دانشکده مهندسی کامپیوتر

استراتژی تخلیه محاسباتی احتمالی برای وظایف ناهمگون

پروژه کارشناسی مهندسی کامپیوتر

محمدمبین داریوش همدانی

استاد راهنما

رضا انتظاری ملکی

خرداد ۱۴۰۱


بسم الله الرحمن الرحيم

تأییدیه‌ی هیأت داوران جلسه‌ی دفاع از پروژه

نام دانشکده: دانشکده مهندسی کامپیوتر

نام دانشجو: محمدمبین داریوش همدانی

عنوان پروژه : استراتژی تخلیه محاسباتی احتمالی برای وظایف ناهمگون

تاریخ دفاع: خرداد ۱۴۰۱

رشته: مهندسی کامپیوتر

ردیف	سمت	نام و نام خانوادگی	مرتبه دانشگاهی	دانشگاه یا مؤسسه	امضـــــــــــا
۱	استاد راهنما	دکتر رضا انتظاری ملکی	استادیار	دانشگاه علم و صنعت ایران	
۲	استاد داور داخلی	دکتر	.....	دانشگاه علم و صنعت ایران	

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب محمدمبین داریوش همدانی به شماره دانشجویی ۹۶۵۲۱۱۹۱ دانشجوی رشته مهندسی کامپیوتر مقطع تحصیلی کارشناسی تأیید می‌نمایم که کلیه‌ی نتایج این پروژه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری‌شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی‌صلاح (اعم از اداری و قضایی) به عهده‌ی اینجانب خواهد بود و دانشگاه هیچ‌گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: محمدمبین داریوش همدانی

تاریخ و امضا:

مجوز بهره‌برداری از پایان‌نامه

بهره‌برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

□ بهره‌برداری از این پایان‌نامه برای همگان بلامانع است.

□ بهره‌برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما، بلامانع است.

□ بهره‌برداری از این پایان‌نامه تا تاریخ ..... ممنوع است.

استاد راهنما: رضا انتظاری ملکی

تاریخ:

امضا:

چکیده

رایانش لبه‌ای یک الگوی محاسبات توزیع شده است که با نزدیک کردن منابع پردازشی به لبه شبکه، سعی دارد تا مشکلاتی مانند زمان پاسخگویی، محدودیت باتری و تحرک‌پذیری را برای کاربران برطرف کند. از زمان معرفی رایانش لبه‌ای و استانداردهای معروف آن مانند رایانش لبه‌ای چند دسترسی[۱]، یکی از چالش‌های مهم این حوزه طراحی استراتژی‌های کارآمد برای تخلیه وظایف بوده است.

علاوه بر این، با رشد روز افزون صنعت تلفن همراه و اینترنت اشیا، انواع زیادی از کاربردهای نرم‌افزاری جدید با نیازمندی‌های پردازشی متفاوت در سطح شبکه به وجود آمده است. بنابراین یک ویژگی مهم در طراحی استراتژی تخلیه وظایف در رایانش لبه‌ای، در نظر گرفتن ناهمگونی کاربردها از نظر میزان منابع مورد نیاز است.

در این مقاله روشی برای بدست آوردن استراتژی تخلیه وظایف با تاخیر کمینه تحت محدودیت توان مصرفی ارائه شده است. روش پیشنهادی شامل دو قسمت می‌باشد. در قسمت اول، سیستم تخلیه وظایف با کمک زنجیره مارکوف گسسته-زمان مدل‌سازی می‌شود و در قسمت دوم، با استفاده از الگوریتمی مبتنی بر برنامه‌ریزی خطی[۲] استراتژی تخلیه بهینه برای مدل ساخته محاسبه می‌شود.

پس از معرفی و شرح استراتژی تخلیه بهینه، چارچوبی عملی در زبان Kotlin ارائه می‌شود که می‌توان با استفاده از آن استراتژی بهینه را برای سیستم مورد نظر بدست آورد و عملکرد آن را با کمک شبیه‌سازی بررسی کرد. مقاله فعلی گسترشی بر پژوهش [۱] است و از روشی مشابه با روش ارائه شده در این پژوهش استفاده می‌کند.

واژگان کلیدی: زمان‌بندی وظایف، زنجیره مارکوف، برنامه‌ریزی خطی، رایانش لبه‌ای، رایانش ابری

[1] Multi-access Edge Computing

[2] Linear Programming

فهرست مطالب

فهرست تصاویر

فهرست جداول

فهرست الگوریتم‌ها

فهرست علایم اختصاری

اجتماع C

رسانش ..... Φ

ماژولاریتی ..... Ω

گراف ..... G

مجموعه گره‌های گراف G V

مجموعه اتصالات گراف G ..... E

تعداد گره‌ها ..... $n = |V|$

تعداد یال‌ها ..... $m = |E|$

درجه گره v ..... $deg(v)$

تعداد گره‌هایی که x یال دارند ..... $N_{deg(x)}$

میانگین تعداد یال‌های هر گره گراف G ..... a_G

Clustering Coefficient CC

قطر حقیقی گراف ..... Dia

قطر موثر گراف ..... Dia_{ef}

مجموعه گره‌های گراف g ..... V_g

مجموعه اتصالات گراف g ..... E_g

فصل ۱

مقدمه

افزایش روز افزون تعداد دستگاه‌های موجود در لبه شبکه در سال‌های اخیر، و همچنین معرفی کاربردهای نرم افزاری جدید که نیازمند منابع محاسباتی بالا هستند باعث شده است که تقاضای زیادی برای خدمات پردازش ابری بوجود بیاید. پردازش ابری این امکان را به دستگاه‌های هوشمند از جمله تلفن همراه و اینترنت اشیا می‌دهد که بخشی از پردازش‌های سنگین خود را به سرورهای قدرتمند «تخلیه» کنند تا بر محدودیت‌های پردازشی خود غلبه کنند و کاربردهای نرم افزاری پیچیده‌ای مانند واقعیت افزوده و خودروهای هوشمند را برای کاربران فراهم کنند.

با این وجود، پیاده‌سازی‌های سنتی پردازش ابری یک ایراد ذاتی دارند، و آن فاصله زیاد سرورهای ابری با دستگاه‌های پایانی است. معماری پردازش لبه‌ای و پیاده‌سازی‌های استاندارد آن مانند پردازش لبه‌ای دسترسی-چندگانه که توسط سازمان ETSI ارائه شده است، سعی دارند تا با آوردن بخشی از منابع محاسباتی به نزدیکی لبه شبکه، این مشکل را تا حدی برطرف کنند. علاوه بر تمایل دستگاه‌های لبه شبکه به کمتر شدن این فاصله و به عبارتی «کشش» منابع محاسباتی توسط آن‌ها به منظور افزایش کیفیت سرویس، شرکت‌های ارائه‌دهنده خدمات ابری نیز تمایل دارند تا با «فشردن» بخشی از منابع محاسباتی خود به لبه شبکه، بار محاسباتی و هزینه‌های تجهیزاتی خود را کاهش دهند. [۲]

یک امر مهم در پیاده‌سازی کارآمد پردازش لبه‌ای، طراحی استراتژی‌های تخلیه وظایف هوشمند و موثر است. این استراتژی‌ها نحوه تخصیص منابع توسط دستگاه کاربر[1] را مشخص می‌کنند و این امکان را به دستگاه کاربر می‌دهند تا درباره تخلیه یا عدم تخلیه وظایف محاسباتی در طول زمان تصمیم بگیرد.

در این مقاله روشی برای بدست آوردن استراتژی تخلیه بهینه ارائه می‌نماییم که مبتنی بر زنجیره مارکوف گسسته-زمان و برنامه‌ریزی خطی می‌باشد. روش پیشنهادی گسترشی بر روش ارائه شده در [۱] می‌باشد. نوآوری و مزیت اصلی روش پیشنهادی نسبت به مقاله ذکر شده قابلیت پشتیبانی از وظایف با نیازمندی‌های پردازشی و شبکه‌ای متفاوت (وظایف ناهمگون) می‌باشد. یک انگیزه اصلی از این گسترش، تنوع وظایف محاسباتی در محیط‌های اینترنت اشیا بوده است. به طوری دقیق‌تر، در بسیاری از پژوهش‌های مرتبط با تخلیه وظایف در اینترنت اشیا، وظایف به دو دسته «سبک» و «سنگین» تقسیم شده اند. [۳] [۴] برای نمونه در یک اتومبیل خودران، وظیفه پردازش اطلاعات تصاویر و سنسورها به منظور راندن خودرو یک وظیفه سنگین محسوب می‌شود. در حالی که وظیفه‌ای مانند روشن کردن سیستم گرمایشی خودرو بر حسب داده‌ی سنسور دما، یک وظیفه سبک محسوب می‌شود.

ادامه این مقاله به چهار فصل تقسیم شده است. در فصل ۲ مروری بر پژوهش‌های انجام شده در حوزه استراتژی‌های تخلیه وظایف در پردازش لبه‌ای صورت گرفته است. در فصل سوم روش پیشنهادی و نحوه پیاده‌سازی آن ارائه شده است. علاوه بر این، ساختار نرم‌افزاری[2] جدیدی مبتنی بر زبان Kotlin ارائه شده است که این امکان را به کاربران و پژوهشگران می‌دهد تا استراتژی تخلیه بهینه را به ازای سیستم دلخواه خود محاسبه نمایند و آن استراتژی را با سایر استراتژی‌های پایه[3] مقایسه نمایند. در فصل چهارم به طور جامع به آزمایش و شبیه‌سازی روش ارائه شده در سناریوهای مختلف پرداخته شده است. در انتها، در فصل ۵ یک جمع‌بندی کلی از تمامی مطالب ارائه شده است و پیشنهاداتی نیز برای گسترش روش پیشنهادی ارائه شده است.

[1] User Equipment

[2] Framework

[3] Baseline

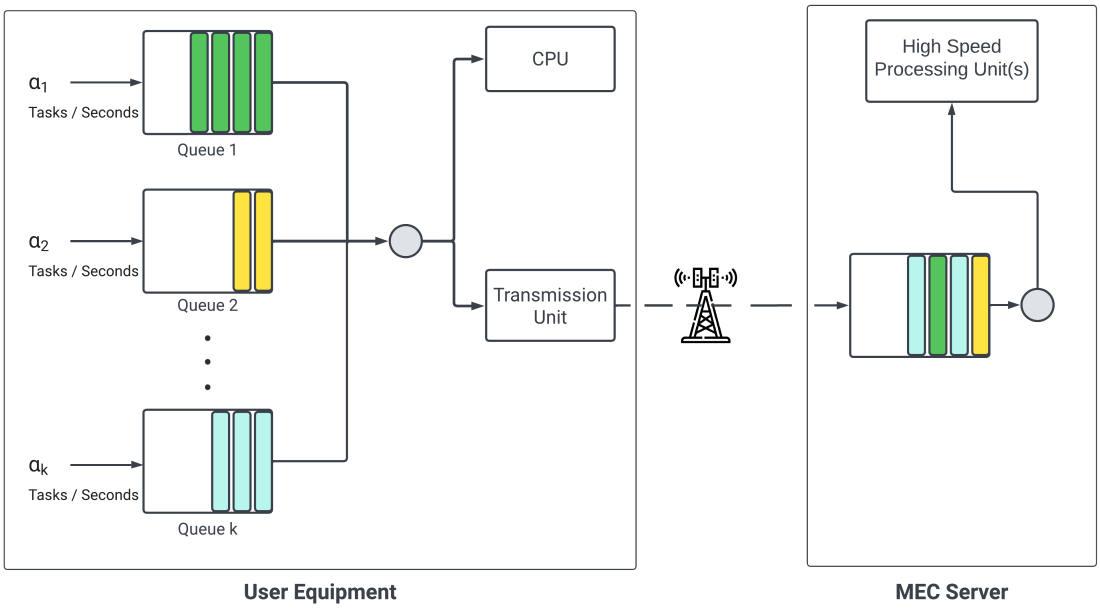
فصل ۲

مروری بر ادبیات و کارهای انجام شده

فصل ۳

روش پیشنهادی

در این مقاله برای حل مسئله پیدا کردن استراتژی تخلیه بهینه یک سیستم رایانش لبه‌ای مطابق با شکل ۳-۱ در نظر می‌گیریم.



شکل ۳-۱: ساختار کلی سیستم تخلیه پردازش

همانطور که در شکل ۳-۱ مشاهده می‌شود، در سامانه مد نظر سه مولفه اصلی زیر وجود دارد:

۱. دستگاه کاربر (User Equipment)

۲. سرور پردازش لبه‌ای چند-دسترسی (Multi-access Edge Computing Server)

۳. کانال بیسیم

در فصل جاری ابتدا نحوه مدل‌سازی هر کدام از این مولفه‌ها شرح داده می‌شود و در انتها الگوریتم و ساختار نرم‌افزاری جدیدی ارائه می‌شود که می‌توان با استفاده از آن استراتژی تخلیه بهینه را به ازای یک سیستم داده شده محاسبه کرد.

۳-۱ مدل وظایف

فرض می‌شود که k نوع وظیفه مختلف در سیستم رایانش لبه‌ای وجود دارد و به ازای هر نوع وظیفه دقیقا یک صف در سیستم موجود است. وظایف نوع i-اُم برای اجرا به صورت محلی[۱] احتیاج به L_i بازه زمانی پردازش توسط واحد پردازنده دارند و به منظور تخلیه به سرور رایانش لبه‌ای احتیاج به M_i واحد زمانی ارسال توسط واحد ارسال[۲] دارند. علاوه بر این فرض می‌شود که وظایف نوع i-اُم در سرور رایانش لبه‌ای به C_i بازه زمانی پردازش توسط سرور دارند. در ادامه این مقاله برای اشاره به یک واحد زمانی اجرا توسط پردازنده از عبارت «**قسمت**»[۳] استفاده می‌کنیم که انتزاعی از قسمت‌های کد اجرایی است. و برای اشاره به یک واحد زمانی ارسال توسط واحد ارسال از عبارت «**بسته**» استفاده می‌شود.

[1] Local

[2] Transmission Unit

[3] Section

۳-۲ مدل دستگاه کاربر

دستگاه کاربر مطابق با شکل ۳-۱ شامل دو مولفه پردازنده و واحد ارسال می‌باشد. همچنین همانطور که اشاره شد k صف مختلف به ازای هر کدام از انواع وظایف در سیستم وجود دارد. ظرفیت هر صف را برابر با مقدار ثابت Q در نظر می‌گیریم.

در هر بازه زمانی، واحد پردازنده یا به اندازه یک قسمت پردازش انجام می‌دهد و یا بیکار[۴] است. اجرای هر قسمت پردازش توسط پردازنده به میزان P_toc توان مصرف می‌کند. به طور مشابه واحد ارسال در هر بازه زمانی یا یک بسته را به شبکه ارسال می‌کند یا بیکار است. نکته قابل توجه در مورد واحد ارسال این است که با توجه به شرایط کانال بیسیم، در یک بازه زمانی خاص ممکن است ارسال موفقیت آمیز باشد یا نباشد. فرض می‌شود که ارسال موفقیت آمیز هر بسته به میزان P_tx توان مصرف می‌کند. توضیحات بیشتر در مورد نحوه کارکرد کانال بی‌سیم در بخش ۳-۴ آورده شده است.

با توجه به توضیحات داده شده می‌توان مدلی برای «حالت دستگاه کاربر»[۵] تعریف کرد. در مقاله [۱] برای مشخص کردن حالت دستگاه در زمان t از یک سه تایی مانند $\boldsymbol{\tau}[t] = (q[t], c_T[t], c_L[t])$ استفاده شده است، که در آن $q[t]$ مشخص کننده تعداد وظایف موجود در صف وظایف، $c_T[t]$ مشخص کننده اندیس بسته ارسالی توسط واحد ارسال، و $c_L[t]$ مشخص کننده اندیس قسمت[۶] اجرایی توسط پردازنده است. همچنین حالت $c_T[t] = 0$ را معادل با بیکار بودن واحد ارسال و $c_L[t] = 0$ را معادل بیکار بودن واحد پردازنده تعریف می‌کنیم. به عنوان نمونه سه تایی $(4, 2, 1)$ به این معنی است که ۴ وظیفه در صف وظایف وجود دارد، واحد پردازش در حال تخلیه وظیفه‌ای است و تا کنون یک بسته از آن وظیفه را ارسال کرده و به عنوان قدم بعدی باید بسته شماره ۲ را ارسال کند. واحد پردازنده نیز در حال اجرای وظیفه‌ای به صورت محلی است و تا کنون یک قسمت از آن وظیفه را اجرا کرده است.

[4] Idle

[5] User Equipment State

[6] Section

با این حال مدل فوق برای مسئله تخلیه وظایف ناهمگون قابل استفاده نیست و نیاز به تغییر دارد. ما در این مقاله برای تعیین حالت دستگاه کاربر از یک چندتایی[۷] به طول $k+4$ مطابق با رابطه ۳-۱ استفاده می‌کنیم. در این رابطه متغیرهای $q_1[t]$ تا $q_k[t]$ تعداد وظایف موجود از هر یک انواع وظیفه را در صف مربوطه مشخص می‌کنند. متغیرهای $c_R[t]$ و $c_L[t]$ مشابه مقاله [۱] تعریف می‌شوند و به ترتیب وضعیت واحد ارسال و واحد پردازنده را مشخص می‌کنند. دو متغیر جدید $T_R[t]$ و $T_L[t]$ به ترتیب مشخص کننده نوع وظیفه در حال ارسال توسط واحد ارسال و نوع وظیفه در حال اجرا توسط پردازنده اند.

$$\tau[t] = (q_1[t], q_2[t], \ldots, q_k[t], c_R[t], c_L[t], T_R[t], T_L[t]) \tag{۳-۱}$$

رابطه ۳-۲ شروط حاکم بر متغیرهای فضای حالت مسئله را عنوان می‌کند و به عبارتی توصیف‌گر فضای حالت مسئله است. در این رابطه عبارت $\tau(X)$ مقدار متغیر X در حالت τ را مشخص می‌کند.

$$\begin{aligned}
&\forall \tau \in S, i \in \{1,2,\ldots,k\} \quad 0 \leqslant \tau\left(q_i[t]\right) \leqslant Q \\
&\forall \tau \in S \quad \tau\left(T_L[t]\right), \tau\left(T_R[t]\right) \in \{0,1,2,\ldots,k\} \\
&\forall \tau \in \{\tau' \in S \mid \tau'\left(T_R\right) = 0\} \quad \tau(C_R[t]) = 0 \\
&\forall \tau \in \{\tau' \in S \mid \tau'\left(T_R\right) \neq 0\} \quad 1 \leqslant \tau\left(C_R[t]\right) \leqslant M_{T_R[t]} \\
&\forall \tau \in \{\tau' \in S \mid \tau'\left(T_L\right) = 0\} \quad \tau(C_L[t]) = 0 \\
&\forall \tau \in \{\tau' \in S \mid \tau'\left(T_L\right) \neq 0\} \quad 1 \leqslant \tau\left(C_L[t]\right) \leqslant L_{T_R[t]} - 1
\end{aligned} \tag{۳-۲}$$

[7] Tuple

۳-۳ مدل زمان

در مدل مسئله وضعیت سیستم در فواصل زمانی[8] با طول ثابت Δ میلی ثانیه بررسی می‌شود. برای مثال حالت دستگاه کاربر را در بازه زمانی t-اُم با $\tau[t]$ مشخص می‌کنیم، و حالت دستگاه در بازه زمانی $t+1$ را با $\tau[t+1]$ مشخص می‌کنیم و فاصله بین این دو بازه زمانی Δ میلی ثانیه است.

بررسی زمان به صورت واحدهای گسسته به منظور ساده‌سازی مسئله و همچنین گسترش پذیری آن به شرایط محیطی مختلف صورت گرفته است. در عمل، یک مقدار قابل استفاده برای Δ طول بازه‌های زمانی شبکه دسترسی[9] مورد نظر است. برای مثال در شبکه‌های LTE طول هر بازه زمانی ۰/۵ میلی‌ثانیه می‌باشد. [۵]

۳-۴ مدل کانال بیسیم

در این مقاله مشابه با [۱] کانال بی‌سیم را به صورت تصادفی مدل می‌کنیم[10] زیرا با توجه به ناپایداری کانال در ارتباطات بیسیم، به خصوص در شبکه‌های تلفن همراه، ارسال بسته‌ها توسط واحد ارسال لزوما موفقیت آمیز نخواهد بود. برای کانال بی‌سیم یک مدل ساده احتمالی به این صورت در نظر می‌گیریم که که ارسال هر بسته با احتمال β موفقیت آمیز خواهد بود و با احتمال $1-\beta$ ناموفق خواهد بود. در عمل مقدار β با توجه به رابطه ۳-۳ (رابطه شنون) محاسبه می‌شود، که در آن R مشخص کننده سایز هر بسته است، B پهنای باند سیستم، $\gamma[t]$ مقدار بهره کانال[11] و N_0 مشخص کننده اندازه نویز کانال است.

$$\begin{aligned} \beta &= P(r(t) \geq R) \\ r(t) &= B \log_r \left(1 + \frac{\gamma[t] P_{\text{tx}}}{N_0 B}\right) \end{aligned} \qquad (۳-۳)$$

[8] Time Slot

[9] Access Network

[10] Stochastic Channel

[11] Channel Gain

۳-۵ مفهوم کنش

یک استراتژی تخلیه در هر بازه زمانی مانند t می‌بایست یک کنش[۱۲] مانند $A[t]$ را برای اجرا توسط دستگاه کاربر انتخاب کند. برای درک مفهوم کنش، ابتدا مشابه [۱] حالتی را در نظر می‌گیریم که تنها یک صف (یعنی یک نوع وظیفه) در سیستم وجود داشته باشد. در این حالت می‌توانیم مجموعه کنش‌ها را با چهار عضو مطابق جدول ۳-۱ مشخص کنیم.

ID	Transmit	Local Execution	Description
1	False	False	No operation
2	False	True	Add to CPU
3	True	False	Add to TU
4	True	True	Add to both units

جدول ۳-۱: لیست کنش‌ها در سیستم با یک صف وظیفه

در حالتی که بیش از یک صف در سیستم وجود دارد به نحو مشابه می‌توان تعداد کنش‌های ممکن را مطابق با جدول ۳-۲ بدست آورد.

ID	Transmit	Local Execution	Description	Count
$\{1\}$	False	False	No operation	1
$\{2, ..., k+1\}$	False	True	Add to CPU	k
$\{k+2, ..., 2k+1\}$	True	False	Add to TU	k
$\{2k+2, ..., 2k+k*k-1\}$	True	True	Add to both units	k^2

جدول ۳-۲: دسته‌بندی کنش‌ها در سیستم با k صف

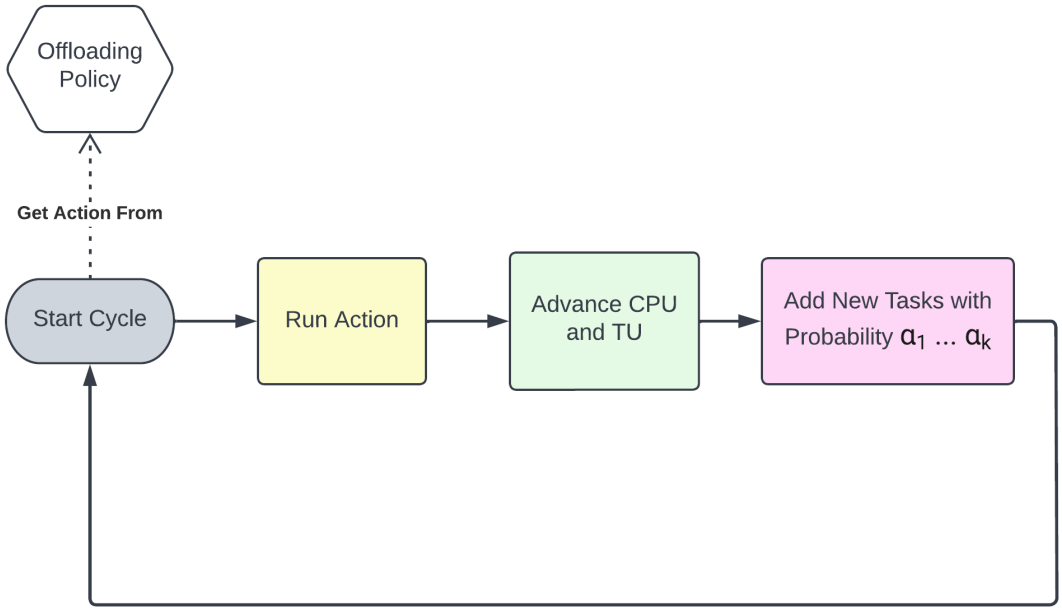
اجرای هر کنش طبعا ممکن است که حالت سیستم را تغییر دهد. به طور مثال اجرای هر عمل نوع ردیف دوم یک وظیفه را از صف مربوطه برداشته بنابراین طول صف به شکل $q_i[t+1] = q_i[t] - 1$

[12] Action

تغییر می‌کند، همچنین وضعیت پردازنده را از $c_L[t] = 0$ یعنی حالت بیکار به $c_L[t+1] = 1$ تغییر خواهد کرد زیرا قسمت اول وظیفه مربوطه در بازه زمانی t حتما انجام می‌شود. به طور مشابه برای سایر کنش‌ها نیز میتوان توابع انتقال[۱۳] مشخص تعریف کرد که با گرفتن یک حالت ورودی، حالت خروجی را محاسبه نماید. به دلیل پیچیدگی و حجم زیاد معادلات این توابع از توضیح بیشتر در این بخش صرف نظر شده است. برای مشاهده منطق دقیق این توابع در قالب کد، به پیوست ۱ مراجعه شود.

۳-۶ روند فعالیت سیستم تخلیه وظایف

دستگاه کاربر در هر سیکل زمانی به ترتیب مشخص شده در شکل ۳-۲ فعالیت می‌کند.



شکل ۳-۲: روند فعالیت دستگاه کاربر

[13] Transition Function

۳-۷ استراتژی تخلیه تصادفی

با استفاده از مدل‌های توصیف شده در بخش‌های قبل، حال می‌توانیم یک تعریف ریاضی از «استراتژی تخلیه تصادفی» داشته باشیم. مشابه با مقاله [۱] یک استراتژی تخلیه تصادفی را به صورت توزیع احتمالی مانند g_τ^a بر روی مجموعه $S \times A$ تعریف می‌کنیم. در اینجا عبارت $S \times A$ نمایانگر ضرب دکارتی مجموعه تمام حالت‌های سیستم در مجموعه تمام کنش‌های ممکن در سیستم است. یک نکته قابل توجه این است که برخی از دو تایی‌های حاصل از این ضرب دکارتی هیچ گاه در واقعیت امکان‌پذیر نیست. برای مثال در حالتی که صف خالی باشد تنها یک کنش امکان پذیر است و آن هم کنش شماره ۱ (No Operation) است. با این حال برای سادگی در توضیح تئوری روش حل مسئله، این دو تایی‌ها را نیز در دامنه تابع توزیع احتمالی استراتژی تخلیه در نظر می‌گیریم تا همواره اندازه دامنه تابع احتمال برابر با مقدار ثابت $|S| \cdot |A|$ باشد اما در پیاده‌سازی عملی چنین دوتایی‌هایی از دامنه حذف می‌شوند و مقدار احتمال ثابتی برابر صفر می‌گیرند تا با کاهش فضای حالت مسئله، سرعت الگوریتم اجرایی بهبود یابد.

همچنین طبق تعریف توزیع احتمال، رابطه ۳-۴ باید برای هر استراتژی تخلیه تصادفی برقرار باشد.

$$\sum_{\tau \in S} \sum_{a \in A} g_\tau^a = 1 \tag{۳-۴}$$

۳-۸ مدل زنجیره مارکوف دستگاه کاربر

به منظور محاسبه معیارهایی مانند توان مصرفی میانگین و تاخیر سرویس میانگین لازم است که بتوانیم درباره وضعیت سیستم تخلیه وظیفه در طولانی‌مدت استنتاج کنیم. در این قسمت ابتدا مدل آماری زنجیره مارکوف گسسته-زمان را معرفی می‌کنیم و سپس توضیح می‌دهیم که چگونه می‌توان با استفاده از این مدل معیارهای تاخیر و توان میانگین را برای یک سیستم تخلیه وظیفه محاسبه کرد.

تعریف ۱.۳. دنباله‌ای از متغیرهای تصادفی $X_1, X_2, \ldots$ را که احتمال تغییر وضعیت از زمان t به $t+1$ مستقل از وضعیت‌های قبلی باشد را یک **زنجیره مارکوف گسسته-زمان** می‌نامند. این گزاره را به بیان متغیرهای تصادفی و تابع احتمال به صورت رابطه زیر نشان می‌دهیم.

$$\Pr\left(X_{t+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_t\right) = \Pr\left(X_{t+1} = x \mid X_t = x_t\right)$$

تعریف ۲.۳. زنجیره مارکوف گسسته زمان $X(t)$ را همگن-زمان می‌گوییم اگر شرط زیر همواره برقرار باشد:

$$P\left(X_{n+1} = j \mid X_n = i\right) = P\left(X_1 = j \mid X_0 = i\right)$$

به عبارت دیگر یعنی احتمالات مربوط به انتقال بین حالت‌ها به زمان t وابسته نیستند. در این حالت احتمال انتقال زنجیره از حالت i به j را با عبارت $p_{ij} = P\left(X_1 = j \mid X_0 = i\right)$ نمایش می‌دهیم و همچنین ماتریس انتقال را با $P = (p_{ij})$ نمایش می‌دهیم. ماتریس انتقال را می‌توان به صورت یک گراف جهت‌دار نیز توصیف کرد به طوری که درایه $p_{i,j}$ در ماتریس معادل یک یال جهت‌دار از راس i به راس j با وزن $p_{i,j}$ است.

طبق تعاریف ۱.۳ و ۲.۳ می‌توان زنجیره مارکوفی برای مدل کردن حالت دستگاه کاربر در طی زمان در نظر گرفت که در آن حالت دستگاه کاربر $\tau[t]$ حالت زنجیره در زمان t را مشخص می‌کند. همچنین ماتریس انتقال χ تعریف می‌شود به طوری که $\chi_{\tau,\tau'}$ احتمال انتقال از حالت τ به τ' را مشخص می‌کند.

ماتریس انتقال را می‌توان به ازای یک استراتژی تخلیه داده شده و پارامترهای سیستمی مانند $\alpha_1, \cdots, \alpha_k$ بدست آورد. در شکل . گراف جهت‌دار معادل زنجیره مارکوف برای یک سیستم با یک صف وظیفه و $Q = 4$

فصل ۴

آزمایش و نتایج

فصل ۵

جمع‌بندی و پیشنهادها

پیوست ۱ – توابع انتقال حالت

تابع انتقال حالت به ازای کنش ورودی

```
fun getNextStateRunningAction(
   sourceState: UserEquipmentState,
   action: Action
): UserEquipmentState {
   return when (action) {
      is Action.NoOperation → {
         sourceState
      }
      is Action.AddToCPU → {
         getNextStateAddingToCPU(sourceState, action.queueIndex)
      }
      is Action.AddToTransmissionUnit → {
         getNextStateAddingToTU(sourceState, action.queueIndex)
      }
      is Action.AddToBothUnits → {
         getNextStateAddingToBothUnits(
            sourceState,
            action.cpuTaskQueueIndex,
            action.transmissionUnitTaskQueueIndex
         )
      }
   }
}
```

تابع انتقال حالت پایه

```
fun getNextStateAddingToCPU(
   sourceState: UserEquipmentState,
   queueIndex: Int
): UserEquipmentState {
   require(sourceState.cpuState == 0)
   require(sourceState.taskQueueLengths[queueIndex] > 0)

   val updatedLengths = sourceState.taskQueueLengths.decrementedAt(queueIndex)

   return sourceState.copy(
      taskQueueLengths = updatedLengths,
      cpuState = -1,
      cpuTaskTypeQueueIndex = queueIndex
   )
}
```

تابع انتقال حالت با کنش ارسال توسط واحد ارسال

```
fun getNextStateAddingToTU(
   sourceState: UserEquipmentState,
   queueIndex: Int
): UserEquipmentState {
   require(sourceState.tuState == 0)
   require(sourceState.taskQueueLengths[queueIndex] > 0)

   val updatedLengths = sourceState.taskQueueLengths.decrementedAt(queueIndex)

   return sourceState.copy(
      taskQueueLengths = updateLengths,
      tuState = 1,
      tuTaskTypeQueueIndex = queueIndex
   )
}
```

تابع انتقال حالت با کنش اجرا و ارسال به طور همزمان

```
fun getNextStateAddingToBothUnits(
   sourceState: UserEquipmentState,
   cpuQueueIndex: Int,
   tuTaskQueueIndex: Int
): UserEquipmentState {
   if (cpuQueueIndex == tuTaskQueueIndex) {
      require(sourceState.taskQueueLengths[cpuQueueIndex] > 1)
   } else {
      require(sourceState.taskQueueLengths[cpuQueueIndex] > 0)
      require(sourceState.taskQueueLengths[tuTaskQueueIndex] > 0)
   }
   return getNextStateAddingToCPU(
      getNextStateAddingToTU(sourceState, tuTaskQueueIndex),
      cpuQueueIndex
   )
}
```

مراجع

[1] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, “Delay-optimal computation task scheduling for mobile-edge computing systems,” in 2016 IEEE International Symposium on Information Theory (ISIT), pp.1451–1455, 2016.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” IEEE Internet of Things Journal, vol.3, no.5, pp.637–646, 2016.

[3] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, “On reducing iot service delay via fog offloading,” IEEE Internet of Things Journal, vol.5, no.2, pp.998–1010, 2018.

[4] H. Tran-Dang and D.-S. Kim, “Frato: Fog resource based adaptive task offloading for delay-minimizing iot service provisioning,” IEEE Transactions on Parallel and Distributed Systems, vol.32, no.10, pp.2491–2508, 2021.

[5] A.-E. M. Taha, N. A. Ali, and H. S. Hassanein, Frame-Structure and Node Identification, pp.147–160. 2011.

واژه‌نامه فارسی به انگلیسی

احتمالی . . . . . Probabilistic
اندازه . . . . . Measure
هیوریستیک . . . . . Heuristic
توپولوژی . . . . . Topology
برش . . . . . Cut
آزمایش . . . . . Experiment
شبکه‌های اجتماعی . . . . . Social Networks
قطعه‌برنامه . . . . . Program Fragment
داده‌کاوی . . . . . Data Mining
گراف . . . . . Graph
یال . . . . . Edge
گره . . . . . Node
مرکزیت . . . . . Centrality
جهانی . . . . . Global
محلی . . . . . Local
لوون . . . . . Lovain
ناپایدار . . . . . Unstable
وبلاگ . . . . . Weblog
پست . . . . . Post
افراز . . . . . Partition
خوشه . . . . . Cluster
همپوشان . . . . . Overlapping
پل . . . . . Bridge
افراز . . . . . Partition
خوشه . . . . . CLuster

واژه‌نامه انگلیسی به فارسی

مولفه . . . . . Component
اجتماع . . . . . Community
شبکه‌های پیچیده . . . . . Complex Networks
توزیع توانی . . . . . Power-Law Distribution
پدیده‌ی دنیای کوچک . . . . . Small-World Phenomenon
احتمالی . . . . . Probabilistic
تصادفی . . . . . Random
محک . . . . . Benchmark
نمودار میله‌ای . . . . . Bar-Chart
انتخاب دانه . . . . . Seed Selection
بسط . . . . . Expansion
سیستم اجتماعی . . . . . Social System
دنیای واقعی . . . . . Real-World
گره دانه . . . . . Seed Node
بسط . . . . . Expansion
پیرامون . . . . . Boundary
کارایی . . . . . Performance
گره برشی . . . . . Articulation-Point
گره برشی . . . . . Cut-Node
یال برشی . . . . . Cut-Edge
متخصص . . . . . Expert
انسجام . . . . . Cohesiveness
جدایی‌پذیری . . . . . Separability

Abstract:

Exploring community structure is an appealing problem that has been drawing much attention in the recent years. One serious problem regarding many community detection methods is that the complete information of real-world networks usually may not be available most of the time, also considering the dynamic nature of such networks(e.g. web pages, collaboration networks and users friendships on social networks), it is most probable possibility that one could detect community structure from a certain source vertex with limited knowledge of the entire network. The existing approaches can do well in measuring the community quality, Nevertheless they are largely dependent on source vertex chosen for the process. Additionally, using unsuitable seed vertices may lead to finding of low quality or erroneous communities for output of many of the algorithms. This paper proposes a method to find better source vertices to be used as seeds to construct community structures locally. Inspired by the fact that many gargantuan real-world networks and respectively their graphs contain a myriad of lightly connected vertices, we explore community structure heuristically by giving priority to vertices which have a high number of links pertaining to the core structure of the network. Experimental results prove that our method can perform effectively for finding high quality seed vertices.

Keywords: Community Detection, Complex Networks, Graph Algorithms

Iran University of Science and Technology

Computer Engineering Department

Community detection on graphs of large real-world complex networks

Bachelor of Science Thesis in Computer Engineering

By:

Hassan Abedi

Supervisor:

Dr. Hassan Naderi

May 2016