عملگرهایی که روی آن ماشین اجرا می‌شوند. بنابراین وزن هر لبه از کلان‌گراف پرس‌وجو برابر است با هزینه اجرای همه عملگرهای مربوط به ماشینی که عملگر گره مقصد این لبه نیز در آن ماشین اجرا می‌شود، طبق رابطه (۱۳):

$$\forall(a,b) \in E' \therefore a = O_k^i \wedge b = O_j^{i+1} ((w(a,b) \leftarrow \sum_{l=1}^{k}(q\_count(O_l^-, O_j^-) \times e_{O_j^-}))) \quad (۱۳)$$

وزن لبه‌ها که تابعی از تعداد المان‌های منتظر در صف ورودی هر عملگر و زمان اجرای آن عملگر روی یک تاپل می‌باشد هم قابل اندازه‌گیری است (با استفاده از شمارنده‌ای برای تعداد المان‌های هر صف) و هم قابل محاسبه (براساس نرخ ورود جریان داده، زمان اجرا و بهگزینی عملگرهای قلبی عملگر). در هر مرحله از الگوریتم زمان‌بندی پیوسته و پویای XYZ که عملگر مبداء یک لبه می‌خواهد مسیریابی پویای XYZ را انجام دهد، وزن لبه‌های متصل خروجی خود را تعیین نموده و از بین لبه‌ها، لبه با وزن مینیمم را برای ارسال تاپل نتیجه خود مورد استفاده قرار می‌دهد.

علاوه بر اینکه مراحل به روزرسانی وزن لبه‌ها، مسیریابی و ارسال تاپل‌ها به ماشین مورد نظر (زمان‌بندی اجرای موازی عملگرها) در روش ارائه شده به صورت پیوسته و پویا در طول زمان پرداش جریان داده توزیع شده‌اند (به جای اجرای رویدادگرای این مراحل در زمان زمان‌بندی مجدد)، در روش موازی‌سازی ارائه شده در بخش ۲-۱-۳، این مراحل به صورت متمرکز و توسط واحد زمانبند سیستم صورت می‌گرفت در حالیکه در روش ارائه شده توسط همه ماشین‌های محاسب اجرا می‌گردد.

۲-۱-۵- تحلیل پیچیدگی زمانی

- **موازی‌سازی با زمانبندی رویدادگرا**

پیچیدگی زمانی الگوریتم پردازش موازی پرس‌وجوی ارائه شده را از دو منظر مورد بررسی قرار می‌دهیم، یکی پیچیدگی زمانی هریک از ماشین‌هایی که به طور موازی عملگرهای نقشه پرس‌وجو (کلان گراف پرس‌وجو) را اجرا می‌کنند و یکی هم پیچیدگی زمانی واحد زمانبند سیستم.

الف) ماشین‌هایی که عملگرهای نقشه پرس‌وجو را اجرا می‌کنند

پیچیدگی زمانی هریک از ماشین‌هایی که عملگرهای نقشه پرس‌وجو را اجرا می‌کنند بسیار کم و در حد اجرای عملگر جریان داده (مانند گزینش، پرتو یا پیوند) روی تاپل‌هایی است که در صفی به اندازه **B** در انتظار هستند. پیچیدگی زمانی ماشینی که گره sink از کلان گراف پرس‌وجو (مرتب کردن بافر خروجی و تحویل نتایج به کاربر) را اجرا می‌کند نیز برابر است با $O(n.\log n)$ که در اینجا n= output_buffer_size.

ب) واحد زمان‌بند

پیچیدگی زمانی واحد زمانبند نیز برابر است با مجموع پیچیدگی زمانی مراحل مختلف اجرا شده توسط این واحد. مراحل ایجاد نقشه پرس‌وجو و ارسال کپی آن به هریک از ماشینها و نیز ایجاد کلان گراف پرس‌وجو، به صورت ایستا و یکبار در آغاز پردازش هر پرس‌وجو اجرا خواهد شد.

پیچیدگی زمانی الگوریتم زمانبندی اولیه نیز برابر است با زمان لازم برای تعیین ماشین مربوط به اجرای هر عملگر نقشه پرس‌وجو بر اساس رابطه (۳) و $O(|V|)$ و سپس ارسال سه‌تایی (p,o,s) به ماشین مربوطه $O(|k|)$.

پیچیدگی مراحلی که واحد زمانبند به طور پویا و پیوسته در زمان اجرا انجام خواهد داد را براساس رابطه (۳) می‌توان اینگونه بیان کرد:

$$O(|E'|) + O(|V'|.\log|V'|) + O(K)$$

که به ترتیب مربوط به به‌روزرسانی وزن لبه‌های کلان گراف پرس‌وجو، یافتن کوتاهترین مسیر در کلان گراف با استفاده از الگوریتم دایجسترا و ارسال سه‌تایی (p,o,s) مسیر جدید به هر یک از ماشین‌ها می‌باشد.

- **موازی‌سازی با زمانبندی پیوسته و پویا**

پیچیدگی زمانی الگوریتم پردازش موازی پرس‌وجو با به کارگیری زمان‌بندی پیوسته و پویای XYZ به جای الگوریتم زمان‌بندی رویدادگرا از سه منظر زیر بررسی و تحلیل می‌شود:

الف) ماشین‌هایی که عملگرهای نقشه پرس‌وجو را اجرا می‌کنند

با به کارگیری الگوریتم زمان‌بندی پیوسته و پویای XYZ در پردازش موازی پرس‌وجوها، هر یک از k ماشین منطقی موازی، علاوه بر اجرای عملگرهای نقشه پرس‌وجوی اصلی، وظیفه زمان‌بندی (مسیریابی) پیوسته و پویای XYZ را نیز بر عهده دارند. پیچیدگی زمانی اجرای عملگرهای جریان داده برای صف ورودی عملگر به طول n تاپل $O(n^2)$ در نظر گرفته شده است. یافتن لبه با وزن مینیمم و ارسال تاپل نتیجه به همراه سرآیند یکی بیشتر از Id عملگر نیز دارای پیچیدگی زمانی $O(K^2)$ خواهد بود (تعداد لبه‌های خروجی هر عملگر حداکثر به تعداد ماشین‌های منطقی یعنی k است که برای یافتن وزن هریک از این لبه‌ها، لبه‌های وارد شونده به ماشین مربوط به عملگر گره مقصد این لبه بررسی می‌شوند).

ب) گره sink

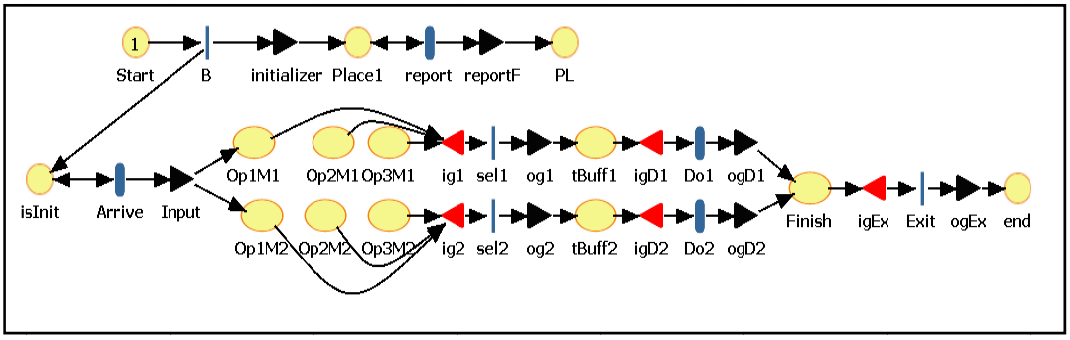
اگرچه با زمان‌بندی پیوسته و پویای XYZ به ظاهر مسیر قدیمی و جدید (مشابه سیستم ارائه شده در بخش ۲-۱-۳) به طور صریح ایجاد نمی‌شود اما تغییر پویای مقصد انتخابی در انتخاب لبه مینیمم در واقع منجر به بکارگیری مسیرهای متمایز برای پیمایش تاپل‌ها خواهد شد. لذا نیاز به بافر کردن و مرتب سازی تاپل‌ها قبل از تحویل به کاربر در گره sink همچنان احساس می‌شود. پیچیدگی زمانی این فرآیند (با فرض n به عنوان اندازه بافر خروجی) برابر است با $O(n.\log n)$.

ج) واحد زمان‌بند

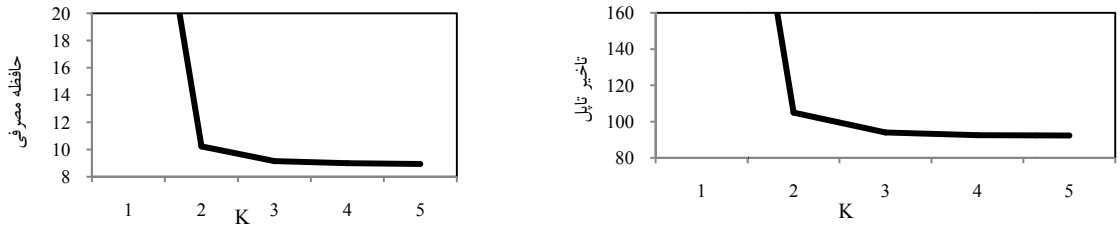
با به کارگیری زمان‌بندی پیوسته و پویای XYZ به جای زمان‌بندی مجدد، واحد زمانبند محدود به ایجادنقشه پرس‌وجو و کلان‌گراف پرس‌وجو و سپس زمان‌بندی اولیه طبق رابطه j=i **mod** k خواهد شد. این مراحل فقط یکبار در آغاز پردازش پرس‌وجو انجام می‌شوند و در زمان اجرای پرس‌وجو واحد زمانبند معماری سیستم ارائه شده در بخش (۲-۱-۳) غیر فعال خواهد بود.

۲-۱-۶- مدل‌سازی و ارزیابی سیستم

به منظور ارزیابی کارایی سیستم مدیریت جریان داده ارائه شده، این سیستم در شبکه‌های پتری[۴۰] مدل شده است (شکل ۹):



شکل ۹- مدل سیستم ارائه شده در شبکه‌های پتری



الف) حافظه مصرفی

ب) تاخیر تاپل

شکل ۱۰- مقایسه عملکرد سیستم پیشنهادی در ازای افزایش تعداد ماشین‌های منطقی

- **تنظیمات آزمایشات**

کارایی مدل سیستم با استفاده از ابزار PDETool [۱۵] که محیطی برای مدل‌سازی و شبیه‌سازی شبکه‌های پتری و بسطهای آن از جمله CSAN [۱۶و۱۷] است، و روی سیستمی با پردازنده دو هسته‌ای با سرعت ۲/۴ گیگاهرتز و ۲ گیگابایت حافظه مورد ارزیابی قرار گرفت که جزئیات و نتایج این ارزیابی در ادامه بیان می‌شود.

ابتدا الگوریتم ارائه شده در این مقاله که سعی در بهبود سرعت پردازش پرس‌وجوهای پیوسته روی جریان داده با به کارگیری موازی‌سازی با زمانبندی رویدادگرای ارائه شده دارد، با روش بدون استفاده از موازی‌سازی (اجرای متوالی و الگوریتم زمانبندی Round Robin) [۸] و نیز با الگوریتم زمانبندی min-latency به کار رفته در سیستم مدیریت جریان داده Aurora که هدف آن به حداقل رساندن تاخیر تاپل‌ها است [۱۸] با انجام آزمایشاتی به شرح زیر مقایسه شده است.

پرس‌وجوهای مورد استفاده در آزمایشات شامل عملگرهای گزینش، پرتو، و پیوند (جریان با رابطه و جریان با جریان) می‌باشند. هر عملگر با دو مشخصه زمان اجرا (با مقادیر بین ۲ تا ۵ میلی ثانیه) و بهگزینی[۴۱] (با مقادیر بین ۰/۸ تا ۱/۲) مدل شده است. یک پرس‌وجو به صورت دنباله‌ای از سه عملگر پیوند، گزینش و پرتو مدلسازی شده که ۱۲ نوع متمایز از ترکیبات مختلف پرس‌وجوهای شامل فقط یک نوع عملگر تا شامل هر سه نوع عملگر ایجاد شد. جریان داده ورودی به صورت تحلیلی[۴۲] (با نرخ ورود دارای توزیع پوآسن و نرخ 1/0.0085 میلی ثانیه) ایجاد و مورد پردازش پرس‌وجوها قرار می‌گیرد.

با توجه به اینکه ماشین‌ها، ماشین‌هایی منطقی هستند از هزینه ارتباطات بین آنها صرفنظر شده است. همچنین اگر در روش پیشنهادی k ماشین هر یک با سرعت x به کارگرفته می‌شوند، در روشهای دیگر مورد مقایسه که یک پردازنده دارند، ماشینی با سرعت k.x استفاده شده است.

پارامترهای اندازه‌گیری شده در این ارزیابی عبارتند از: تاخیر تاپل (زمان پاسخگویی[۴۳])، میزان حافظه مصرفی[۴۴] ، گذردهی[۴۵] سیستم و از دست دادن تاپل‌ها[۴۶]. همچنین میانگین پارامترهای فوق محاسبه و مورد مقایسه قرار گرفته‌اند.

- **نتایج ارزیابی**

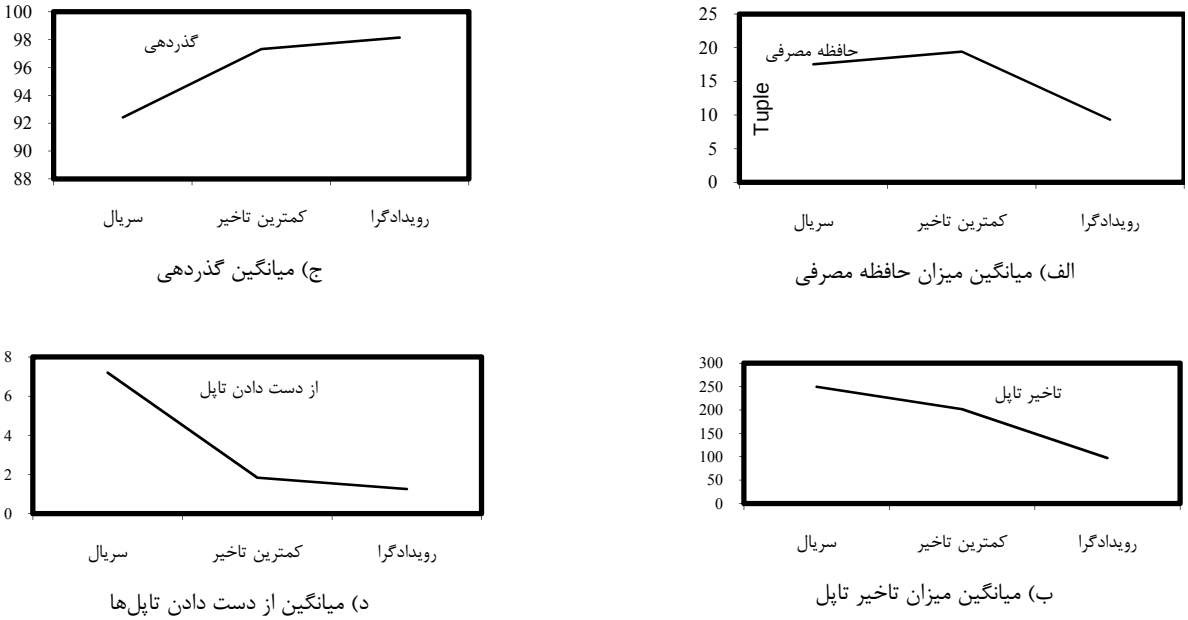
قبل از تحلیل کارایی سیستم پیشنهادی و مقایسه آن با سایر روشهای مورد نظر، کارایی سیستم را به ازای مقادیر مختلف برای K مقایسه می‌کنیم. شکل ۱۰ حافظه مصرفی و تاخیر تاپل را برای تعداد ۱ الی ۵ ماشین منطقی نشان می‌دهد.

همانطور که در شکل (۱۰) دیده می‌شود، اختلاف بین حالت تک ماشینی (بدون موازی‌سازی) و حالت دو ماشینی (انجام موازی‌سازی) بسیار زیاد و چشمگیر است و در بقیه موارد نسبت مستقیمی بین افزایش تعداد ماشینها و بهبود کارایی سیستم وجود دارد. به رغم اینکه با انتخاب تعداد ماشین منطقی بیشتر کارایی سیستم بهتر خواهد شد، اما در ارزیابی صورت گرفته، حداقل منابع را در نظر گرفته‌ایم. لذا نتایج ارائه شده در نمودارهای بعدی با فرض K=2 است. با انتخاب تعداد ماشین بیشتر از ۲، کارایی سیستم پیشنهادی بهتر از مقادیر ارائه شده خواهد بود.

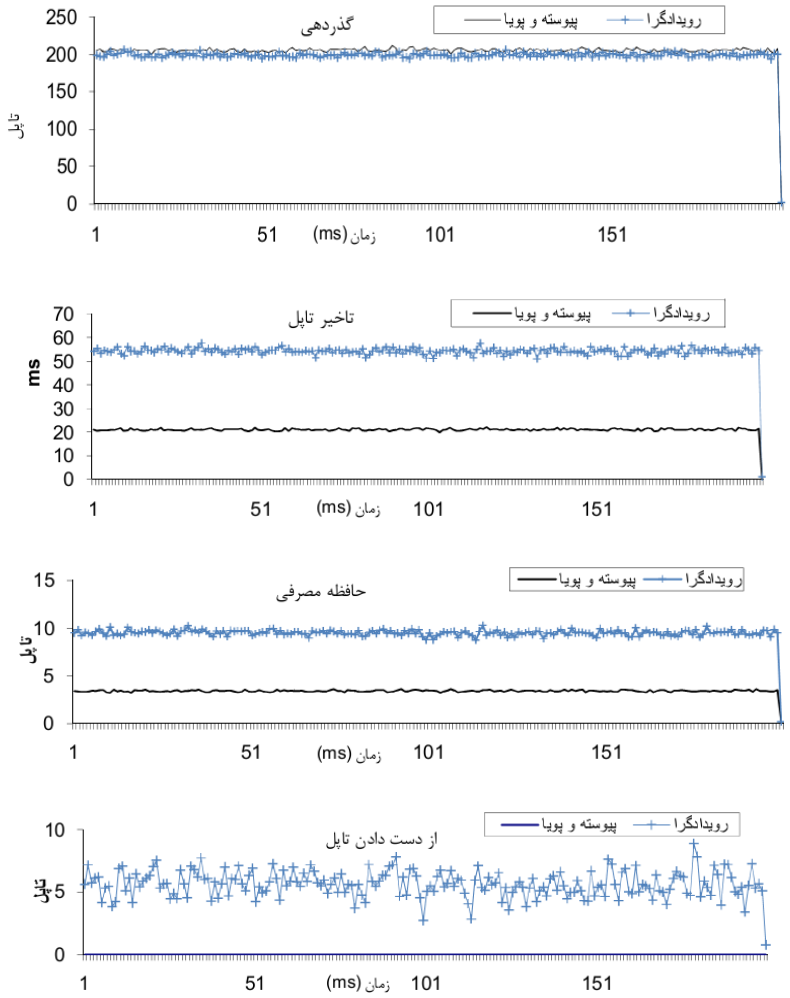
سیستم پیشنهادی (پردازش موازی پرس‌وجو با الگوریتم زمانبندی رویدادگرا با دو ماشین) به نام رویدادگرا با روش اجرای سریال پرس‌وجو و استفاده از الگوریتم زمانبندی عملگرهای Round Robin (به نام روش سریال) و روش اجرای سریال با الگوریتم زمانبندی min-latency (روش کمترین تاخیر) مقایسه می‌شود.

میانگین مقادیر مربوط به پارامترهای حافظه مصرفی، تاخیر تاپل، گذردهی و از دست دادن تاپل‌ها که مربوط به پنجبار اجرای روش‌های مورد مقایسه است در شکل (۱۱) نمایش داده شده است.

نتایج حاصل از نمودارهای شکل (۱۱) بیانگر آن است که روش پیشنهادی بهبود قابل توجهی نسبت به روش فاقد موازی‌سازی و نیز الگوریتم کمترین تاخیر از نظر حافظه مصرفی و تاخیر تاپل دارد.



شکل ۱۱- میانگین پارامترهای مورد اندازه‌گیری حاصل از ۵ بار اجرای روش‌های مورد مقایسه



شکل ۱۲- مقادیر پارامترهای مورد ارزیابی نسبت به زمان

همچنین، اگرچه گذردهی و از دست دادن تاپل‌ها در الگوریتم کمترین تاخیر بسیار بهتر از روش سریال است اما همچنان روش پیشنهادی عملکرد بهتری نسبت به آن دارد.

براساس نتایج حاصل از ارزیابی‌ها، به خصوص نمودارهای شکل (۱۱) می‌توان نتیجه گرفت که روش پیشنهادی که در پردازش پرس‌وجوها از موازی‌سازی (با دو ماشین منطقی) استفاده می‌کند کارایی بسیار بهتری نسبت به روش بدون موازی‌سازی (اجرای سریال) و روش استفاده از الگوریتم کمترین تاخیر دارد.

در ادامه، الگوریتم زمان‌بندی پیوسته و پویای ارائه شده در بخش ۲-۱-۴ با روش زمان‌بندی رویدادگرا ارائه شده در بخش ۲-۱-۳، با انجام آزمایشاتی که نتایج آنها به شرح زیر است، مقایسه شده‌اند.

شکل (۱۲) نمودارهای مقادیر پارامترهای تاخیر تاپل، گذردهی، حافظه مصرفی و از دست دادن تاپل‌ها برای روش‌های مورد مقایسه نسبت به زمان را نشان می‌دهد.

براساس نمودارهای شکل (۱۲)، روش زمان‌بندی پیوسته و پویای پیشنهادی بهبود بسیار چشمگیری در کارایی سیستم پردازش موازی پرس‌وجوی ارائه شده از نظر تاخیر تاپل، گذردهی، حافظه مصرفی و نیز از دست دادن تاپل‌ها فراهم می‌کند. میانگین مقادیر مربوط به هر یک از پارامترهای کارایی سیستم در شکل (۱۳) نمایش داده شده است.

علاوه بر این بهبود قابل توجه در مورد تمامی پارامترهای کارایی سیستم، نتایج نمودارهای شکل (۱۴) نشان می‌دهند که همانگونه که انتظار می‌رفت در روش پیشنهادی، میزان نوسانات پارامترهای کارایی سیستم بسیار کمتر از روش زمان‌بندی رویدادگرا است.

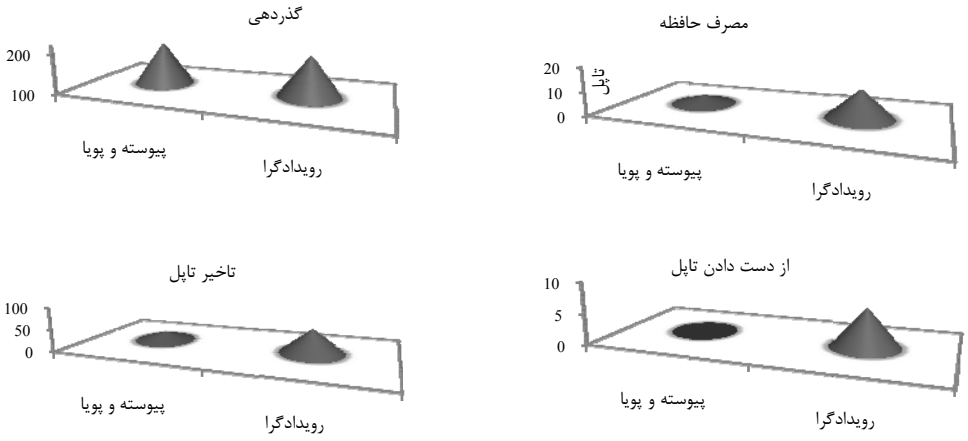
به منظور تعیین میزان دقیق این بهبود در زمینه یکنواخت کردن عملکرد سیستم و کاهش نوسانات مربوط به پارامترهای کارایی سیستم، نمودارهای مجموع انحراف از میانگین مقادیر پارامترهای مختلف در شکل (۱۴) بررسی شده‌اند.

کاهش میزان انحراف از میانگین در روش زمان‌بندی پیوسته و پویای پیشنهادی به معنی کاهش نوسانات در مقادیر هر یک از پارامترهای کارایی سیستم است. لذا سیستم موازی پردازش پرس‌وجو که از الگوریتم زمان‌بندی پیوسته و پویای ارائه شده استفاده می‌نماید عملکرد یکنواخت و نرمال‌تری خواهد داشت. این ویژگی سیستم پیشنهادی را می‌توان به منزله آن دانست که تغییرات عملکرد و کارایی سیستم نسبت به تغییرات جریان داده ورودی و وضعیت سیستم بسیار ناچیز است. به عبارت دیگر، تطبیق‌پذیری سیستم پیشنهادی نسبت به تغییرات بسیار بیشتر از سیستم با زمان‌بندی رویدادگرا خواهد بود. همچنین، می‌توان حالت پایدار[۴۷] را برای سیستم پیشنهاد شده تعیین نمود.

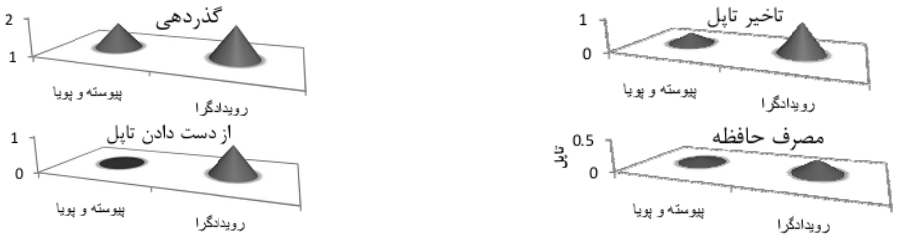
۲-۲- رویکرد افراز داده‌ها (مدل محاسباتی نگاشت-کاهش)

رویکرد و مدل محاسباتی نگاشت-کاهش به دلیل سادگی و در عین‌حال رسا و گویا بودن، موفقیت‌های فراوانی به ویژه در زمینه امکان موازی‌سازی داده‌ها به دست آورده است. با این وجود، در مورد کاربردهای پیوسته و جریانی هنوز به تکامل نرسیده است. علت این ضعف نیز اتّکا این مدل به بافرهای دیسک برای نتایج میانی، عدم بهره‌گیری از توازی خط‌لوله‌ای و ناتوانی در به حداقل رساندن تاخیر به ویژه برای کارهای پیچیده می‌باشد.

در این بخش، روشی برای پردازش برخط پرس‌وجوهای پیوسته روی جریان‌های داده پیوسته با بهره‌گیری از مدل برنامه‌نویسی و رفع چالش‌ها و مشکلات این مدل برای پردازش برخط و پیوسته جریان‌های داده ارائه شده است.



شکل ۱۳- میانگین مقادیر پارامترهای کارایی سیستم



شکل ۱۴- مجموع انحراف از میانگین برای مقادیر پارامترهای مورد ارزیابی

۲-۲-۱- مدل محاسباتی نگاشت-کاهش[۴۸]

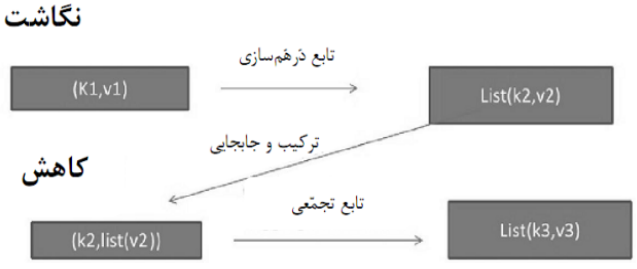
مدل محاسباتی نگاشت-کاهش [۲۱و۲۲] می‌تواند درجه نسبتاً بالایی از موازی‌سازی داده‌ها را ارائه نماید. بدین منظور، با الهام از رویکرد تقسیم‌وغلبه[۴۹]، بار کاری به وظائفی شکسته و خُرد می‌شوند که بتوانند به طور مستقل از یکدیگر روی هر گره محاسباتی عمومی[۵۰] پردازش شوند. البته ایده اولیه نگاشت-کاهش برای پردازش دسته‌ای[۵۱] و غیربرخط ارائه شده است اما برای سازگارنمودن آن با طبیعت پیوسته سیستم‌های جریان داده، داده‌های باید تغییراتی در مدل محاسباتی نگاشت- کاهش پایه انجام گیرد. مهمترین چالش‌ها و موانع در راه به کارگیری مدل محاسباتی نگاشت-کاهش برای پردازش جریان‌های داده در بخش ۲-۲-۲ مورد بررسی قرار گرفته و راه حل این موانع در بخش ۲-۲-۳ در قالب روش پیشنهادی ارائه شده است.

به طور کلی، مدل محاسباتی نگاشت-کاهش سنّتی مبتنی بر رویکرد تقسیم و غلبه و تفکّری داده- محور بنا شده است که مرکّب از دو گام اصلی می‌باشد:

الف) گام نگاشت: با اعمال تابع نگاشت مورد نظر بر روی رکوردهای ورودی، آنها را به گروه‌هایی شکسته و تقسیم می‌نمائیم. به عبارتی، در این گام، داده‌ها از یک فایل سیستم توزیع‌شده که روی مجموعه‌ای از گره‌های محاسباتی در دسترس افراز شده است، خوانده می‌شود و به صورت مجموعه‌ای از زوج (کلید، مقدار) ارسال می‌گردد. وظیفه‌های گام نگاشت هریک از رکوردهای ورودی را به صورت مستقل از سایر رکوردها پردازش می‌کنند و برای هر رکورد، نتایج میانی مربوطه را از تابع list(key, value) می‌سازند. داده‌های میانی تا هنگامی که گره‌ها پردازشِ داده‌های تخصیص داده شده به آنها در وظائف گام نگاشت را به پایان برسانند، در دیسک‌های محلی آنها بافر خواهند شد.

ب) گام کاهش: پس از آنکه وظائف گام نگاشت در تمامی گره‌های پردازشی خاتمه یافت، گام کاهش آغاز می‌گردد که در آن همه داده‌های میانی با مقدار یکسان کلید، در همان گره جمع‌آوری می‌شوند تا عملیات تجمّعی[۵۲] مناسب (که توسط برنامه‌نویس تعریف می‌شود) روی آنها اجرا گردد. گره‌هایی که وظیفه‌های گام کاهش به آنها تخصیص داده شده است ابتدا با یک گره مرکزی ارتباط برقرا می‌کنند تا مجموعه‌ای از کلیدها بین آنها تخصیص داده شود. سپس، داده‌های مربوطه را از هر گرهی که چنین کلیدی را به عنوان خروجی گام نگاشت خودش تولید نموده است، دریافت و جمع‌آوری می‌نمایند. این داده‌ها بازیابی شده و به طور محلی ذخیره می‌شوند و سپس پردازش انجام شده و نتایج به صورت list(value) به ازای هر کلید تولید می‌گردند.

البته می‌توان یک مرحله *ترکیب*[۵۳] نیز انجام داد که در آن یک تجمّع شبیه به کاهش روی نتایج میانی با کلیدهای یکسان روی خروجی‌های گام نگاشت (قبل از ارائه شدن به گره کاهش) انجام گیرد. این روش از نظر مصرف پهنای باند و توزیع نمودن بارِکاری تجمّع مربوط به یک وظیفه‌ی کاهش مناسب است.



شکل ۱۵- گام‌ها در مدل محاسباتی نگاشت-کاهش

این مدل محاسباتی اولیه و ابتدایی عمدتاً برای انجام محاسبات دسته‌ای روی مجموعه‌های خیلی بزرگ داده‌ها مناسب است. اما در کاربردهای پردازش برخط پرس‌وجوها که محاسبات تجمّعی برخط[۵۴] و پردازش جریان‌های داده[۵۵] مهمترین مصداق‌های ان هستند، توسعه‌هایی از مدل اولیه و سنّتی نگاشت-کاهش از جمله نگاشت-کاهشِ خط‌لوله‌ای[۵۶] ارائه و به کارگیری شد که در آن گام نگاشت روی داده‌های نامتناهی عمل کرده (پردازش جریان داده) و گام کاهش هم می‌تواند نتایج زودهنگام را تولید نماید (محاسبات تجمّعی برخط). وظائف نگاشت و کاهش به طور پیوسته اجرا می‌شوند [۳۹]. اما برای پردازش پیوسته پرس‌وجوها روی جریان‌های داده‌ی پیوسته، نامتناهی، سریع، غیرقابل پیش‌بینی و انفجاری ملاحظات دیگری نیز لازم است که با بررسی چالش‌های به کارگیری مدل محاسباتی نگاشت-کاهش برای پردازش جریان‌های داده می‌توان به آنها پی‌بُرد.

۲-۲-۲- چالش‌های به کارگیری مدل نگاشت-کاهش برای پردازش جریان‌داده‌ها

مهمترین چالش‌ها و موانع در به کارگیری مدل نگاشت-کاهش برای پردازش جریان‌داده‌ها را می‌توان اینگونه برشمرد [۳۱]:

- **پردازش دسته‌ای**

همانگونه که بیان شد، ایده‌ی اصلی مدل محاسباتی نگاشت-کاهش برای به کارگیری در کاربردهای داده- محور اما به صورت غیربرخط و دسته‌ای بود. به عبارت دیگر، مهمترین وجه تمایز قابلیت‌های این مدل محاسباتی با محیط یک سیستم جریان داده آنست که مدل محاسباتی نگاشت-کاهش فرض بر در اختیار داشتن مجموعه همه داده‌های ورودی دارد و با این فرض سعی در انجام موازی‌سازی در انجام وظائف با رویکرد موازی‌سازی داده‌ای (افراز داده‌ها) دارد؛ حال آنکه در سیستم‌های جریان داده عملاً این فرض منتفی بوده و داده‌ها به صورت نامتناهی و پیوسته در طول زمان دریافت خواهند شد.

- **اتکاء به بافرهای دیسک**

چالش و مسئله دیگر در به‌کارگیری مدل محاسباتی نگاشت-کاهش برای پردازش جریان‌داده‌ها این است که نمونه‌هایی که از این مدل محاسباتی برای پردازش جریان‌های داده پیاده‌سازی و ارائه شده‌اند عمدتاً متکی بر استفاده از بافرهای دیسک هستند که تاخیر تاپل ناشی از این مسئله برای پرداش برخط جریان‌های داده قابل چشم‌پوشی نیست.

به عنوان مثال، نمونه‌های آزمایشگاهی نظیر HOP[۵۷] [۲۱] به خاطر حفظ سازگاری با Hadoop و به دلیل اتکاء به بافرهای میانی مبتنی بر دیسک و سیاست‌های زمانبندی وظیفه‌گرا مانع از پشتیبانی موثر از کاربردهای جریان داده خواهند بود [۳۹]. از طرفی، موتورهای پردازش جریان داده موجود نیز امکاناتی در خصوص موازی‌سازی خط‌لوله‌ای داده و وظائف ارائه می‌کنند اما این پشتیبانی در سطوح خیلی کلان و دُرشت‌دانه صورت می‌گیرید و لذا از نظر بهره‌وری از منابع محاسباتی و به ویژه تاخیر تاپل چندان موثر و کارامد نیستند [۳۹].

- **تجریدی۵۸ بودن مدل**

همانطور که از نام آن پیداست، مدل نگاشت-کاهش یک مدل محاسباتی است که البته داده‌محور بوده، برای محیط‌های توزیع‌شده و کاربردهای پردازش دسته‌ای و

غیربرخط مناسب است. این مفهوم و مدل تجریدی برای آنکه بتواند در عمل به کار آید باید بسیار جزئی، مشخص و عملیاتی گردد. به عنوان مثال، توابع نگاشت و کاهش که باید توسط برنامه‌نویس و بنابه کاربرد تعریف شوند، نیاز به آن دارند که به طور کاملاً مشخص تعریف و تشریح گردند. یکی از چالش‌های مهم در به کارگیری مدل محاسباتی نگاشت-کاهش در پردازش جریان داده همین مسئله یعنی تعیین وظائف لازم در گام نگاشت و گام کاهش در پردازش برخط جریان‌های داده است.

- **نابالغ بودن مدل**

مدل محاسباتی نگاشت-کاهش از حدود سال ۲۰۱۰ به طور جدی معرفی و به کارگیری شد و با توجه به اینکه اصولاً برای پردازش دسته‌ای مناسب بود، در حوزه پردازش جریان داده هنوز به در اوان راه تکامل به سر می‌برد. لذا کارهایی که در زمینه به کارگیری مدل محاسباتی نگاشت-کاهش در پردازش جریان داده ارائه شده‌اند بسیار معدود و مقدماتی محسوب می‌شوند. اما به هر حال، با توجه به مزایای این مدل از جمله سادگی، مناسب بودن برای محیط‌های توزیع‌شده، امکان توازن بار روی خوشه‌ها، مقیاس‌پذیری بالا، و انعطاف پذیری نسبتاً قابل قبول، سفارشی نمودن این مدل و بهره‌برداری از آن در مسئله پردازش برخط جریان‌های داده بسیار مورد توجه و اقبال متخصصان و پژوهشگران این حوزه قرار گرفته است.

- **عدم وجود زبان پایگاه‌داده**

با توجه نوپا بودن مدل محاسباتی نگاشت- کاهش برای پردازش جریان داده، هنوز زبان سطح بالایی نظیر SQL برای استفاده از این مدل در سیستم‌های جریان داده ارائه نشده است.

براساس چالش‌ها، مسائل و موانعی که در راه به کارگیری مدل محاسباتی نگاشت-کاهش در پردازش جریان‌های داده وجود دارد، در این مقاله روشی برای رفع این چالش‌ها و بهره‌گیری از مدل محاسباتی نگاشت-کاهش در پردازش جریان‌های داده، با هدف اِعمال توازی داده‌ها (رویکرد افراز داده‌ها) ارائه شده است.

۲-۲-۳- روش پیشنهادی: مدل نگاشت-کاهش پیوسته نامتوازن

همانگونه که در چالش‌ها و موانع به کارگیری مدل محاسباتی نگاشت-کاهش برای پردازش جریان داده‌ها عنوان شد، مهمترین و اصلی‌ترین چالش این است که مدل نگاشت-کاهش برای کاربردهای پردازش دسته‌ای و غیربرخط ارائه شده و مناسب است، حال آنکه پردازش برخط یکی از نیازمندی‌های اساسی و پایه در کاربردهای جریان داده محسوب می‌شود.

در روش پیشنهادی، برای حل این مُعضل و دیگر موانع راه حل‌هایی به شرح زیر ارائه شده‌اند که ترکیب آنها، روشی مبتنی بر مدل محاسباتی نگاشت- کاهش برای پردازش موازی جریان‌های داده فراهم می‌آورد.

به منظور برطرف نمودن چالش دسته‌ای و غیربرخط بودن مدل محاسباتی نگاشت-کاهش، روش پیشنهادی با تعریف متناسب برای وظائف نگاشت و کاهش، و پردازش خط‌لوله‌ای[۵۹] و افزایشی[۶۰] ارائه شده است. برای رفع چالش نامتناهی بودن جریان داده نیز از پنجره‌بندی جریان داده استفاده خواهد شد. به منظور حذف بافرهای دیسک در مدل و نمونه‌های نگاشت-کاهش نیز استفاده از حافظه‌های اشتراکی که در معماری سیستم‌های متداول چندهسته‌ای مرسوم است، جایگزین شده است.

در مدل محاسباتی نگاشت-کاهش، داده‌های با کلید یکسان روی ماشین‌های یکسانی افراز و توزیع می‌شوند (نگاشت) و پس از پردازش، داده‌های با کلید یکسان ادغام شده و تابع تجمّعی مناسب روی آنها اِعمال می‌گردد (کاهش).

براساس معماری سیستم پیشنهادی (شکل ۵)، با فرض داشتن k ماشین محاسباتی منطقی یکسان[۶۱]، برای اینکه بتوان وظیفه و کار یکسانی (پرس‌وجوی پیوسته ثبت شده) را به طور موازی روی داده‌های افراز شده اجرا نمود (رویکرد توازی داده‌ها)، راه حل پیشنهادی بدین شکل خواهد بود؛ جریان داده ورودی (که نامتناهی، پیوسته، سریع، غیرقابل پیش‌بینی و انفجاری است) باید به مجموعه‌های موازی و مستقل افراز گردد و هر مجموعه به یک ماشین محاسب (یک ماشین منطقی از معماری پیشنهادی) تخصیص یابد.

هر ماشین منطقی با دریافت بخش[۶۲] داده‌های خود، وظیفه یکسانی با سایر ماشین‌ها را روی داده‌های بخش خود اِعمال می‌نماید و در نهایت نتایج میانی پردازش‌ها جمع‌آوری و تجمّع متناسب روی آنها انجام می‌گیرد.

در این حالت که با اصول مدل محاسباتی نگاشت-کاهش نیز سازگاری و همخولنی دارد باید وظائف گام‌های نگاشت و کاهش به طور مشخص و با تطابق با مدل نگاشت-کاهش از یک سو، و ماهیت پیوسته پرس‌وجوها (وظائف) و جریان داده‌ها از سوی دیگر تعیین گردد.

به منظور تبدیل جریان داده ورودی نامتناهی به زیر مجموعه‌هایی متناهی و مستقل، گره مبداء در معماری سیستم پیشنهادی (شکل ۵) به عنوان یک *جداکننده*[۶۳] عمل نموده و جریان داده را به پنجره‌هایی مستقل و مجزّا تفکیک و افراز می‌نماید (*پنجره‌بندی*[۶۴]).

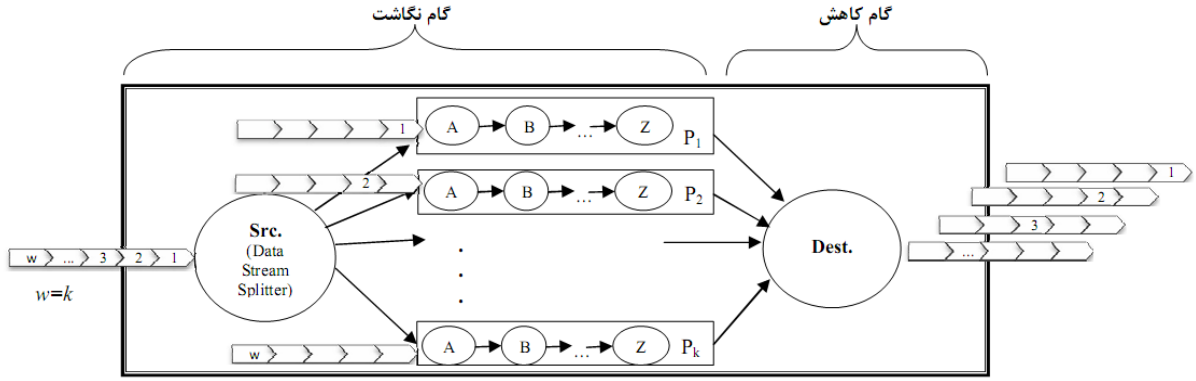
در مدل محاسباتی نگاشت-کاهش که با رویکرد توازی داده‌ای ارائه شده است، کار و وظیفه‌ی یکسانی روی داده‌های موازی اجرا می‌گردد. لذا هر یک از ماشین‌های منطقی کار یکسانی (پرس‌وجوی ثبت‌شده) را روی بخش داده‌ای خودشان اجرا خواهند کرد.

براین اساس، وظائف گام‌های نگاشت و کاهش در روش پیشنهادی به شرح زیر تعیین می‌شوند:

گره مبداء در معماری سیستم پیشنهادی به عنوان جداکننده، تاپل‌های جریان داده ورودی را براساس مقدار کلید به زیربخش‌هایی افراز می‌نماید. این کلید می‌تواند مقدار برچسب زمانی تاپل‌ها باشد؛ در واقع شماره‌ی نِسبی[۶۵] هر تاپل در پنجره مربوطه‌اش به عنوان کلید در گام نگاشت تلقّی می‌شود. در اینصورت، گره مبداء، تاپل شماره iاُم از پنجره را به ماشین منطقی iاُم تخصیص می‌دهد و بدین ترتیب، جریان داده (هر پنجره از جریان داده) به w بخش افراز می‌گردد و هر ماشین، پرس‌وجوی ثبت‌شده را روی بخشِ (تاپل‌های) خود اجرا می‌کند.

چنانچه تعداد ماشین‌ها بیشتر از تعداد تاپل‌های پنجره باشد (k>w)، می‌توان تاپل‌های بیش از یک پنجره را به ماشین‌ها تخصیص داد (مثلاً تاپل اول از پنجره اول را به ماشین شماره ۱ و تاپل wاُم به ماشین شماره w، و تاپل اول از پنجره بعدی به ماشین w+1اُم و به همین ترتیب تا تاپل k-w اُم که به ماشین kاُم تخصیص می‌یابد). در چنین حالتی، می‌توان تمام تاپل‌های با کلید یکسان (مثلاً تاپل‌های با برچسب ۱ یا همان اولین تاپل پنجره‌ها) را از ماشین‌های مختلف جمع‌آوری نمود (مشابه آنچه که در مدل نگاشت- کاهش مبنا تعریف می‌شود).

اما در جریان داده‌ها که ترتیب تاپل‌ها اهمیت پیدا می‌کند، این تاپل‌ها باید به نوبت پردازش شوند (مثلاً تاپل دوم تا wاُم از پنجره اول باید قبل از تاپل اول از پنجره دوم پردازش و ارائه شوند). لذا در مدل نگاشت-کاهش پیشنهادی اندازه پنجره را برابر با اندازه تعداد ماشین‌های منطقی در معماری سیستم در نظر می‌گیریم (k=w).



شکل ۱۶- گام‌های کاهش و نگاشت در مدل پیشنهادی و وظایف هر گام

بدین ترتیب، در گام نگاشت مدل نگاشت-کاهش پیشنهادی گره مبداء معماری (جداکننده)، تاپل‌های ورودی جریان داده را به ترتیب شماره‌ی نِسبی آنها در پنجره (همان کلید در مدل نگاشت)، بین ماشین‌های مُحاسِب تقسیم می‌نماید (تاپل iاُم از پنجره، به ماشین iاُم تخصیص می‌یابد). سپس، هر ماشین به طور موازی و مستقل، پرس‌وجوی ثبت‌شده را روی تاپل خود اجرا می‌نماید (شکل ×).

در گام کاهش، نتایج میانیی حاصل از پردازش ماشین‌های محاسب باید جمع‌آوری شده و عملیات تجمّعی متناسب روی آنها انجام گیرد که در روش پیشنهادی، وظائف این گام شامل جمع‌آوری نتایج اجرای پرس‌وجو از ماشین‌ها، مرتّب کردن نتایج به ترتیب ورودی تاپل‌ها، و ارائه آنها در قالب جریان داده به کاربر به عنوان خروجی می‌باشد. با توجه به اینکه تخصیص تاپل‌ها به ماشین‌ها در گام نگاشت براساس ترتیبِ نسبی تاپل‌ها در پنجره بوده است، برای اطمینان از حفظ ترتیب ورودی تاپل‌ها در خروجی، در گام کاهش کافیست تاپل‌ها به ترتیب شماره از ماشین‌ها جمع‌آوری گردند (ابتدا نتیجه ماشین اول، سپس دوم، و به همین ترتیب).

از آنجا که ماشین‌های محاسب در معماری سیستم پیشنهادی ماشین‌های یکسانی[۶۶] در نظر گرفته شده‌اند، و از طرف دیگر محاسبات آنها (پردازش پرس‌وجوی ثبت‌شده) نیز یکسان است، حداقل به صورت تئوری انتظار می‌رود ماشین‌ها به ترتیبی که تاپل‌ها را دریافت می‌نمایند خروجی‌های خود را ارائه نمایند که این منجر به رعایت و حفظ ترتیب در آماده‌شدن نتایج و گام کاهش خواهد شد.

به عبارتی، گام کاهش نیاز به اجرای وظیفه‌ی سنگینی نخواهد داشت و صرفاً شامل ارائه نتایج آماده شده به صورت دنباله‌ای و در قالب جریان خواهد بود.

شکل (۱۶) گام‌های کاهش و نگاشت در مدل پیشنهادی و وظایفی که در هرکدام انجام می‌شود را نشان می‌دهد.

با توجه به اینکه در روش پیشنهادی و طبق آنچه که توسط مدل محاسباتی نگاشت-کاهش و ماهیت جریان داده و پرس‌وجوهای پیوسته به روش پیشنهادی دیکته شده است، گام نگاشت و کاهش که به طریقه‌ی مذکور تعیین شده‌اند از نظر حجم و پیچیدگی عملکرد متعادل و متوازن نیستند (گام نگاشت سنگین در برابر گام کاهش سَبُک). لذا روش پیشنهادی که درآن وظائف این گام‌ها به طور پیوسته اجرا خواهند شد به نام روش *نگاشت-کاهش پیوسته نامتوازن (UCMR)*[۶۷] نامگذاری شده است.

معماری سیستم موازی پیشنهادی می‌تواند بر روی بستر چندهسته‌ای[۶۸] و همانگونه که در [۲۵] معرفی و ارائه کرده‌ایم پیاده‌سازی گردد. در اینصورت، ماشین‌های منطقی که در قالب هسته‌های سیستم عمل می‌کنند با استفاده از حافظه اشتراکی سیستم فرآیند میانگیری[۶۹] را انجام می‌دهند و لذا معضلِ بافرهای دیسک در مدل سنتی و پایه‌ی نگاشت-کاهش و نمونه‌های پیاده‌سازی شده، جای خود را به استفاده از بافرهای حافظه‌ی اشتراکی خواهد داد.

۳- کارهای مرتبط

کارهای تحقیقاتی فراوانی در حوزه سیستم‌های جریان داده صورت گرفته است [۱-۸] و نمونه‌های اولیه[۷۰] متعددی از سیستم‌های مدیریت جریان داده نیز ارائه شده‌اند که STREAM [۲] و Aurora [۳] نمونه‌هایی از این دست هستند.

استراتژی‌های زمان‌بندی عملگرها برای پردازش پرس‌وجوهای پیوسته روی جریان‌های داده طیف وسیعی را شامل می‌شود؛ از استراتژهای ساده نظیر Round Robin، chain و Greedy [۸] تا الگوریتم‌های پیچیده‌تر [۱۹] و [۲۰]. برخی از این الگوریتم‌ها با هدف بهینه‌سازی یک معیار کارایی ارائه شده‌اند [۵۱] در حالیکه برخی سعی در بهینه سازی چند معیار یا یک معیار ترکیبی دارند [۳۳]. برای داشتن مروری بر تکنیک‌های پردازش پرس‌وجو در پایگاه داده و پردازش موازی پرس‌وجو، خوانندگان محترم را به [۱۱]، [۱۲]، [۲۵] و [۲۶] ارجاع می‌دهیم. موازی سازی اجرای پرس‌وجوهای پیوسته سیستم‌های جریان داده نیز در مراجع زیر به چالش کشیده شده‌اند.

تطبیق‌پذیری در سیستم‌های جریان داده توزیع‌شده در مراجع متعددی مورد توجه قرار گرفته است [۲۸-۳۰]. به منظور بهبود تطبیق‌پذیری و مقیاس‌پذیری، شاه و همکاران در [۳۷] عملگر flux را برای افراز عملگرهای وضعیت‌گرا و حساس به محتوی (مانند پیوند و گروهبندی) ارائه کردند. این عملگر تطبیق‌پذیری افقی و درون عملگری را با افراز مجدد و توزیع برخط بارکاری و وضعیت عملگرها در بین گره‌های پردازشی به انجام می‌رساند. معماری Eddy [۳۸] با از بین بردن نقشه پرس‌وجو و استفاده از عملگر Eddy، تطبیق‌پذیری را با دانه‌بندی ریز و بین عملگرها انجام می‌دهد. سربارهای بسیار زیاد این معماری مهمترین نقطه ضعف آن محسوب می‌شود. مسئله گلوگاه بودن این عملگر واحد در نسخه توزیع‌شده Eddy [۳۹] رفع شده است اما سربارهای ذخیره‌سازی (مثلاً برای ذخیره نقشه‌بیتی Done به ازای هر تاپل) و پردازشی (مثلاً به روزرسانی نقشه بیتی و نیز جمع‌آوری اطلاعات آماری مانند بهگزینی و هزینه عملگرها توسط هر عملگر) همچنان باقی است که با توجه به توزیع شده بودن آن در یک شبکه محلی، هزینه‌های ارتباطی قابل توجه و چالش برانگیز خواهد بود. در [۵۵] براساس داده‌های آماری راجع‌به داده‌ها، برای هر زیرمجموعه از داده‌ها که ویژگی‌های متمایزی دارند، با استفاده از یک طبقه‌بندی کننده مسیرجداگانه‌ای محاسبه و مورد استفاده قرار می‌گیرد. این رویکرد، راه‌حلی برای تطبیق‌پذیری است که بین دو استراتژی اصلی می‌گنجد: استراتژی استفاده از یک نقشه اجرای واحد و یکپارچه و استراتژی فاقد نقشه مانند معماری Eddy. در [۴۱] با کمک مفهوم concept-drift در یادگیری ماشین، تطبیق‌پذیری query mesh بهبود داده شده است. روش‌های چیدمان پویای عملگرها در سیستم‌های پردازش پرس‌وجوی توزیع شده در مراجعی مانند

[۴۲ و ۴۳] بررسی شده‌اند. ژو و همکاران با توجه به توزیع شده بودن سیستم در یک شبکه محلی، در [۴۲] توازن بار با تمرکز بر کاهش هزینه ارتباطات را مورد بررسی قرار دادند. تیان و دویت در [۳۹] استراتژی‌های مختلفی برای مسیریابی معرفی و مقایسه کرده‌اند که استراتژی WSCQ (Weighted Selectivity-Cost-Qlength) که در آن هر سه معیار بهگزینی، هزینه اجرای عملگر و وزن صف ورودی عملگر به حساب می‌آیند به عنوان بهترین استراتژی شناخته شده است. در مسیریابی مورد استفاده در زمان‌بندی پیوسته و پویای XYZ، به روزرسانی وزن لبه‌ها برای انتخاب لبه با وزن مینیمم با توجه به بار کاری ماشین مقصد لبه انجام می‌شود. از آنجا که ماشین مقصد باید همه تاپل‌های منتظر در صف خود را پردازش کند، بهگزینی آن عملگر را در محاسبه وزن لحاظ نمی‌کنیم.

در زمینه رویکرد موازی‌سازی افراز داده‌ها، مقاله Dean و Ghemawat در مورد نگاشت-کاهش گوگل [۲۲] به عنوان یک مرجع استاندارد شناخته می‌شود که پایه‌های پیاده‌سازی متن‌باز Hadoop را نیز شکل داده است. البته طراحی نگاشت- کاهشِ گوگل برای خوشه‌های خیلی بزرگ از داده‌هایی که احتمال خرابی در آنها نیز بالاست ارائه شده بود. به طور کلی، اخیراً ایده و مدل نگاشت-کاهش برای به کارگیری در پردازش جریان‌های داده به طور وسیعی مورد استفاده و اقبال محققان این حوزه قرار گرفته است [۴۰، ۴۳، ۴۵ و ۴۶]. در [۳۱] مروری کلی بر پردازش موازی جریان‌های داده توسط مدل محاسباتی نگاشت-کاهش ارائه شده است که علاوه بر اشاره به قابلیت‌ها و نقاط قوّت این مدل به چالش‌ها و موانع به کارگیری آن در پردازش جریان داده نیز پرداخته است. پیاده‌سازی یک چارچوب از مدل نگاشت-کاهش برای پردازش داده‌های سریع برای کاربردهایی نظیر رسانه‌های اجتماعی یا تجارت الکترونیک در [۳۲] ارائه شده است.

نسخه‌ای تغییریافته از چارچوب نگاشت-کاهش Hadoop برای پشتیبانی از توابع تجمّعی برخط در [۳۴] و [۲۷] ارائه شده است. چالش‌های مدل نگاشت-کاهش برای به کارگیری در پردازش تک‌گذره و افزایشی به ویژه از منظر معماری سیستم موازی در [۳۸] مورد بررسی قرار گرفته است. اجرای پیوسته جریان‌های کاری در مدل نگاشت-کاهش روی سیستم‌های چندهسته‌ای شامل معماری سیستم، واسط‌های برنامه‌نویسی لازم و نحوه زمان‌بندی وظائف نیز در [۲۴] ارائه و تشریح شده است.

۴- نتیجه‌گیری

طبیعت پیوسته جریان‌های داده به همراه پیوسته بودن پرس‌وجوهای متعارف سبب می‌شود که پردازنده واحد به کار رفته در سیستم مدیریت جریان داده قادر به پردازش سریع تاپل‌ها نباشد. به کارگیری مکانیزم‌های کاهش بار و کنترل پذیرش کیفیت خروجی را کاهش می‌دهد و استفاده از بافرهای بزرگتر نیز تاخیر تاپل را افزایش می‌دهد که با نیازهای بی‌درنگی کاربردهای جریان داده مغایرت دارد. به طور کلی، دو رویکرد برای موازی‌سازی معرفی می‌شود: کنترل موازی و داده‌های موازی. در این مقاله راه‌حل‌هایی برای پردازش موازی جریان داده‌ها هم با رویکرد کنترل موازی (بخش ۲-۱) و هم با رویکرد داده‌های موازی (افراز داده‌ها با استفاده از مدل محاسباتی نگاشت-کاهش، بخش ۲-۲) ارائه شده است.

در رویکرد کنترل موازی، اجرای موازی پرس‌وجو با به کارگیری چندپردازنده (چندپردازه) راه حلی است که برای مشکل ناشی از گلوگاه تک‌پردازنده‌ای بودن چنین سیستم‌هایی در این مقاله ارائه شده است. زمانبندی اجرای موازی عملگرها، از طریق یافتن کوتاهترین مسیر در گراف وزن‌دار "کلان گراف پرس‌وجو" که یک دید منطقی از k ماشین است، صورت می‌گیرد. به مرور زمان، تعداد تاپل‌های منتظر در صف عملگرهای مختلف ممکن است بسیار متفاوت شوند. در صورتی که صف ورودی عملگری پر شود، زمانبندی مجدد صورت می‌گیرد که با به‌روزرسانی وزن لبه‌های کلان گراف پرس‌وجو، در مسیر انتخابی جدید از ماشین‌هایی که بار کاری زیادی دارند کمتر استفاده خواهد نمود. حفظ ترتیب ورودی تاپل‌ها در خروجی چالشی است در سیستم موازی ارائه شده که راه حل تضمین حفظ ترتیب در این مقاله معرفی و بحث شد. الگوریتم پردازش موازی پرس‌وجوها، تحلیل پیچیدگی زمانی این الگوریتم و نحوه تخصیص منابع در پردازش چند پرس‌وجو ارائه شد و کارایی سیستم پیشنهادی با روش اجرای سریال و الگوریتم Min_Latency مورد ارزیابی و مقایسه قرار گرفت. اگرچه افزایش تعداد ماشین‌های منطقی با بهبود کارایی سیستم پیشنهادی نسبت مستقیم دارد، اما در ارزیابی کارایی سیستم موازی ارائه شده حداقل منابع به کار گرفته شده است. سیستم ارائه شده ضمن صحت در عملکرد، بهبود قابل توجهی در کارایی به ویژه از نظر کاهش تاخیر تاپل‌ها به ارمغان می‌آورد.

طبیعت پیوسته، نامتناهی، سریع، غیرقابل‌پیش‌بینی و انفجاری جریان‌های داده به همراه پیوسته بودن پرس‌وجوها سبب می‌شود که سیستم‌های مدیریت جریان داده تک‌پردازنده‌ای قادر به تامین نیازمندی‌هایی نظیر زمان پاسخگویی نباشند. در بخش ۲ این مقاله، سیستم مدیریت جریان داده‌ای ارائه کردیم که با انجام موازی‌سازی در اجرای پرس‌وجوهای پیوسته در یک محیط چندپردازنده‌ای (ماشین‌های منطقی) کارایی سیستم (به خصوص تاخیر تاپل) را بهبود می‌دهد. زمانبندی ارائه شده اولیه به صورت رویدادگرا (اجرای زمان‌بندی با پرشدن صف عملگر) بود که به منظور رعایت سازگاری بیشتر با ماهیت پیوسته جریان‌های داده و پرس‌جوهای پیوسته، در قسمت بعدی همان بخش، روش زمان‌بندی پیوسته‌ای ارائه شده است. در این روش زمان‌بندی، هر عملگر با دریافت هر تاپل، از بین ماشین‌های منطقی موجود، ماشینی که کمترین بار کاری را دارد برای پردازش عملگر بعدی روی تاپل نتیجه انتخاب می‌کند. این فرآیند به طور پیوسته تکرار می‌شود و مسیر عملگرهایی که برای پردازش تاپل‌ها به کار می‌رود به طور پویا تغییر می‌کند.

صحت عملکرد روش‌های زمان‌بندی ارائه شده به صورت رسمی اثبات شده است. نتایج مدل‌سازی و ارزیابی حاکی از آنست که روش زمانبندی پیوسته و پویا بهبود چشمگیری در کارایی سیستم (تاخیر تاپل، گذردهی، حافظه مصرفی و از دست دادن تاپل‌ها) نسبت به روش زمان‌بندی رویدادگرا ایجاد می‌کند. علاوه بر بهبود در کارایی، میزان نوسانات در پارامترهای کارایی در روش پیشنهادی بسیار کمتر از روش رویدادگرا است. این بدان معنی است که تغییرات در عملکرد سیستم در صورت تغییرات در ویژگی‌های جریان داده ورودی و ویژگی‌های سیستم بسیار ناچیز است. به عبارت دیگر، روش زمان‌بندی پیوسته و پویای ارائه شده، با توزیع پیوسته و پویای بارکاری روی ماشین‌ها، تطبیق‌پذیری بالایی در سیستم ایجاد می‌نماید. همچنین، حالت پایدار سیستم را می‌توان تحلیل و تعیین نمود. توزیع شده بودن زمان‌بندی در طول زمان پردازش پرس‌وجو و توزیع شده بودن فرآیند زمان‌بندی روی همه ماشین‌های منطقی (به جای متمرکز بودن در واحد زمان‌بند) که مشکلات گلوگاه بودن و اتکاء به یک ماشین[۷۱] را رفع می‌کند، از دیگر مزایای روش زمان‌بندی پیوسته و پویای ارائه شده است.

از طرف دیگر، اخیراً مدل محاسباتی نگاشت- کاهش برای کاربردهای داده‌محور با پردازش‌های دسته‌ای ارائه شده است که به پردازش موازی با رویکرد افراز داده تحقق می‌بخشد. اگرچه نسخه‌های پایه و ابتدایی این مدل چالش‌های متعددی به منظور به کارگیری برای پردازش برخط پرس‌وجوهای پیوسته روی جریان‌های داده دارند، در بخش ۲-۲ این مقاله با بررسی این چالش‌ها و موانع، روشی مبتنی بر مدل نگاشت-کاهش برای پردازش موازی جریان داده‌ها ارائه شده است.

از جمله کارهایی که برای تکمیل و بهبود سیستم ارائه شده می‌تواند در ادامه صورت گیرد، می‌توان به موارد زیر اشاره نمود: کاهش سربار الگوریتم‌های ارائه شده، تخصیص بهینه منابع با استفاده از روش‌هایی مانند تئوری بازی‌ها، به کارگیری موتور پردازش موازی پرس‌وجوی ارائه شده برای سیستم‌های مدیریت

جریان داده بی‌درنگ، ارائه سیستم موازی به صورت توزیع شده و تحمل‌پذیری خطا در سیستم پیشنهادی.

مراجع

[1] B. Babcock, and et. al., "Models and Issues in Data Stream Systems," *Proc. twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 1-16, 2002.

[2] A. Arvind, and et. al., "STREAM: The Stanford Data Stream Management System," *Proc. SIGMOD*, pp. 665-665. ACM 2003.

[3] D. Abadi, and et. al., "Aurora: A New Model and Architecture for Data Stream Management," *VLDB Journal* vol. 12, no. 2, pp. 120-139, 2003.

[4] B. Babcock, and et. al., "Models and Issues in Data Stream Systems," *Proc. twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 1-16, 2002.

[5] L. Golab, and M. T. Ozsu, "Issues in Data Stream Management," *ACM Sigmod Record*, vol. 32, no. 2, pp. 5-14, 2003.

[6] N. Tatbul, and et. al., "Load Shedding in a Data Stream Manager." *Proc. VLDB*, pp. 309-320, 2003.

[7] C. Chekuri, and et. al.,"Scheduling problems in parallel query optimization," *Proc. fourteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, *ACM*, pp. 255-265, 1995.

[8] B. Babcock, and et. al.,"Operator Scheduling in Data Stream Systems," *VLDB Journal*, vol. 13, no. 4, pp. 333–353, 2004.

[9] D. J. DeWitt, and J. Gray, "Parallel Database Systems: The Future of High Performance Database Processing," *Communications of the ACM*, vol. 36, no. 6, pp. 85-98, 1992.

[10] W. Hong, *Parallel Query Processing Using Shared Memory Multiprocessors and Disk Arrays*, Ph. D. Dissertation, University of California, Berkeley, 1992.

[11] W. Hasan, and et. al., "Open issues in parallel query optimization," *ACM SIGMOD Record*, vol. 25 no. 3, pp. 28-33, 1996.

[12] G. Graefe, and et al., "Extensible Query Optimization and Parallel Execution in Volcano," *in Query Processing for Advanced Database Applications, J. C. Freytag, G. Vossen and D. Maier (ed.), Morgan-Kaufman,* San Mateo, CA, 1994.

[13] S. Babu, and J. Widom, "Continuous Queries over Data Streams," *ACM Sigmod Record,* vol. 30, no. 3, pp. 109-120, 2001.

[14] J. Chen, DJ. DeWitt and JF. Naughton, "Design and Evaluation of Alternative Selection Placement Strategies in Optimizing Continuous Queries," *Proc. 18th International Conference on Data Engineering,* pp. 345-356, 2002.

[15] A. Khalili, A. Jalaly Bidgoly, and M. Abdollahi Azgomi, "PDETool: A Multi-formalism Modeling Tool for Discrete-Event Systems Based on SDES Description," *Applications and Theory of Petri Nets. Springer Berlin Heidelberg,* pp. 343-352, 2009.

[16] A. Movaghar, *Performability Modeling with Stochastic Activity Networks*, Ph. D. Dissertation, University of Michigan, 1985.

[17] M. Abdollahi Azgomi, and A. Movaghar, "Coloured Stochastic Activity Networks: Definitions and Behavior," *Proc. 20th Annual UK Performance Engineering Workshop (UKPEW'04)*, pp. 297-308, 2004.

[18] D. Carney, and et. al., "Operator scheduling in a Data Stream Manager," *Proc. 29th international conference on Very large data bases,* pp. 838-849, 2003.

[19] 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, 2010.

[20] A. A. Safaei, and M. S. Haghjoo, "Dispatching of Stream Operators in Parallel Execution of Continuous Queries," *Journal of Supercomputing,* vol. 61, no. 3, 619-641, 2012.

[21] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. "Mapreduce online," *Proc. 7th USENIX conference on Networked systems design and implementation, NSDI'10,* pp. 21, 2010.

[22] J. Dean, and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Communications of the ACM,* vol. 51, no. 1, pp. 107-113, 2008.

[23] N. Backman, K. Pattabiraman, R. Fonseca, and U. Cetintemel, "CMR: Continuously Executing MapReduce Workflows on Multi-core Processors," *Proc. 3rd International Workshop on MapReduce and its Applications,* 2012.

[24] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "C-MR: Continuously Executing MapReduce Workflows on Multi-core Processors," *Proc. third international workshop on MapReduce and its Applications Date. ACM,* pp. 1-8, 2012.

[25] A. A. Safaei, A. Sharifrazavian, M. Sharifi, and M. S. Haghjoo, "Dynamic Routing of Data Stream Tuples among Parallel Query Plan Running on Multi-core Processors," *Journal of Distributed and Parallel Databases*, vol. 30, no. 2, pp. 145-176, 2012.

[26] G. Graefe, "Volcano-An Extensible and Parallel Query Evaluation System," *IEEE Transactions on Knowledge and Data Engineering,* vol. 6, no. 1, pp. 120-135, 1994.

[27] T. Condie, and et. al., "MapReduce Online," *Proc. 7th USENIX conference on Networked systems design and implementation*, pp. 21-21, 2010.

[28] Y. Zhou, and et. al., "Toward massive query optimization in large-scale distributed stream systems," *Middleware 2008, Springer Berlin Heidelberg,* pp. 326-345, 2008.

[29] Y. Zhou, and et. al., "An adaptable distributed query processing architecture," *Data and Knowledge Engineering,* vol. 53, no. 3, pp. 283–309, 2005.

[30] Y. Zhu, and et al., "Dynamic Plan Migration for Continuous Queries Over Data Streams," *Proc. ACM SIGMOD international conference on Management of data*, pp. 431-442, 2004.

[31] K. H. Lee, and et. al., "Parallel data processing with MapReduce: a survey," *ACM SIGMOD Record,* vol. 40, no. 4, pp. 11-20, 2012.

[32] W. Lam, and et. al., "Muppet: MapReduce-style processing of fast data," *Proc. VLDB Endowment,* vol. 5, no. 12, pp. 1814-1825, 2012.

[33] M. Ghalambor, and et. al., "DSMS scheduling regarding complex QoS metrics," *Proc. IEEE/ACS International Conference on Computer Systems and Applications (AICCSA),* pp. 10–13, 2009.

[34] T. Condie, and et. al., "Online Aggregation and Continuous Query support in MapReduce," *Proc. international conference on Management of data, ACM,* pp. 1115-1118, 2010.

[35] E. Mazur, and et. al., "Towards Scalable One-Pass Analytics Using MapReduce," *Proc. International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW),* pp. 1102-1111, 2011.

[36] Z. Brakerski, V. Dreizin, and B. Pattshamir "Dispatching in perfectly-periodic schedules," *Journal of Algorithms,* vol. 49, no. 2, pp. 219-239, 2003.

[37] M. A. Shah, and et. al., "Flux: An Adaptive Partitioning Operator for Continuous Query Systems," *Proc. 19th International Conference on Data Engineering (ICDE),* pp. 25-36, 2003.

[38] R. Avnur, and J. M. Hellerstein, "Eddies: Continuously Adaptive Query Processing," *ACM Sigmod Record,* vol. 29, no. 2, pp. 261-272, 2000.

[39] F. Tian, and D. J. DeWitt, "Tuple Routing Strategies for Distributed Eddies," *Proc. 29th international conference on Very large data bases-Volume (VLDB Endowment),* vol. 29, pp. 333-344, 2003.

[40] R. Nehme, and et. al., "Query Mesh: Multi-Route Query Processing Technology," *Proc. VLDB Endowment,* vol. 2, no. 2, pp. 1530-1533, 2009.

[41] R. Nehme, and et. al., "Self-tuning Query Mesh for Adaptive Multi-Route Query Processing," *Proc. 12th International Conference on Extending Database Technology: Advances in Database Technology, ACM*, pp. 803-814, 2009.

[42] C. Min, and D. Guo, "Operator Placement Techniques for Distributed Stream-Processing Systems," 2006.

[43] Z. Yongluan, B. C. Ooi, and K. L. Tan. "Dynamic load management for distributed continuous query systems," *Proc. 21st International Conference on Data Engineering (ICDE)*, pp. 322-323, 2005.

علی اصغر صفائی متولد سال ۱۳۵۷، کارشناسی و کارشناسی ارشد خود را در رشته مهندسی کامپیوتر- گرایش نرم‌افزار به ترتیب در سال‌های ۱۳۸۰ و ۱۳۸۳ به پایان رسانید. او در سال ۱۳۹۰ موفق به دریافت درجه دکتری در همین رشته و در حوزه‌ی پایگاه داده‌ها از دانشگاه علم و صنعت ایران گردید. وی هم‌اکنون به عنوان عضو هیئت علمی گروه مهندسی کامپیوتر، دانشکده مهندسی برق و کامپیوتر دانشگاه صنعتی خواجه نصیرالدین طوسی مشغول به فعالیت می‌باشد. زمینه‌های پژوهشی ایشان شامل سیستم‌های پایگاه داده و جریان داده، پردازش موازی و بی‌درنگ پرس‌وجوها، حافظه‌های نهان معنائی، امنیت در سیستم‌های پایگاه داده و مدیریت اطلاعات در مقیاس وب می‌باشد. از ایشان کتاب و مقالات متعدّدی نیز در همین زمینه‌ها به چاپ رسیده است.

آدرس پست‌الکترونیکی ایشان عبارت است از:

safaei@eetd.kntu.ac.ir

مصطفی سانیجی حق‌جو متولد سال ۱۳۳۴، دانش‌آموخته‌ی کارشناسی ریاضی از دانشگاه شیراز، کارشناسی ارشد و دانشوری علوم کامپیوتر از دانشگاه جورج واشنگتن آمریکا و دکتری کامپیوتر از دانشگاه ملی استرالیا به ترتیب در سال‌های ۱۳۵۶، ۱۳۵۸، ۱۳۵۹ و ۱۳۷۴ می‌باشند. ایشان علاوه بر همکاری در سازمان‌های نظیر سازمان سنجش آموزش کشور، و وزارت کار و امور اجتماعی در گذشته، از سال ۱۳۶۳ به عنوان عضو هیئت علمی دانشکده مهندسی کامپیوتر دانشگاه علم و صنعت ایران مشغول به فعالیت هستند و هدایت آزمایشگاه پایگاه داده‌ها و جریان داده‌ها را نیز برعهده دارند. از ایشان کتب و مقالات بسیاری در زمینه پایگاه داده‌ها به چاپ رسیده است. علاوه بر آن، تا کنون کتب متعدّدی در زمینه شعر نیز از ایشان منتشر شده است.

آدرس پست‌الکترونیکی ایشان عبارت است از:

haghjoom@iust.ac.ir

اطلاعات بررسی مقاله:

تاریخ ارسال: ۸۸/۱۰/۸

تاریخ اصلاح: ۹۱/۶/۱۳

تاریخ قبول شدن: ۹۱/۸/۱۷

نویسنده مرتبط: دکتر علی اصغر صفائی، دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی خواجه نصیرالدین طوسی، تهران، ایران.

[1] Data Base Management System (DBMS)
[2] Data Stream Management System (DSMS)
[3] Monitoring
[4] Load Shading
[5] Admission Control
[6] Performance
[7] Availability
[8] Extensibility
[9] Operator Path
[10] Data Partitioning یا Parallel Data
[11] Parallel Control
[12] Overload
[13] Logical
[14] Cluster
[15] Core
[16] Multi-Processing
[17] Parallel
[18] Query Mega Graph (QMG)
[19] Source (SRC)
[20] Sink
[21] Level
[22] Ldentifier
[23] Predecessor
[24] Successor
[25] Work Load
[26] Balanced
[27] Selection
[28] Projection
[29] Jjoin
[30] Snapshot
[31] Handle
[32] Load Shedding
[33] Load Balancing
[34] Load Balancing
[35] Event-Driven
[36] Stage
[37] Step
[38] Header
[39] Tuple-Based Continuous Scheduling
[40] Perti-Nets
[41] Selectivity
[42] Synthetic
[43] Response Time
[44] Memory Usage
[45] Throughput
[46] Tuple Loss
[47] Steady State
[48] Map Reduce
[49] Devide and Conqure
[50] Generic Computational Node
[51] Batch Processing
[52] Aggregation
[53] Combine
[54] Online Aggregation
[55] Stream Processing
[56] Pipelined Map Reduce
[57] Hadoop Online Prototype
[58] Abstract
[59] Pilpelined
[60] Incremental
[61] Identical
[62] Portion
[63] Data Stream Splitter
[64] Windowing
[65] Offset
[66] Identical
[67] Unbalanced Continous Map Reduce
[68] Multicore
[69] Buffering
[70] Prototype
[71] Single Source of Failure