Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: www.jscit.nit.ac.ir
Volume 7, Number 1, Spring & Summer 2018, pp. 42-55
Received: 2017/12/19; Revised: 2018/07/14; Accepted: 2018/11/03

A Scalable Traffic Engineering Method in an SDN-based Data Center Network Using Decomposition Technique

Mostafa Bastam[1*], Masoud Sabaei[2] and Rouhollah Yousefpour[3]
1- *Computer Engineering Department, University of Mazandaran, Babolsar, Iran.
2- Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran.
3- Mathematics Department, University of Mazandaran, Babolsar, Iran.
[1*]Bastam@aut.ac.ir, [2]Sabaei@aut.ac.ir, and [3]Yousefpour@umz.ac.ir

Corresponding author address: Mostafa Bastam, Faculty of Computer Engineering, University of Mazandaran, Babolsar, Iran.

Abstract- Today's data centers consist of thousands of servers hosting a variety of cloud-based services. In this paper, a scalable and new technique is presented for traffic engineering in software-defined data center networks helping to design an optimal demand-path mapping with a tolerable computational complexity. The proposed method is based on linear programming model and attempts to minimize the maximum link utilization that causes minimum link congestion. The method focuses on an optimal solution to load balancing in networks. It proposes a new decomposition technique that can limit the search space of the original Linear Programming problem, such that the time required to solve the problem can be reduced substantially. To reduce time complexity, a decomposition technique is used to divide the problem model into smaller sub-problems, which are then solved simultaneously by applying parallelizing techniques (multiple-core computing and OpenMP).Simulation results show that solving time and load balancing are considerably improved.

Keywords- Data center networks, Software-defined networks, Traffic engineering, Congestion control, Linear programming decomposition, Parallel solution.

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: www.jscit.nit.ac.ir
جلد ۷، شماره ۱، فصل بهار و تابستان ۱۳۹۷، صفحه ۴۲-۵۵
دریافت: ۱۳۹۶/۰۹/۲۸، بازنگری: ۱۳۹۷/۰۴/۲۳، پذیرش: ۱۳۹۷/۰۸/۱۲

ارایه یک روش مهندسی ترافیک مقیاس پذیر در شبکه‌های نرم‌افزار محور مراکز داده با استفاده از تکنیک تجزیه مسائل بزرگ

مصطفی بستام*[۱]، مسعود صبائی[۲]، روح ا.. یوسف پور[۳]
۱*- دانشکده فنی و مهندسی، گروه مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه مازندران، بابلسر، ایران.
۲- دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران.
۳- دانشکده ریاضی، دانشگاه مازندران، بابلسر، ایران.
[1*] Bastam@umz.ac.ir, [2]Sabaei@aut.ac.ir, and [3]Yousefpour@umz.ac.ir

* نشانی نویسنده مسئول: مصطفی بستام، دانشگاه مازندران، دانشکده فنی و مهندسی، گروه مهندسی کامپیوتر و فناوری اطلاعات.

چکیده- مراکز داده امروزی، از هزاران سرویس‌دهنده تشکیل شده‌اند که هر یک از آنها از سرویس‌های متنوع مبتنی بر ابر میزبانی می‌نماید. در این مقاله، یک روش جدید و مقیاس‌پذیر مهندسی ترافیک در شبکه‌های نرم‌افزار محور مراکز داده، با هدف تخصیص بهینه درخواست‌ها به مسیرها، با پیچیدگی محاسباتی قابل قبول ارائه شده است. روش ارائه شده مبتنی بر برنامه‌ریزی خطی است و تلاش می‌کند حداکثر میزان بار ترافیکی بر روی لینک‌ها حداقل شود. حاصل این عمل کاهش ازدحام بر روی لینک‌های شبکه خواهد بود. این روش، بر روی ارائه یک راه‌حل بهینه به منظور موازنه بار ترافیکی در شبکه متمرکز شده است و یک روش جدید تجزیه به منظور محدود نمودن فضای جستجوی مساله برنامه‌ریزی خطی پیشنهاد می‌نماید. روش تجزیه به گونه‌ای است که زمان حل مساله به میزان قابل توجهی کاهش یابد. به منظور کاهش پیچیدگی زمانی، یک روش تجزیه استفاده شده است که مدل مساله را به زیرمساله‌های مجزا تقسیم می‌کند. با استفاده از روش‌های موازی‌سازی (محاسبات بر روی چندین هسته محاسباتی و OpenMP) می‌توان این زیرمساله‌ها را به صورت همزمان حل نمود. نتایج شبیه‌سازی نشان دادند که در روش پیشنهادی، زمان حل و موازنه بار ترافیکی هر دو به میزان چشمگیری بهبود یافته‌اند.

واژه‌های کلیدی: شبکه‌های مرکز داده، شبکه‌های نرم‌افزار محور، مهندسی ترافیک، کنترل ازدحام، تجزیه برنامه‌ریزی خطی، حل موازی.

۱- مقدمه

شبکه‌های نرم‌افزار محور (SDN) یک معماری نوین در شبکه است که در آن انتقال ترافیک، از کنترل شبکه مجزا شده است. این مجزا سازی، برنامه‌ریزی مستقیم شبکه را راحت‌تر و سریع‌تر ساخته است. همه تصمیمات در کنترل‌کننده مرکزی صورت می‌پذیرد و دستورات لازم بوسیله آن به تجهیزات شبکه تزریق می‌شود. در نتیجه کنترل‌کننده مرکزی شبکه، دارای یک دید متمرکز و جامع از وضعیت شبکه است. منظور از دید جامع از شبکه آن است که در هر لحظه وضعیت توپولوژی و ترافیک کل شبکه در دسترس باشد. ویژگی‌ها و امکانات شبکه‌های نرم‌افزار محور، مدیریت متمرکز را به دنبال دارد که این امکان اخذ تصمیم‌های بهینه را مقدور می‌سازد. از آنجایی که ما در این پژوهش به دنبال بهبود مکانیزم‌های مهندسی ترافیک در شبکه‌های مراکز داده هستیم، چنین تکنولوژی که دید

جامع در اختیار ما قرار دهد، از اساسی‌ترین نیازمندی‌ها به شمار می‌رود [۱, ۲].

تکنیک‌های مهندسی ترافیک کنونی برای شبکه‌های نسل بعدی به دو دلیل عمده مطلوب نیستند. اول، برنامه‌های کاربردی اینترنتی بر روی مراکز داده نیازمند تکنیک‌هایی هستند که بتواند به صورت بلادرنگ به وقایع واکنش نشان دهد و برای ترافیک‌های حجیم، مقیاس‌پذیر باشد. دوم، با توجه به رشد سریع محاسبات ابری و نیاز به مراکز داده با مقیاس بزرگ، یک مدیریت مناسب شبکه باید قادر به بهبود استفاده موثر از منابع، در جهت ارتقا کارایی سیستم نیز باشد. بنابراین، تکنیک‌های مهندسی ترافیک جدیدی مورد نیاز است که نقاط ضعف ذکر شده را مرتفع نماید. درSDN، کنترل‌کننده مرکزی به صورت سراسری وضعیت شبکه را براساس سیاست‌های شبکه تنظیم می‌نماید و به سبب دسترسی کامل به کلیه اجزای شبکه و منابع، سیاست‌های شبکه قادرند با توجه به رفتار ترافیک جاری واکنش مناسب داشته باشند.

پس انتظاری که هم‌اکنون از الگوریتم‌های مسیریابی و روش‌های مهندسی ترافیک داریم این است که به منظور نگاشت بهینه درخواست‌ها به مسیرها، تصمیم‌های بهینه اتخاذ نمایند ولی متاسفانه این مهم هنوز مقدور نگشته است. دلیل این نقیصه آن است که با وجود اطلاع جامع و پویا از وضعیت شبکه، بدلیل آن که فضای جستجوی مساله تصمیم‌گیری (تعداد لینک‌ها و تعداد درخواست‌ها) در شبکه‌های مرکز داده امروزی بسیار بزرگ است، یافتن جواب و راه حل بهینه در زمان کم و عملیاتی مقدور نیست. مقصود از فضای جستجوی مساله تصمیم‌گیری بهینه، همان یافتن بهترین مسیرها برای درخواست‌های وارد شده به شبکه است. لذا اکثر روش‌های ارائه شده از الگوریتم‌های مکاشفه‌ای بهره برده‌اند تا معدود روش‌هایی که سعی کرده- به راه‌حل‌های بهینه نزدیک شوند. اند به جواب بهینه با پیچیدگی زمانی اندک برسند تکنیک تجزیه را بکار برده‌اند ولی زمان حل مساله باز هم عملیاتی نیست.

در این مقاله قصد داریم روشی پویا به منظور مهندسی ترافیک در شبکه‌های مرکز داده نرم‌افزار محور ارائه دهیم که علاوه بر یافتن جواب بهینه مساله تخصیص درخواست‌ها به مسیرها، پیچیدگی محاسباتی آن نیز قابل قبول و عملیاتی باشد. در نتیجه نیاز است با توجه به دید جامع و بروز شده‌ای که در بازه‌های زمانی کوچک بوسیله SDN در اختیار کنترل‌کننده مرکزی قرار می‌گیرد، تخصیص بهینه درخواست‌ها به مسیرها به گونه‌ای که اهداف مهندسی ترافیک تامین شود، صورت پذیرد. در اینجا منظور از اهداف مهندسی ترافیک حداقل کردن ازدحام در شبکه می‌باشد که به دلیل کاهش ریزش بسته‌ها، افزایش گذردهی[۱] را نیز در پی خواهد داشت. برای این مهم از برنامه‌ریزی خطی به منظور بیان فرمال مساله بهره می‌بریم. مساله را به عنوان MCF[۲] مبتنی بر مسیر مدل می‌کنیم. از آنجایی که زمان حل این مدل برای شبکه‌های مرکز داده امروزی بسیار زمانبر و غیرعملیاتی است، با استفاده از تکنیک تجزیه مساله را به زیرمساله-های کوچک‌تر و مستقل از یکدیگر تقسیم می‌نماییم. با حل این زیرمساله‌ها به صورت موازی، امیدواریم زمان یافتن جواب مساله کاهش چشم‌گیری داشته باشد. این زیرمساله‌ها، با استفاده از تکنیک‌های حل موازی (چندین هسته محاسباتی و OpenMP) به صورت همزمان حل خواهند شد و بعد از آن، جواب‌های زیرمساله‌ها تجمیع خواهند شد.

سازمان‌دهی مقاله بدین قرار است که: در قسمت دوم، تکنولوژی و تعاریف به کار رفته در این مقاله به طور مختصر بیان خواهد شد. در قسمت سوم به بررسی پیشینه نظری و کارهای مرتبط خواهیم پرداخت. قسمت چهارم، جزئیات روش پیشنهادی را در بر می‌گیرد و در قسمت انتهایی نتیجه‌گیری را خواهیم داشت.

۲- تکنولوژی و تعاریف بکار رفته

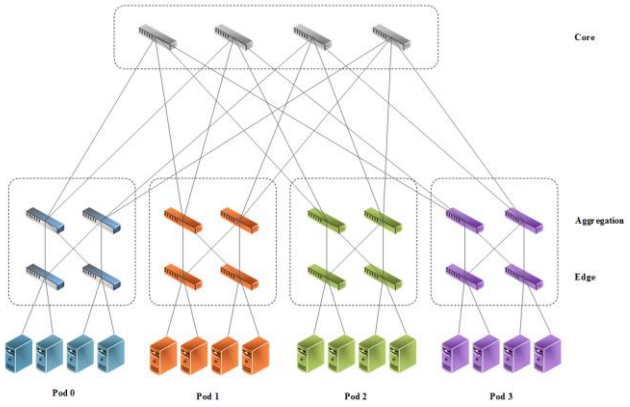
۲-۱- معماری Fat-Tree

امروزه مراکز داده شامل هزاران سرور با نیازمندی پهنای باند تجمعی قابل توجه هستند. معماری شبکه، معمولا از یک سلسله مراتب مبتنی بر درخت تبعیت می‌نماید. هر چه به سمت ریشه حرکت نماییم تجهیزات گران‌تر و خاص‌تر خواهند شد[۳]. Fat-Tree یکی از معروفترین توپولوژی‌های شبکه مراکز داده امروزی در مقیاس هزاران سرور است. این معماری از سوئیچ‌های اترنت معمولی و مرسوم تشکیل شده است [۴]. پارامتر مهم این معماری مقدار k است. همه سوئیچ‌های به کار رفته در Fat-Tree دارای k پورت هستند. این معماری سوئیچ‌ها را در سه لایه تقسیم می‌نماید: لایه هسته[۳]، لایه تجمیع[۴] و لایه لبه[۵]. علاوه بر این، در Fat-Tree سوئیچ‌ها به k قسمت مساوی بنام Pod تقسیم می‌شوند. در هر Pod، تعداد $\frac{k}{2}$

سوئیچ در لایه لبه و $\frac{k}{2}$ سوئیچ در لایه تجمیع قرار می‌گیرد. در سوئیچ‌های لبه، $\frac{k}{2}$ از پورت‌ها به میزبان‌ها و $\frac{k}{2}$ پورت دیگر به سوئیچ-های لایه تجمیع در همان Pod متصل شده‌اند. در سوئیچ‌های لایه تجمیع $\frac{k}{2}$ پورت باقیمانده به سوئیچ‌های لایه هسته وصل می‌شوند. شکل۱، یک مدل از توپولوژی Fat-Tree با مقدار $k = 4$ را نشان می‌دهد. تعداد میزبان‌ها در هر Pod، $\left(\frac{k}{2}\right)^2$ می‌باشد. تعداد سوئیچ‌های لایه هسته $\left(\frac{k}{2}\right)^2$ عدد است، که iامین پورت هر یک از آن‌ها به Pod i وصل می‌شود، تعداد کل میزبان‌ها برابر با $\frac{k^3}{4}$، تعداد کوتاه‌ترین مسیرهای مساوی بین میزبان‌های داخل یک Pod، $\frac{k}{2}$ و تعداد کوتاهترین مسیر مساوی بین دو میزبان در دو Pod مجزا، $\left(\frac{k}{2}\right)^2$ است.

۲-۲- بررسی رفتار ترافیک در شبکه مرکز داده

مکانیزم‌های مهندسی ترافیک به میزان قابل توجهی به ماهیت رفتار ترافیک وابسته هستند. لذا از آنجا که قصد داریم بر روی این مکانیزم‌ها در شبکه‌های مرکز داده متمرکز شویم، نیاز است که رفتار ترافیک و ویژگی‌های ترافیک را در مراکز داده بررسی نماییم. ترافیک در مراکز داده عمدتا به دو نوع جریان فیل[۶] و موش[۷] تقسیم می‌شود. مقصود از جریان، بسته‌های داده‌ای است که شبکه با آنها رفتار یکسانی دارد (مثلا بسته‌های مربوط به یک کاربرد خاص، یا بسته‌های بین یک مبدا و یا مقصد مشخص). جریان‌های فیل بوسیله برنامه‌هایی همچون پشتیبان‌گیری از داده‌ها و مهاجرت ماشین‌های مجازی تولید می‌شود، در نتیجه نیازمند پهنای باند زیادی هستند (تقریبا ۸۰ درصد حجم ترافیک مراکز داده را حمل می‌کنند) ولی در عوض مدت زمان تکمیل این درخواست‌ها خیلی سختگیرانه نیست. طول عمر این نوع جریان‌ها نسبتا طولانی است. در طرف مقابل جریان‌های موش با طول عمر کوتاه که بوسیله برنامه هایی همچون جستجوی صفحات وب تولید می‌شوند قرار گرفته است که حجم آنها کم ولی در عوض به تاخیر بسیار حساس هستند. نیازمندی‌های تاخیر این برنامه‌ها تاثیر قابل توجهی در کیفیت دریافتی کاربران دارد. در نتیجه جریان‌های موش باید با در اولویت قرار گرفتن نسبت به جریان‌های فیل گرانتی شوند [۵]. Kandula و همکاران [۹] اطلاعات مرتبط با ۱۵۰۰ سرور موجود در یک مرکز داده را به مدت ۲ ماه جمع‌آوری کردند. یافته‌های آنها بدین قرار بود، که ۸۰ درصد جریان ها کمتر از ۱۰ ثانیه طول می‌کشند. مدت زمان کمتر از ۰,۱ درصد از جریان ها بیش از ۲۰۰ ثانیه است و بیش از ۵۰ درصد بایت‌ها در جریان‌های با طول عمر کمتر از ۲۵ ثانیه قرار دارند. آنها همچنین مشاهده



شکل۱- نمایی از یک معماری Fat-Tree با $k = 4$

کردند که حجم ترافیک خیلی سریع تغییر می‌کند و چندین بار در در طول روز همه لینک‌های شبکه به اندازه‌ای نزدیک به حداکثر ظرفیتشان استفاده می‌شوند. Faringtion و همکارانش [۱۰]، اقدام به بررسی معماری شبکه مرکز داده فیسبوک نموده‌اند. آنها دریافتند که به طور مثال به منظور ارائه پاسخ به یک درخواست HTTP ، به اندازه ۹۳۰ برابر درخواست، ترافیک در شبکه مرکز داده تولید شده است. در نتیجه در مراکز داده بیشتر تمرکز بر روی میزان ترافیک داخلی است تا ترافیک ارتباطی با دنیای بیرون.

۲-۳- مهندسی ترافیک

تکنیک‌های مهندسی ترافیک با ارزیابی و آنالیز پویای توپولوژی و ترافیک شبکه، پیش‌بینی، کنترل و دستکاری رفتار ترافیک، قصد بهینه‌سازی کارایی شبکه‌های ارتباطی را دارند [۱۱]. مدیران شبکه می‌توانند مساله مهندسی ترافیک را براساس نیازشان مبتنی بر یک یا مجموعه ای از معیارهای ارزیابی، بهینه نمایند. در عمل، بهینه‌سازی کیفیت سرویس بر اساس کلیه معیارها ارزیابی با همدیگر به طور همزمان غیرممکن است، چرا که بهبود در یکی می‌تواند تاثیر منفی بر روی معیارهای دیگر داشته باشد. بهینه‌سازی تک‌هدفه، بهترین جواب و راه‌حل را بر اساس یک معیار بدست می-آورد در حالیکه در بهینه‌سازی چند هدفه، بهترین راه حل بر اساس

مصالحه بین چند معیار ارزیابی حاصل خواهد شد [۱۲]. میزان استفاده موثر از لینک، نسبت بار ترافیکی[۸] لینک به ظرفیت در دسترس آن لینک، تعریف شده است. مجموع ترافیک جریان‌هایی که از روی یک لینک عبور می‌کنند، بار ترافیکی آن لینک را مشخص می‌نمایند. مهمترین هدف در اکثر مسائل مهندسی ترافیک، اجتناب و یا حداقل کردن ازدحام است. این مهم معمولا با حداقل کردن حداکثر میزان استفاده موثر از لینک[۹]، ارزیابی می‌شود. در نتیجه در این مقاله ما نیز بر این هدف تمرکز نموده‌ایم.

۳- کارهای مرتبط

با مطالعه نیازمندی‌ها و ویژگی‌های مرکز داده و همچنین شناخت رفتار ترافیک در آن‌ها، و ویژگی‌های منحصر بفردی که شبکه‌های نرم‌افزار محور فراهم نموده است، در این قسمت قصد داریم برخی از روش‌ها و مکانیزم‌های مهندسی ترافیک پر استناد در مراکز داده را بررسی نماییم.

Hedera یک سیستم مسیریابی جریان به صورت پویاست که به منظور هدایت جریان‌های فیل از روش متمرکز استفاده می‌نماید [۱۳]. هر جریانی که بیش از ۱۰ درصد پهنای باند کارت شبکه را اشغال نماید به عنوان جریان فیل مشخص می‌شود. به منظور ارسال مابقی جریان‌ها یعنی جریان‌هایی با طول عمر کوتاه (جریان‌های موش) از روش ایستای ECMP [۱۴] استفاده می‌شود. Hedera قصد دارد با قرار دادن جریان‌ها با بیشترین ترافیک در مسیرهای جایگزین مناسب، موازنه بار را در شبکه انجام دهد. در واقع سعی می‌نماید جریان‌های حجیم را به گونه‌ای در شبکه توزیع کند که با یکدیگر بر روی لینک‌های شبکه تلاقی نداشته باشند. این روش را می‌توان جزو نخستین روش‌های ارائه شده در زمینه مورد پژوهش به شمار آورد. که به مرور زمان روش‌های دیگر برای بهبود آن ارائه شده‌اند و غالبا به عنوان مدل پایه به منظور قیاس بکار رفته است. عمده‌ترین محدودیت این روش آن است که جریان‌های موش که تشکیل دهنده ۸۰ درصد جریان‌های ترافیکی مرکز داده هستند را با روش ECMP ایستا مسیریابی می‌نماید. یعنی این جریان‌ها را برحسب آدرس هدر به یکی از مسیرها هدایت می‌نماید و وضعیت جاری شبکه در آن‌ها لحاظ نمی‌گردد، در عوض مکانیزم مهندسی ترافیک را تنها برای ۲۰ درصد جریان‌ها (جریان‌های فیل) اعمال می‌نماید. معمولا جریان-های موش نیازمندی‌های کیفیت سرویس حساس‌تری نیز دارند.

پروتکل Mahout [۱۵] برای تشخیص جریان‌های بزرگ (فیل) از یک برنامه مقیم در سرورها (یک لایه بر روی سیستم عامل) بهره می‌برد. با مانیتور کردن بافر سوکت‌ها، وجود جریان فیل را تشخیص می‌دهد. اگر میزان بسته‌های موجود در بافر از یک حد آستانه تجاوز کنند به عنوان جریان فیل شناخته خواهد شد. سپس بسته‌های این جریان را نشانه‌گذاری می‌نماید. (مثلا با استفاده از فیلدDSCP هدر لایه IP). این بسته‌های علامت‌دار بوسیله اولین سوئیچ به کنترل‌کننده مرکزی فرستاده می‌شود. با این کار دیگر تشخیص جریان، سرباری بر روی سوئیچ‌ها نخواهد داشت. در کنترل‌کننده مرکزی با استفاده از یک الگوریتم مسیریابی مکاشفه‌ای یک مسیر با کمترین ازدحام برای جریان مزبور تخصیص داده می-شود. به منظور ارسال جریان‌های موش از مکانیزم ECMP استفاده می‌شود. مهمترین عیب این روش آن است که باید در سرورها تغییراتی ایجاد نمود و مسیر محاسبه شده برای جریان‌های فیل، مسیرهایی نزدیک به بهینه بدست خواهند آورد نه بهینه.

MicroTE یک روش متمرکز است که با پیش‌بینی ترافیک در بازه‌های زمانی خیلی کوتاه خود را با تغییرات ترافیک داخل مرکز داده سازگار کرده است [۱۶]. بنابراین، به صورت دائم تغییرات ترافیک را مانیتور می‌کند تا تشخیص دهد که کدام جفت از سوئیچ های موجود در رک‌ها دارای ترافیک قابل بیش‌بینی است و سپس این میزان ترافیک بیش‌بینی شده را به مسیرهای بهینه هدایت نماید تا حداکثر میزان استفاده موثر از لینک‌ها را حداقل نماید. مابقی ترافیک پیش‌بینی نشده با استفاده از روش ECMP وزن‌دار، در شبکه جاری خواهند شد. وزن‌ها در واقع منعکس کننده میزان ظرفیت در دسترس مسیرها پس از تخصیص ترافیک پیش‌بینی شده است. مهمترین ایرادات microTE نیاز به تغییر در میزبان‌ها به منظور جمع‌آوری اطلاعات ترافیک در سطح ریزدانه است. مسیر بهینه برای همه جریان‌ها محاسبه نمی‌شود و مسیرهای محاسبه شده نزدیک به بهینه هستند. برای محاسبه آن‌ها، از روش-های مکاشفه‌ای استفاده شده است.

Dynamic Load Balancing (DLB) یک الگوریتم متمرکز است که جریان‌ها را بر روی مسیرهای جایگزین به‌گونه‌ای توزیع می‌کند که میزان ترافیک دریافت شده در هریک از مسیرهای جایگزین یکسان باشد. این روش برای مراکز داده با توپولوژی Fat-Tree و مبتنی بر

پروتکل OpenFlow طراحی شده است. الگوریتم از ویژگی سلسله مراتبی Fat-Tree به منظور جستجو به صورت بازگشتی برای یافتن مسیرهای موجود بین یک مبدا و مقصد مشخص بهره برده است. سپس مبتنی بر اطلاعات آماری ترافیک که به صورت بلادرنگ هر ۵ ثانیه بوسیله OpenFlow از سوئیچ‌ها بدست می‌آورد، تصمیم لازم به منظور انتخاب مسیر مناسب را اتخاذ می‌نماید. در این روش، انتخاب مسیر با توجه به شرایط محلی صورت می‌پذیرد در نتیجه تخصیص مسیر یک راه‌حل بهینه محلی است. در نهایت و مهم‌تر از همه، DLB برای تعداد زیاد جریان مشکل مقیاس‌پذیری و پاسخگویی دارد. بنابراین این الگوریتم تنها زمانی که جریان جدید ظاهر می‌شود باید در کنترل‌کننده مرکزی فراخوانی شود [۱۷].

در [۱۸] مساله با MCF مبتنی بر لینک مدل شده است. پس از تجزیه مساله به زیرقسمت‌های مجزا و حل آنها به صورت موازی، مدت زمان اجرا را به میزان قابل توجهی کاهش داده است. از آنجایی که در این روش، اندازه زیرقسمت‌ها مساوی نیستند و اندازه یکی از زیر قسمت‌ها (از لحاظ لینک و گره) از بقیه زیر قسمت‌ها بزرگتر است، مدت زمان اجرا در هر تکرار وابسته به زمان حل این زیرقسمت می‌باشد. دیگر آن‌که مدل LMCF[۱۰] به طور ذاتی فضای جستجوی بزرگی دارد. نویسندگان اقدام به مقایسه مدت زمان بدست آوردن جواب بهینه در دو روش اولیه (بدون تجزیه) و توزیع شده نموده‌اند. مدت زمان اجرای الگوریتم اولیه، با فرض تعداد یک جریان در هر میزبان برای مقدار $k = 16$ بیش از ۴۵۰۰ ثانیه می‌شود. در واقع با افزایش سایز شبکه (متغیرها و محدودیت‌ها) زمان حل مساله افزایش چشمگیری دارد و می‌توان گفت که الگوریتم اولیه مقیاس‌پذیر نیست. مدت زمان اجرای مدل توزیع شده الگوریتم برای شرایط مطرح شده در بالا، در حدود چند دقیقه می‌باشد. درست است که این زمان کاهش قابل توجهی یافته است ولی هنوز هم برای استفاده در مراکز داده امروزی زیاد است.

در [۱۹] مساله نگاشت بهینه درخواست‌ها به لینک‌ها، به صورت برنامه‌ریزی عدد صحیح (ILP) مدل شده است. از آنجایی‌که تجزیه به صورت پی در پی صورت می‌پذیرد، در نتیجه امکان موازی سازی نیست. اگر مدل مساله به گونه‌ای مطرح می‌شد که قابلیت تقسیم به زیرمساله‌های مجزا فراهم می‌شد، با حل این زیرمساله‌ها به صورت موازی و سپس تجمیع جواب آنها، زمان حل مساله بسیار کاهش پیدا می‌کرد.

به طور کلی می‌توان روش‌های مورد بررسی قرار گرفته شده در این قسمت را به دو گروه تقسیم نمود. یک، الگوریتم‌هایی که از روش‌هایی مکاشفه‌ای برای تخصیص مسیر مناسب به درخواست‌ها استفاده می‌کنند. دو، روش‌هایی که سعی می‌کنند که برای مساله تخصیص منابع به درخواست‌ها از مکانیزم‌های بهینه‌سازی بهره ببرند. علت آنکه روش‌های دسته اول به دنبال جواب نزدیک به بهینه هستند آن است که به دلیل بزرگ بودن فضای جستجوی مساله تصمیم‌گیری در مراکز داده امروزی با مقیاس هزاران سرور، بدست آوردن جواب در زمان معقول قابل حصول نیست، در نتیجه سعی می‌نمایند از روش‌های مکاشفه‌ای بهره ببرند و باز هم برای کاهش فضای تصمیم‌گیری معمولا این کار را بر روی برخی از جریان‌ها (معمولا جریان‌های فیل) اعمال می‌نمایند. مهمترین ویژگی روش‌های گروه اول، زمان محاسباتی معقول به قیمت فاصله گرفتن از جواب بهینه است. در سوی دیگر روش‌های گروه دو، سعی دارند با مدل کردن مساله به صورت برنامه‌ریزی خطی، و بهره‌گیری از تکنیک‌های تجزیه مسائل بزرگ به زیر قسمت‌های کوچک‌تر بر مشکل پیچیدگی زمانی فائق آیند. روش‌های ذکر شده به طور مختصر در جدول ۱ نشان داده شده است. همچنین ویژگی‌های روش پیشنهادی این مقاله نیز در آن آورده شده است. برخی از مهم‌ترین مطالبی که می‌توان از این جدول استنباط نمود به قرار ذیل است:

هدف اکثر روش‌ها، حداقل کردن ازدحام در شبکه است. اکثر این روش‌ها از کنترل‌کننده مرکزی استفاده می‌نمایند که این یک روش متمرکز است. روش‌های مکاشفه‌ای در زمان معقول حل می‌شوند ولی از جواب بهینه فاصله دارند، در عوض روش‌های [۱۸, ۱۹] که دارای جواب بهینه سراسری هستند، قادر به ارائه راه‌حل در زمان معقول نیستند و این به دلیل بزرگ بودن فضای جستجوی مساله تصمیم‌گیری است. روش‌های مهندسی ترافیک نیازمند داشتن یک تخمین خوب از وضعیت ترافیکی درخواست‌ها هستند. روش‌های مهندسی ترافیک باید بتوانند با ریزدانگی در مقیاس جریان راه‌حل ارائه نمایند. اکثر این روش‌ها فرض نموده‌اند که معماری شبکه مرکز داده Fat-Tree است.

با توجه به مطالب ذکر شده، روش پیشنهادی مهندسی ترافیک به گونه‌ای ارائه خواهد شد که بتواند در یک شبکه نرم افزار محور مرکز داده با معماری شبکه Fat-Tree، بدون نیاز به تغییر در میزبان‌ها، با ریزدانگی در مقیاس جریان، درخواست‌ها را به صورت بهینه به مسیرها با پیچیدگی زمانی عملیاتی هدایت نماید به گونه‌ای که ازدحام در شبکه حداقل شود.

۴- روش پیشنهادی

با توجه به مطالب ذکر شده تاکنون این نکته حائز اهمیت است که استفاده از شبکه‌های نرم‌افزار محور امکان بهره‌مندی از دید جامع را برای روش‌های مهندسی ترافیک در شبکه‌های مراکز داده فراهم می‌سازد، تا علاوه بر تامین نیازمندی‌های برنامه‌های متنوعی که بر روی این مراکز داده در حال اجرا هستند، میزان بهره‌وری شبکه نیز افزایش یابد. انتظار می‌رود دید جامع منجر به تصمیم‌های بهینه گردد، به طوری‌که تخصیص جریان‌ها به مسیرها به بهترین نحو ممکن صورت پذیرد. این عمل تخصیص، باید در بازه زمانی بسیار کوچک متناسب با تغییرات ترافیک در شبکه‌های مرکز داده صورت پذیرد. در واقع برای مجموعه درخواست‌هایی که در هر دوره تصمیم‌گیری به سیستم وارد شده است، الگوریتم باید اجرا شود.

روش‌ها و مطالعاتی که در قسمت های قبل به آنها اشاره شد سعی کرده‌اند که به این خواسته دست یابند ولی عمدتا به یکی از علت‌های زیر موفقیت حداکثری بدست نیاورده‌اند: ۱)محاسبه مسیر بهینه برای برخی از جریان‌ها نه همه جریان‌ها (چرا که در صورت لحاظ کردن همه جریان‌ها، فضای مساله بسیار بزرگ می‌شود)، ۲) انتخاب مسیر به صورت ثابت(ECMP)، ۳) به علت بزرگ بودن فضای مساله، امکان حل مسائل بهینه‌سازی در زمان معقول دور از دسترس بوده است، ۴) عدم استفاده از تکنیک‌های تجزیه مسائل بزرگ بهینه‌سازی و حل موازی آنها به صورت کارا به صورتی که، امکان تقسیم به زیرقسمت-های مجزا به صورت مقیاس پذیر فراهم باشد.

در این مقاله قصد داریم برای موارد ذکر شده در بالا راه‌حل ارائه نماییم. ابتدا، مساله به صورت فرمال مدل می‌کنیم. سپس، مدل به دست آمده را با توجه به توپولوژی شبکه تجزیه می‌نماییم، حال با استفاده از تکنیک‌های حل موازی، زیر مساله‌های حاصل شده از تجزیه مدل به صورت موازی حل خواهد شد.

جدول ۱- مقایسه روش‌های مهندسی ترافیک در شبکه مرکز داده

روش پیشنهادی	DLB	microTE & Mahout	Hedera	[۱۹]	[۱۸]	
Min MLU	Balancing Load	Min MLU	Max BisectionBW	-	Min T_f	**هدف مساله**
متمرکز	متمرکز	متمرکز	متمرکز	متمرکز	متمرکز	**روش**
Per-flow	Per-flow	Per-predicted flow	Per-elephant flow	Per-flow	Per-flow	**ریزدانگی**
بله	خیر	خیر	خیر	بله	بله	**بهینه‌سازی سراسری**
عملیاتی	عملیاتی	عملیاتی	عملیاتی	زیاد	کم ولی غیرعملیاتی	**پیچیدگی زمانی**
لازم	-	لازم	لازم	لازم	لازم	**تخمین ماتریس ترافیکی**
خیر	خیر	بله	خیر	خیر	خیر	**تغییر در میزبان**
Fat-tree	Fat-tree	Typical Tree	Fat-tree	XGFT	Fat-tree	**توپولوژی شبکه**

۴-۱- مدل ریاضی مساله

برنامه‌ریزی خطی به منظور حل مساله تخصیص منابع در مراکز داده در برخی از مقالات استفاده شده است که تابع هدف آنها عمدتا حداقل نمودن حداکثر میزان استفاده از لینک‌ها در شبکه است. در واقع با این کار قصد کاهش ازدحام بر روی لینک‌ها را دارند. تکنیک‌های جریان در شبکه [۲۰] در جهت مهندسی ترافیک در مراکز داده کاربرد دارند. اغلب از آنجایی‌که حل این‌گونه مسائل با توجه به مقیاس شبکه از لحاظ زمانی، عملیاتی نیست معمولا به عنوان روشی برای یافتن یک حد بالا یا حد پایین برای ارزیابی الگوریتم‌های مکاشفه‌ای مهندسی ترافیک در شبکه بکار می روند.

۴-۲- مساله جریان در شبکه

MCF یک مساله جریان در شبکه است که به ارسال جریان‌های ترافیکی بین چندین مبدا و مقصد می‌پردازد. مدل مساله جریان در شبکه مبتنی بر لینک (LMCF) با هدف حداقل کردن، حداکثر استفاده موثر از لینک‌ها در شبکه به قرار زیر است:

$$min \quad m$$

s.t.

$$\sum_{i=1}^{k} f_i(u,v) \le m \times c(u,v)\,, \quad \forall (u,v) \epsilon E \qquad (۱)$$

$$\sum_{(u,v)\in E} f_i(u,v) = \sum_{(v.w)\in E} f_i(v,w)\,, \quad \forall i, \forall v \in V - \{s_i.\, t_i\} \qquad (۲)$$

$$\sum_{w\in V} f_i(s_i,w) = \sum_{w\in V} f_i(w,t_i) = d_i \ , \forall i \qquad (۳)$$

ورودی‌ها: ۱- توپولوژی شبکه : G(V, E) یک گراف با V گره و E یال. ۲- ماتریس ترافیک: T، در واقع در این ماتریس مشخص می‌شود که کدام مبدا به کدام مقصد چه میزان ترافیک ارسال خواهد کرد. ۳-ظرفیت لینک‌ها: $c(u,v)$ به ازای هر لینک (u,v) متعلق به E.

متغیرهای تصمیم‌گیری: ۱- m: Maximum Link Utilization، ۲- نرخ جریان موجود بر روی هر لینک$f_i(u,v)$ به ازای هر درخواست i و هر لینک (u,v).

محدودیت‌ها: ۱- محدودیت ظرفیت (محدودیت ۱)، نرخ جریانی که از یک لینک می‌گذرد همواره کمتر و یا مساوی ظرفیت آن لینک است. ۲- حفاظت از جریان (محدودیت ۲)، به ازای هر گره‌ای که مبدا و یا مقصد جریان نباشد، نرخ جریان ورودی یک درخواست برابر با نرخ جریان خروجی آن درخواست است. ۳-رضایتمندی تقاضا (محدودیت ۳)، نرخ جریانی که یک گره مبدا به ازای هر درخواست تولید می‌کند مساوی است با نرخ جریانی که در گره مقصد دریافت می‌شود که این میزان برابر است با میزان تقاضای جریان (d_i).

ثابت شده است که اگر همین مساله را در حالت عدد صحیح در نظر بگیریم *np-complete* خواهد بود [۲۱, ۲۲]. در صورتی که اگر شکستن جریان‌ها مجاز باشد با استفاده از برنامه‌ریزی خطی در زمان چندجمله‌ای مساله حل می‌شود. البته باز هم اگر سایز مساله و فضای جستجوی آن (شامل محدودیت‌ها و متغیرها) بزرگ باشد این مدت زمان حل در عمل کاربردی نیست.

به دلیل کارایی که عمدتا به رفتار TCP در شبکه مرتبط می‌شود، باید از شکسته شدن یک جریان بر روی چندین مسیر اجتناب شود. برای این منظور یک متغیر دودویی تعریف می‌شود. $r_i(u,v)$ که بیانگر آن است که آیا جریان i از لینک (u,v) استفاده می‌کند یا خیر. سپس محدودیت ۴ به محدودیت‌های مساله اضافه می‌شود [۲۳].

$$f_i(u,v) = d_i \times r_i(u,v)\,, \forall i, \quad \forall (u,v) \in E \qquad (۴)$$

این محدودیت تضمین می‌نماید که ترافیکی که بوسیله جریان i بر روی لینک (u,v) گذاشته می‌شود یا صفر است یا به اندازه کل نیاز جریان. در واقع کسری از نرخ جریان بر روی لینک هرگز وجود نخواهد داشت. اضافه شدن این مساله مدل را تبدیل به یک مدل برنامه‌ریزی آمیخته[۱۱] می‌نماید. که در این‌صورت دیگر حتی با در نظر گرفتن مقادیر پیوسته برای تقاضاها، مساله در زمان چندجمله‌ای قابل حل نیست.

البته اخیرا روش‌هایی ارائه شده است که می‌توان مدل را همچنان LP فرض نمود و محدودیت عدم شکستن جریان به چندین مسیر را نادیده گرفت. چرا که با استفاده از پروتکل MPTCP[۱۲]، امکان مدیریت پدیده خارج از ترتیب رسیدن بسته‌های متعلق به یک جریان که بر روی چند مسیر تقسیم شده‌اند فراهم شده است [۲۴]. در نتیجه هر بسته جریان می‌تواند هر یک از مسیرهای در دسترس را مستقل از سایر بسته‌های آن جریان انتخاب نماید. در حال حاضر MPTCP یک پیش‌نویس در IETF[۱۳] (RFC[۱۴] 6824) می‌باشد. MPTCP امکاناتی فراهم می‌سازد تا یک نشست TCP قادر به استفاده از مسیرهای چندگانه باشد. اگر بسته‌های مرتبط با یک جریان بر روی مسیرهای مختلف جاری شوند، با در نظر گرفتن بسته‌های روی هر مسیر به عنوان یک زیر جریان، MPTCP مکانیزم‌های لازم را برای مرتفع نمودن مساله خارج از ترتیب رسیدن بسته‌ها را مدیریت خواهد نمود.

۱-۲-۴ مساله جریان در شبکه مبتنی بر مسیر[۱۵]

همانگونه که اشاره شد، جواب مساله پایه MCF تخصیص بهینه جریان به هر لینک است به گونه‌ای که هدف مساله، بهینه شود. در نتیجه تعداد متغیرهای تصمیم‌گیری $f_i(u,v)$ با افزایش اندازه توپولوژی شبکه و یا تعداد جریان‌ها در شبکه، رشد چشمگیری خواهند داشت. با افزایش تعداد متغیرهای تصمیم‌گیری، حتی مدل برنامه‌ریزی خطی این مساله، جواب را در مدت زمان معقول بدست نخواهد آورد. در مدل مبتنی بر مسیر فرض بر این است که کلیه مسیرها بین گره‌های شبکه از قبل در دسترس هستند. البته این فرض با توجه به رفتار مدل معماری Fat-Tree به راحتی قابل حصول

است. مقصود تخصیص جریان‌ها به مسیرهای موجود به گونه‌ای است که هدف مساله بهینه گردد. مدل مبتنی بر مسیر به میزان قابل توجهی، زمان محاسباتی مساله را نسبت به MCF پایه (مبتنی بر لینک) کاهش می‌دهد. مدلPMCF در زیر آورده شده است.

$$\text{minimize } m$$

$$s.t.$$

$$\sum_{i=1}^{D} \sum_{p \in P_{T_i}^{(u,v)}} f_i(p) \le m \times c(u,v), \forall (u,v) \in E \quad (۵)$$

$$\sum_{p \in P_{T_i}} f_i(p) = d_i, \qquad \forall i \quad (۶)$$

ورودی‌ها: توپولوژی شبکه: $G(V,E)$، ماتریس ترافیکی: T، ظرفیت لینک‌ها: $\forall (u.v) \in E,\ c(u,v)$، مجموعه مسیرها به ازای تقاضاها(جریان‌ها):

$$P_{T_i} = \{p_{i,0}, \dots, p_{i,j}, \dots, p_{i,l}\}, \qquad \forall i$$

مجموعه همه مسیرهایی که لینک(u,v) را شامل می‌شوند:

$$, \forall i\ P_{T_i}^{(u,v)} \subseteq P_{T_i}$$

D، بیانگر مجموعه درخواست ها.

متغیرهای تصمیم‌گیری: m: Maximum Link Utilization، نرخ جریان بر روی هر مسیر به ازای درخواست i:

$$f_i(p), \qquad \forall i, p \in P_{T_i}$$

هدف مساله، حداقل نمودن حداکثر میزان استفاده موثر از هرلینک است. محدودیت ۵ به این نکته اشاره دارد که به ازای هر درخواست، اگر جریانی بر روی مسیری که لینک (u,v) متعلق به آن است، جاری شود، این میزان ترافیک از لینک (u,v) نیز خواهد گذشت، لذا مجموع جریان‌های مسیرهایی که لینک (u,v) به آنها متعلق است باید کمتر از ظرفیت آن لینک باشد. محدودیت ۶ به این نکته اشاره دارد که میزان ترافیکی که از روی کلیه مسیرهای متعلق به یک جریان می گذرد همواره برابر است با میزان درخواست آن جریان.

ورودی این مساله کل مسیرهای ممکن در شبکه است که در شبکه Fat-Tree با $k > 8$ ، زیاد است. ورودی دیگر، مسیرهایی است که هر لینک به آنها تعلق دارد. این ورودی‌ها تنها نیاز است یکبار محاسبه شوند و در جایی ذخیره گردند و در هر بار اجرای مساله نیازی به محاسبه مجدد آنها نیست. نکته قابل توجه آن است که در هر بار اجرای مساله تنها آن قسمت از مسیرهایی که امکان گذر درخواست‌ها از آنها وجود دارد به مساله وارد می‌شود نه همه مسیرها. در نتیجه در همان ابتدای مساله فضای جستجوی مساله تا حد امکان کوچک خواهد شد. در واقع، محدودیت ۷ بیانگر آن است که میزان ترافیک موجود بر روی لینک (u,v) که بین مسیرهای مختلف می‌تواند مشترک باشد نباید از یک حد آستانه ظرفیت لینک بیشتر باشد. هدف نیز کم کردن این حد آستانه (m) است. این هدف موجب کنترل ازدحام در شبکه خواهد شد.

یکی از مهمترین جنبه‌های مهندسی ترافیک، موازنه بار در شبکه است. با این کار سعی می‌شود مقداری از ترافیک موجود بر روی لینک‌های پربار به لینک‌های کم بار منتقل شود. نتیجه این عمل اجتناب از ازدحام بر روی هر لینک است. پس می‌توان از تاخیرهای طولانی و ریزش بسته‌ها در شبکه اجتناب نمود. مطالب ذکر شده می‌تواند علت انتخاب تابع هدف این مساله را نیز توجیه نمایند.

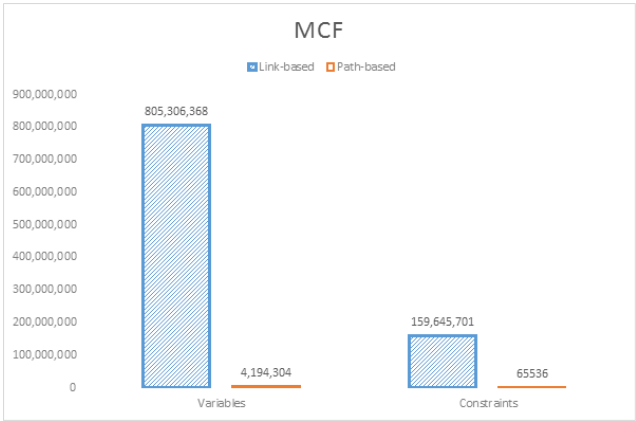
۴-۲-۲ مقایسه PMCF با LMCF

تابع هدف دو مساله PMCF و LMCF یکسان است. مهم-ترین تفاوت این دو مدل در متغیرهای تصمیم‌گیری می‌باشد. در روش LMCF قصد بر این است تا درخواست‌ها به گونه‌ای به لینک‌ها تخصیص داده شود تا علاوه بر محاسبه مسیرهای بهینه برای هر درخواست، محدودیت‌های مساله نیز رعایت شوند. در صورتی‌که در روش PMCF، فرض بر این است که مسیرها از قبل در دسترس می‌باشد و نیازی نیست که مدل، مسیریابی نیز انجام دهد و تنها کافی است که از بین مسیرهای موجود آن مسیرهایی را برگزیند که علاوه بر رعایت محدودیت‌های مساله، تابع هدف مساله را نیز حداقل نمایند.

به منظور درک بهتر اندازه متغیرها و محدودیت‌های دو مساله و زمان مورد نیاز برای هر یک از این دو روش، یک شبکه Fat-Tree با $k = 16$ در نظر می گیریم. این شبکه دارای ۱۰۲۴ میزبان، ۱۳۴۴ گره (شامل کلیه سوئیچ‌ها و میزبان‌ها)، ۶۱۴۴ لینک جهت‌دار، ۶۴ مسیر برای هر درخواست (اگر مبدا و مقصد هر درخواست در دو Pod مجزا قرار داشته باشند) است. با فرض اینکه هر میزبان دارای دو درخواست (جریان) باشد تعداد کل جریان‌ها در شبکه برابر است با ۱۰۲۴×۲=۲۰۴۸.

در LMCF، تعداد متغیرهای تصمیم‌گیری مساله (نرخ جریان بر روی هر لینک به ازای هر درخواست) برابر است با تعداد لینک‌ها × تعداد درخواست‌ها = ۱۲۵۸۲۹۱۲ و تعداد محدودیت‌های مساله (تعداد لینک‌ها + (تعداد تقاضاها× تعداد گره‌ها) + تعداد تقاضاها) برابر است با ۲۷۵۸۶۵۶. این تعداد محدودیت و متغیر فضای مساله را بسیار وسیع کرده است و حل این مساله در زمان کم و عملیاتی امکانپذیر نیست. تعداد متغیرهای تصمیم‌گیری در PMCF (نرخ جریان ترافیکی بر روی هر مسیر به ازای هر درخواست) در بدترین حالت برابر است با تعداد درخواست‌ها ضربدر تعداد مسیر به ازای هر درخواست که برابر است با ۲۰۴۸×۶۴=۱۳۱۰۷۲و تعداد محدودیت-های مساله (تعداد لینک‌ها بعلاوه تعداد تقاضاها) ۸۱۹۲ قید است. این مقادیر در شکل ۲ نشان داده شده است.

به منظور نیل به تخصیص بهینه مسیرها به جریان‌ها به گونه‌ای که ازدحام در شبکه حداقل شود، اعمال زیر باید صورت پذیرد: ۱) انتخاب روش مناسب بهینه‌سازی به منظور تخصیص بهینه جریان‌ها به مسیرها، ۲) فرموله کردن الگوریتم و نوشتن مجدد فرمول به گونه‌ای که قابل تجزیه باشد. ۳) تجزیه مساله به زیرمساله‌های مجزا بگونه‌ای که به صورت موازی قابل حل باشد. ۴) حل این زیرمساله‌ها به صورت موازی بر روی چندین هسته محاسباتی با بهره گیری از OpenMp [۲۵]. ۵) ارائه الگوریتم تجمیع جواب زیرمساله‌ها و بدست آوردن جواب مساله اصلی.



شکل۲- مقایسه فضای جستجوی روشMCF مبتنی بر لینک و مسیر در یک Fat-Tree با $k = 16$ و دو جریان در هر میزبان

۴-۳ انتخاب روش بهینه سازی

طبق مقایسه صورت پذیرفته شده بین LMCF و PMCF مشاهده گردید که روش PMCF دارای فضای جستجوی کوچک‌تری است. در نتیجه ما نیز این روش را در نظر می‌گیریم. با این کار در همان ابتدا فضای جستجوی مساله تصمیم‌گیری را به میزان قابل توجهی کاهش می‌دهیم. درست است که این مدل یک مساله LP است ولی حل این مساله برای شبکه Fat-Tree در مقیاس واقعی به دلیل فضای جستجوی بسیار بزرگ آن (متغیر و محدودیت)، هنوز هم در زمان معقول مقدور نیست.

بدنبال روشی هستیم که بتوانیم مساله را به زیرمساله‌های مجزا بشکنیم، سپس آن‌ها را به صورت موازی حل نماییم. با این کار فضای جستجوی هر زیرقسمت باید به میزان قابل توجهی نسبت به مساله اولیه، کوچک‌تر باشد. سپس با تجمیع جواب و راه‌حل این زیرمساله‌ها، جواب مساله اصلی بدست خواهد آمد.

۴-۳-۱ تجزیه مدل

دوگان مساله PMCF (DPMCF) در ادامه آورده شده است:

$$maximize \quad \sum_{i=1}^{D} y_i d_i$$

$$s.t.$$

$$y_i + \sum_{(u,v)\in p} X(u,v) \le 0, \quad \forall i \in D, \qquad p \in P_{T_i} \qquad (۷)$$

$$1 + \sum_{(u,v)\in E} X(u,v) \times C(u,v) \ge 0 \qquad (۸)$$

$$X(u,v) \le 0$$

D همان مجموعه درخواست‌ها، $X(u,v)$، متغیر دوگان متناظر با محدودیت ۵ و y_i متغیردوگان متناظر با محدودیت ۶ در مساله PMCF است.

$E_{l,l+1}$ مجموعه کلیه یال‌ها بین دو سطح l و $l+1$، مجموعه کلیه یال ها بین دو سطح l و $l+1$ است. که منظور از سطح l، لایه‌های core, Aggregation, edge در توپولوژی Fat-Tree است که به ترتیب با شماره های ۰و۱ و۲ مشخص می‌شوند. حال، مساله را به n زیرمساله مجزا تقسیم می‌نماییم. لینک‌های شبکه را به صورت $E = \cup_{j=1}^{n} A_j$ افراز می کنیم به طوری که به ازای $i \neq j$، $A_j \cap A_i = \emptyset$. و برای هر $i \in demands$ و هر $E_{r,s}$:

$$\exists\, j \;\; s.t. \;\; p \cap A_j \neq \emptyset, \;\; p \cap E_{r,s} \subset A_j, \qquad (۹)$$
$$r,s = 0,1,2 \;, r \neq s \;, \; |r-s| = 1 \quad \forall p \in P_{T_i}$$

رابطه ۹ نشان می‌دهد که همه مسیرهای مربوط به درخواست iام بین دو سطح متوالی $l+1$ و l یا بالعکس، باید در یک A_j قرار گیرد. حال با توجه به تقسیم بندی فضای مساله، زیر مساله j (sub_j) را تعریف می کنیم:

$$\phi_j(Z^j) = \max R^j = \sum_{i=1}^{D} y_i d_i$$

$s.t.$

$$y_i + \sum_{(u,v)\in p\cap A_j} X(u,v) \le 0, \qquad \forall i \in D, p \in P_{T_i} \qquad (۱۰)$$

$$Z^j + \sum_{(u,v)\in A_j} X(u,v) \times C(u,v) \ge 0 \qquad (۱۱)$$

$$X(u,v) \le 0, \qquad 0 \le Z^j \le 1$$

در صورتی که $p \cap A_j = \emptyset$ آنگاه معادله ۱۰ به صورت $y_i \le 0$ خلاصه می گردد. از آنجایی‌که، مساله بالا حداکثرسازی است، مقدار y_i در جواب بهینه برای مساله sub_j برابر صفر می‌شود. بر این اساس مجموعه I_j به صورت زیر تعریف می‌شود:

$$I_j = \{i \in D: \quad \forall p \in P_{T_i}\ , p \cap A_j \neq \emptyset\}$$

حال در مساله sub_j، محدودیت (۱۰) را با محدودیت (۱۲) بازنویسی می کنیم:

$$y_i + \sum_{(u,v)\in p\cap A_j} X(u,v) \le 0, \qquad \forall i \in I_j, p \in P_{T_i} \qquad (۱۲)$$

دوگان sub_j به صورت زیر است:

$$min\ \Pi^j Z^j$$

$s.t.$

$$\sum_{i\in I_j} \sum_{p\in P_{T_i}^{(u,v)}} f_i(p) \le \Pi^j \times c(u,v), \quad \forall (u,v) \in A_j \qquad (۱۳)$$

$$\sum\nolimits_{p\in P_{T_i}} f_i(p) = d_i, \quad \forall i \in I_j \qquad (۱۴)$$

Π^j متغیر دوگان متناظر با محدودیت ۱۱ و $f_i(p)$ متغیر دوگان متناظر با محدودیت ۱۲ در مساله sub_j است. کلیه زیرمساله‌های تعریف شده مستقل از هم بوده و می‌توانند به صورت مجزا حل گردند. براساس زیر مساله های بالا، مساله اصلی زیر را تعریف می کنیم:

$$max\ \Phi(Z) = \sum_{j=1}^{n} \phi_j(Z^j)$$

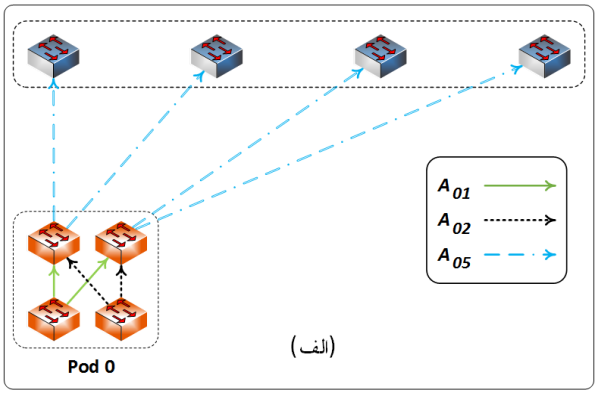
$s.t.$

$$\sum_{j=1}^{n} Z^j = 1\ .\ Z^j \ge 0 \qquad (۱۵)$$

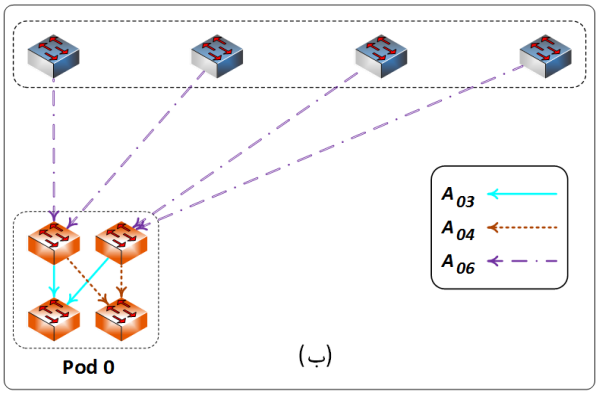
که در آن $Z = (Z^1, \dots, Z^n)$، برداری متشکل از Z^j هاست. یک جواب شدنی برای مساله اصلی به صورت زیر بدست می‌آید:

$$Z^j = \frac{1}{n} \quad \forall j = 1, \dots, n$$

ما بدنبال آن هستیم که بتوانیم لینک‌های شبکه را به زیرمجموعه-های مستقل بیشتری افراز نمایم. این عمل منجر به افزایش بیشتر تعداد زیرمساله‌ها خواهد شد و ما می‌توانیم با حل موازی این زیرمساله‌ها، زمان حل مساله را کاهش دهیم. در نتیجه، لینک‌های موجود در هر Pod_j را به $k+2$ زیرمجموعه مجزا تقسیم خواهیم نمود. سوئیچ e را در لایه لبه در Pod_j در نظر بگیرید.



شکل ۳. (الف) مجموعه همه یال‌های خروجی از pod_0



شکل ۳. (ب) مجموعه همه یال‌های ورودی به pod_0

برای هر $e = 1, \dots, k/2$، A_{ej} را به عنوان مجموعه همه لینک-هایی که از این e خارج می‌شوند، تعریف می‌نماییم. اکنون مجموعه $A_{j(k+1)}$ که شامل همه لینک‌های خروجی از همه سوئیچ‌های لایه تجمیع به لایه هسته، در Pod_j می‌شوند را لحاظ می‌کنیم. در یک روند مشابه، $A_{j(k+2)}$ و $A_{(e+k/2)j}$ به ترتیب بیانگر مجموعه همه لینک‌های ورودی به گره e و به سوئیچ‌های لایه تجمیع از لایه هسته در Pod_j هستند. این مجموعه‌ها در شکل ۳ (الف و ب) نمایش داده شده است. در این روش تجزیه، تعداد لینک‌ها در همه زیرمساله‌ها با همدیگر برابر نیست. در واقع به ازای هر گره در لایه لبه دو زیرمساله داریم. یکی مربوط است به لینک‌های خارج شده از این گره و دیگری لینک‌های ورودی به گره مزبور. تعداد لینک‌ها در این

نوع زیرمساله‌ها برابر $\frac{k}{2}$ است. نوع دیگر زیرمساله، لینک‌های ارتباطی بین لایه تجمیع با لایه هسته را در یک جهت شامل می‌شوند. که تعداد لینک‌ها در این زیرمساله‌ها $\frac{k^2}{4}$ می‌باشد. در نتیجه به ازای هر Pod، $\frac{k}{2}$ زیرمساله برای لینک‌ها خروجی از هر گره لایه لبه و به همین تعداد زیرمساله برای لینک‌های ورودی داریم. یک زیرمساله دیگر شامل کلیه لینک‌های گره‌های لایه تجمیع در جهت خروجی و یک زیر مساله شامل همین تعداد لینک در جهت ورودی است (شکل ۳). پس به ازای هر pod، $\frac{k}{2}+\frac{k}{2}+1+1=k+2$ زیرمساله مجزا داریم که این تعداد ضربدر تعداد podها، مجموع زیرمساله‌ها یعنی $n=(k+2)*k$ را تشکیل می‌دهند.

۲-۳-۴ حل مساله اصلی.

در این بخش، برای حل مساله اصلی از یک الگوریتم تکراری بر اساس تصویر زیرگرادیان[۱۶]، استفاده می‌شود. این الگوریتم را در مقاله [۳۰] ارائه و بررسی نموده‌ایم. $\phi_j(Z^j)$ یک تابع مقعر[۱۷] و ناهموار[۱۸] است. در گزاره ۱ نشان داده‌ایم که چگونه یک زیرگرادیان آن بدست می‌آید:

گزاره ۱: برای $Z^j>0$ فرض کنیم Π^j جواب بهینه دوگان متناظر با محدودیت ۱۱ در مساله $\phi_j(Z^j)$ باشد. در آن صورت Π^j زیرگرادیان تابع $\phi_j(Z^j)$ در نقطه Z^j است.

برای حل مساله اصلی با توجه به محدودیت‌های آن از روش تصویر زیرگرادیان استفاده می‌کنیم. در این روش، گرادیان تابع هدف در فضای پوچ[۱۹] محدودیت‌های فعال تصویر می‌شود [۲۶]. حال نشان می‌دهیم که چگونه ماتریس فضای پوچ محدودیت‌های فعال برای مساله اصلی محاسبه می‌شود. مجموعه $I(Z)=\{j: Z^j=0\}$ را تعریف می‌نماییم. فرض کنیم، ξ_r ، زیرگرادیان تابع اصلی در نقطه Z_r در تکرارrام باشد، با استفاده از مجموعه $I(Z_r)$ تصویر زیرگرادیان، ξ_{proj} ، به صورت (۱۶) محاسبه می شود:

$$(\xi_{proj})^j=\begin{cases}\xi_r^j-\dfrac{\sum_{i\in I(Z_r)}\xi_r^i}{|I(Z_r)|} & \forall j\in I(Z_r)\\ 0 & else\end{cases} \qquad (۱۶)$$

αگام جستجو است که براساس تصویر زیرگرادیان به صورت (۱۷) محاسبه می شود:

$$\alpha=\min\left\{-\frac{Z_r^j}{(\xi_{proj})^j}: \forall j\in I(Z_r)\ s.t.\ (\xi_{proj})^j<0\right\} \qquad (۱۷)$$

مقدار جدید Z با معادله ۱۸ بدست می آید:

$$Z_{r+1}^j=Z_r^j+\alpha(\xi_{proj})^j \qquad (۱۸)$$

براساس تعریف، تصویر زیرگرادیان نقطه جدید، نقطه شدنی است، یعنی $\sum_{j=1}^{n}Z_{r+1}^j=1,\ \ Z_{r+1}^j\geq 0 \quad \forall j$.

برای محاسبه $\phi_j(Z_{r+1}^j)$ در تکرار $r+1$ ، نیازی به حل مجدد زیر مساله jام نیست بلکه با استفاده از تحلیل حساسیت در برنامه‌ریزی خطی و جواب بهینه مساله در تکرار قبل ($\phi_j(Z_r^j)$) بدون صرف زمان محاسباتی نه چندان زیاد جواب بهینه در این تکرار بدست می‌آید. فرآیند تکرار ادامه می یابد تا شرط $\|\xi_{proj}\|\leq\epsilon$ برقرار گردد. ϵ یک عدد آستانه کوچک نزدیک به صفر است. الگوریتم ۱ حل مساله را نشان می‌دهد.

در لم ۱، نشان می‌دهیم زمانی‌که یک متغیر مساله اصلی فعال است (یعنی $z^j=0$) ضریب لاگرانژ متناظر با محدودیت $z^j\geq 0$ همواره منفی است. درنتیجه z^j در تکرار بعدی غیرمثبت (فعال) باقی می‌ماند.

الگوریتم۱: حل مساله اصلی

گام۲۰ (مقدار دهی): $\epsilon\in(0,1)$ ، $Z_1=\left\{\frac{1}{n},\dots,\frac{1}{n}\right\}$ ، $r=1$

گام ۱ (حل زیر مساله ها): مقدار $\phi(Z_r^j)$ را به ازای $j=1,\dots,n$ حل کن.

گام ۲ (محاسبه زیرگرادیان): زیرگرادیان تابع اصلی را به صورت زیر بدست آور: $\xi_r=\left(\phi(Z_r^1),\dots,\phi(Z_r^n)\right)$

گام ۳ (محاسبه تصویر زیرگرادیان): تصویر زیرگرادیان تابع اصلی را برای هر $j=1,\dots,n$ مطابق با رابطه (۱۶) بدست آور.

گام ۴ (شرط توقف): اگر $\|\xi_{proj}\|\leq\epsilon$ آنگاه متوقف شو.

گام ۵ (بروزرسانی مقادیر): گام جستجو α را براساس رابطه (۱۷) محاسبه کن و نقطه جدید را با رابطه (۱۹) بدست آور.

$$Z_{r+1}=Z_r+\alpha\xi_{Proj} \qquad (۱۹)$$

قرار ده $r=r+1$ و به گام ۱ برو.

لم ۱. در الگوریتم MPSA، اگر $z_r^j=0$ شود، آنگاه ضریب لاگرانژ محدودیت $z^j\geq 0$ در تکرار بعدی همچنان منفی باقی خواهد ماند.

روش زیرگرادیان تصویر شده در نقطه‌ای که شرط اول بهینگی KKT در آن صدق می‌کند، همگرا می‌شود. اگر چه هنوز تضمینی به منظور تحقق شرط کافی وجود ندارد. با لحاظ نمودن ساختار

مساله مورد ارزیابی، قضیه ۱ اثبات می‌کند که الگوریتم ۱ در جواب حل بهینه مساله اصلی همگرا خواهد شد.

نتایج عددی حاصل از شبیه‌سازی‌ها صحت ادعای ما را نیز تایید می‌نمایند. همه متغیرها، مقادیر دوگان و مقادیر هدف مدل تجزیه شده و مدل تجزیه نشده DPMCF یکسان هستند.

۵- پیاده‌سازی و ارزیابی تکنیک تجزیه پیشنهادی

به کمک تکنیک تجزیه پیشنهادی، توانستیم مساله اولیه DPMCF را به تعدادی زیرمساله مجزا تقسیم نماییم. در هر لحظه، بر روی هر هسته محاسباتی یک زیرمساله در حال اجرا است. در تکرار[۲۱] اول (مرحله ۱ الگوریتم MPSA)، برنامه‌ریزی خطی و ابزار OpenMP برای حل موازی زیرمساله‌ها به صورت همروند بر روی هسته‌های محاسباتی در دسترس، استفاده می‌شوند. در تکرارهای بعدی، تا زمانی که الگوریتم ۱ به جواب نهایی همگرا شود، نیازی به حل مجدد زیرمساله‌ها توسط برنامه‌ریزی خطی نیست. جواب‌های زیرمساله‌ها در تکرارهای بعد توسط تحلیل حساسیت قابل حصول است. از آنجایی‌که این فرآیند پیچیدگی محاسباتی خاصی ندارد می‌توان از زمان محاسباتی آن چشم‌پوشی نمود. نتایج بررسی‌های ما بر روی عملکرد اجرای الگوریتم بر روی سناریوهای مختلف نشان داد که این الگوریتم در تعداد تکراری برابر با تعداد زیرمساله‌ها به جواب بهینه همگرا می‌شود.

به منظور بررسی پیچیدگی زمانی و کارایی تکنیک تجزیه پیشنهادی، شبکه مرکز داده با مقیاس‌های مختلف را در نظر گرفته‌ایم. به عنوان نمونه، در یک شبکه با معماری Fat-Tree و مقدار $k = 16$، تعداد میزبان‌ها ۱۰۲۴عدد است که هر میزبان تعداد ۱۰ درخواست به مقصد ۱۰ میزبان در Podهای دیگر ارسال می‌کند. مقصدها به صورت تصادفی انتخاب شده‌اند. نوع و حجم ترافیک تولیدی توسط درخواست‌ها مشابه با مقاله [۲۹] در نظر گرفته شده است. مطابق با تکنیک تجزیه تعداد ۲۸۸ زیرمساله داریم. سخت‌افزار و نرم‌افزار بکار گرفته شده در سیستم ارزیابی دارای ویژگی‌های زیر است:

OS Version: Windows 2012 R2;
CPU: Quad 3.47 GHz Intel Xeon® X5690;
CPU Cores: 48;
Memory: 48 GB;
LP Solver: Cplex 12.6.2;

در جدول۲ پارامترهای مدل بهینه‌سازی برای مدل اولیه DPMCF و تکنیک تجزیه آورده شده است. در این آزمایش، تنها درخواست‌های بین دو Pod مجزا در نظر گرفته شده است. علت این انتخاب بدان سبب بوده است که ترافیک بر روی لینک‌ها در لایه بالاتر (که بیش‌تر ازدحام در شبکه بر روی این لینک‌ها ظاهر می شود) متاثر از این نوع درخواست‌ها است. بین هر مبدا و مقصد در دو Pod مجزا تعداد $\left(\frac{k}{2}\right)^2$ وجود دارد. در اولین مرحله از الگوریتم ، sub_j که یک مدل LP است، بوسیله Cplex حل می‌شود. ما آزمایش‌ها را بر روی شبکه با اندازه مختلف $k = 8, 12, 16$ انجام داده‌ایم. در شکل ۴ زمان حل مدل اولیه (بدون تکنیک تجزیه) در مقیاس‌های ذکر شده نمایش داده شده است. شکل ۵ زمان حل بدست آمده با استفاده از روش پیشنهادی است. همانگونه که مشاهده می‌شود، زمان حل مساله برای $K = 16$، از ۳۰۲۲ ثانیه به حدود ۱ ثانیه تقلیل چشم‌گیری یافته است. در شکل‌ها، محور عمودی بیانگر مدت زمان اجرای روش‌ها، در مقیاس ثانیه است.

جدول۲- فضای جستجو مدل بهینه‌سازی در شبکه Fat-Tree با $k = 16$

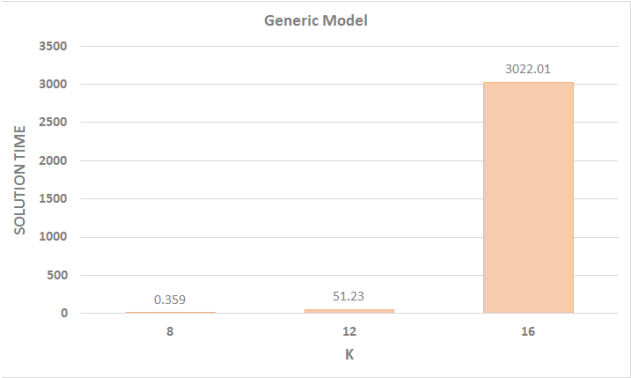
روش	مدل اولیه DPMCF (بدون تجزیه)	زیر مساله sub_j در تکنیک تجزیه
تعداد لینک	۴۰۹۶	۶۴
تعداد درخواست	۱۵۳۶۰	۹۶۰
تعداد مسیر در دسترس برای هر درخواست	۶۴	۶۴
تعداد متغیر	۱۹۴۵۶	۱۰۲۴
تعداد محدودیت	۹۸۳۰۴۱	۶۱۴۴۱

در مدل تجزیه نشده، با افزایش k، فضای جستجوی مساله گسترش می‌یابد که این منجر به افزایش نمایی زمان حل مساله می‌شود. در مدل پیشنهادی، هر چه که اندازه شبکه بزرگتر می‌شود تعداد زیر مساله‌ها نیز افزایش می‌یابد. به منظور حصول حداکثر کارایی روش پیشنهادی باید هر یک از زیرمساله‌ها بر روی یک هسته اجرا شود. با اجرای موازی زیرمساله‌ها، زمان حل، در واقع برابر می‌شود با زمان حل یک زیرمساله. در سه سناریوی تعریف شده برای $k = 8, 12, 16$ به ترتیب ۸۰، ۱۶۰ و ۲۸۸ زیرمساله داریم. پس برای داشتن حداکثر کارایی روش پیشنهادی در هر سناریو باید به تعداد زیرمساله‌ها هسته محاسباتی در اختیار داشته باشیم. شایان ذکر است، این کاهش چشم‌گیر که بوسیله تکنیک تجزیه پیشنهادی

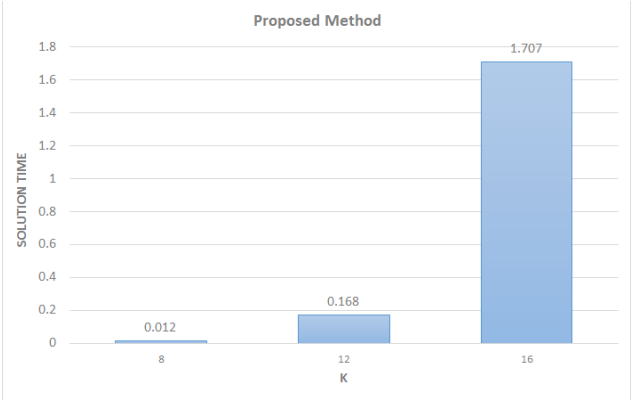
حاصل شد، تنها با ۴۸ هسته محاسباتی که در سیستم تست موجود بود، بدست آمد. شکل ۶ بیشترین زمان حل یک زیرمساله، در سناریوهای مختلف را نشان می‌دهد که این حداقل زمان حل مساله تخصیص درخواست‌ها به مسیرها در روش مهندسی ترافیک پیشنهادی خواهد بود اگر به تعداد زیرمساله‌ها هسته محاسباتی در اختیار داشتیم. به عنوان نمونه، در $k = 16$، تعداد زیرمساله‌ها در تکنیک تجزیه ۲۸۸ است. از آنجای‌که ۴۸ هسته محاسباتی داریم، در مرحله اول الگوریتم ۱ هر $\frac{288}{48} = 6$ زیرمساله باید بر روی هر هسته محاسباتی به صورت سریال اجرا شوند، چرا که حداکثر ۴۸ زیرمساله در هر زمان می‌توانند به صورت موازی و همروند اجرا شود. در واقع در هر زمان تنها یک نخ بر روی هر هسته اجرا خواهد شد. هر چه تعدد زیرمساله‌ها بیش از تعداد هسته‌های محاسباتی باشد، این تعداد زیرمساله‌ای که باید بر روی یک هسته اجرا شوند، برای در اختیار گرفتن حافظه نهان رقابت خواهند نمود که این منجر به پدیده false sharing و افزایش زمان محاسبات می‌شود. نتایج عددی حاصل از شبیه‌سازی‌ها صحت ادعای ما را نیز تایید می‌نمایند. همه متغیرها، مقادیر دوگان و مقادیر هدف مدل تجزیه شده DPMCF و مدل تجزیه نشده DPMCF یکسان هستند. شایان ذکر است که نویسندگان این مقاله، در [۳۰]، یک روش دیگر از تجزیه مسائل بزرگ بهینه‌سازی را ارائه داده‌اند که در آن مقاله میزان کارایی و دقت روش بهینه‌سازی با برخی از روش‌های مطرح مهندسی ترافیک در شبکه‌های مرکز داده مقایسه شده است. از آنجایی که روش پیشنهادی ما نیز جواب بهینه را نتیجه می‌دهد می‌توان گفت از لحاظ دقت مشابه روش ارائه شده در [۳۰] عمل خواهد نمود.

۶- نتیجه‌گیری

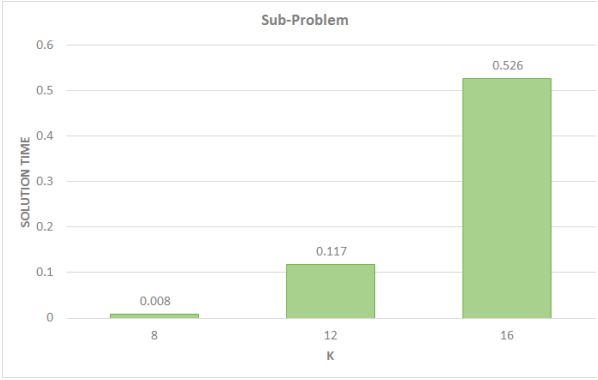
در این مقاله، به منظور تخصیص بهینه درخواست‌ها به مسیرها در شبکه‌های نرم‌افزار محور مراکز داده، یک روش مهندسی ترافیک مقیاس‌پذیر ارائه شد که با استفاده از تکنیک تجزیه می‌تواند فضای جستجو این مساله را به منظور یافتن راه‌حل بهینه در زمان کم و عملیاتی تقلیل دهد. در نتیجه مساله نگاشت بهینه درخواست‌ها به مسیرها در شبکه‌ای با معماری Fat-Tree به $(k + 2) \times k$ زیرمساله مجزا تقسیم شده است، با حل موازی این زیرمساله‌ها، زمان یافتن جواب بهینه به میزان قابل توجهی کاهش می‌یابد. یکی از مهمترین چالش‌های این تکنیک تجزیه به منظور حصول حداکثر کارایی در اختیار داشتن تعداد هسته‌های محاسباتی به تعداد زیرمساله‌ها است. لذا، پیاده‌سازی تکنیک ارائه شده با استفاده از ابزار CUDA و بهره-مندی از GPU از جمله کارهای آتی در جهت بیشتر عملیاتی شدن روش پیشنهادی می‌تواند مورد توجه واقع شود.



شکل ۴- مقایسه زمان حل مساله اولیه(بدون تجزیه) در مقیاس‌های $k = 8, 12, 16$



شکل ۵- مقایسه زمان حل روش پیشنهادی در مقیاس‌های $k = 8, 12, 16$



شکل ۶- مقایسه حداکثر زمان حل یک زیرمساله مدل تجزیه شده در مقیاس‌های $k = 8, 12, 16$

مراجع

[1] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, pp. 14-76, 2015.

[2] H. Farhady, H. Lee, and A. Nakao, "Software-defined networking: A survey," *Computer Networks*, vol. 81, pp. 79-95, 2015.

[3] T. Chen, X. Gao, and G. Chen, "The features, hardware, and architectures of data center networks: A survey," *Journal of Parallel and Distributed Computing*, 2016.

[4] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 63-74, 2008.

[5] W. Wang, Y. Sun, K. Zheng, M. A. Kaafar, D. Li, and Z. Li, "Freeway: Adaptively Isolating the Elephant and Mice Flows on Different Transmission Paths," in *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*, 2014, pp. 362-367.

[6] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 267-280.

[7] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Computer Communication Review*, vol. 40, pp. 92-99, 2010.

[8] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2001, pp. 1300-1309.

[9] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, 2009, pp. 202-208.

[10] N. Farrington and A. Andreyev, "Facebook's data center network architecture," in *IEEE Optical Interconnects Conf*, 2013.

[11] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, pp. 1-30, 2014.

[12] H. Viet, Y. Deville, O. Bonaventure, and P. Francois, "Traffic engineering for multiple spanning tree protocol in large data centers," in *Teletraffic Congress (ITC), 2011 23rd International*, 2011, pp. 23-30.

[13] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks," in *NSDI*, 2010, pp. 19-19.

[14] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," 2000.

[15] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 1629-1637.

[16] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, 2011, p. 8.

[17] Y. Li and D. Pan, "OpenFlow based load balancing for fat-tree networks with multipath support," in *Proc. 12th IEEE International Conference on Communications (ICC'13), Budapest, Hungary*, 2013, pp. 1-5.

[18] E.-S. Jung, V. Vishwanath, and R. Kettimuthu, "Distributed multipath routing algorithm for data center networks," in *Data Intensive Scalable Computing Systems (DISCS), 2014 International Workshop on*, 2014, pp. 49-56.

[19] B. Prisacari, G. Rodriguez, C. Minkenberg, and T. Hoefler, "Fast pattern-specific routing for fat tree networks," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 10, p. 36, 2013.

[20] T. C. Hu, "Multi-commodity network flows," *Operations research*, vol. 11, pp. 344-360, 1963.

[21] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Foundations of Computer Science, 1975., 16th Annual Symposium on*, 1975, pp. 184-193.

[22] M. Handley, O. Bonaventure, C. Raiciu, and A. Ford, "TCP extensions for multipath operation with multiple addresses," 2013.

[23] S.-s. Seo, "Dynamic Traffic Engineering for Improving Energy Efficiency and Network Utilization of Data Center Networks," 2013.

[24] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, Implementation and Evaluation of Congestion Control for Multipath TCP," in *NSDI*, 2011, pp. 8-8.

[25] OpenMP. (2016). Available: https://www.openmp.org

[26] D. G. Luenberger and Y. Ye, *Linear and nonlinear programming*, 4 ed.: Springer International Publishing, 2016.

[27] J. L. Kennington and R. V. Helgason, *Algorithms for network programming*: John Wiley & Sons, Inc., 1980.

[28] D. P. Bertsekas, "Nonlinear programming," 1999.

[29] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, *et al.*, "Data center tcp (dctcp)," in *ACM SIGCOMM computer communication review*, 2010, pp. 63-74.

[30] M. Bastam, M. Sabaei, and R. Yousefpour. "A scalable traffic engineering technique in an SDN-based data center network." *Transactions on Emerging Telecommunications Technologies 29*, no. 2 (2018): e3268

زیرنویس‌ها:

[1] Throughput

[2] Multi Commodity Flow(MCF)

[3] Core Layer

[4] Aggregation Layer

[5] Edge Layer

[6] Elephant Flow

[7] Mice Flow

[8] Traffic Load

[9] Minimization of the Maximal Link Utilization

[10] Link based MCF (LMCF)

[11] Mixed Integer Linear Programing

[12] Multi Path TCP

[13] Internet Engineering Task Force

[14] Request For Comments

[15] Path-based MCF(PMCF)

[16] Gradient Projection Method

[17] Concave

[18] Non-smooth

[19] Null space

[20] Step

[21] Iteration