علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۹، شماره ۱، ۱۴۰۰
صفحات ۸۳- ۹۳
مقاله عادی

زمان‌بندی آگاه از تداخل زمانی در سیستم‌های مبتنی بر پردازنده گرافیکی

نگارسادات علیزاده[۱]، محمود ممتازپور[۲*]

[*]نویسنده مسئول، دریافت: ۱۴۰۰/۰۴/۲۸، بازنگری: ۱۴۰۰/۰۶/۱۴، پذیرش: ۱۴۰۰/۰۶/۱۹

[۱] فارغ‌التحصیل کارشناسی ارشد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران، ایران
[۲] استادیار، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران، ایران

چکیده

معماری‌های جدید پردازنده‌های گرافیکی برای جلوگیری از هدررفت منابع امکان اجرای همزمان چند کرنل را فراهم می‌کنند. بااین‌حال، اگر دو یا چند کرنل بخواهند همزمان به منابع اشتراکی دسترسی داشته باشند تداخل ایجاد شده و باعث کاهش کارایی می‌شود. در سال‌های اخیر پژوهش‌های متعددی در زمینه پیش‌بینی تداخل اجرای کرنل‌ها انجام شده است، منتهی پیش‌بینی تداخل بر اساس الگوی زمانی مصرف منابع تابه‌حال مورد بررسی قرار نگرفته است. در این مقاله روشی برای پیش‌بینی تداخل زمانی اجرای کرنل‌ها با استفاده از مدل‌های یادگیری ماشین ارائه شده است که با استفاده از الگوی زمانی مصرف منابع هر کرنل، تداخل اجرای همزمان دو کرنل را تخمین می‌زند. همچنین یک روش زمان‌بندی مکاشفه‌ای با استفاده از مدل تخمین تداخل پیشنهادی برای اجرای همزمان کرنل‌ها ارائه شده است. ارزیابی‌های انجام شده بر روی کاربردهای واقعی نشان می‌دهد که روش پیشنهادی در اجرای سریع‌تر کرنل‌ها به طور میانگین ۶۷٪ نسبت به حالت اجرای سریال، بیش از ۱۷٪ نسبت به به‌روزترین زمان‌بند آگاه از تداخل پیشین و بیش از ۲۷٪ نسبت به زمان‌بند ناآگاه از تداخل بهبود داشته است. همچنین نسبت به این روش‌ها به ترتیب به میزان ۲۶٪، ۱۰٪ و ۸٪ نیز در مصرف انرژی صرفه‌جویی شده است.

کلمات کلیدی: اجرای همزمان کرنل، پردازنده گرافیکی، تداخل زمانی، یادگیری ماشین.

۱- مقدمه

رشد روزافزون کاربرد[۱]های داده‌محور مانند یادگیری عمیق، پردازش تصویر، تحلیل کلان داده و محاسبات فوق سریع استفاده از پردازنده‌های گرافیکی را رونق بخشیده است. از طرفی در سال‌های اخیر شاهد رشدی چشم‌گیر در قابلیت‌های محاسباتی و کاربردهای این پردازنده‌ها بوده‌ایم که منجر به افزایش توان مصرفی آن‌ها نیز شده است [۱]. با این وجود گزارش‌ها حاکی از آن است که در بسیاری از کاربردها از حداکثر ظرفیت پردازشی پردازنده‌های گرافیکی استفاده نمی‌شود [۲]. برای رفع این مشکل در معماری‌های جدید فناوری‌هایی برای اجرای همزمان چند کاربرد بر روی یک پردازنده گرافیکی معرفی شده‌اند که این به نوبه خود رقابت بر سر منابع مشترک و تداخل اجرا و در نتیجه کاهش کارایی را به دنبال دارد.

تاکنون پژوهش‌های مختلفی در زمینه پیش‌بینی تداخل اجرای همزمان کاربردها و تخمین میزان کاهش کارایی بر اثر تداخل در پردازنده‌های گرافیکی انجام شده است [۳]. هدف ارائه روشی است که با پیش‌بینی هرچه دقیق‌تر تداخل قبل از رخداد از وقوع آن اجتناب کند و به این ترتیب، با زمان‌بندی اجرای همزمان چند کاربرد غیرمتداخل بر روی یک پردازنده گرافیکی، در عین حفظ کارایی بهره‌وری پردازنده گرافیکی بهبود یافته و انرژی مصرفی کاهش یابد. با این وجود، پیش‌بینی تداخل بر اساس الگوی زمانی[۲] مصرف منابع کرنل[۳]ها تابحال مورد بررسی قرار نگرفته است و پژوهش‌های پیشین اغلب از میانگین مصرف منابع که توسط ابزارهای نمایه‌ساز[۴] [۴،۵] یا روش‌های تحلیل و تغییر کد منبع [۶] به‌دست می‌آید استفاده کرده‌اند. استفاده از میانگین مصرف منابع برای تشخیص تداخل دقت کافی را ندارد. زیرا ممکن است در اجرای همزمان دو کاربرد، الگوی مصرف یک منبع مشترک به نحوی باشد که زمان‌های دسترسی به آن منبع توسط کرنل‌های مختلف همپوشانی نداشته باشند و به واسطه این الگوی خاص مصرف، تداخلی در اجرا نخواهند داشت. با این وجود چه در صورت وجود همپوشانی زمانی دسترسی به منابع و چه در صورت عدم وجود آن، میانگین مصرف منابع می‌تواند یکسان باشد و روش‌های مبتنی بر آن از تفاوت الگوی مصرف ناآگاه هستند.

هدف اصلی این مقاله استفاده از الگوی زمانی مصرف منابع برای پیش‌بینی تداخل در اجرای همزمان دو کرنل است. به این منظور، ابتدا با استفاده از رابط

برنامه‌نویسی کوپتی[5] شرکت انویدیا[6] نمایه‌سازی توسعه داده شده است که می‌تواند میزان استفاده یک کرنل از هر یک از منابع درون پردازنده گرافیکی را در طول زمان به صورت یک سری زمانی گزارش کند. سپس با استخراج معیار[7]های مفید از نمایه[8] زمانی کرنل‌ها و ارائه یک مدل دسته‌بند[9]، تداخل یا عدم تداخل دو کاربرد برای اجرای همزمان بر روی یک پردازنده گرافیکی پیش‌بینی می‌شود. همچنین یک مدل پیش‌بینی‌کننده رگرسیون[10] ارائه شده است که می‌تواند میزان افت کارایی اجرای همزمان کرنل‌ها را تخمین بزند. در نهایت یک الگوریتم حریصانه زمان‌بندی آگاه از تداخل ارائه شده است که با استفاده از مدل‌های مذکور می‌تواند کرنل‌های غیرمتداخل یا کمترمتداخل را برای اجرای همزمان به پردازنده‌های گرافیکی نگاشت کند. نوآوری‌های این مقاله به اختصار عبارتند از:

۱. توسعه یک ابزار نمایه‌ساز برای استخراج سری زمانی مصرف منابع پردازنده گرافیکی
۲. ارائه یک مدل مبتنی بر یادگیری ماشین برای پیش‌بینی تداخل و همچنین تخمین میزان افت کارایی اجرای همزمان دو کرنل بر اساس الگوی زمانی مصرف منابع
۳. ارائه یک زمان‌بند آگاه از تداخل زمانی برای زمان‌بندی و نگاشت کرنل‌ها به پردازنده گرافیکی

نتایج ارزیابی بر روی کاربردهای محک واقعی که از بسته توسعه نرم‌افزار انویدیا [۷] انتخاب شده‌اند نشان می‌دهد که روش ارائه شده برای زمان‌بندی، به طور میانگین ۶۷٪ نسبت به حالت سریال، بیش از ۱۷٪ نسبت به به‌روزترین زمان‌بند آگاه از تداخل پیشین و بیش از ۲۷٪ نسبت به زمان‌بندهای ناآگاه از تداخل تسریع داشته است. میزان صرفه‌جویی در مصرف انرژی نیز به همین ترتیب ۲۶٪، ۱۰٪ و ۸٪ بوده است.

این مقاله در پنج بخش تنظیم شده است. در بخش دوم مرتبط‌ترین و جدیدترین پژوهش‌های پیشین در زمینه زمان‌بندهای آگاه از تداخل و زمان‌بندهایی که با هدف کاهش انرژی مصرفی ارائه شده‌اند، مرور شده است. در بخش سوم، مدل‌های پیشنهادی برای پیش‌بینی و تخمین تداخل زمانی به همراه زمان‌بند آگاه از تداخل زمانی ارائه شده‌اند. نحوه ارزیابی مدل‌ها و زمان‌بند پیشنهادی، نتایج ارزیابی و تحلیل آن‌ها نیز در بخش چهارم ارائه شده است. در نهایت بخش پنجم شامل نتیجه‌گیری و پیشنهاداتی به منظور کارهای آتی است.

۲- پژوهش‌های پیشین

در این بخش به مرور زمان‌بندهای ارائه شده در پژوهش‌های پیشین برای کارهای محول شده به پردازنده گرافیکی می‌پردازیم. برخی از این زمان‌بندها با در نظر گرفتن تداخل، بهره‌وری پردازنده گرافیکی را با اجرای همزمان چند کاربرد غیرمتداخل روی یک پردازنده گرافیکی بهبود می‌دهند. برخی دیگر نیز با استفاده از تکنیک‌های کاهش انرژی، زمان‌بندی با هدف بهینه‌سازی انرژی مصرفی ارائه کرده‌اند.

۲-۱- زمان‌بندی آگاه از تداخل

در پژوهش‌های پیشین روش‌هایی برای اجرای هم‌زمان چند کاربرد بر روی یک پردازنده گرافیکی ارائه شده است که هرکدام تأثیر بسزایی در بهبود کارایی داشته‌اند. شکفته و همکاران [۵] با ابزار نمایه‌ساز انویدیا گزارشی از میانگین میزان مصرف کاربردها تهیه کرده‌اند و با بهره‌گیری از مدل‌های یادگیری ماشین به دسته‌بندی کاربردها به دو کلاس حافظه‌محور و محاسبه‌محور پرداخته‌اند. سپس در حین زمان‌بندی سعی شده است کاربردها از دو کلاس متفاوت برای اجرای همزمان انتخاب شوند تا احتمال بروز تداخل کاهش یابد. ایده این روش در انتخاب حداقل معیارهایی است که بتوان با آن‌ها دسته‌بندی را با دقت قابل قبولی انجام داد. نتایج این پژوهش نشان‌دهنده بهبود ۱۰٪ کارایی در اجرای همزمان کاربردها است، اما انرژی مصرفی گزارش نشده است.

فال و همکاران [۶] با به‌کارگیری ابزارهای نمایه‌سازی نمایه‌ای از کاربردها حین اجرای تکی تهیه کرده‌اند و با اطلاعات حاصل از آن کاهش کارایی در حالت اجرای هم‌زمان دو کار بر روی یک پردازنده گرافیکی را پیش‌بینی کرده‌اند. در حین اجرا، در صورتی که افت کاراییِ ناشی از تداخل پیش‌بینی شده از حد آستانه بیش‌تر باشد، کار از ابتدا بر روی پردازنده گرافیکی دیگری اجرا می‌شود و به این صورت از رخداد تداخل جلوگیری می‌شود. به‌کارگیری این روش سبب بهبود ۲۵٪ بهره‌وری پردازنده گرافیکی، کاهش ۳۹٪ زمان انتظار کارها در صف و بهبود ۲۰٪ تأخیر کارها شده است و انرژی مصرفی اجرا گزارش نشده است.

اوکیداو و همکاران [۴] از روی شباهت منابع مورد درخواست دو کاربرد، امکان ایجاد تداخل را تخمین زده‌اند. به این صورت که هرچه شباهت منابع درخواستی دو کاربرد بیشتر باشد، احتمال ایجاد تداخل در اجرای هم‌زمان آن‌ها بر روی یک پردازنده گرافیکی بیشتر است. در این روش به محض ورود یک کاربرد برای به حداقل رساندن سربار زمانی، نمایه کوتاهی از منابع مورد نیاز آن تهیه می‌شود. سپس با استفاده از تاریخچه نمایه کامل کاربردهایی که به صورت برون‌خط[11] اجرا شده‌اند (به عنوان مجموعه آموزش) و روش فیلترکردن مشارکتی[12]، موارد ناقص نمایه‌های کوتاه پیش‌بینی و تکمیل می‌شوند. در نهایت در صورت وجود منبع بیکار، کاربرد جدید به آن منبع اختصاص می‌یابد و در غیر این صورت از روی شباهت نمایه کاربرد جدید و کاربردهای در حال اجرا، میزان تداخل آن را با کاربردهای جاری تخمین زده و کاربرد را به منبعی با کمترین تداخل ممکن نگاشت می‌کند. روش مورد استفاده باعث ۲۷/۵٪ افزایش در گذرداد و ۱۶/۳٪ افزایش در بهره‌وری پردازنده گرافیکی نسبت به زمان‌بندهای ناآگاه از تداخل شده است.

زو و همکاران [۸] با استفاده از ابزارهای نمایه‌سازی، پیش از اجرای کاربرد و از روی کد برنامه آن، گراف فراخوانی توابع کرنل را به دست آورده و سپس الگوی استفاده از منابع پردازنده گرافیکی را استخراج کرده‌اند و به این ترتیب معیارهای هر کاربرد را به دست آورده‌اند. سپس از روی شباهت معیارهای کاربرد جدید و مقایسه آن با معیارهای کاربردهای در حال اجرا تداخل را تشخیص داده و کاربرد جدید را به مناسب‌ترین پردازنده گرافیکی تخصیص داده‌اند. ژو و همکاران [۹] نیز به بررسی تداخل بین پردازنده مرکزی و پردازنده گرافیکی که روی یک تراشه مجتمع شده‌اند پرداخته‌اند. با اینکه تأخیر ارتباطات بین پردازنده مرکزی و پردازنده گرافیکی در تراشه‌های مجتمع کمتر شده است، اما در عین‌حال اشتراک منابع سخت‌افزاری بین آن‌ها منجر به پیچیده‌تر شدن تداخل در حین اجرای هم‌زمان برنامه‌ها بر روی پردازنده مرکزی و پردازنده گرافیکی شده است. به‌کارگیری این روش پیش‌گیرانه سبب ۴۶٪ بهبود در گذرداد[13] و عدم تجاوز میزان مصرف توان پردازنده از حد مورد نظر شده است. اما تداخل ناشی از اجرای همزمان کاربردها در پردازنده گرافیکی در نظر گرفته نشده است.

هوگو و همکاران [۱۰] افزونه‌ای[14] برای کتابخانه معماری‌های ناهمگن ارائه نموده‌اند. این کتابخانه به چندین کد موازی اجازه می‌دهد که همزمان و با کمترین تداخل اجرا شوند. این کدهای موازی درون زمینه زمان‌بندی[15] اجرا می‌شوند. زمینه زمان‌بندی، محیط اجرایی محدودی را با امکان پارتیشن‌بندی منابع محاسباتی (مثل پردازنده‌های مرکزی و پردازنده‌های گرافیکی) فراهم می‌کند و خود می‌تواند اندازه محیط اجرا را در راستای بهینه‌سازی نحوه تخصیص منابع به صورت پویا تغییر دهد. محققان در این پژوهش ناظر ارشدی[16] طراحی نموده‌اند که با نظارت بر وقایع درون زمینه زمان‌بندی و با توجه به بازخوردهایی که از سیستم در حال اجرا می‌گیرد (مانند میزان بهره‌وری منابع)، اندازه زمینه محیط اجرایی را گسترش داده و یا محدود می‌نماید. نتایج نشان داده است که تغییر اندازه زمینه

اجرا منجر به بهره‌وری بهتر از منابع محاسباتی می‌شود و زمان اجرای کاربردها را تا ۳۴٪ کاهش می‌دهد.

آگویلرا و همکاران [۱۱] به اجرای هر هسته با بیشینه فرکانس کاری و نیز اجرای همزمان چند کاربرد برروی هسته‌های متفاوت پرداخته‌اند. در روش پیشنهادی در این پژوهش با در نظر گرفتن تداخل در دسترسی همزمان به منابع در اجرای کاربردها بر روی یک پردازنده گرافیکی، سعی شده پردازش‌های محاسبه‌محور برای اجرا به هسته‌های سریع‌تر و پردازش‌های حافظه‌محور به هسته‌های کندتر تخصیص یابند. این روش به‌طور میانگین منجر به افزایش کارایی تا ۴۶٪ شده است. این پژوهش از مکانیزم اولیه‌ای که برخی سازندگان پردازنده گرافیکی برای اجرای همزمان مکانی در نظر گرفته‌اند، استفاده نموده است. پتنک و همکاران [۱۲] با استفاده از معیارهایی نظیر تعداد دستورات محاسباتی، تعداد دستورات حافظه، تعداد نخ‌ها و معیارهای دیگر مدل دسته‌بندی ارائه نموده‌اند که کارها را به دو دسته پردازش درون حافظه‌ای و پردازش درون هسته‌ای دسته‌بندی می‌کند و از نتایج آن برای تخصیص یک کار حافظه‌محور روی یک پردازنده گرافیکی و تخصیص هم‌زمان چند کار محاسبه‌محور روی پردازنده دیگر استفاده می‌کند. نتایج حاصل کارایی کاربردها را ۲۵٪ تا ۴۲٪ افزایش داده است. اما دستیابی به معیارهای مورد نیاز دسته‌بند به تغییر کد منبع نیاز دارد. کرمی و همکاران [۱۳] نیز به شناسایی گلوگاه‌های کارایی پرداخته‌اند زیرا معتقدند که برطرف کردن این گلوگاه‌های می‌تواند کاربردها را به بهترین کارایی ممکن سوق دهد. در این پژوهش نیز نقدی بر نمایه‌ساز انویدیا وارد شده و یک مدل کارایی آماری برای یافتن گلوگاه‌ها و ارتباط بین آن‌ها ارائه شده است. سپس با اندازه‌گیری شمارنده‌های کارایی در پردازنده‌های گرافیکی رفتار کاربردها را ثبت نموده و با استفاده از آن‌ها یک مدل رگرسیون را برای پیش‌بینی رفتار کاربردها با دقت ۹۱٪ آموزش داده‌اند.

۲-۲- زمان‌بندی با هدف بهبود انرژی مصرفی

دو تکنیک رایج کاهش انرژی مصرفی در مراکز داده مقیاس‌بندی پویای ولتاژ و فرکانس و خواباندن پویای منابع است. می و همکاران [۱۴] با استفاده از تکنیک‌های مقیاس‌بندی پویای ولتاژ و فرکانس[۱۷] و خواباندن پویای منابع[۱۸] خاص پردازنده گرافیکی، یک الگوریتم زمان‌بندی برخط[۱۹] اکتشافی برای تخصیص چند کار به خوشه[۲۰] ارائه داده‌اند، به این ترتیب که کارها را به دو دسته مهلت-مقدم و انرژی-مقدم تقسیم می‌کنند. دسته اول کارهایی هستند که باید به مهلت اجرای خود برسند و دسته دوم کارهایی هستند که می‌توانند برای صرفه‌جویی در مصرف انرژی در مدت بیشتری اجرا شوند. الگوریتم پیشنهادی با توجه به بار کاری فعلی هر سرور و بار کاری جدید، فرکانس و ولتاژ پردازنده گرافیکی را برای کاهش انرژی مصرفی، مجددا تنظیم می‌کند. از طرفی با تجمیع بار کاری بر روی یک سرور و خواباندن سرورهای بی‌بار به کاهش توان ایستا کمک می‌کند. فرض بر آن است که هرکار توسط یک زوج پردازنده مرکزی-پردازنده گرافیکی انجام می‌شود و هر زوج بیش از یک کار در حال اجرا ندارد. این روش سبب بهبود ۳۰٪ الی ۳۶٪ انرژی مصرفی در مقایسه با الگوریتم سنتی بین‌پکینگ[۲۱] شده است. اما تداخل ناشی از اجرای چند کار روی سرور را نادیده گرفته است. چاو و همکاران [۱۵] نیز به مطالعه زمان‌بندی کارها در سیستم‌های ناهمگن پرداخته و از رویکرد مقیاس‌بندی پویای ولتاژ و فرکانس در سیستم‌های مبتنی بر پردازنده‌های مرکزی و گرافیکی استفاده کرده‌اند. ابتدا با فرض یکسان بودن زمان اجرای کارها بر روی تمامی پردازنده‌ها، یک الگوریتم حریصانه جهت کاهش انرژی مصرفی معرفی نموده‌اند و سپس با فرض غیریکسان بودن زمان اجرای کارها دو الگوریتم تقریبی و حریصانه پیشنهاد کرده‌اند. در این پژوهش نیز به بررسی واگذاری همزمان کارها به پردازنده گرافیکی پرداخته نشده است. نتایج نشان داده است که این روش ۱۶٪ الی ۲۰٪ بیشتر از مقدار بهینه انرژی مصرف می‌کند.

۳- روش پیشنهادی

گسترش روزافزون استفاده از پردازنده‌های گرافیکی و توان مصرفی زیاد آن‌ها باعث شده است تا به‌کارگیری روش‌هایی برای کاهش مصرف انرژی حتی به میزان کم، منجر به کاهش قابل‌توجهی در هزینه‌ها شود. در همین راستا، فناوری‌هایی برای اجرای همزمان چند کاربرد بر روی یک پردازنده گرافیکی به منظور افزایش بهره‌وری آن معرفی و چالش تداخل به عنوان مانعی در جهت حفظ کارایی مطرح شد. از آن‌جایی که پژوهش‌های گذشته تداخل را با توجه به میانگین مصرف هر کاربرد در نظر می‌گرفتند و توجهی به الگوی مصرف در طول زمان نداشتند، در این مقاله روشی برای نمایه‌سازی زمانی کاربردها و استخراج الگوی زمانی مصرف منابع جهت تشخیص تداخل زمانی معرفی شده است.

روش ارائه شده در این تحقیق قابلیت زمان‌بندی موثری در دو سناریو زیر را دارد:

۱. بخشی از مجموعه کاربردهایی که قرار است زمان‌بندی شوند، از پیش معلوم باشند. در این سناریو هر بارکاری جدیدی که وارد سیستم می‌شود، به صورت دسته‌ای از کاربردهای تک هسته‌ای یکسان است که قرار است به صورت مکرر اجرا شوند. در این حالت سربار یک بار نمایه‌سازی در ازای چندین بار تکرار کاربرد، سرشکن می‌شود.

۲. می‌توان فرض معلوم و تکراری بودن کاربردها را در نظر نگرفت و در عوض در بخش زمان‌بندی کاربردهای هر بارکاری، نمایه کوتاهی از کاربرد جدید تهیه کرد که فقط شامل چند معیاری باشد که در دسته‌بندی داده‌ها نقش موثرتری دارند. سپس با روش‌های مشابهت‌گیری سیستم‌های توصیه‌گر[۲۲] از روی مجموعه نمایه‌های کاملی که قبلا به صورت برون‌خط جمع‌آوری شده‌اند، آن را نیز کامل نمود تا به این ترتیب سربار نمایه‌سازی کاهش یابد تا بتوان کاربرد جدید را به صورت برخط نمایه‌سازی نمود.

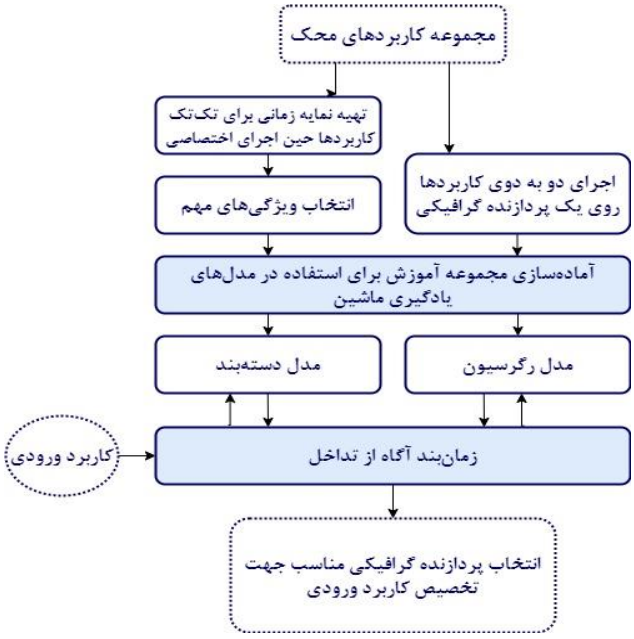
در این پژوهش به پیاده‌سازی روش زمان‌بندی سناریو اول پرداخته شده و بررسی سناریو دوم به پژوهش‌های آتی سپرده شده است.

نمایه‌ساز توسعه داده شده در این پژوهش از رابط برنامه‌نویسی کوپتی شرکت انویدیا استفاده می‌کند. این رابط برنامه‌نویسی اجازه می‌دهد تا بتوان شمارنده‌های کارایی[۲۳] پردازنده گرافیکی را در هر لحظه از زمان استخراج کرده و به عنوان مجموعه داده آموزش استفاده کرد. همچنین برای تکمیل مجموعه داده آموزش مدل تشخیص تداخل، کاربردهای مجموعه آموزش هم به صورت تکی و هم دو به دو با هم بر روی پردازنده گرافیکی اجرا شده و میزان افزایش زمان اجرای آن‌ها که ناشی از تداخل اجرای همزمان است اندازه‌گیری می‌شود و داده‌ها برچسب‌گذاری می‌شوند.

برای تشخیص تداخل، معیارهای به‌دست آمده از نمایه‌ساز به همراه برچسب‌های به‌دست آمده از اجرای کاربردهای مجموعه آموزش به یک مدل دسته‌بند و یک مدل رگرسیون داده می‌شود تا به ترتیب برای پیش‌بینی تداخل دو کاربرد و پیش بینی میزان افت کارایی آموزش داده شوند. سپس برای زمان‌بندی کارها، مدل‌ها در یک گره[۲۴] مرکزی که گره زمان‌بند نامیده می‌شود مورد استفاده قرار می‌گیرند. با ورود کاربرد جدید، نمایه‌اش به همراه نمایه کاربردی که بر روی پردازنده گرافیکی در حال اجرا است، به عنوان داده آزمون به مدل‌ها داده می‌شوند. به این صورت قبل از نگاشت کاربرد ورودی به پردازنده گرافیکی تشخیص داده می‌شود که این دو کاربرد برای اجرا روی آن پردازنده گرافیکی مشترک مناسب هستند یا خیر. اگر کاربرد ورودی از نظر دسته‌بند با همه کاربردهای در حال اجرا بر روی همه پردازنده‌های گرافیکی تداخل داشته باشد، با استفاده از نتیجه مدل رگرسیون، آن را در کنار کاربردی که اجرای هم‌زمانشان کمترین افت سرعت را در پی دارد، قرار می‌دهد. لزوم استفاده از مدل رگرسیون، افزایش میزان بهره‌وری

پردازنده‌های گرافیکی و پذیرش حداکثری بار کاری ضمن اجتناب از تداخل است. کلیات مراحل انجام شده در چارچوب پیشنهادی در شکل ۱ آورده شده است.

نمایه‌ساز زمانی، مدل‌های دسته‌بند و رگرسیون و همچنین زمان‌بند پیشنهادی در ادامه با جزئیات بیشتر توصیف شده‌اند.



شکل ۱: کلیات مراحل انجام شده در چارچوب روش پیشنهادی

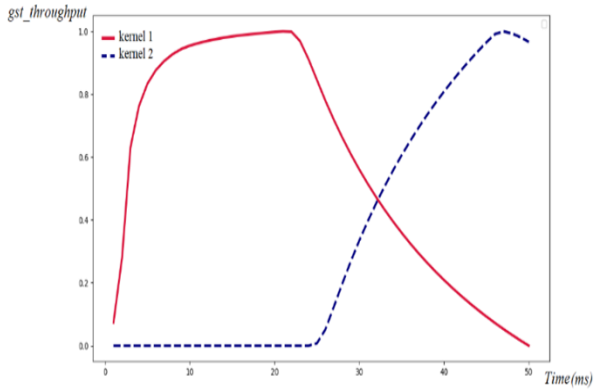
۳-۱- نمایه‌ساز زمانی

تمرکز این پژوهش بر روی کاربردهایی است که به زبان کودا نوشته شده باشند. می‌توان با کمک ابزارهای نمایه‌ساز انویدیا (به عنوان مثال nvprof یا nsight)، اطلاعات کرنل کاربردهایی که به زبان کودا هستند را استخراج نمود. در پژوهش‌های پیشین، عمدتا از این ابزارها برای نمایه‌سازی استفاده شده است. این ابزارها میانگین مقدار شمارنده‌های کارایی (مثلاً میزان مصرف منابع) در کل زمان اجرا را در اختیار کاربر قرار می‌دهند. در این پژوهش با استفاده از رابط کوپتی روشی برای نمایه‌سازی ارائه شده است که در حین اجرای کرنل، با نرخ معینی که توسط کاربر مشخص می‌شود از شمارنده‌های کارایی نمونه‌برداری می‌کند. این نمایه‌ساز در یک نخ جداگانه به صورت موازی با نخی که در حال اجرای کرنل‌ها است اجرا می‌شود تا بتواند همزمان با اجرای کرنل‌های کاربرد در همان زمینه کودای[۲۵] مورد استفاده به نمایه‌سازی کاربرد بپردازد، بنابراین نیاز است درون کد کاربرد نهفته‌سازی شود. نمایه‌ساز با فراخوانی توابع رابط‌های برنامه‌نویسی رخداد[۲۶] و بازگشت فراخوانی[۲۷] کوپتی در نخ ایجاد شده برای آن و انجام عملیات همگام‌سازی با فراخوانی‌های کرنل‌ها در نخ‌های اصلی کاربرد، به طور همزمان و همگام با اجرای کرنل‌ها به استخراج شمارنده‌های کارایی و معیارهای مورد نظر کرنل‌ها می‌پردازد. سپس نتایج نمایه‌سازی کرنل‌ها را در قالب سری‌های زمانی (یک سری زمانی به ازای هر معیار استخراج شده از هر کرنل) ذخیره می‌کند. جدول ۱ لیست ۱۵ معیار استخراج شده از نمایه‌سازی در این پژوهش را نشان می‌دهد که در پژوهش‌های پیشین به اهمیت آن‌ها در تشخیص تداخل اجرای کرنل‌ها اشاره شده است [۵،۴]. در واقع این معیارها طوری انتخاب شده‌اند که میزان مصرف منابع مشترک پردازنده گرافیکی از جمله حافظه و واحدهای پردازشی را که می‌توانند در تشخیص تداخل اجرا نقش داشته باشند پوشش دهند.

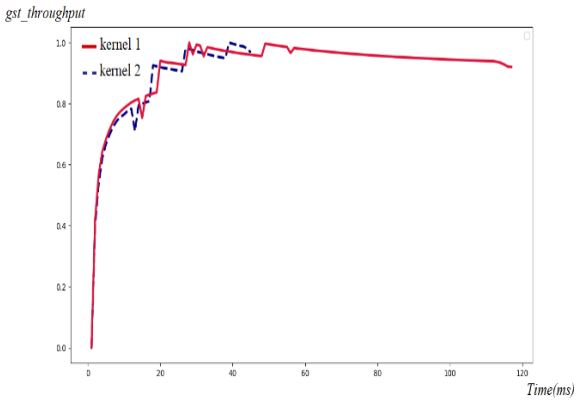
برای درک بهتر اهمیت استخراج الگوی زمانی مصرف منابع در تشخیص تداخل نسبت به روش مرسوم استخراج میانگین مصرف منابع، نمونه‌هایی از اجرای دو به دو کرنل‌ها در ادامه آورده شده است. در شکل ۲، شکل ۴ و شکل ۶ مثال‌هایی از دو کرنل غیر متداخل نشان داده شده است که میانگین مصرف هر دو در معیارهای متناظر یکسان است، درحالی‌که یکی در اوایل اجرا و دیگری در اواخر اجرا، منابع بیشتری مصرف کرده است؛ بنابراین عملاً اجرای همزمان آن‌ها بر روی یک پردازنده گرافیکی تداخل چندانی نخواهد داشت. از طرفی دیگر، شکل ۳، شکل ۵ و شکل ۷ الگوی مصرف دو کرنل متداخل را نشان می‌دهد. مشاهده می‌شود که نمایه‌ساز زمانی به خوبی تفاوت جفت کرنل‌های متداخل و غیر متداخل را نشان می‌دهد. اما نتایج میانگین مصرف گزارش شده از نمایه‌ساز انویدیا قادر به تفکیک این حالت نسبت به حالت قبلی (غیرمتداخل) نیست.

جدول ۱: معیارهای نمایه شده

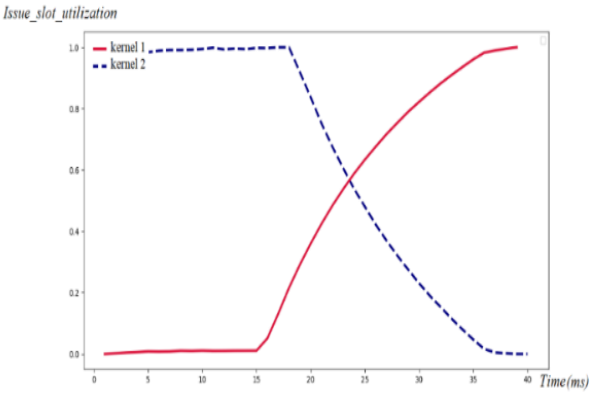
معیار	توضیح
achieved_occupancy	نسبت میانگین تعداد رسیمان‌های فعال در هر چرخه[۲۸] به بیشینه تعداد ریسمان در هر SM
dram_read_throughput	نسبت تعداد دسترسی خواندن از DRAM به زمان اجرای کرنل
dram_read_transactions	میانگین تعداد دسترسی خواندن از DRAM
dram_write_throughput	نسبت تعداد دسترسی نوشتن در DRAM به زمان اجرای کرنل
dram_write_transactions	میانگین تعداد دسترسی نوشتن در DRAM
gld_throughput	نسبت میانگین تعداد دسترسی خواندن از حافظه سراسری به زمان اجرای کرنل
gst_throughput	نسبت میانگین تعداد دسترسی نوشتن در حافظه سراسری به زمان اجرای کرنل
IPC	میانگین تعداد کل دستورات اجرا شده به کل چرخه‌های فعال
issue_slot_utilization	میانگین درصد حفره‌های زمانی[۲۹] که حداقل یک دستور برای اجرا داشته‌اند در طول همه چرخه‌ها
l2_read_transactions	میانگین تعداد دسترسی خواندن از حافظه نهان سطح ۲
l2_write_transactions	میانگین تعداد دسترسی نوشتن در حافظه نهان سطح ۲
sm_efficiency	نسبت چرخه‌های فعال به کل چرخه‌های اجرای کرنل
dram_utilization	نسبت سطح بهره‌وری از حافظه به بیشترین بهره‌وری
tex_cache_throughput	نسبت میانگین دسترسی به حافظه نهان به زمان اجرای کرنل
shared_store_throughput	نسبت میانگین دسترسی نوشتن در حافظه مشترک به زمان اجرای کرنل



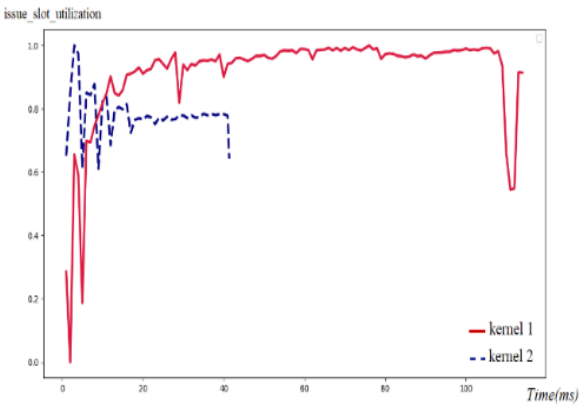
شکل ۲: الگوی زمانی معیار gst_throughput برای دو کرنل غیر متداخل



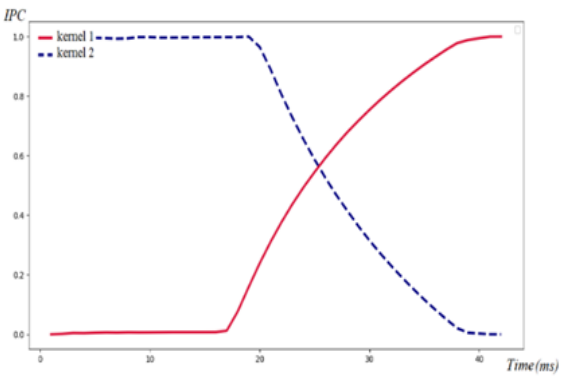
شکل ۳: الگوی زمانی معیار gst_throughput برای دو کرنل متداخل



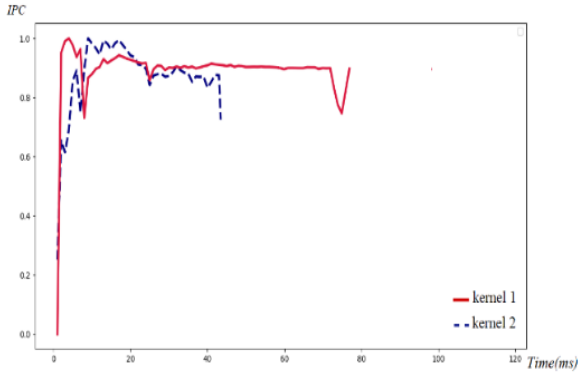
شکل ۴: الگوی زمانی معیار issue_slot_utilization برای دو کرنل غیر متداخل



شکل ۵: الگوی زمانی معیار issue_slot_utilization برای دو کرنل متداخل



شکل ۶: الگوی زمانی معیار IPC برای دو کرنل غیر متداخل



شکل ۷: الگوی زمانی معیار IPC برای دو کرنل متداخل

۳-۲- مدل‌های دسته‌بند و رگرسیون

مدل‌های مورد استفاده در این پژوهش هر دو از نوع با سرپرست هستند، بنابراین داده‌های مجموعه آموزش با ثبت زمان اجرای دو به دو کاربردها و محاسبه میزان افزایش زمان اجرا برچسب‌گذاری می‌شوند. به‌ازای هر معیار مستخرج از هر کاربرد، یک سری زمانی با نمونه‌برداری با نرخ ۷۰ میلی‌ثانیه یک‌بار به‌دست آمده است. برای سنجش شباهت الگوی زمانی مصرف منابع هر دو کاربرد دلخواه از شاخص همبستگی متقابل[۳۰] [۱۶] استفاده شده است که برای محاسبه میزان شباهت سری‌های زمانی بسیار پرکاربرد است. مقدار همبستگی متقابل دو سری زمانی به ازای یک معیار مشخص (مثلاً IPC) بین دو کاربرد مورد نظر محاسبه شده و سپس از نتایج حاصل میانگین‌گیری شده است. این روند در معادله (۱) نشان داده شده است که در آن x و y سری‌های زمانی یک معیار مشخص، $cross\_corr$ تابع محاسبه همبستگی متقابل، N تعداد نمونه‌های زمانی نمایه و Sim میزان مشابهت دو سری زمانی است.

$$Sim(x,y) = \frac{1}{N}\sum cross\_corr(x,y) \quad (۱)$$

پس از این مرحله، برای هر دو کاربرد دلخواه آرایه‌ای ۱۵تایی از معیارها داریم که مقدار هر عنصر آن، شباهت دو کاربرد در آن معیار مشخص را نشان می‌دهد. در واقع استفاده از شباهت الگوی زمانی مصرف منابع کمک می‌کند تا برخلاف ~~معیار~~ میانگین مصرف بتوان وجود یا عدم وجود تداخل در لحظه را بهتر تشخیص داد. به عبارت دیگر، اگر دو کاربرد در یک لحظه از زمان از یک منبع مشترک استفاده کنند، مقدار شباهت به دست آمده با استفاده از همبستگی متقابل معیار متناظر افزایش خواهد یافت، و بر عکس اگر استفاده از منبع مشترک زیاد ولی در لحظات متفاوتی از زمان باشد، این شباهت کاهش یافته و به تشخیص عدم تداخل کمک می‌کند.

برای تعیین برچسب تداخل یا عدم تداخل دو کاربرد در مدل دسته‌بند، از افت کارایی اجرای دو به دو آن‌ها استفاده می‌شود. به این صورت که اگر میزان افت کارایی (افزایش زمان اجرا) حداکثر بیشتر از ۲۰٪ زمان اجرای اختصاصی کاربردها باشد، این دو کاربرد متداخل با یکدیگر شناخته می‌شوند و با برچسب ۱ علامت‌گذاری می‌شوند، در غیر این‌صورت با برچسب ۰ علامت‌گذاری می‌شوند. داده‌های آموزش دسته‌بند را می‌توان با یک ماتریس دوبعدی نمایش داد که $n + \binom{n}{2}$ ردیف (تعداد انتخاب‌های دو کاربرد متفاوت یا دو کاربرد یکسان از n کاربرد) و ۱۶ ستون دارد (۱۵ ستون معادل ۱۵ معیار نمایه‌شده و ستون برچسب). برای مدل رگرسیون نیز تنها ستون برچسب با مقادیر واقعی افت کارایی جایگزین می‌شود.

در این پژوهش دو مدل رایج دسته‌بندی که در کاربردهای بسیاری دقت مناسبی داشته‌اند مورد آزمایش قرار گرفته‌اند: مدل جنگل تصادفی[۳۱] [۱۷] و

ماشین بردار پشتیبان[۳۲] [۱۸] دسته‌بندهایی هستند که نحوه آموزش و آزمون آن‌ها در بخش بعد توضیح داده خواهد شد. جنگل تصادفی یک روش دسته‌بندی ترکیبی است و زمانی که تعداد داده‌های آموزش کم است، دقت خوبی دارد [۱۹]. همچنین در این مقاله از دو مدل ترکیبی رگرسیون بگینگ[۳۳] [۲۰] و رگرسیون آدابوست[۳۴] [۲۱،۲۲] استفاده شده است و نتایج آن‌ها با یکدیگر مقایسه شده‌اند.

پارامترهای هر مدل با کمک مجموعه آموزش و جستجو بین مقادیر ممکن[۳۵] تنظیم شده است. در گام بعد میزان اهمیت هر معیار در بهتر تمیز دادن داده‌های دو کلاس بررسی شده است. در اغلب الگوریتم‌های یادگیری ماشین افزایش ویژگی[۳۶]ها منجر به بهبود کیفیت مدل می‌شود و بیشتر این الگوریتم‌ها از توانایی بهره‌برداری از هزاران و حتی میلیون‌ها ویژگی برخوردارند. بنابراین افزایش تعداد ویژگی‌های مرتبط می‌تواند منجر به بهبود کیفیت مدل شود. اما این امر لزوماً همیشه برقرار نیست. با افزایش تعداد ویژگی‌ها، مدل با جزئیات بیشتری مقدار متغیر هدف را محاسبه می‌کند. در نتیجه این خطر وجود دارد که بیش‌برازش[۳۷] رخ دهد و دقت مدل برای تشخیص داده‌های آزمون و دیده نشده در مجموعه آموزش کاهش یابد. در نتیجه لازم است تا با انتخاب بهترین ویژگی‌ها دقت مدل را افزایش دهیم. از طرف دیگر، کاستن تعداد ویژگی‌ها باعث افزایش سرعت در زمان آموزش و زمان آزمودن مدل می‌شود.

برای انتخاب ویژگی در این مقاله از روش حذف بازگشتی ویژگی با اعتبارسنجی متقابل[۳۸] استفاده شده است. این روش در مرحله اول با استفاده از یک الگوریتم یادگیر پایه با حذف هر کدام از n ویژگی یک مدل می‌سازد و با اعتبارسنجی متقابل کیفیت این مدل‌ها را محاسبه می‌کند. سپس اقدام به حذف ویژگی کم‌اهمیت‌تر (که متناظر با بیشترین امتیاز اعتبارسنجی متقابل است) می‌کند. در مراحل بعد همین عمل با $n-1$ ویژگی باقی‌مانده انجام می‌شود و به همین ترتیب ادامه می‌یابد. در نهایت بر اساس تمام دقت‌های به دست آمده، مجموعه ویژگی‌هایی که بیشترین دقت را تولید کرده‌اند انتخاب می‌شوند.

۳-۳- زمانبند آگاه از تداخل زمانی

هدف از این بخش ارائه زمان‌بندی است که با در نظر گرفتن تداخل، به گونه‌ای کاربردها را زمان‌بندی کند که کاربردهای محول شده به یک پردازنده گرافیکی، کمترین میزان تداخل را داشته باشند و به این ترتیب، ضمن استفاده بهینه و اشتراکی از پردازنده‌های گرافیکی، میزان مصرف انرژی نیز کاهش یابد. جایگاه این زمان‌بند در خوشه‌ای از پردازنده‌های گرافیکی، بر روی گره مرکزی (گره ورودی یا گره زمان‌بند) است.

برای انجام زمان‌بندی، فرض شده است کرنل‌ها به صورت دسته‌ای زمان‌بندی می‌شوند، بنابراین الگوریتم زمان‌بندی در هر بار اجرا با دسته‌ای از کرنل‌ها روبروست. مراحل ذکر شده برای آموزش مدل‌های دسته‌بند و رگرسیون یک‌بار به صورت برون خط بر روی بارهای کاری مجموعه داده آموزش انجام می‌شود. از آن پس می‌توان مدل‌های آموزش یافته را حین زمان‌بندی همه بارهای کاری استفاده کرد. برای این کار، ابتدا لیستی از کرنل‌هایی که قرار است اجرا شوند، در اختیار گره مرکزی قرار می‌گیرد. سپس هر کرنل یک‌بار نمایه‌سازی شده و نمایه‌اش ذخیره می‌شود. پس از تخصیص هر کرنل به پردازنده‌های گرافیکی نیز سطری در جدول نگاشت کاربردها به پردازنده‌های گرافیکی اضافه می‌گردد که مشخص می‌کند هر پردازنده در حال اجرای کدام کاربردها است. این سطر از جدول به محض اتمام اجرای کاربرد حذف می‌شود. برای درک بهتر نحوه کار زمان‌بند ارائه شده، شبه‌کد آن در شکل ۸ نشان داده شده است. در الگوریتم پیشنهادی، زمان‌بند مادامی که همه کرنل‌ها زمان‌بندی نشده‌اند به ترتیب زیر عمل می‌کند: ابتدا یک کرنل را از لیست کرنل‌های تازه‌وارد برداشته و سپس در صورت وجود یک پردازنده گرافیکی بیکار (در صورت داشتن منابع کافی) به آن تخصیص می‌دهد. در صورتی که همه پردازنده‌های گرافیکی مشغول اجرای کرنل باشند، با استفاده از مدل دسته‌بند برچسب تداخل اجرای همزمان کرنل ورودی با همه کرنل‌های در حال اجرا را به دست می‌آورد. سپس اگر پردازنده گرافیکی وجود داشت که کرنل در حال اجرای آن دارای برچسب عدم تداخل با کرنل ورودی باشد، به کرنل ورودی اختصاص می‌یابد. در غیر این صورت، الگوریتم به سراغ مدل رگرسیون می‌رود و این‌بار میزان افت کارایی اجرای همزمان کرنل ورودی را با همه کرنل‌های در حال اجرا را به دست می‌آورد. در این مرحله، اگر پردازنده گرافیکی وجود داشته باشد که کرنل در حال اجرای آن دارای میزان افت کارایی پیش‌بینی شده کمتر از آستانه باشد به کرنل ورودی تخصیص داده می‌شود. در غیر این صورت، الگوریتم زمان‌بندی کرنل ورودی را به تعویق انداخته و به سراغ کرنل دیگری از لیست کرنل‌های تازه‌وارد می‌رود.

```
Input: List of kernels and their profiles
Output: Kernel assignment
  K ← List of kernels with their profiles
  while K ≠ Ø do
    for all k_i in K do
      idle_GPU ← Get an idle GPU.
      if (idle_GPU ≠ Ø) then
        Assign k_i to idle_GPU.
      else
        L ← Use the classifier to obtain the interference labels of
            running k_i along with every running kernels
        j ← Index of the first zero label in L.
        if (j ≠ Ø) then
          Assign k_i to the GPU running k_j.
        else
          S ← Use the regression model to predict the slow
              down of running k_i along with every running kernels
          j ← Index of the first entry in S with SD < threshold
          if (j ≠ Ø) then
            Assign k_i to GPU_j.
          else
            Postpone k_i for the next assignment round.
          end if
        end if
      end if
    end for
  end while
```

شکل ۸: شبه کد الگوریتم زمان‌بندی پیشنهادی

۴- ارزیابی و نتایج

۴-۱- نحوه ارزیابی و بستر پیاده‌سازی

همان‌طور که اشاره شد، تمرکز این پژوهش بر روی کاربردهایی است که به زبان کودا نوشته شده باشند. جدول ۲ لیست کاربردهای محک مورد استفاده و مشخصات آن‌ها (محاسبه‌محور یا حافظه‌محور بودن) را نشان می‌دهد. این کاربردهای محک از بسته توسعه نرم‌افزار کودا[۳۹] استخراج شده‌اند. هفت کاربرد محک اصلی به گونه‌ای انتخاب شده‌اند که از نظر حافظه‌محور و محاسبه‌محور بودن تنوع لازم را داشته باشند. از طرفی هر کاربرد با سه اندازه ورودی متفاوت مورد استفاده قرار گرفته است تا علاوه بر اینکه مدل‌ها از تعداد داده کافی برخوردار باشند، تنوع کاربردها از نظر ابعاد نیز فراهم شود. برای ارزیابی روش پیشنهادی، از دو پردازنده گرافیکی GTX 1070 با معماری پاسکال و ظرفیت حافظه ۸ گیگابایت استفاده شده است. همچنین سیستم مورد استفاده شامل پردازنده مرکزی اینتل Core i7-9800X و ۱۶ گیگابایت حافظه است.

برای اجرای همزمان دو کاربرد بر روی یک پردازنده گرافیکی، می‌توان از جریان‌های کودا[۴۰] استفاده کرد. توابع و کرنل‌هایی که از یک زمینه هستند اما در جریان‌های متفاوتی قرار دارند، می‌توانند همزمان اجرا شوند. اما این روش نیاز به

تغییرات اساسی کد دارد که در روش پیشنهادی، سعی بر آن است تا کمترین تغییر در کد کاربر ایجاد شود. لذا برای اجرای همزمان کرنل‌ها، از سرویس چندروندی کودا (ام‌پی‌اس[۴۱]) استفاده شده است [۲۳]. این سرویس فراخوانی کرنل از دو زمینه متفاوت را در یک زمینه مشترک اما در جریان‌های متفاوت قرار می‌دهد و به پردازنده گرافیکی می‌فرستد. به این ترتیب کرنل‌هایی که از دو پروسه مختلف برای اجرا بر روی یک پردازنده گرافیکی ارسال می‌شوند می‌توانند همزمان اجرا شوند. از آنجا که سرویس ام‌پی‌اس تنها از سیستم‌عامل لینوکس پشتیبانی می‌کند، تمام مراحل پیاده‌سازی بر روی سیستم‌عامل ابونتو ۱۸/۰۴ انجام شده است. همچنین نسخه راه‌انداز انویدیا ۴۱۰ و نسخه راه‌انداز کوپتی ۱۰ بوده است.

جدول ۲: مشخصات کاربردهای محک استفاده شده

نام کاربرد	اندازه گرید/بلوک نخ	توضیح	کلاس
Interval	(۵۱۲،۱،۱)/(۴،۱،۱) (۲۵۶،۱،۱)/(۶،۱،۱) (۶۴،۱،۱)/(۱۰،۱،۱)	عملیات حسابی بازه‌ای	محاسبه‌محور
Sobol QRNG	(۱۲۸،۱،۱)/(۱،۱۵،۱) (۶۴،۱،۱)/(۱،۲۵،۱) (۳۲،۱،۱)/(۱،۱۲،۱)	تولید دنباله شبه تصادفی Sobol	
Binomial Options	(۶۴،۱،۱)/(۱،۱،۳۲) (۱۲۸،۱،۱)/(۱،۱،۳۲) (۱۲۸،۱،۱)/(۱،۱،۱۶)	تعیین قیمت بازخرید با مدل دوجمله‌ای	
Matrix Multiplication	(۲۸،۲۸،۱)/(۱،۱،۲) (۲۴،۲۴،۱)/(۱،۱،۲) (۳۲،۳۲،۱)/(۱،۲،۱)	محاسبه ضرب دو ماتریس دو بعدی	
Vector Add	(۱۰۲۴،۱،۱)/(۱،۱،۱) (۲۵۶،۱،۱)/(۴،۱،۱) (۵۱۲،۱،۱)/(۴،۱،۱)	محاسبه مجموع دو بردار	حافظه‌محور
Transpose Naive	(۱۶،۱۶،۱)/(۱،۲،۲) (۳۲،۳۲،۱)/(۱،۱،۱) (۸،۸،۱)/(۱،۴،۴)	محاسبه ترانهاده ماتریس به روش ساده	
Transpose Coalesced	(۱۶،۱۶،۱)/(۱،۲،۲) (۳۲،۳۲،۱)/(۱،۱،۱) (۸،۸،۱)/(۴،۴،۱)	محاسبه ترانهاده ماتریس به روش تلفیق شده	

۴-۲- ارزیابی ابزار نمایه‌سازی

نمایه‌ساز ارائه شده با زبان C++ پیاده‌سازی شده است. به منظور اعتبارسنجی عملکرد نمایه‌ساز زمانی، میانگین مقادیر سری زمانی به‌دست آمده از آن ابزار نمایه‌ساز انویدیا مقایسه شده است. این نتایج در جدول ۳ نشان داده شده است. همانطور که در این جدول مشاهده می‌شود، خطای نتایج نمایه‌ساز زمانی نسبت به نمایه‌ساز انویدیا حداکثر در حدود ۱۱٪ و به طور میانگین در حدود ۴/۵٪ است.

۴-۳- ارزیابی نتایج دسته‌بندی و رگرسیون

با دردست داشتن زمان اجرای هر کاربرد در دوحالت اجرای همزمان با یک کاربرد دیگر و اجرا به صورت اختصاصی، می‌توان میزان افت سرعت (SD) را با استفاده از معادله (۱) محاسبه کرد که در آن T_p زمان اجرای همزمان و T_s زمان اجرای تکی یک کرنل است.

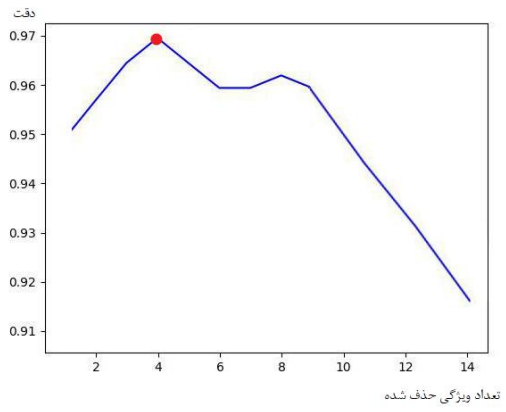
$$SD = {T_p}/{T_s} \quad (۱)$$

صورت معادله (۱) از میانگین ده بار زمان اجرای دو به دوی کاربردها و مخرج آن نیز از میانگین ده بار زمان اجرای اختصاصی آن‌ها به‌دست آمده است. چنانچه زمان اجرای یکی از کاربردها در اجرای همزمان بیش از ۲۰٪ زمان اجرای اختصاصی‌اش شده باشد، این دو کاربرد متداخل تشخیص داده می‌شوند و برچسب واقعی آن‌ها مقدار ۱ در نظر گرفته می‌شود. درصورتی که تعداد داده‌های یک برچسب نسبت به دیگری بسیار کم باشد لازم است مجموعه آموزش را متوازن نمود. بدین منظور، در این مقاله از روش بیش‌نمونه‌برداری تصادفی ساده[۴۲] استفاده شده است. این روش برای دسته‌ای که داده‌های کمتری دارد به اندازه دسته با داده بیشتر، از روی نمونه‌های موجود، به صورت تصادفی نمونه جدید می‌سازد. برای عادلانه بودن ارزیابی، بیش‌نمونه‌برداری فقط بر روی مجموعه داده آموزش (به غیر از بخش اعتبارسنجی) انجام شده است.

همانطور که در بخش ۳-۲ ذکر شد، برای انتخاب ویژگی در این مقاله از روش حذف بازگشتی ویژگی با اعتبارسنجی متقابل استفاده شده است. با توجه به شکل ۹ مشاهده می‌شود که دقت دسته‌بند با حذف ۳ ویژگی به بیشترین مقدار خود یعنی ۹۷٪ می‌رسد. سه ویژگی حذف شده، سه معیار dram_utilization، tex_cache_throughput و shared_store_throughput بوده‌اند.

جدول ۳: درصد خطای روش نمایه‌ساز زمانی نسبت به نمایه‌ساز انویدیا

معیار	نمایه‌ساز زمانی	نمایه‌ساز انویدیا	خطا (٪)
achieved_occupancy	۰/۴۹۹	۰/۴۹۷	۰/۵۱
dram_read_throughput	۱۱۱/۷۳۶	۱۱۲/۹۴۸	۳/۲۹۵
dram_read_transactions	۱۹۵۸۵	۲۱۰۹۵	۷/۱۵۸
dram_utilization	۱	۱	۰
dram_write_throughput	۲۷۹	۲۶۷	۴/۴۹۴
dram_write_transactions	۱۷	۱۹	۱۰/۵۲۶
gld_throughput	۷/۲۱۸	۷/۸۲	۷/۷۰۷
gst_throughput	۲۳۶/۷۲۸	۲۱۲/۸۵	۱۱/۲۱۵
IPC	۰/۵۳۵	۰/۵۳۴	۰/۲۳۹
issue_slot_utilization	۱۳/۰۶	۱۳/۰۵۱	۰/۰۶۶
l2_read_transactions	۷۴۲۳۴۰۲۹۶۸۰	۸۰۱۵۶۰۳۷۹۲۱	۷/۳۸۸
l2_write_transactions	۳۶۹۴۲۷۵۹۶۱۲	۴۰۹۶۰۰۰۲۶۳۱	۹/۸۰۷
sm_efficiency	۳۰/۹۵۳	۳۰/۹۳	۰/۰۷۶
tex_cache_throughput	۳۷۳/۸۲۲۹۳	۳۵۳/۳۸۹۳۵	۵/۷۸۲
shared_store_transactions	۰	۰	۰



شکل ۹: بیشترین دقت دسته‌بند با حذف ۳ ویژگی غیرمهم

در انتها با روش اعتبارسنجی متقابل ۵ بخشی[۴۳] [۲۴،۲۵]، به آزمودن هر یک از مدل‌های دسته‌بند پرداخته شده است. در این پژوهش از دو مدل دسته‌بند ماشین بردار پشتیبان به عنوان یک دسته‌بند پرکاربرد ساده و جنگل تصادفی به عنوان یک دسته‌بند ترکیبی استفاده شده است و نتایج آن‌ها با همدیگر مقایسه شده است. نتایج حاصل (معیار F-measure و صحت[۴۴]) در جدول ۴ آورده شده است. با توجه به این‌که دسته‌بند جنگل تصادفی در مجموع دارای عملکرد بهتری

نسبت به ماشین بردار پشتیبان بوده است، در زمان‌بند پیشنهادی از دسته‌بند جنگل تصادفی استفاده شده است.

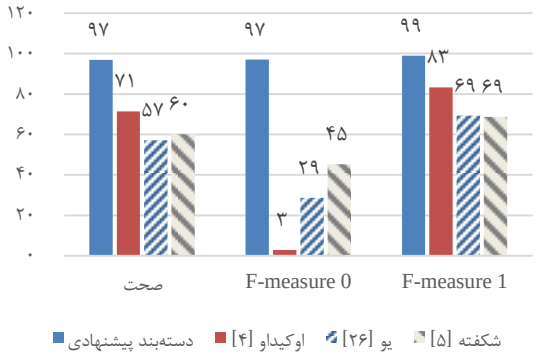
جدول ۴: مقایسه عملکرد دو دسته‌بند جنگل تصادفی و بردار پشتیبان

معیار	ماشین بردار پشتیبان	جنگل تصادفی
معیار F-measure برای کلاس ۰	٪۹۲/۴	٪۹۷
معیار F-measure برای کلاس ۱	٪۹۱/۲	٪۹۸/۹
صحت	٪۹۱/۸	٪۹۶/۹

عملکرد دسته‌بند جنگل تصادفی پیشنهادی با سه روش تشخیص تداخل پیشین نیز مقایسه شده است:

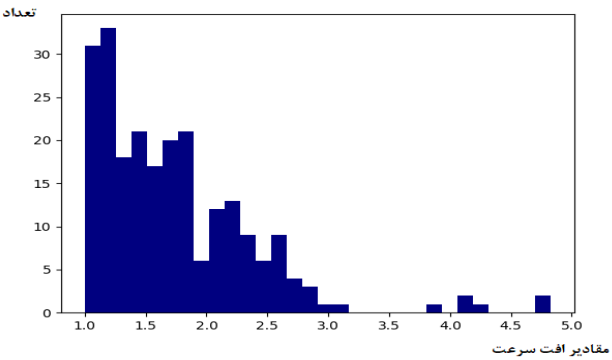
- شکفته و همکاران [۵] که کرنل‌هایی که هر دو محاسبه‌محور یا هر دو حافظه‌محور هستند را متداخل فرض کرده و با هم اجرا نمی‌کند.
- اوکیداو و همکاران [۴] که دو کرنل را متداخل می‌داند اگر شباهت حداقل ۳ معیار از کرنل‌ها بیشتر از ٪۴۰ باشد.
- یو [۲۶] که دو کرنل را متداخل می‌داند اگر میانگین افت کارایی اجرای آن‌ها با دیگر کرنل‌ها حداقل ٪۲۰ باشد.

نتایج مقایسه در شکل ۱۰ آورده شده است و نشان می‌دهد دسته‌بند ارائه شده دقت بسیار بالاتری نسبت به روش‌های مذکور دارد.



شکل ۱۰: مقایسه دسته‌بند پیشنهادی و روش‌های تشخیص تداخل پیشین

در مرحله آزمون دو مدل رگرسیون استفاده شده در این مقاله یعنی بگینگ و آدابوست از روش کنارگذاری یک نمونه استفاده شده است. به عنوان معیار ارزیابی این دو مدل معیار میانگین مربعات خطا استفاده شده است. خطای میانگین مربعات روشهای بگینگ و آدابوست به ترتیب ۰/۲۲۹ و ۰/۲۳۴ بوده است. هیستوگرام مقادیر افت کارایی در شکل ۱۱ نشان داده شده است. همان‌طور که مشاهده می‌شود، خطای هر دو مدل نسبت به مقادیر افت سرعت، مقدار قابل قبولی است. در ادامه کار از مدل بگینگ استفاده شده است که خطای کمتری نسبت به مدل آدابوست دارد.



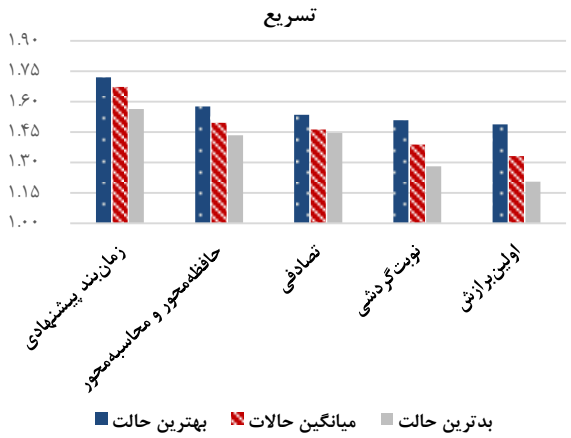
شکل ۱۱: هیستوگرام مقادیر افت کارایی

۴-۴- ارزیابی نتایج زمان‌بند

در این بخش به ارزیابی روش زمان‌بندی پیشنهای که از مدل‌های دسته‌بند جنگل تصادفی و رگرسیون بگینگ استفاده می‌کند می‌پردازیم. برای ارزیابی این زمان‌بند از دو پردازنده گرافیکی یکسان GTX 1070 استفاده شده است. در این پژوهش برای سادگی فرض شده است که حداکثر دو کاربرد می‌توانند با یکدیگر بر روی یک پردازنده گرافیکی اجرا شوند.

به این منظور هر بار یک لیست ۲۰تایی از میان کاربردهای محک موجود به صورت تصادفی انتخاب شده است. هنگام اعزام هر کاربرد برای اجرا، ابتدا برچسب پیش‌بینی شده آن وکاربردهای درحال اجرا بررسی شده و در صورت وجود پردازنده گرافیکی مناسب، به آن تخصیص می‌یابد. در غیر این‌صورت مقدار افت سرعت ناشی از اجرای هم‌زمان کاربرد منتخب با کاربردهای در حال اجرا از نتایج مدل رگرسیون استخراج شده و کمینه آن‌ها با یک مقدار حد آستانه مقایسه می‌شود. اگر کمینه افت سرعت از حد آستانه کمتر باشد، کار به آن پردازنده تخصیص می‌یابد. در غیر این صورت کاربرد دیگری از لیست انتخاب می‌شود. حد آستانه با توجه به شکل ۱۱ مقدار ۱/۹ در نظر گرفته شده است، به این معنی که زمان‌بند در بدترین حالت، افت سرعت کمتر از ۹۰ درصد زمان اجرای اختصاصی را می‌پذیرد تا بتواند ضمن حفظ کارایی، حتی‌الامکان از اجرای انفرادی یک کرنل بر روی یک پردازنده گرافیکی خودداری کند.

عملکرد زمان‌بند پیشنهادی با سه روش زمان‌بندی بی‌توجه به تداخل یعنی روش‌های اولین برازش، نوبت گردشی و تخصیص تصادفی و یک روش آگاه از تداخل که در پژوهش [۵] ارائه شده است، از منظر زمان بازگشت مقایسه شده است. زمان بازگشت معادل مدت زمان بین اعزام اولین کرنل تا اتمام اجرای آخرین کرنل در نظر گرفته شده است. لازم به ذکر است که مراحل فوق برای ده لیست تصادفی متفاوت از کرنل‌ها با ده جایگشت مختلف هر لیست تکرار شده و میانگین زمان بازگشت آن‌ها در سه حالت بهترین، بدترین و میانگین زمان بازگشت هر جایگشت، به عنوان زمان بازگشت نهایی گزارش شده است. نتایج حاصل از مقایسه روش‌های زمان‌بندی ذکر شده در قالب شاخص تسریع (نسبت زمان بازگشت هر کدام به میانگین زمان بازگشت اجرای سریال) در شکل ۱۲ آورده شده است.



شکل ۱۲: مقایسه تسریع حاصل از زمان‌بندی کارها با استفاده از زمان‌بند پیشنهادی با روش‌های دیگر

مشاهده می‌شود زمان‌بند پیشنهادی به طور میانگین ٪۶۷ نسبت به حالت سریال، بیش از ٪۱۷ نسبت به زمان‌بند آگاه از تداخل پژوهش [۵] و بیش از ٪۲۷ نسبت به زمان‌بندهای بی‌توجه به تداخل تسریع داشته است. به‌علاوه تسریع حاصل از زمان‌بند پیشنهادی در حالت میانگین نزدیک به بهترین حالت خود است. همچنین فاصله بدترین حالت و میانگین نیز کم است. این امر نشان می‌دهد که روش ارائه شده می‌تواند در اکثر موارد موثر واقع شود. روش آگاه از تداخل

حافظه‌محور-محاسبه‌محور [۵] اگرچه تداخل حین اجرای هم‌زمان را کاهش می‌دهد اما به دلیل سخت‌گیری در انتخاب جفت کاربردها، منجر به اجرای اختصاصی کاربردهای زیادی می‌شود. لذا نه تنها تسریع کمتری می‌گیرد، بلکه بهره‌وری از پردازنده گرافیکی را نیز به خوبی افزایش نمی‌دهد.

به دلیل مصرف زیاد انرژی در پردازنده‌های گرافیکی، صرفا کاهش زمان بازگشت، جز در کاربردهای بی‌درنگ، برای یک روش زمان‌بندی کافی نیست و نیاز است با جلوگیری از هدررفت منابع پردازنده گرافیکی در مصرف انرژی نیز صرفه جویی نماید. از آنجا که زیرساخت‌های مجهز به پردازنده‌های گرافیکی با چالش مصرف زیاد انرژی این پردازنده‌ها مواجه هستند، ارائه راه‌حلی در راستای کاهش انرژی مصرفی پردازنده‌های گرافیکی، می‌تواند بسیار موثر واقع شود. برای ارزیابی میزان انرژی مصرفی زمان‌بند ارائه شده و مقایسه آن با روش‌های پیشین، از توان مصرفی هریک از پردازنده‌های گرافیکی حین اجرای زمان‌بندهای مذکور با نرخ هر ۱۰ میلی‌ثانیه یک‌بار نمونه‌برداری شده است. در نهایت میزان انرژی مصرفی با استفاده از معادله (۲) محاسبه شده است.

$$E = \sum_{i=0}^{n} P_i \cdot t_i \qquad (۲)$$

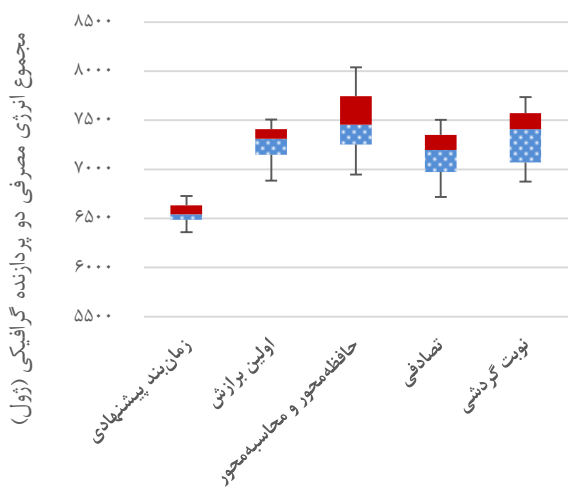
نتایج حاصل از اندازه گیری میزان مصرف انرژی بستر مورد استفاده در جدول ۵ ارائه شده است. مشاهده می‌شود که زمان‌بند پیشنهادی ۲۶٪ نسبت به اجرای سریال، بیش از ۸٪ نسبت به زمان‌بندهای ناآگاه از تداخل و بیش از ۱۰٪ نسبت به به‌روزترین زمان‌بند آگاه از تداخل ارائه شده در پژوهش [۵]، در مصرف انرژی صرفه‌جویی می‌کند.

با توجه به نمودار جعبه‌ای شکل ۱۳، مشاهده می‌شود که نه تنها میزان انرژی مصرفی زمان‌بند پیشنهادی نسبت به سایر زمان‌بندها کمتر است، بلکه پراکندگی مقادیر انرژی مصرفی نیز ناچیز و در اکثر موارد نزدیک به میانگین است. مصرف زیاد انرژی در زمان‌بند حافظه‌محور-محاسبه‌محور به این دلیل است که پژوهش‌گران در این پژوهش توجهی به میزان انرژی مصرفی نداشته‌اند و همان‌طور که در بخش قبل اشاره شد، تلاشی برای کاهش مواردی که یک کاربرد به تنهایی روی پردازنده گرافیکی اجرا می‌شود، نشده است. پراکندگی مقدار مصرف انرژی نیز در این روش به دلیل وابسته بودن انتخاب درست به محل قرارگیری کاربردها در صف است.

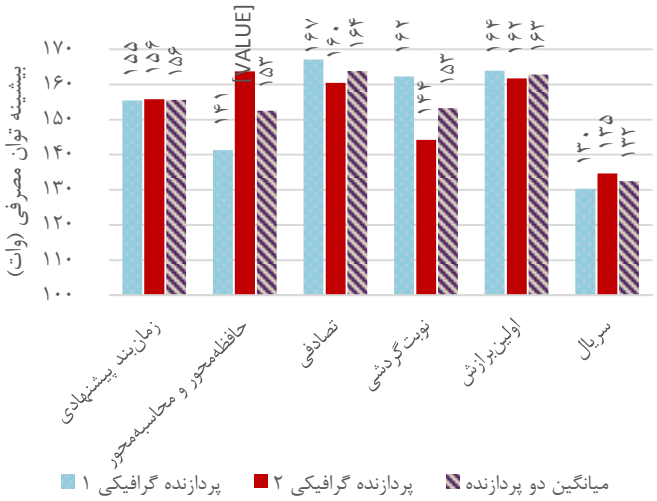
همانطور که در نمودار شکل ۱۴مشاهده می‌شود، بیشینه توان در روش سریال از سایر روش‌ها کمتر است که این امر به دلیل بیکار بودن بخشی از پردازنده گرافیکی است که منجر به اتلاف قابلیت‌های آن و بهره‌وری پایین می‌شود. اما در روش پیشنهادی ضمن استفاده حداکثری از قابلیت پردازنده‌های گرافیکی، بیشینه توان نیز از حد قابل قبول تجاوز نمی‌کند. به طوری که بیشینه توان مصرفی از روش‌های تصادفی و اولین برازش کمتر و میانگین آن در حد دو روش حافظه‌محور-محاسبه‌محور و نوبت گردشی حفظ شده است. لذا در حین اجرای برنامه‌های کاربردی، هوشمندی روش پیشنهادی منجربه بهره‌وری حداکثری از پردازنده گرافیکی و کاهش زمان بازگشت کاربردها و درنتیجه، با توجه به معادله ۲، منجر به صرفه جویی در مصرف انرژی شده است.

جدول ۵: انرژی مصرفی پردازنده‌های گرافیکی با زمان‌بندی به روش‌های مختلف

زمان‌بند	میانگین انرژی مصرفی (کیلو ژول)	درصد ذخیره انرژی نسبت به حالت سریال
زمان‌بند پیشنهادی	۶/۵۵۵	۲۶/۰۳
تصادفی	۷/۱۶۷	۱۹/۱۲
اولین برازش	۷/۲۹	۱۷/۷۴
نوبت گردشی	۷/۳۴۲	۱۷/۱۵۳
حافظه‌محور و محاسبه‌محور	۷/۵۲۱	۱۵/۱۳
سریال	۸/۸۶۳	-



شکل ۱۳: نمودار جعبه‌ای مصرف انرژی پردازنده‌های گرافیکی با زمان‌بندهای مختلف



شکل ۱۴: مقایسه بیشینه توان مصرفی هر دو پردازنده‌های گرافیکی با زمان‌بندهای مختلف

۵- جمع‌بندی

تابه‌حال در پژوهش‌های پیشین روش‌های متعددی برای تشخیص تداخل اجرای همزمان کرنل‌ها در پردازنده‌های گرافیکی معرفی شده است که برخی مبتنی بر تکنیک‌های یادگیری ماشین بوده‌اند. منتهی در این روش‌ها اغلب از معیارهای میانگین مصرف منابع برای تخمین و تشخیص تداخل استفاده شده است. در این مقاله، روشی برای تشخیص تداخل اجرای همزمان کرنل‌ها بر روی پردازنده‌های گرافیکی معرفی شد که مبتنی بر نمایه‌سازی زمانی الگوی مصرف منابع کرنل‌ها و اندازه‌گیری میزان مشابهت معیارهای زمانی مستخرج از آن است. سپس برای تشخیص تشابه الگوهای مصرف از شاخص همبستگی متقابل استفاده شد. همچنین یک مدل دسته‌بند جنگل تصادفی برای تشخیص تداخل یا عدم تداخل کرنل‌ها و یک مدل رگرسیون برای تخمین میزان افت کارایی اجرای همزمان کرنل‌ها معرفی شد. در نهایت روشی برای زمان‌بندی کرنل‌ها در خوشه‌های مبتنی بر پردازنده‌های گرافیکی با استفاده از مدل‌های یادگیری ماشین معرفی شده ارائه شد.

نتایج ارزیابی روش‌ها و مدل‌های معرفی شده در این مقاله با کارهای مرتبط پیشین نشان می‌دهد که استفاده از زمان‌بند پیشنهادی می‌تواند منجر به تسریع

بیش از ۱۷٪ نسبت به به‌روزترین زمان‌بند آگاه از تداخل پیشین و بیش از ۲۷٪ نسبت به زمان‌بندهای ناآگاه از تداخل شود. از طرفی پژوهش‌های بررسی شده در حوزه زمان‌بندهای آگاه از تداخل، هیچکدام انرژی مصرفی را در نظر نگرفته‌اند. نتایج همچنین نشان می‌دهد زمان‌بند پیشنهادی ۲۶٪ نسبت به اجرای سریال، بیش از ۸٪ نسبت به زمان‌بندهای ناآگاه از تداخل و بیش از ۱۰٪ نسبت به به‌روزترین زمان‌بند آگاه از تداخل، در مصرف انرژی صرفه‌جویی می‌کند.

با وجودی که مدل‌های پیشنهادی از دقت قابل قبولی برخوردار بوده‌اند، منتهی آزمایش‌های صورت گرفته تنها به استفاده از یک نوع پردازنده گرافیکی محدود بوده است. به منظور افزایش دقت و جامعیت روش پیشنهادی می‌توان نمایه‌سازی کرنل‌ها را بر روی معماری‌های مختلف پردازنده گرافیکی انجام داده و اثر تغییرات معماری خانواده‌های مختلف پردازنده گرافیکی بر تشخیص تداخل را مطالعه نمود. بر این اساس می‌توان در کارهای آتی مشخصات معماری پردازنده گرافیکی را نیز جهت تشخیص دقیق‌تر تداخل لحاظ کرد که این مورد از محدوده این پژوهش خارج بوده است.

۶- مراجع

[1] S. Mittal and J. S. Vetter, "A Survey of Methods for Analyzing and Improving GPU Energy Efficiency." *Computing Surveys*, vol. 47, no. 2, pp. 1-23, 2015.

[2] Q. Xu, H. Jeon, K. Kim, W. W. Ro, and M. Annavaram, "Warped-slicer: Efficient Intra-SM Slicing Through Dynamic Resource Partitioning for GPU Multiprogramming," *International Symposium on Computer Architecture*, pp. 230-242, 2016.

[3] N.R. RamMohan and E. Baburaj, "Resource Allocation Using Interference Aware Technique in Cloud Computing Environment," *Journal of Digital Content Technology and its Applications*, vol. 8, no.1, pp. 35-46, 2014.

[4] Y. Ukidave, X. Li, and D. Kael, "Mystic: Predictive Scheduling for GPU-Based Cloud Servers Using Machine Learning," *Parallel and Distributed Processing Symposium*, pp. 353-362, 2016.

[5] S. Shekofteh, H. Noori, M. Naghibzadeh, H. Sadoghi Yazdi, and H. Froning, "Metric Selection for GPU Kernel Classification," *Transactions on Architecture and Code Optimization*, vol. 15, no. 4, pp. 1-27, 2019.

[6] R. Phull, C. Li, K. Rao, H. Cadambi, and S. Chakradhar, "Interference-Driven Resource Management for GPU-Based Heterogeneous Clusters." *International Symposium on High-Performance Parallel and Distributed Computing*, pp. 109-120, 2012.

[7] CUDA Toolkit 10.1, https://developer.nvidia.com/cuda-downloads/, July 2021.

[8] Z. Xu, F. Dong, J. Jin, J. Luo, and J. Shen, "GScheduler: Optimizing Resource Provision by Using GPU Usage Pattern Extraction in Cloud Environment," *International Conference on Systems, Man and Cybernetics*, pp. 3225-3230, 2017.

[9] Q. Zhu, B. Wu, X. Shen, L. Shen, and Z. Wang, "Co-Run Scheduling with Power Cap on Integrated CPU-GPU Systems," *International Parallel and Distributed Processing Symposium*, pp. 967-977, 2017.

[10] A. Hugo, A. Guermouche, P. Wacrenier, and R. Namyst, "Composing Multiple StarPU Applications over Heterogeneous Machines: a Supervised Approach," *International Symposium on Parallel and Distributed Processing*, pp. 1050-1059, 2013.

[11] P. Aguilera, J. Lee, A. Farmahini-Farahani, K. Morrow, M. Schulte, and N. S. Kim, "Process Variation-aware Workload Partitioning Algorithms for GPUs Supporting Spatial-Multitasking," *Design, Automation and Test in Europe Conference and Exhibition*, pp. 1-6, 2014.

[12] A. Pattnaik, X. Tang, A. Jog, O. Kayiran, A. K. Mishra, M. T. Kandemir, O. Mutlu, and C. R. Das, "Scheduling Techniques for GPU Architectures with Processing-in-Memory Capabilities," *International Conference on Parallel Architectures and Compilation*, pp. 31–44, 2016.

[13] A. Karami, F. Khunjush, and S. A. Mirsoleimani, "A Statistical Performance Analyzer Framework for OpenCL Kernels on NVIDIA GPUs," *Journal of Supercomputing*, vol. 71, no. 8, pp. 2900–2921, 2015.

[14] X. Mei, X. Chu, H. Liu, Y. Leung, and Z. Li, "Energy Efficient Real-Time Task Scheduling on CPU-GPU Hybrid Clusters," *IEEE Conference on Computer Communications*, pp. 1-9, 2017.

[15] V. Chau, X. Chu, H. Liu, and Y. W. Leung, "Energy Efficient Job Scheduling with DVFS for CPU-GPU Heterogeneous Systems," *International Conference on Future Energy Systems*, pp. 1-11, 2017.

[16] T.R. Derrick, and J.M. Thomas, *Time Series Analysis: The Cross-Correlation Function*. In Innovative Analyses of Human Movement, pp. 189-205, 2004.

[17] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[19] M. Mohandes, M. Deriche, and S. O. Aliyu, "Classifiers Combination Techniques: A Comprehensive Review," *IEEE Access,* vol. 6, pp. 19626-19639., 2018.

[23] CUDA Multi-Process Service (MPS), https://docs.nvidia.com/, July 2021.

[18] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.

[24] E. Alpaydin, *Introduction to Machine Learning*, MIT Press, 2004.

[25] A Gentle Introduction to k-fold Cross-Validation, https://machinelearningmastery.com/k-fold-cross-validation, July 2021.

[20] T. Hastie, *Elements of Statistical Learning*, Springer Science & Business Media, 2009.

[21] A. Sharafati, S. B. H. S. Asadollah, and M. Hosseinzadeh, "The Potential of New Ensemble Machine Learning Models for Effluent Quality Parameters Prediction and Related Uncertainty," *Process Safety and Environmental Protection*, vol. 140, pp. 68-78, 2020.

[22] H. Drucker, "Improving Regressors using Boosting Techniques," *International Conference on Machine Learning*, vol. 97, pp. 107-115, 1997.

[26] L. Yu, *Multilevel Interference-aware Scheduling on Modern GPUs*. PhD Thesis, April 2019.

نگار سادات علیزاده فارغ‌التحصیل رشته معماری سیستم‌های کامپیوتری از دانشگاه امیرکبیر است. ایشان دوره کارشناسی خود را در رشته مهندسی کامپیوتر-نرم‌افزار در دانشگاه خواجه نصیرالدین طوسی گذرانده و چندین سال سابقه فعالیت در صنعت به عنوان توسعه دهنده نرم‌افزار را دارد. علاقه او به حوزه یادگیری ماشین وی را در مسیر تحقیق بیشتر در این حوزه قرار داد که در نهایت به استفاده از الگوریتم‌های یادگیری ماشین برای ارائه زمان‌بند در سیستم‌های مبتنی بر پردازنده‌های گرافیکی انجامید. آدرس پست الکترونیکی ایشان عبارت است از:

Negar.alizadeh@aut.ac.ir

محمود ممتازپور عضو هیأت علمی دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر است. تحصیلات ایشان در رشته مهندسی برق در سه مقطع کارشناسی، کارشناسی ارشد و دکتری در دانشکده مهندسی برق دانشگاه صنعتی شریف بوده و در سال ۱۳۹۳ به دانشگاه صنعتی امیرکبیر پیوسته است. ایشان همچنین در حال حاضر در حال همکاری با مرکز تحقیقات پژوهش‌های فوق سریع دانشگاه صنعتی امیرکبیر می‌باشد. حوزه تحقیقاتی وی شامل سیستم‌های موازی و ابری، اینترنت اشیاء، پردازنده‌های گرافیکی و ابررایانه‌ها است. آدرس پست الکترونیکی ایشان عبارت است از:

momtazpour@aut.ac.ir

1. Application
2. Temporal Pattern
3. Kernel
4. Profiler
5. CUPTI API
6. NVIDIA
7. Metric
8. Profile
9. Classifier
10. Regression
11. Offline
12. Collaborative Filtering
13. Throughput
14. Extension
15. Scheduling contexts
16. Hypervisor
17. Dynamic voltage and frequency scaling (DVFS)
18. Dynamic Resource Sleep (DRS)
19. Online
20. Cluster
21. Bin-Packing
22. Recommender System
23. Performance Counter
24. Node
25. CUDA Context
26. Event API
27. Callback API
28. Cycle
29. Time Slot
30. Cross-Correlation
31. Random Forest
32. Support Vector Machine
33. Bagging
34. Adaboost
35. Grid Search
36. Feature
37. Overfit
38. Recursive Feature Elimination, Cross-Validated (RFECV)
39. CUDA Software Development Kit (SDK)
40. CUDA Streams
41. CUDA Multi-Process Service (MPS)
42. Naive Random Oversampling
43. 5-Fold Cross Validation
44. Accuracy

The CSI Journal on Computing Science and Information Technology
Vol. 19, No.1, 2021

Temporal Interference-aware Scheduling for GPU-based Systems

Negar Sadat Alizadeh[1], Mahmoud Momtazpour[2]

[1,2] Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran

Abstract

Recent advancements in graphics processing unit (GPU) architectures allow concurrent kernel execution on a single GPU, to avoid under-utilization of its resources. However, this may cause resource contention and performance degradation if both kernels try to access a shared resource simultaneously. There have been many studies in the field of interference prediction in concurrent kernel execution. However, predicting the interference based on the temporal pattern of resource usage was not investigated. This paper proposes a machine learning-based approach to predict temporal interference in concurrent kernel execution using temporal resource usage profile of kernels. In addition, a heuristic kernel scheduling method is proposed to use this prediction model to improve GPU utilization and reduce energy consumption. Experimental results on real-world benchmarks show that the proposed method, with respect to speed-up, outperforms serial scheduling, previous interference-aware scheduling, and interference-oblivious scheduling approaches by 67%, 17%, and 27% respectively, and also improves energy consumption by 26%, 10% and 8% respectively.

Keywords: Concurrent Kernel Execution, GPU, Temporal Interference, Machine Learning.