علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۸، شماره ۲، ۱۳۹۹
صفحات ۱۰۵- ۱۱۵
مقاله عادی

نشریه علوم رایانش و فناوری اطلاعات

بهبود کارایی پردازنده‌های گرافیکی از طریق به کارگیری مناسب کنترل‌کننده‌های حافظه

هژیر باخویشی[۱]، نگار اکبرزاده[۲]، سید محمد صدرالساداتی[۳*]، حمید سربازی‌آزاد[۴،۵]

[*]نویسنده مسئول، دریافت: ۱۴۰۰/۰۲/۱۴، بازنگری: ۱۴۰۰/۰۲/۲۱، پذیرش: ۱۴۰۰/۰۳/۰۱

[۱] دانش‌آموخته کارشناسی ارشد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی‌شریف، تهران، ایران
[۲] دانشجوی دکترا، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی‌شریف، تهران، ایران
[۳] پژوهشگر پسادکترا، پژوهشگاه دانش‌های بنیادی، تهران، ایران
[۴] استاد، دانشکده مهندسی کامپیوتر ، دانشگاه صنعتی شریف، تهران، ایران
[۵] استاد پژوهشگر، پژوهشگاه دانش‌های بنیادی، تهران، ایران

چکیده

امروزه از پردازنده‌های گرافیکی به عنوان بستری مناسب برای پردازش موازی همه منظوره استفاده می‌شود. روند پیکربندی پردازنده‌های گرافیکی در طول سال‌های گذشته به این شکل بوده است که همواره با افزایش ظرفیت پردازشی و زیاد شدن منابع رو تراشه‌ی پردازنده گرافیکی ، پهنای باند آن نیز افزایش می‌یابد. در نگاه اول این نسبت مستقیم بین قدرت پردازنده‌ی گرافیکی و پهنای باند آن کاملاً منطقی به نظر می‌رسد. چرا که بارهای کاری به نسبت مقدار داده‌ای که از حافظه می‌خوانند، پردازش انجام می‌دهند. اما با نگاهی عمیق‌تر به بارهای کاری مختلف مشاهده می‌کنیم که برای تعداد قابل توجهی از بارهای کاری میزان بهره‌وری پهنای باند بسیار پایین است. در حالی که بهره‌وری هسته‌های پردازشی و منابع روی تراشه‌ی پردازنده‌ی گرافیکی همچنان بالا است. بنابراین، برای همه‌ی بارهای کاری رعایت نسبت مستقیم بین قدرت پردازنده‌های گرافیکی و پهنای باند آن‌ها لازم نیست. در این پژوهش، ابتدا بر مبنای مشاهدات مذکور رویکرد جدیدی برای طراحی پردازنده‌های گرافیکی با قدرت پردازشی بالا و پهنای باند کمتر ارائه می‌شود و در ادامه، فرصت‌های برآمده برای استفاده از مساحت به دست آمده از کاهش پهنای باند، بررسی شده و روش‌های مناسب استفاده از این مساحت ارائه می‌شود.

کلمات کلیدی: پردازنده گرافیکی عام‌منظوره، پهنای باند، ظرفیت پردازشی، افزایش کارایی.

۱- مقدمه

در گذشته‌ی نه چندان دور، برنامه‌نویسی موازی به عنوان موضوعی ناشناخته و جدا از علم کامپیوتر مورد مطالعه قرار می‌گرفت. این خط مشی در سال‌های اخیر به کلی تغییر کرد به گونه‌ای که در سال ۲۰۱۰ تقریباً همه‌ی کاربران کامپیوترهای شخصی از سیستم‌هایی استفاده می‌کردند که دارای پردازنده‌های چند هسته‌ای بودند. در ادامه‌ی سیر فرآیند چند هسته‌سازی و بهره‌گیری هر چه بیشتر از مزیت‌های آن، پردازنده‌های گرافیکی مورد توجه قرار گرفتند و واحدهای پردازش گرافیکی برای بارهای کاری با قابلیت موازی‌سازی بالا به منظور افزایش کارایی طراحی شدند [۱].

پردازنده‌های گرافیکی به دو صورت نخ[۱]ها را موازی می‌کنند. اول آن‌که نخ‌هایی که کد یکسانی اجرا می‌کنند را در داخل یک دسته با نام ریسمان[۲] قرار می‌دهند. این ریسمان‌ها روی هسته‌های پردازشی قرار گرفته و از قابلیت تک دستور-چند داده[۳] استفاده می‌کنند. نخ‌های داخل ریسمان به طور موازی بر روی هسته‌های پردازشی اجرا می‌شوند. دوم آن که پردازشگرهای گرافیکی تعداد زیادی ریسمان را روی یک چندپردازنده‌ی جریانی[۴] اجرا می‌کنند، به این صورت که هرگاه

یک ریسمان به هر دلیلی متوقف شد، ریسمان دیگری برای اجرا زمان‌بندی می‌شود. با این روش، مشکلات وابستگی داده، شرط‌های پرش و تأخیر عملوندها پوشش داده می‌شود. اما با تمام این فنون، پردازنده‌های گرافیکی هم‌چنان نمی‌توانند از منابع محاسباتی به صورت بهینه استفاده کنند [۳،۲].

در سال‌های گذشته، همواره شرکت‌های تولید کننده‌ی پردازنده‌های گرافیکی و در صدر آن‌ها شرکت NVIDIA، در طراحی و پیکربندی پردازنده‌های گرافیکی از یک خط مشی مشخص پیروی کرده‌اند. به این معنا که همواره ظرفیت پردازشی پردازنده‌ی گرافیکی با ظرفیت پهنای باند آن نسبت مستقیم دارد. به عبارت دیگر، هرچه منابع پردازشی و ذخیره‌ای روی تراشه‌ی پردازنده‌های گرافیکی افزایش یابد، پهنای باند نیز به دنبال آن بیشتر می‌شود. دلیل این امر این است که برنامه‌های کاربردی به ازای مقدار داده‌ای که از حافظه می‌خوانند، محاسبات انجام می‌دهند و در نتیجه منابع پردازشی را درگیر محاسبات خود می‌کنند. به بیان دیگر، پردازنده‌های گرافیکی حول یک محور طراحی می‌شوند و آن این است که "پردازش بیشتر نیاز به پهنای باند بزرگتری دارد" [۵،۴]. اگرچه این رویکرد طراحی برای بیشتر بارهای کاری قابل توجیه است، اما برای آن دسته از بارهای کاری که تعداد دسترسی به حافظه‌ی بسیار کمتری در مقایسه با تعداد دستورهای محاسباتی دارند، به دلیل بهره‌وری پایین پهنای باند، منجر به کاهش کارایی می‌شود. این بارهای کاری را از این پس CILMA[۵] می‌نامیم.

باتوجه به مسائل عنوان شده، گلوگاه پردازنده‌ی گرافیکی برای بارهای کاری CILMA، به هیچ وجه پهنای باند پردازنده‌ی گرافیکی نیست. بلکه مانع اصلی رسیدن به حداکثر ظرفیت پردازشی این برنامه‌ها کمبود منابع پردازشی و ذخیره‌سازی اطلاعات روی تراشه است. نتایج شبیه‌سازی‌ها نشان می‌دهد که برای این بارهای کاری، کاهش پهنای باند تاثیری بر روی کارایی پردازنده‌های گرافیکی ندارد. از این رو، نیاز به مدلی از پردازنده‌های گرافیکی که دارای ظرفیت پردازشی معادل و یا حتی بیشتر از پردازنده‌های گرافیکی پر قدرت و پهنای باند کمتر باشند، کاملاً مشهود است. کاهش پهنای باند نتایجی به دنبال خواهد داشت که هرکدام از آن‌ها فرصت‌های جدیدی برای بهبود کارایی مدل پردازنده‌ی گرافیکی ارائه شده را به همراه دارد. این نتایج عبارتند از:

۱- کاهش هزینه‌ی ساخت پردازنده‌ی گرافیکی نسبت به پیکربندی قبلی با ظرفیت پردازشی زیاد و پهنای باند بالا

۲- کاهش توان مصرفی

۳- ایجاد فضای خالی بر روی تراشه (هرکدام از کانال‌های حافظه‌ی بیرون تراشه، در داخل تراشه به یک واحد کنترل‌کننده‌ی حافظه متصل است که در صورت حذف هر کانال، کنترل کننده‌ی متناظر نیز حذف خواهد شد)

چالش اصلی این پژوهش، مشخص کردن نحوه‌ی بهره‌وری از مساحت بدست آمده است، با این هدف که بیشترین مقدار کارایی حاصل شود. برای حصول این هدف، نقطه ضعف و قوت پردازنده‌های گرافیکی آماج بررسی‌های ما قرار گرفته است. ساختار پردازنده‌های گرافیکی به صورتی طراحی شده است که به ازای هر پردازنده در کنار نخ‌های در حال اجرا، تعداد نخ‌های بسیار زیادی آماده به اجرا هستند و در صورتی که هر یک از ریسمان‌های در حال اجرا به دلیل درخواست برای دسترسی به حافظه معلق شود، سریعاً و با کمترین سربار ریسمان دیگری جایگزین خواهد شد. این ویژگی نقطه قوت پردازنده‌های گرافیکی است. از طرف دیگر در اوایل ظهور پردازنده‌های گرافیکی به دلیل پایین بودن فرکانس کاری هسته‌های پردازنده‌ی گرافیکی، تاخیر دسترسی به حافظه مشهود نبود و برای سال‌ها اکثر تحقیقات مربوط به پردازنده‌ی گرافیکی در جهت افزایش سطح موازات در این پردازنده‌ها بوده است. اما با رشد فرکانس کاری پردازنده‌ی گرافیکی به تدریج تاثیر تاخیر دسترسی به حافظه مشهود گردید. بنابراین، درحال حاضر یکی از موانع اصلی و نقاط ضعف پردازنده‌های گرافیکی، تاخیر دسترسی به حافظه است. از این رو، بهترین راه پیشنهادی برای بهره‌وری از مساحت بدست آمده، راه‌حلی میانی است که در آن نه تنها سطح موازات در پردازنده‌های گرافیکی به عنوان نقطه قوت این پردازنده‌ها افزایش یابد، بلکه تا جایی که ممکن است تاخیر دسترسی به حافظه به عنوان مهمترین نقطه ضعف پردازنده‌های گرافیکی کاهش داده شود [۶].

۲- پیش‌زمینه

در این بخش، مروری بر معماری پردازنده‌های گرافیکی عام‌منظوره می‌شود. ابتدا تاریخچه و سیر تحول در این پردازنده‌ها بیان شده، سپس معماری پردازنده‌های گرافیکی عام‌منظوره‌ی نوین را بررسی خواهیم کرد.

۲-۱- تاریخچه‌ی پردازنده‌های گرافیکی عام‌منظوره

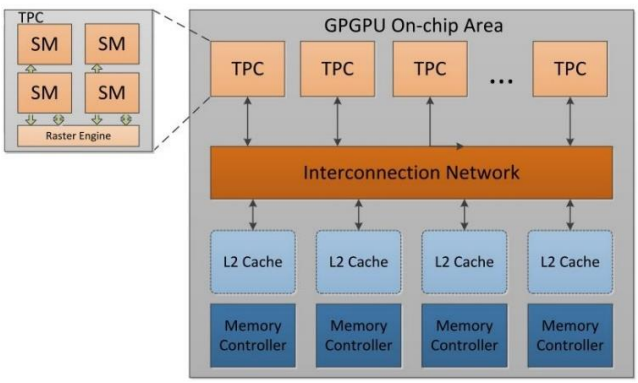
در دهه‌های ۸۰ و ۹۰ رشد محبوبیت سیستم‌عامل‌های با واسط گرافیکی مانند ویندوز باعث ایجاد بازاری پررونق برای نسل تازه‌ای از پردازنده‌ها شد. در همین دوره بود که کمپانی Silicon Graphic پردازنده‌های گرافیکی دارای توانایی عرضه تصویر سه بعدی را برای اولین بار وارد بازار کرد. در سال ۱۹۹۲ واسط برنامه‌نویسی را درقالب کتابخانه‌ی OpenGL برای سخت‌افزاری که اخیراً تولید کرده بود در اختیار برنامه‌نویسان قرار داد. این بازار پر رونق به ظهور شرکت‌هایی مانند NVIDIA، ATI و dfx۳ منجر شد [۱].

ورود کارت گرافیکی NVIDIA GeForce 256 موجب افزایش چشم‌گیر توانایی پردازنده‌های گرافیکی گردید. برای اولین بار پردازش نورپردازی و انتقال تصاویر مستقیماً روی پردازنده‌ی گرافیکی انجام شد. از دیدگاه بسیاری از پژوهشگران، عرضه‌ی کارتهای گرافیکی نسل GeForce 3 توسط کمپانی NVIDIA در سال ۲۰۰۱ نقطه عطفی در تاریخچه‌ی تکنولوژی GPU می‌باشد. GPUهای سری GeForce 3 اولین پردازنده‌های گرافیکی بودند که توانستند واسط برنامه‌نویسی گرافیکی استاندارد DirectX 8.0 را پشتیبانی کنند.

حدود ۵ سال بعد از انتشار سری GeForce 3 پردازش از طریق GPU وارد عصر تازه‌ای شد. در نوامبر ۲۰۰۶ شرکت NVIDIA با معرفی معماری CUDA برای کارت گرافیکی‌های خود برگ تازه‌ای را در برنامه‌نویسی GPU گشود. کاربران دیگر برای تعامل با GPU نیازی به یادگیری و تسلط بر واسط‌های OpenGL و DirectX نداشتند. در سال ۲۰۰۷ در بسیاری از صنایع و علوم تحقیقاتی برنامه‌هایی به زبان CUDA C نوشته شدند که نتایج درخشانی را در استفاده از GPU به عنوان پردازنده‌ی همه‌منظوره به دنبال داشت. دیری نپایید که برنامه نویسی به زبان CUDA به زمینه‌ای پر رونق در بازار برنامه‌نویسی تبدیل شد [۱].

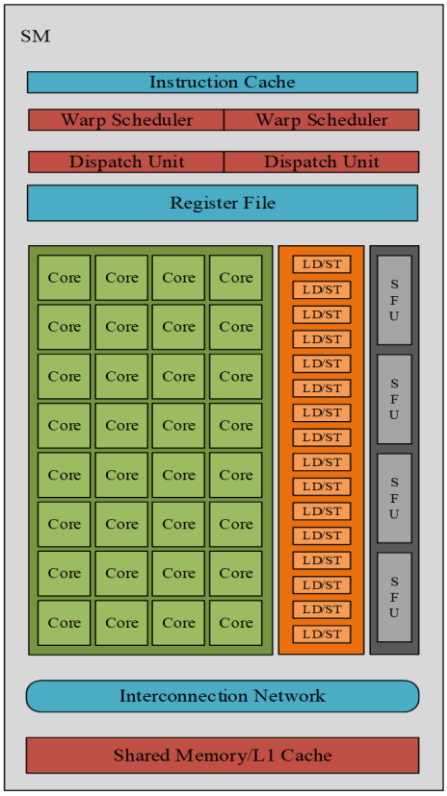
۲-۲- معماری پردازنده‌های گرافیکی

در این بخش در مورد معماری پردازنده‌های گرافیکی امروزی توضیح می‌دهیم. اکثر توضیحات در این بخش مربوط به معماری نسل فرمی[۶] شرکت NVIDIA است. شکل ۱ نمای کلی ساختار پردازنده‌های امروزی را نشان می‌دهد.



شکل ۱- نمایی کلی ساختار نسل جدید پردازنده‌های گرافیکی [۱]

در معماری پردازنده‌های گرافیکی، مجموعه‌ای از چند پردازنده‌های جریانی تک‌دستور-چندین نخ[۷] در قالب یک خوشه‌ی پردازنده-نخ قرار گرفته و چندین خوشه‌ی پردازنده-نخ با یک شبکه‌ی میان‌ارتباطی به چندین بانک حافظه نهان[۸] سطح دو متصل شده‌اند. علاوه بر حافظه‌ی اصلی پردازنده‌ی مرکزی، حافظه‌ای خارج از تراشه موسوم به حافظه‌ی سراسری نیز مختص پردازنده‌ی گرافیکی وجود دارد که وظیفه‌ی ارتباط با بانک‌های حافظه‌ی سراسری برعهده‌ی کنترل‌کننده‌ی حافظه است. برنامه‌ای که برای پردازنده‌ی گرافیکی نوشته می‌شود به تعداد زیادی بلوک تقسیم می‌شود و هر کدام از این بلوک‌ها برای اجرا به یکی از چند پردازنده‌های جریانی تخصیص داده می‌شود.



شکل ۲- ساختار داخلی چندپردازنده‌ی جریانی [۱]

شکل ۲ ساختار داخلی یک چند پردازنده‌ی جریانی را نشان می‌دهد. در داخل چند پردازنده‌ی جریانی سه نوع واحد پردازشی وجود دارد:

۱. هسته‌های پردازشی که وظیفه‌ی انجام محاسبات مقدار صحیح و اعشاری را برعهده دارند.
۲. واحد ذخیره و بازیابی که وظیفه‌ی اجرای دستورات خواندن/نوشتن از/در حافظه را برعهده دارد.
۳. واحدهای اجرای توابع مخصوص، که برای انجام دستورات خاص‌منظوره و پیچیده‌تر استفاده می‌شوند.

۲-۲-۱- سیستم حافظه در پردازنده‌های گرافیکی

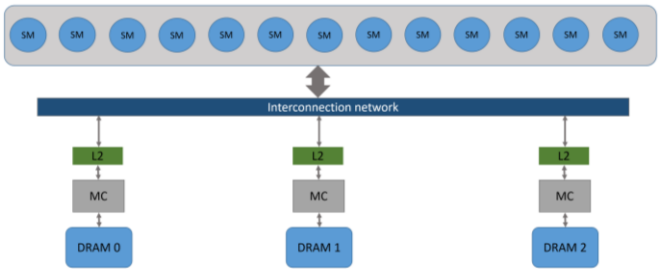
بلوک‌ها در چند پردازنده‌های جریانی برای اجرا به دسته‌های ۳۲ تایی از نخ‌ها تقسیم می‌شوند. به هرکدام از این دسته‌ها اصطلاحاً ریسمان می‌گویند. هر ریسمان و متغیرهایی که نیاز دارد، در داخل فایل ثبات ذخیره خواهند شد. برای مثال، اگر به یک چندپردازنده‌ی جریانی ۱۲۸ نخ تخصیص داده شود و هر نخ از ۶۴ ثبات استفاده کند، در مجموع ۶۴ × ۱۲۸ ثبات برای هر چندپردازنده‌ی جریانی مورد نیاز است. فایل ثبات در پردازنده‌های گرافیکی علاوه بر دارا بودن ظرفیت بسیار زیاد، باید پهنای باند زیادی را نیز فراهم کند. زیرا در غیر این صورت، واحدهای اجرایی خوب به کار گرفته نمی‌شوند. ایجاد پهنای باند بالا برای فایل ثبات یکی از چالش‌های مهم در طراحی پردازنده‌های گرافیکی است که روش‌های گوناگونی برای آن ارائه شده است. به همین دلیل، یکی از عواملی که سطح موازات را محدود می‌کند، فایل ثبات است.

در داخل چند پردازنده‌ی جریانی علاوه بر موارد مذکور، یک حافظه‌ی نهان سطح یک و یک حافظه‌ی مشترک نیز تعبیه شده‌اند که به صورت یکپارچه در یک فضای حافظه قرار دارند. دلیل این امر آن است که برنامه نویس بتواند بسته به نیاز خود اکثریت این فضا را به حافظه نهان سطح یک و یا حافظه‌ی مشترک اختصاص دهد. حافظه‌ی مشترک، یک فضای آدرس دهی هم‌سرعت با حافظه نهان سطح یک را در اختیار برنامه‌نویس قرار می‌دهد و برنامه‌نویس می‌تواند متغیرهایی را که به صورت متداوم در طول اجرای برنامه استفاده می‌کند، در این حافظه ذخیره کند تا نیازی نباشد که برای دسترسی به آن به حافظه مراجعه کرد. لازم به ذکر است که برنامه‌نویس به ازای هر بلوک، می‌تواند یکی از دو حالت زیر را برای اجرای برنامه خود تعیین کند.

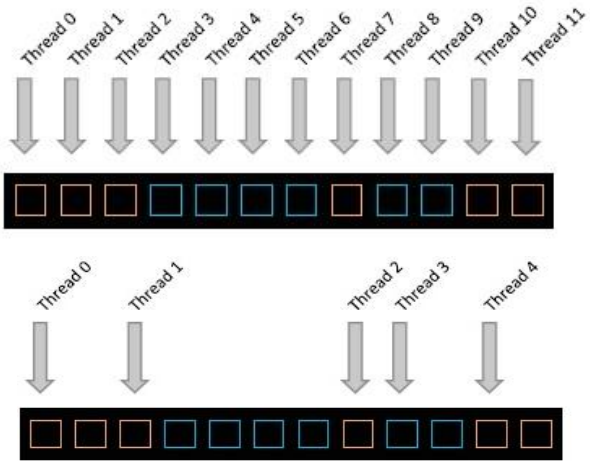
۱- ۱۶ کیلوبایت حافظه مشترک و ۴۸ کیلوبایت حافظه نهان
۲- ۴۸ کیلوبایت حافظه مشترک و ۱۶ کیلوبایت حافظه نهان

از این رو، حافظه‌ی مشترک نیز یکی دیگر از عوامل محدودکننده‌ی سطح موازات در پردازنده‌ی گرافیکی است.

از نسل فرمی به بعد، یک حافظه‌ی نهان سطح دوم مشترک نیز در معماری پردازنده‌های گرافیکی تعبیه شده است. این حافظه مطابق معماری NUCA به تعدادی بانک تقسیم شده است و این بانک‌ها تحت یک شبکه‌ی روی تراشه[۹] به چند پردازنده‌های جریانی متصل هستند. همچنین، هر بانک با یک کنترل‌کننده‌ی حافظه[۱۰] مرتبط شده است به این معنی که تنها محدوده مشخصی از آدرس‌ها را شامل می‌شود. شکل ۳ نمای کلی یک پردازنده‌ی گرافیکی را نشان می‌دهد که دارای سه کنترل‌کننده‌ی حافظه و سه واحد حافظه‌ی جداگانه است. برگ‌برگ‌سازی در پردازنده‌های گرافیکی به صورت متوالی است. یعنی آدرس‌های متوالی در واحدهای حافظه‌ای جداگانه قرار دارند تا از این طریق دسترسی به آدرس‌های متوالی را بتوان با بهره‌گیری از توازی دسترسی سرعت بخشید.



شکل ۳- نمای کلی پردازنده‌ی گرافیکی و سیستم دسترسی به حافظه‌ی آن



شکل ۴- دسترسی‌های منظم و نامنظم به حافظه [۱]

پردازنده‌های گرافیکی از حافظه‌ی GDDR برای حافظه‌ی برون تراشه[11] استفاده می‌کنند. حافظه‌ی GDDR یک حافظه‌ی DRAM با پهنای باند بالا است. GDDR از تعداد بسیار بیشتر بانک DRAM نسبت به DDR بهره می‌جوید و به صورت مستقیم به پایه‌های تراشه‌ی پردازنده گرافیکی متصل است. در حالی که در پردازنده‌های عام‌منظوره، DRAM از طریق DIMM به پردازنده متصل می‌شود. به دلیل پهنای باند بالای حافظه‌ی GDDR، در معماری پردازنده‌های گرافیکی تلاش می‌شود تا درخواست‌های متفاوت حافظه در کنار هم قرار گیرد و تعداد کمتری دسترسی به حافظه ایجاد شود. همان طور که در شکل ۴ نشان داده شده است، اگر آدرس دسترسی به حافظه برای نخ‌های گوناگون یک ریسمان به ترتیب باشد، تعداد دسترسی به حافظه کاهش یافته و در نتیجه، بهره‌وری پهنای باند حافظه افزایش می‌یابد. اما اگر دسترسی‌ها به ترتیب نباشد، دسترسی‌هایی با بهره‌وری پایین پهنای باند به سمت حافظه ارسال می‌شود.

جدول ۱- فضاهای متفاوت آدرس‌دهی در پردازنده‌ی گرافیکی به همراه مکان دسترسی و تاخیر دسترسی

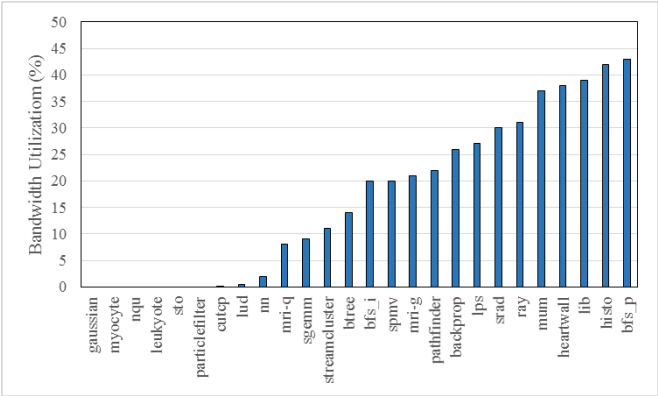
فضای آدرس‌دهی	مکان دسترسی	تاخیر دسترسی
حافظه محلی	DRAM و حافظه نهان سخت‌افزاری	صدها سیکل (فقدان حافظه نهان)
حافظه مشترک	حافظه نهان نرم‌افزاری	۴ تا ۳۲ سیکل
حافظه سراسری	DRAM و حافظه نهان سخت‌افزاری	صدها سیکل (فقدان حافظه نهان)
حافظه ثابت	DRAM و حافظه نهان ثابت	صدها سیکل (فقدان حافظه نهان)
حافظه بافت	DRAM و حافظه نهان بافت	صدها سیکل (فقدان حافظه نهان)

در پردازنده‌ی گرافیکی برای مدیریت هرچه بهتر آدرس، فضاهای متفاوتی برای آن در نظر گرفته شده است. در جدول ۱ فضاهای متفاوت آدرس‌دهی و همچنین، تاخیر دسترسی به آن‌ها بیان شده است.

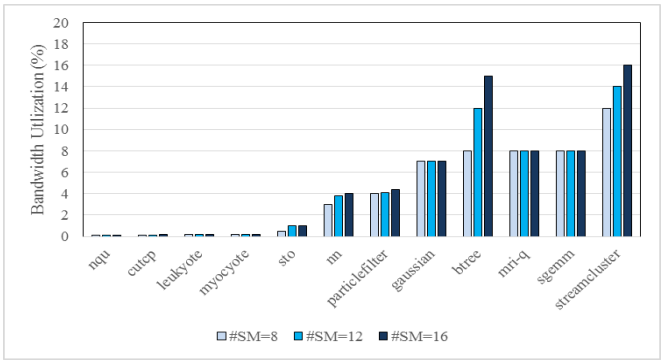
۳- طرح مسئله و اهمیت آن

چنان که در فصل مقدمه توضیح داده شد، تولید کنندگان پردازنده‌های گرافیکی به شکل ثابتی همواره بین قدرت محاسباتی پردازنده‌های گرافیکی و پهنای باند آن‌ها نسبت مستقیم را رعایت کرده‌اند. به این معنی که با افزایش قدرت پردازشی پردازنده‌های گرافیکی، پهنای باند را نیز افزایش داده‌اند. در نگاه اول به نظر می‌رسد که پردازش بیشتر نیاز به پهنای باند بزرگتر نیز دارد. بنابراین، رعایت این نسبت منطقی به نظر می‌رسد. اما، بررسی‌های انجام شده بر روی تعداد زیادی از بارهای کاری موجود نشان می‌دهد که برای بخش اعظمی از برنامه‌های محک، بهره‌وری پهنای باند به شدت پایین است. شکل ۵ نرخ بهره‌وری را برای تعداد زیادی از بارهای کاری نشان می‌دهد.

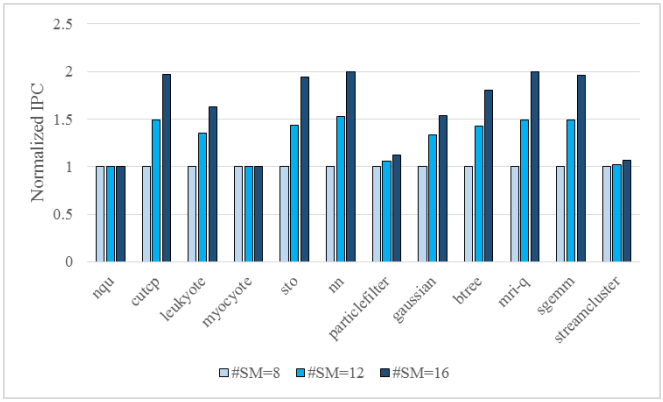
چنان که در شکل ۵ مشاهده می‌کنیم، به ازای ۵۷٪ بارهای کاری نرخ بهره‌وری زیر ۲۰٪ است و در سایر موارد نرخ بهره‌وری در بیشترین مقدار خود، به ۴۴ درصد می‌رسد. بررسی‌های ما نشان می‌دهد که دلیل بهره‌وری پایین پهنای باند این است که این بارهای کاری، نسبت به زمان انتقال داده به/ از حافظه، زمان بسیار بیشتری را صرف پردازش می‌کنند. به عبارت دیگر تعداد دستورات محاسباتی آن‌ها بسیار بیشتر از دستورات حافظه‌ای است. مشخص است که هرچه یک بارکاری تعداد سیکل‌های بیشتری را صرف عملیات اجرایی و پردازشی کند، طبیعتاً سیکل‌های دسترسی به حافظه و به دنبال آن بهره‌وری از پهنای باند نیز در آن کاهش می‌یابد.



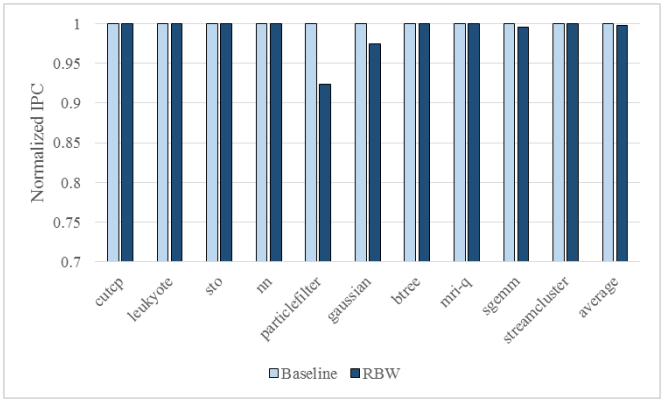
شکل ۵- نرخ بهره‌وری پهنای باند به ازای همه‌ی بارهای کاری موجود



شکل ۶- نرخ بهره‌وری پهنای باند به ازای تعداد چند پردازنده‌های جریانی



شکل ۷- کارایی به ازای تعداد چند پردازنده‌های جریانی



شکل ۸- کارایی پردازنده‌ی گرافیکی در ازای کاهش پهنای باند

نمودار شکل ۵ به وضوح نشان می‌دهد که صرف پهنای باند بزرگتر برای بارهای کاری CILMA، مقرون به صرفه نیست. شکل ۶ نمودار نرخ بهره‌وری را به ازای تعداد مختلف چند پردازنده‌های جریانی برای این بارهای کاری نشان می‌دهد.

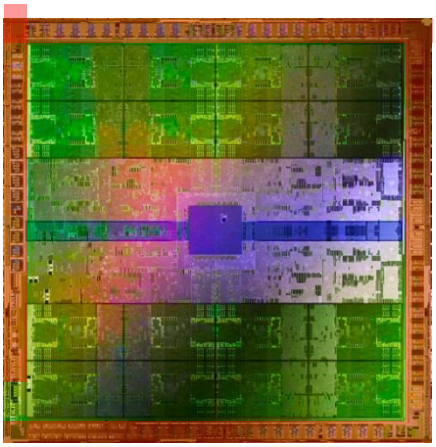
چنان که در این شکل نشان داده شده است، با تغییر تعداد چندپردازنده‌های جریانی و ثابت نگه داشتن پهنای باند برای این بارهای کاری، نرخ بهره‌وری پهنای باند چندان تغییری نمی‌کند. اما همان‌طور که در شکل ۷ مشاهده می‌شود، با افزایش تعداد چند پردازنده‌های جریانی، کارایی این بارهای کاری به میزان چشمگیری بهبود می‌یابد. در واقع شکل‌های ۶ و ۷ نشان می‌دهند که افزایش پهنای باند برای این بارهای کاری منجر به بهبود نخواهد شد بلکه نیاز اصلی این بارهای کاری فراهم کردن قدرت محاسباتی بیشتر توسط افزایش تعداد چند پردازنده‌های جریانی است.

می‌توان نتیجه گرفت که نسبت مستقیم موجود بین قدرت پردازنده‌های گرافیکی و پهنای باند برای بارهای کاری CILMA که حساس به پهنای باند پردازنده‌های گرافیکی نیستند، رویه‌ی مناسبی نیست. بنابراین، این بارهای کاری به پردازنده‌های گرافیکی پرقدرت با توان پردازشی بالا و پهنای باند کمتر نیاز دارند. به منظور بررسی بیشتر ظرفیت موجود برای تولید پردازنده‌های گرافیکی با پهنای باند کمتر، می‌بایست کارایی را پس از کاهش پهنای باند[۱۲] نیز برای این بارهای کاری ارزیابی کنیم. شکل ۸ نتایج ارزیابی این آزمایش را نشان می‌دهد.

چنان که از نتایج مطرح شده در شکل ۸ پیداست، کارایی برای دو مدل پیکربندی پردازنده‌ی گرافیکی با قدرت برابر اما پهنای باند متفاوت، در اکثر موارد تقریباً یکسان است. همچنین، نتیجه‌ی حالت میانگین نیز نشان می‌دهد که کاهش پهنای باند پردازنده‌ی گرافیکی، کارای این بارهای کاری را چندان تحت تاثیر قرار نداده است.

۳-۱- انگیزه‌ی راه‌کار پیشنهادی

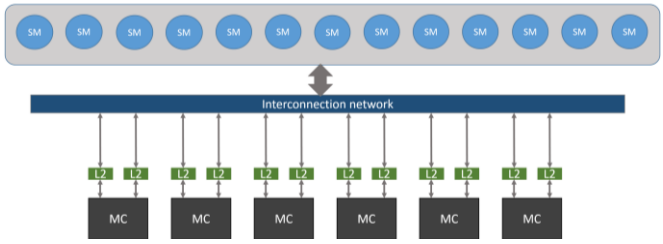
چنان که در بخش قبل توضیح داده شد، کارایی پردازنده‌ی گرافیکی تحت بارهای کاری CILMA پس از کاهش پهنای باند چندان تغییری نمی‌کند. بنابراین، نیاز به مدلی از پردازنده‌های گرافیکی با قدرت پردازشی زیاد و پهنای باند کمتر نسبت به مدل‌های کنونی آشکار است. کاهش پهنای باند از طریق حذف کانال‌های گذرگاه حافظه انجام می‌شود. این کانال‌ها در خارج تراشه به واحدهای حافظه‌ی بیرون تراشه و در داخل تراشه به کنترل‌کننده‌های حافظه متصل می‌باشند. بنابراین، کاهش پهنای باند به حذف کنترل‌کننده‌های اضافی روی تراشه می‌انجامد. بررسی‌های ما حاکی از این است که مساحت هرکدام از این کنترل‌کننده‌ها بسیار بزرگ است. در این پژوهش، پردازنده‌ی گرافیکی GTX 480 مورد آزمایش قرار گرفته است. اگرچه در صورت انتخاب معماری‌های دیگر نیز نتیجه مشابه حاصل خواهد شد.



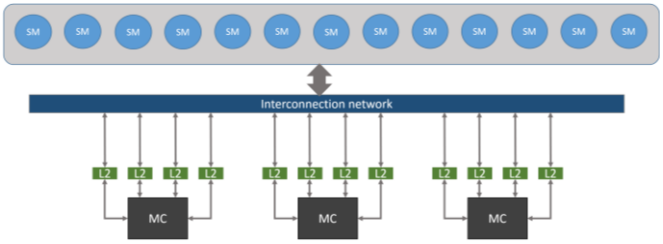
شکل ۹- عکس قالب پردازنده‌ی گرافیکی GTX 480 [۷]

شکل ۹ عکس قالب پردازنده‌ی گرافیکی مدل GTX 480 را نشان می‌دهد. در این شکل، ناحیه‌ای که به رنگ قرمز است، مساحت اشغال شده توسط کنترل‌کننده‌های حافظه را نشان می‌دهد. اندازه‌گیری‌های ما نشان می‌دهد که مساحت هر یک از این کنترل‌کننده‌ها تقریباً برابر با ۱۰ mm^2 است. نصف شدن پهنای باند منجر به حذف نیمی از کنترل‌کننده‌های حافظه خواهد شد. این مسئله به تنهایی باعث می‌شود بدون تغییر محسوسی در کارایی پردازنده‌ی گرافیکی برای بارهای کاری CILMA، توان مصرفی در حد قابل توجهی کاهش یابد.

همان‌طور که در شکل ۱۰ مشخص شده است، در پردازنده‌ی گرافیکی مدل GTX 480 شش عدد کنترل‌کننده‌ی حافظه موجود است. بنابراین، با نصف کردن پهنای باند می‌توان به مقدار ۳۰ mm^2 مساحت آزاد به دست آورد. شکل ۱۱ وضعیت داخل تراشه‌ی پردازنده‌ی GTX 480 را پس از کاهش پهنای باند نشان می‌دهد. بدست آمدن این فضای آزاد بر روی تراشه نوید حصول فرصت‌هایی است که می‌توانند در جهت بهبود کارایی پردازنده‌های گرافیکی به کار گرفته شوند. به این ترتیب که از این فضای آزاد می‌توان به شکل بهتری بهره‌برداری کرد و به گونه‌ای منابع روی تراشه‌ی پردازنده‌ی گرافیکی را توسعه داد که نهایتاً منجر به افزایش کارایی این پردازنده‌ها به خصوص حین اجرای بارهای کاری CILMA شود. در بخش بعدی این فرصت‌ها را بررسی نموده و بهترین راه‌کار استفاده از مساحت به دست آمده را معرفی خواهیم کرد.



شکل ۱۰- نمای کلی پردازنده‌ی گرافیکی GTX 480 قبل از کاهش پهنای باند



شکل ۱۱- نمای کلی پردازنده‌ی گرافیکی GTX 480 پس از نصف شدن پهنای باند

چالش اصلی این پژوهش تعیین نحوه‌ی بهره‌برداری از فضای آزاد به‌دست آمده از حذف کنترل‌کننده‌های حافظه است، با این هدف که بیشترین تسریع برای بارهای کاری CILMA به دست بیاید. چنانچه پیش‌تر عنوان شد، نقطه‌ی قوت پردازنده‌های گرافیکی بهره‌وری از سطح موازات بسیار بالا چه در سطح نخ و چه در سطح بلوک است. بنابراین، به نظر می‌رسد بهترین انتخاب توسعه‌ی منابع پردازشی و ذخیره‌سازی با هدف افزایش سطح موازات باشد.

۴- راه‌کار پیشنهادی

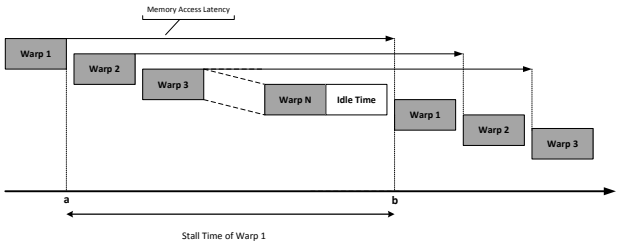
سطح موازات را در پردازنده‌ی گرافیکی می‌توان به یکی از دو روش زیر افزایش داد:

۱. افزایش تعداد چند پردازنده‌های جریانی
۲. افزایش ظرفیت چند پردازنده‌های جریانی برای نگهداری نخ‌های آماده به اجرا

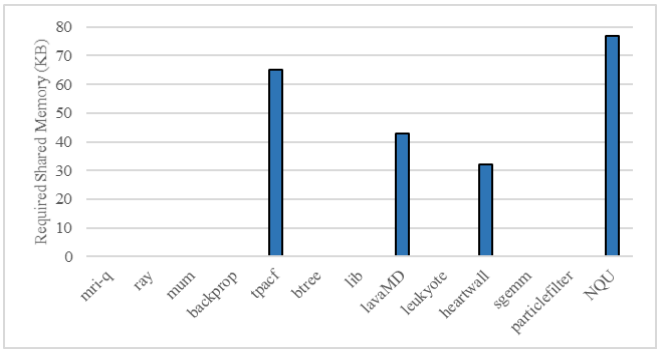
افزایش تعداد چند پردازنده‌های جریانی باعث می‌شود بستر موازی سازی اجرای همزمان بلوک‌ها افزایش یافته و در نتیجه، پردازنده‌ای قدرتمندتر و سریع‌تر داشته باشیم. منتهی چنان که از عکس قالب پردازنده‌ی گرافیکی پیداست مساحت لازم برای اضافه کردن تنها یک چند پردازنده‌ی جریانی به تنهایی برابر با ۱۸ mm^2 است. بنابراین، مساحت به دست آمده از حذف کنترل‌کننده‌های حافظه تنها می‌تواند صرف اضافه‌کردن یک چندپردازنده‌ی جریانی شود که تاثیر چشم‌گیری بر

سطح موازات بلوک نخواهد داشت. بنابراین، بهره‌وری از مساحت به‌دست آمده از طریق افزایش تعداد چندپردازنده‌های جریانی در عمل غیرممکن است.

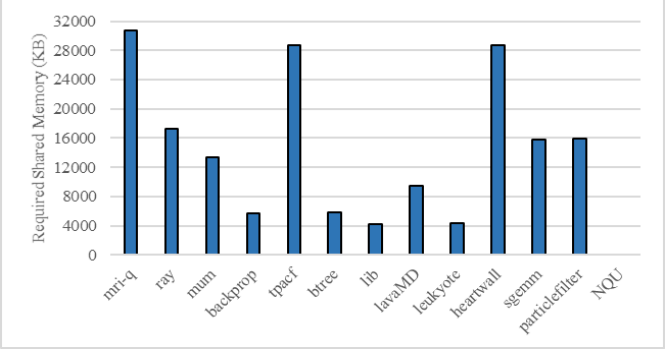
در روش دوم، بهبود سطح موازات از طریق افزایش ظرفیت چند پردازنده‌های جریانی برای نگهداری نخ‌های آماده به اجرا، میسر است. همان طور که در بخش‌های قبلی ذکر شد، پردازنده‌های گرافیکی عموماً از موازی‌سازی استفاده می‌کنند تا تاخیر دسترسی به حافظه را پوشش دهند. یعنی زمانی که یک ریسمان به دلیل دسترسی به حافظه متوقف می‌شود، یک ریسمان دیگر از میان مجموعه ریسمان‌های آماده به اجرا انتخاب شده و جایگزین ریسمان فعلی می‌شود. به این ترتیب، نه تنها از بیکار ماندن چندپردازنده‌ی جریانی جلوگیری می‌شود، بلکه تاخیر دسترسی به حافظه نیز پوشش داده می‌شود. شکل ۱۲ نحوه انجام این ساز و کار را در داخل یک چندپردازنده‌ی جریانی نشان می‌دهد. هرچه سطح موازات نخ و به عبارت دیگر تعداد ریسمان‌های آماده به اجرا بیشتر باشد، توانایی چندپردازنده‌ی جریانی برای کاهش سیکل‌های بیکاری بیشتر می‌شود. کاهش سیکل‌های بیکاری افزایش بهره‌وری چندپردازنده‌ی جریانی و درنهایت افزایش کارایی را در پی خواهد داشت. تعداد بلوک‌های تخصیص داده شده به هر چندپردازنده‌ی جریانی به سه عامل مختلف بستگی دارد که در ادامه به تشریح آن‌ها خواهیم پرداخت.



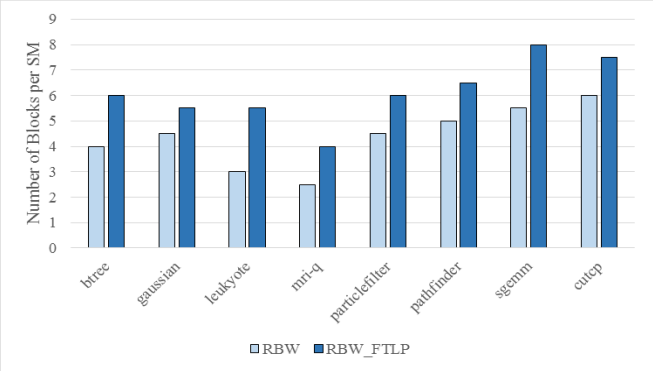
شکل ۱۲- چگونگی نوبت‌دهی ریسمان در چندپردازنده‌ی جریانی (بلوک‌های خاکستری سیکل‌های مشغول و بلوک‌های سفید زمان‌های بیکاری یک پردازنده را نشان می‌دهند)



شکل ۱۳- مقدار حافظه مشترک مورد نیاز برای رفع محدودیت سطح موازات به ازای هریک از بارهای کاری



شکل ۱۴- تعداد ثبات مورد نیاز برای رفع محدودیت سطح موازات به ازای هریک از بارهای کاری



شکل ۱۵- وضعیت تعداد بلوک‌های تخصیص داده شده به هر چندپردازنده‌ی جریانی قبل و بعد از رفع محدودیت کمبود فایل ثبات

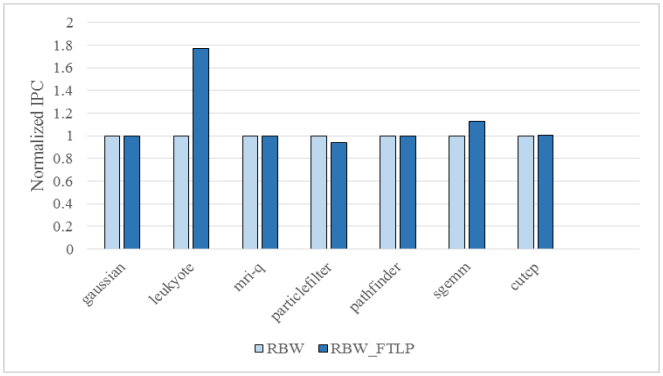
برای افزایش سطح موازات نخ چهار راه حل پیش رو داریم:

۱. افزایش حافظه‌ی مشترک
۲. افزایش فایل ثبات
۳. افزایش حافظه‌ی نهان سطح یک
۴. افزایش حافظه نهان سطح دو

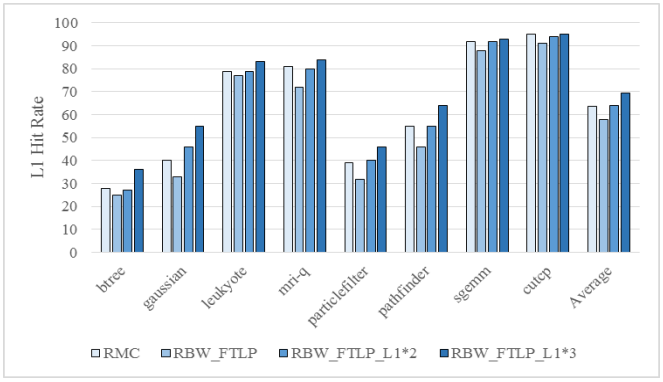
به منظور تصمیم‌گیری در این زمینه که با توجه به مساحت موجود به چه مقدار از منابع یاد شده نیاز است، دوباره رفتار بارهای کاری موجود را بررسی کرده و به ازای هرکدام از این بارها محدودیت اصلی سطح موازت نخ را استخراج می‌کنیم. شکل ۱۳ مقدار حافظه‌ی مشترک مورد نیاز برای رفع محدودیت سطح موازات به ازای بارهای کاری مختلف را نشان می‌دهد. برای آن دسته از بارهای کاری که مقداری برای آن‌ها گزارش نشده است، یعنی عامل محدود کننده حافظه‌ی مشترک نبوده و باید عوامل دیگر را مورد بررسی قرار دهیم. با توجه به بررسی‌های صورت گرفته، تنها حدود ۴٪ از هسته‌ها به دلیل کمبود حافظه‌ی مشترک با محدودیت افزایش سطح موازات روبه رو هستند. همچنین، مقدار حافظه‌ی مشترکی که نیاز است تا محدودیت همه‌ی بارهای کاری از بین برود، تقریباً برابر با KB ۷۸ به ازای هر چندپردازنده‌ی جریانی است که در مجموع برابر با KB ۱۱۷۰= ۱۵×۷۸ می‌باشد. این مقدار به تنهایی حدود ۱۰ درصد مساحت به دست آمده از حذف کنترل‌کننده‌های حافظه است. با توجه به دو دلیل فوق بهره‌برداری از مساحت به دست آمده از طریق افزایش حافظه‌ی مشترک به هیچ وجه مقرون به صرفه نیست. بنابراین افزایش حافظه‌ی مشترک از گزینه‌های موجود برای بهره‌وری از مساحت به دست‌آمده به کلی حذف می‌شود.

عامل بعدی افزایش فایل ثبات است. شکل ۱۴ تعداد ثبات مورد نیاز برای رفع محدودیت سطح موازات به ازای هر بار کاری را نشان می‌دهد. بررسی‌های ما نشان می‌دهد که بر خلاف حافظه‌ی مشترک، تقریباً ۴۵٪ هسته‌های بارهای کاری موجود از کمبود فایل ثبات رنج می‌برند. بنابراین، افزایش فایل ثبات به دلیل پوشش محدودیت سطح موازات درصد زیادی از هسته‌های موجود، به عنوان گزینه‌ی اصلی بهره‌برداری از مساحت موجود به حساب می‌آید. شکل ۱۵ نشانگر وضعیت تعداد بلوک‌های تخصیص داده شده به هر چندپردازنده‌ی جریانی تحت بارهای کاری CILMA است. همان‌طور که مشاهده می‌شود، تعداد بلوک‌های تخصیصی به هر چندپردازنده‌ی جریانی به درستی پس از افزایش حجم فایل ثبات، بیشتر شده است. دقت کنید که بارهای کاری که در شکل ۱۵ آورده شده است، مواردی هستند که در آن‌ها کمبود فایل ثبات مانع اصلی افزایش سطح موازات بوده است. همچنین، شکل ۱۶ وضعیت کارایی پردازنده‌ی گرافیکی را قبل و بعد از افزایش حجم فایل ثبات، برای همان بارهای کاری نشان می‌دهد. چنان که در این شکل مشاهده می‌شود، با وجود افزایش حجم فایل ثبات و به دنبال آن بالا رفتن سطح موازات در اکثر موارد به جز دو مورد کارایی پردازنده‌ی گرافیکی کاهش یافته است. علت این مسئله را باید در داخل چندپردازنده‌ی جریانی جستجو کرد.

افزایش سطح موازات سبب می‌شود که استخر ریسمان بزرگتر شده و در نتیجه تعداد بیشتری از نخ‌های مختلف به صورت گردشی و در زمان‌های کوتاه، خط اجرا را به دست گیرند. با توجه به اینکه هر کدام از این نخ‌ها می‌توانند درخواست‌های متنوعی به آدرس‌های مختلف حافظه داشته باشند، افزایش تعداد نخ‌ها سبب افزایش تنوع دسترسی به حافظه در چندپردازنده‌ی جریانی می‌شود. این مسئله سبب جایگذاری‌های بیهوده در حافظه نهان سطح یک شده و نرخ برخورد را در این واحد به شدت پایین می‌آورد. کاهش نرخ برخورد در حافظه نهان سطح یک در برخی از بارهای کاری به حدی شدید است که افزایش سطح موازات نمی‌تواند سربار تاخیر اضافه شده را جبران کند. این پیامد را در اصطلاح Cache Thrashing می‌گویند. بنابراین، برای اینکه افزایش سطح موازات برای این دسته از بارهای کاری هم، موثر باشد می‌بایست همزمان حجم حافظه نهان سطح یک نیز افزایش یابد.

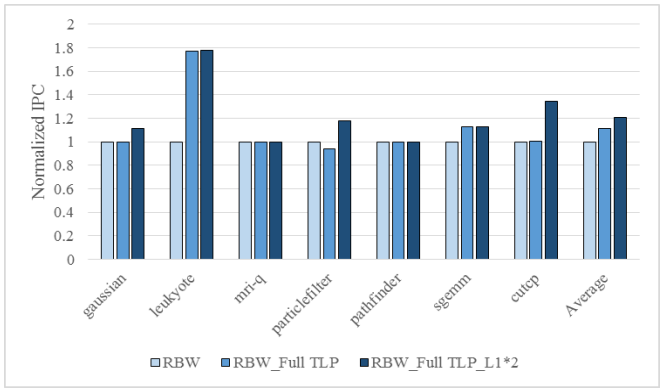


شکل ۱۶- کارایی پردازنده‌ی گرافیکی قبل و بعد از افزایش حجم فایل ثبات

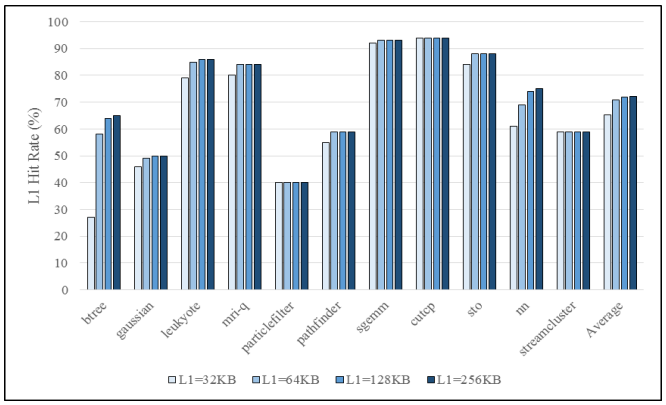


شکل ۱۷- تغییرات نرخ برخورد حافظه‌ی نهان سطح یک از چپ به راست: ۱) در حالت عادی، ۲) پس از افزایش حجم فایل ثبات، ۳) پس از افزایش حجم فایل ثبات و دو برابر کردن حافظه‌ی نهان سطح یک و ۴) پس از افزایش حجم فایل ثبات و سه برابر کردن حافظه‌ی نهان سطح یک

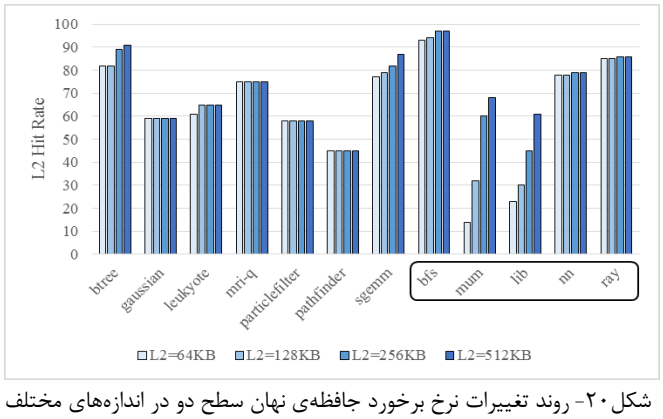
به همین منظور برای بررسی تاثیر حافظه نهان سطح یک، در آزمایش دیگری علاوه بر افزایش حجم فایل ثبات، حافظه‌ی نهان سطح یک را به ترتیب دو و سه برابر کردیم. شکل ۱۷ نمودار نرخ برخورد حافظه‌ی نهان سطح یک را در حالت‌های مختلف نشان می‌دهد. به ازای هر بار کاری، نمودار اول نرخ برخورد در حالت پایه، نمودار دوم پس از افزایش حجم فایل ثبات و نمودارهای سوم و چهارم پس از افزایش حجم فایل ثبات و همچنین، دو و سه برابر کردن حجم حافظه‌ی نهان سطح یک را نشان می‌دهند. چنان که مشاهده می‌شود، با دو برابر کردن حجم حافظه‌ی نهان سطح یک، نرخ برخورد با حالت اولیه تقریباً برابر شده و مشکل Cache Trashing پوشش داده می‌شود. همچنین، شکل ۱۸ تغییرات کارایی پرازنده‌ی گرافیکی را پس از افزایش حجم فایل ثبات و دوبرابر کردن حافظه‌ی نهان سطح یک در قیاس با حالت عادی را نشان می‌دهد. همان‌طور که مشاهده می‌شود، کارایی پردازنده‌ی گرافیکی برای بارهای کاری CILMA که دچار کمبود فایل ثبات بودند، نزدیک به ۲۰٪ افزایش یافته است. بررسی‌های ما نشان می‌دهد با افزایش حجم فایل ثبات به میزان مورد نظر و همچنین، دو برابر کردن حجم حافظه‌ی نهان سطح یک برای حل مشکل Cache Trashing در کل حدود ۲۲ درصد از مساحت حاصل از کاهش پهنای باند مصرف می‌شود.



شکل ۱۸- تغییرات کارایی پردازنده‌ی گرافیکی از چپ به راست: ۱) پس از کاهش پهنای باند، ۲) پس از افزایش حجم فایل ثبات و ۳) پس از افزایش حجم فایل ثبات و دوبرابر کردن حجم حافظه‌ی نهان سطح یک



شکل ۱۹- تغییرات نرخ برخورد حافظه‌ی نهان سطح یک به ازای اندازه‌های مختلف حجم این واحد



شکل ۲۰- روند تغییرات نرخ برخورد جافظه‌ی نهان سطح دو در اندازه‌های مختلف این واحد (بارهای کاری که دور آن‌ها خط کشیده شده است بارهایی هستند که دسترسی بیشتری به حافظه دارند)

روش سوم افزایش حجم حافظه‌ی نهان سطح یک است. این روش در واقع، با کاهش تاخیر دسترسی به حافظه تعداد سیکل‌های بیکاری چندپردازنده‌ی جریانی را کاهش می‌دهد. شکل ۱۹ تغییرات نرخ برخورد حافظه‌ی نهان سطح یک را به ازای مقادیر متفاوت حجم این واحد نشان می‌دهد. توجه شود که مقادیر اعلام شده

برای حجم حافظه‌ی نهان به ازای یک چندپردازنده‌ی جریانی است. چنان که مشاهده می‌شود، به ازای افزایش اندازه‌ی حافظه‌ی نهان سطح یک، نرخ برخورد نیز بیشتر می‌شود. زمانی که این حافظه‌ی نهان به حجم KB ۶۴ به ازای هر چندپردازنده‌ی جریانی می‌رسد، تقریباً بهترین شرایط مهیا می‌شود. در حجم KB ۶۴ نرخ برخورد به طور میانگین ۵ درصد افزایش نسبت به مقدار قبلی خود یعنی KB ۳۲ داشته است. از طرف دیگر، حجم بالاتر یعنی KB ۱۲۸ و KB ۲۵۶ تنها ۰/۶ درصد نسبت به حجم قبلی بهبود نرخ برخورد داشته است. بنابراینً با توجه به توان مصرفی بسیار بیشتر حافظه‌های نهان با حجم KB ۱۲۸ و KB ۲۵۶ ، افزایش حجم حافظه‌ی نهان سطح یک از مقدار KB ۶۴ به هیچ وجه معقول و مقرون به صرفه نیست.

روش نهایی، افزایش حجم حافظه‌ی نهان سطح دو و به دنبال آن بالا بردن نرخ برخورد در این حافظه است. این روش نیز از طریق کاهش تاخیر دسترسی به حافظه به بهبوئ سطح موازات کمک می‌کند. نکته‌ی اصلی در پیش‌برد این هدف این است که برای دیدن تاثیر افزایش حجم حافظه‌ی نهان سطح دو می‌بایست وضعیت نرخ برخورد حافظه‌ی نهان سطح دو تحت بارهای کاری به جز بارهای کاری CILMA نیز در نظر گرفته شود. چرا که از طرفی تعداد زیادی از این بارهای کاری به شدت پردازشی هستند و دسترسی‌ها به حافظه‌ی نهان سطح دو در این بارهای کاری به حدی پایین است که افزایش حجم حافظه‌ی نهان سطح دو نمی‌تواند بهبود چشمگیری در نرخ برخورد آن در این بارهای کاری ایجاد کند. از طرف دیگر درست است که هدف این پژوهش بهبود کارایی پردازنده‌ی گرافیکی تحت بارهای کاری CILMA است اما باید توجه داشته باشیم که تغییرات انجام شده نباید کارایی پردازنده‌ی گرافیکی را برای سایر بارهای کاری به شکل محسوسی پایین بیاورد. باتوجه به این نکات شکل ۲۰ روند افزایش نرخ برخورد را برای دسته‌ای از بارهای کاری مختلف (چه بارهای کاری CILMA و چه سایر بارهای کاری) نشان می‌دهد.

چنان که در شکل ۲۰ مشخص است، با افزایش حجم حافظه‌ی نهان سطح دو نرخ برخورد نیز بیشتر می‌شود. افزایش نرخ برخورد باعث کاهش تعداد دسترسی‌های مستقیم به حافظه‌ی بیرون تراشه شده و تاخیر دسترسی به حافظه را کاهش می‌دهد. از طرف دیگر، باید خاطر نشان کرد که بزرگ کردن حافظه‌ی نهان سطح دو بیشتر در مواردی باعث افزایش نرخ برخورد می‌شود که تعداد دسترسی‌های به حافظه زیاد باشد. به همین دلیل، تغییرات نرخ برخورد در بارهای کاری که در شکل ۲۰ دور آن‌ها خط کشیده شده است، محسوس‌تر است. واضح است، زمانی که حجم حافظه‌ی نهان سطح دو به KB ۵۱۲ می‌رسد، در بارهای کاری‌ای که تعداد دسترسی زیاد به حافظه دارند، کمبود پهنای باند به شکل قابل توجهی جبران شده و کاهش کارایی نسبت به حالت اولیه که پهنای باند دست نخورده بود، بسیار ناچیز است. اندازه‌ی بانک‌های حافظه‌ی نهان سطح دو نمی‌تواند مقداری بزرگتر از KB ۵۱۲ باشد چرا که در صورت افزایش حجم این واحد تاخیر دسترسی بیشتر از حدی می‌شود که بتوان در یک پالس ساعت به بانک‌های حافظه‌ی نهان دسترسی داشت. تعیین دقیق اندازه‌ی بانک‌های حافظه‌ی نهان سطح دو می‌بایست با توجه به توان مصرفی کل پردازنده‌ی گرافیکی تعیین شود. شبیه‌سازی‌های ما نشان می‌دهد که برای حجم KB ۵۱۲ به ازای هر بانک حافظه‌ی نهان سطح دو، مقدار افزایش توان مصرفی کل پس از اعمال تغییرات قابل قبول نیست. بنابراین، برای بانک‌های حافظه‌ی نهان سطح دو، مقدار KB ۲۵۶ در نظر گرفته می‌شود.

۵- محیط شبیه‌سازی

برای ارزیابی ایده‌ی مطرح‌شده در این پژوهش، از شبیه‌ساز GPGPU-Sim [۱۲] استفاده شده است. از مزایای مهم به‌کارگیری این شبیه‌ساز، تطبیق ۹۸/۳۷ درصدی با پردازنده‌های گرافیکی رایج از نظر تعداد دستورالعمل صادر شده در هر پالس ساعت است. این شبیه‌ساز کد دودویی واحد پردازشگر مرکزی و گرافیکی را اجرا می‌کند اما زمان‌بندی واحد پردازش مرکزی و واسط PCI-express را مدل نمی‌کند. شبیه‌ساز GPGPU-Sim ریزدستورات پردازنده‌ی گرافیکی را شبیه‌سازی کرده و شامل چهار دامنه پالس ساعت، پالس ساعت هسته‌های پردازشی، شبکه‌های میان ارتباطی، حافظه‌ی نهان سطح دو و حافظه‌ی سراسری است. از طرف دیگر، این شبیه‌ساز به دو روش عملکردی و عملیاتی شبیه‌سازی را انجام می‌دهد. در روش عملیاتی، فقط خروجی برنامه‌ی CUDA یا OpenCL نمایش داده می‌شود که بیشتر برای صحت‌سنجی کد نوشته شده است. در روش عملکردی، اطلاعات از قبیل زمان اجرا، نرخ فقدان حافظه‌های نهان، تعداد دستورات، نسبت سیکل ساعت بر دستور و در صورت نیاز کاربر، تخصیص‌های بلوک‌های نخ، تمامی اختصاص‌ها و دسترسی‌ها به حافظه، دستورات اجراشده، محتوای ثبات‌ها و تغییرات آن و کد PTX اجراشده نمایش داده می‌شود.

نحوه‌ی عملکرد شبیه‌ساز ذکرشده، به این صورت است که ابتدا کتابخانه‌هایی کامپایل می‌شود که در زمان اجرا به صورت پویا به برنامه‌های CUDA یا OpenCL مرتبط می‌شوند. این کتابخانه‌ها از صدازدن توابعی که باعث می‌شود برنامه‌ها توسط محیط اجرایی CUDA اجرا شوند، جلوگیری کرده و به جای آن شبیه‌ساز را راه‌اندازی می‌کند. سپس، کد مربوطه به جای این که روی سخت‌افزار اجرا شود توسط این کتابخانه‌ها روی شبیه‌ساز اجرا می‌شود. برای محاسبه‌ی توان نیز از ابزار GPU-Wattch [۸] استفاده می‌شود که ابزار MCPAT [۹] درون آن مجتمع‌سازی شده است. این ابزار با استفاده از ضریب فعالیت گزارش‌شده از ابزار شبیه‌ساز و خروجی توان MCPAT توان پویا را به صورت دقیق برای تمامی واحدها به صورت جداگانه محاسبه می‌نماید. از طرف دیگر یکی از خروجی‌های ابزار MCPAT همان توان ایستا است که در این پروژه استفاده می‌شود.

۵-۱- مجموعه برنامه‌های محک

برای آزمون شبیه‌سازی‌ها از سه دسته بارکاری واقعی استفاده می‌شود. دسته اول دسته بارکاری معرفی‌شده همراه شبیه‌ساز، دسته‌ی دوم بارکاری موسوم به Rodinia [۱۰]، و دسته‌ی سوم دسته برنامه‌ی محک Parboil [۱۱] هستند. این بارهای‌کاری به زبان‌های CUDA و OpenCL هستند. در جدول ۲، خلاصه‌ی برنامه‌های محک توضیح مختصری در رابطه با برنامه‌های محک مورد استفاده ارائه شده است.

جدول ۲- خلاصه‌ای از خصوصیات برنامه‌های محک (بارهای کاری)

Benchmark	Description	Instruction Executed (M)
lib	LIBOR Monto Carlo	999
mum	MUMmerGPU	85
nn	A Neural Network on GPU	123
backprop	Back Propagation	190
bfs	Breadth First Search	460
guassian	Gaussian Elimination	3853
heartwall	Heart Wall traking	26428
kmeans	Dense Linear Algebra	21052
srad	Speckle Reducing Anisotropic	8292
btree	Graph Traversal	-
cutcp	Cuttoff-limited Coulombic Potential	33507
lbm	Lattice-Boltzman Method simulation	10068
mri-gridding	MRI Cartesian Gridding	15665
sgemm	Dense matrix operation	4926

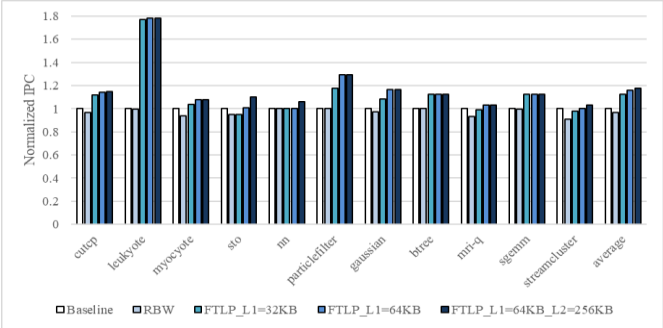
۶- نتایج شبیه‌سازی

شکل ۲۱ نتایج کارایی برای بارهای کاری CILMA را در حالت پایه، پس از کاهش پهنای باند، پس از افزایش حجم فایل ثبات، پس از افزایش حجم حافظه‌ی نهان سطح یک و افزایش حجم حافظه‌ی نهان سطح دو نشان می‌دهد. همان‌طور که مشاهده می‌شود، با استفاده از تکنیک‌های اعمال شده به طور متوسط نزدیک به ۱۸ درصد بهبود حاصل شده است. آن چه که مشخص است برای بارهای کاری CILMA افزایش حجم فایل ثبات نسبت به افزایش حجم حافظه‌های نهان سطح یک و دو تاثیر بیشتری در بهبود کارایی دارد. دلیل این موضوع این است که بارهای کاری CILMA اساساً یا دسترسی اندکی به حافظه دارند و یا دسترسی‌های آن‌ها به حافظه از هیچ الگوی خاصی پیروی نمی‌کند. این دو دلیل باعث می‌شود که سطوح حافظه‌ی نهان به خصوص حافظه‌های نهان سطح دو کارایی خود را در بهبود تاخیر دسترسی به حافظه از دست بدهد و نرخ برخورد در حافظه‌های نهان سطح دو پایین باشد. همچنین، شکل ۲۲ وضعیت کارایی بارهای کاری‌ای که وابسته به حافظه هستند را پس از اعمال هرکدام از مراحل تغییر معماری پردازنده‌ی گرافیکی، نشان می‌دهد. چنان که مشخص است در بدترین حالت برای برنامه‌ی LIB روش ما کارایی را فقط ۷ درصد کاهش داده است. این در حالی است که کارایی حتی برای بارهای کاری وابسته به حافظه نه تنها کاهش قابل توجه نداشته بلکه در اکثر موارد رشد اندکی نیز داشته است. دلیل این اتفاق این است که با این که پهنای باند حین اجرای این برنامه‌ها بهره‌وری بالایی دارد و کاهش پهنای باند به سرعت اجرای این برنامه لطمه می‌زند اما، بزرگ کردن حجم سطوح مختلف حافظه‌ی نهان به شکل قابل توجهی اجرای این برنامه‌ها را تسریع می‌کند.

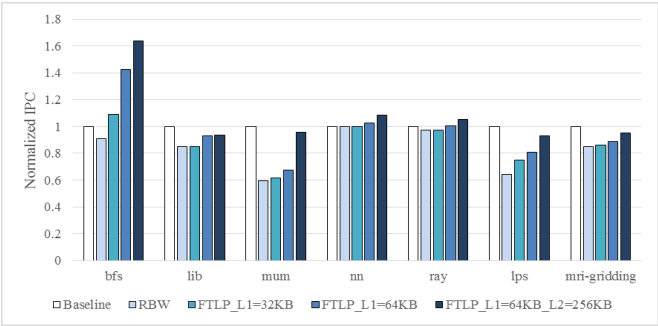
عامل دیگری که مورد بررسی قرار می‌گیرد، توان مصرفی است. در این پروژه، توان مصرفی اصلی‌ترین پارامتر محدود کننده‌ی استفاده از مساحت حاصل از کاهش پهنای باند است که باید در نظر گرفته شود. به عبارت دیگر، علاوه بر بودجه‌ی مساحت یک بودجه برای توان مصرفی داریم و تخصیص منابع می‌بایست نه تنها با توجه به مقدار مساحت موجود بلکه با در نظر گرفتن تغییرات توان مصرفی صورت گیرد. بنابراین، لازم به ذکر است که محدودیت توان مصرفی پردازنده‌ی گرافیکی مانع اصلی افزایش حجم حافظه‌ی نهان سطح دو به مقداری بیشتر از ۲۵۶ KB به ازای هر بانک این واحد بوده است. شکل ۲۳ نمودار تغییر توان مصرفی کل پردازنده‌ی گرافیکی تحت بارهای کاری مختلف را در حالت پایه و پس از کاهش پهنای باند و افزایش حجم فایل ثبات، حافظه‌ی نهان سطح یک و حافظه‌ی نهان سطح دو نشان می‌دهد. در این آزمایش علاوه بر بارهای کاری CILMA، بارهای کاری وابسته به حافظه نیز مورد بررسی قرار گرفته‌اند. همان‌طور که مشاهده می‌شود، روش پیشنهادی توان مصرفی کل را حدود ۸ درصد افزایش می‌دهد. با توجه به این که در این حالت هم کارایی و هم انرژی پردازنده‌ی گرافیکی بهبود یافته است، افزایش ۸ درصدی توان مصرفی قابل چشم پوشی است.

شکل ۲۴ نمودار توان مصرفی بخش به بخش را برای منابعی که در این پژوهش مورد بررسی قرار گرفته است، نشان می‌دهد. همان‌طور که در این شکل دیده می‌شود، توان مصرفی حافظه بیرون تراشه (DRAM) به دلیل کاهش دسترسی‌های به حافظه و همچنین کمتر شدن واحدهای مستقل کاهش چشم‌گیری داشته است. لازم به ذکر است توان DRAM، علاوه بر توان تراشه‌ی حافظه توان مصرفی کانال‌های اتصال به DRAM را نیز شامل می‌شود. از طرف دیگر توان منابع فایل ثبات (RF)، حافظه‌ی نهان سطح یک (L1) و حافظه‌ی نهان سطح دو (L2) به دلیل زیاد شدن حجم آن‌ها، افزایش پیدا کرده است. نکته‌ی قابل توجه افزایش توان مصرفی واحد اجرایی (EXE) است که این مسئله نوید افزایش توان مصرفی مفید پردازنده‌ی گرافیکی را به ما می‌دهد. در ادامه این مسئله دقیق‌تر بررسی می‌گردد. شکل ۲۵ نمودار انرژی را در حالت پایه و پس از کاهش پهنای باند و افزایش حجم فایل ثبات، حافظه‌ی نهان سطح یک و حافظه‌ی نهان سطح دو نشان می‌دهد. با اینکه تغییرات اعمال شده سبب افزایش توان مصرفی پردازنده‌ی گرافیکی شده است اما، کاهش چشم‌گیر زمان اجرا موجب شده است که در نهایت انرژی مصرفی که حاصل‌ضرب توان در زمان اجرای می‌باشد به طور متوسط ۱۴ درصد کاهش یابد.

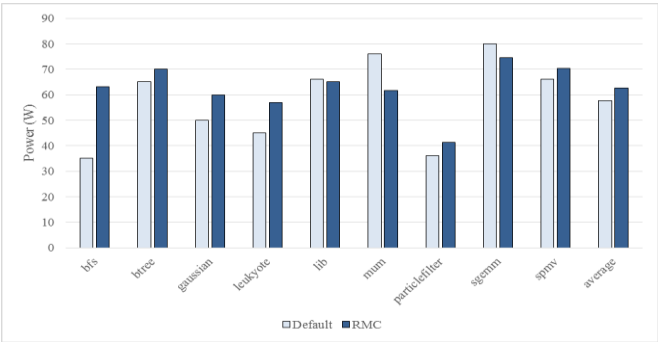
بررسی ما نشان می‌دهد که با افزایش حجم فایل ثبات، حافظه‌ی نهان سطح یک و حافظه‌ی نهان سطح دو به ترتیب به اندازه‌ی ۱.۸۷۵ MB، ۰.۷۰۳ MB و ۲.۲۵ MB در مجموع ۱۶.۶ mm^2 مساحت مورد نیاز است که این مقدار از بودجه‌ی مساحتی موجود (۳۰ mm^2) کمتر است. در واقع، بر خلاف فایل ثبات، برای انتخاب سایز مناسب حافظه‌های نهان سطح یک و دو توان مصرفی عامل اصلی محدود کننده بوده و به همین دلیل، امکان بزرگ‌تر کردن این واحدها وجود ندارد.



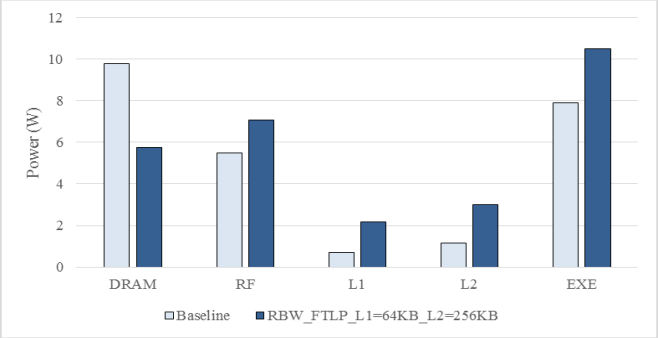
شکل ۲۱- کارایی پردازنده‌ی گرافیکی تحت بارهای کاری CILMA، از چپ به راست: ۱) در حالت پایه، ۲) پس از کاهش پهنای باند، ۳) پس از افزایش حجم فایل ثبات، ۴) پس از افزایش حجم حافظه‌ی نهان سطح یک و ۵) افزایش حجم حافظه‌ی نهان سطح دو



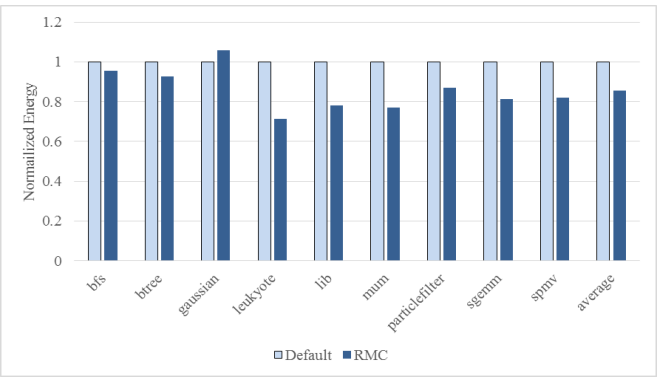
شکل ۲۲- کارایی پردازنده‌ی گرافیکی تحت بارهای کاری وابسته به حافظه، از چپ به راست: ۱) در حالت پایه، ۲) پس از کاهش پهنای باند، ۳) پس از افزایش حجم فایل ثبات، ۴) پس از افزایش حجم حافظه‌ی نهان سطح یک و ۵) افزایش حجم حافظه‌ی نهان سطح دو



شکل ۲۲- توان مصرفی کل پردازنده‌ی گرافیکی در حالت پایه و پس از کاهش پهنای باند و افزایش حجم فایل ثبات، حافظه‌ی نهان سطح یک و حافظه‌ی نهان سطح دو



شکل ۲۳- جزئیات توان مصرفی پردازنده‌ی گرافیکی در حالت پایه و پس از کاهش پهنای باند و افزایش حجم فایل ثبات، حافظه‌ی نهان سطح یک و حافظه‌ی نهان سطح دو



شکل ۲۴- تغییرات انرژی مصرفی پردازنده‌ی گرافیکی در حالت پایه و پس از کاهش پهنای باند و افزایش حجم فایل ثبات، حافظه‌ی نهان سطح یک و حافظه‌ی نهان سطح دو

۷- کارهای پیشین

در مقاله‌ی [۱۳]، به منظور کاهش تاخیر دسترسی به حافظه با استفاده از حافظه‌های STT-RAM یک معماری جدید برای حافظه نهان سطح دو ارائه شده است؛ به این ترتیب که برای افزایش ظرفیت حافظه‌ی نهان سطح دو و طبعاً کاهش تاخیر دسترسی به حافظه از حافظه‌های STT-RAM به دلیل داشتن تراکم (Density) بالا استفاده شده است. با این وجود، عملیات نوشتن در حافظه‌های STT-RAM دارای تاخیر زیاد و نیازمند انرژی بیشتری هستند. برای کاهش این سربار، با مطالعه‌ی دقیق بارهای کاری مختلف در پردازنده‌ی گرافیکی ترکیبی از STT-RAM و SRAM معمولی برای ساختار حافظه‌ی نهان سطح دو به شکلی به کار گرفته شد که نهایتاً کارایی و همچنین توان مصرفی بهبود یابد.

مقاله‌ی [۱۴] با هدف افزایش نرخ اصابت حافظه‌ی نهان سطح دو، ابتدا ثابت کرده است که روش‌های معمول پیش واکشی مانند: Stride Prefetching و Next-Line Prefetching که در CPU استفاده می‌شود، برای افزایش نرخ اصابت در پردازنده‌ی گرافیکی جوابگو نیست. سپس، با ارائه‌ی یک روش جدید پیش واکشی تلاش کرده است که نرخ اصابت حافظه‌ی نهان سطح دو و به دنبال آن کارایی پردازنده‌ی گرافیکی را افزایش دهد.

مقاله‌ی [۱۵] نشان می‌دهد که سیاست کنونی نوبت‌دهی ریسمان، قابل ترکیب با پیش واکشی نیست. دلیل اصلی این امر آن است که پردازنده‌ی گرافیکی، ریسمان‌ها را به ترتیب و پشت سر هم اجرا می‌کند که در نتیجه دسترسی‌ها به ترتیب به بلوک‌های کناری حافظه نهان می‌باشد. بنابراین، در سیکل‌های متوالی، واکشی بعدی دقیقاً پشت واکشی کنونی است. این مسئله باعث می‌شود که ۱- واکشی درخواست شده توسط یک ریسمان دارای آدرس‌هایی باشد که اندکی قبل‌تر توسط یک ریسمان دیگر درخواست شده است. ۲- نیاز داشته باشیم که Prefetcher به اندازه‌ی کافی دقیق باشد که بتواند آدرس دسترسی‌های آینده‌ی دور را پیش‌بینی کند. ایده اصلی این است که زمان اجرای ریسمان‌های پشت سر هم را جدا کرده به شکلی که این ریسمان‌ها در زمان‌های دورتر از هم اجرا شوند. پس از ایجاد این تغییر، پیش واکشی در پردازنده‌ی گرافیکی می‌تواند به شکل موثر، کارایی را افزایش دهد.

مقاله‌ی [۱۶] در ابتدا ثابت می‌کند که یکی از دلایل نرخ فقدان زیاد در سطوح حافظه‌ی نهان پردازنده‌ی گرافیکی، مشکل اشباع حافظه‌ی نهان و جایگذاری‌های بیهوده به دلیل تعداد زیاد ریسمان‌های آماده به اجرا بر روی چند پردازنده‌ی جریانی است. سپس، با این فرض تلاش نموده است که با بررسی بارهای کاری مختلف، نقطه‌ی بهینه‌ای برای تعداد این ریسمان‌های تخصیص داده شده به چند پردازنده‌ی جریانی پیدا کند به طوری که نرخ برخورد سطوح مختلف حافظه‌ی نهان به بزرگترین مقدار خود برسد.

مقاله‌ی [۱۷] در ابتدا بارهای کاری مختلف را در پردازنده‌ی گرافیکی بررسی کرده و در یافته است که نیازهای برنامه‌های مختلف به عناصر ذخیره‌ای مانند حافظه‌ی نهان، فایل ثبات و حافظه‌ی مشترک متفاوت است. سپس، به جای همه‌ی عناصر ذخیره‌ای مذکور، یک حافظه‌ی یکپارچه تعیین نموده که بسته به بارکاری پردازنده‌ی گرافیکی به حافظه‌ی نهان، فایل ثبات و حافظه‌ی مشترک مقادیر مختلفی از این حافظه تخصیص داده می‌شود.

مقاله‌ی [۱۸] با به کارگیری حافظه‌های STT-RAM تلاش نموده است که ظرفیت فایل ثبات را افزایش داده تا از این طریق تعداد ریسمان‌هایی را که همزمان به چند پردازنده‌ی جریانی تخصیص داده می‌شود، افزایش دهد. با این فرض که با بزرگتر شدن اندازه‌ی فایل ثبات، تاخیر دسترسی به هر نوع حافظه‌ای بیشتر می‌شود، در این کار مصالحه‌ی بین ظرفیت فایل ثبات و تاخیر دسترسی در نظر گرفته شده است تا بهینه‌ترین اندازه‌ی فایل ثبات تعیین گردد.

مقاله‌ی [۱۹]، با استفاده از یک ایده‌ی کامپایلری عمر متغیرها را حساب کرده و به اندازه‌ی زمان لازم به این متغیرها، ثبات اختصاص می‌دهد و به محض این که پیش‌بینی شود که دیگر به آن متغیر نیاز نیست، ثبات مربوطه آزاد می‌شود. به این ترتیب در مصرف تعداد زیادی ثبات صرفه‌جویی خواهد شد. از این ثبات‌های صرفه‌جویی شده می‌توان استفاده کرده و سطح موازات را افزایش داد.

۸- نتیجه‌گیری

پردازنده‌های گرافیکی در طول سالیان اخیر به عنوان بستر مناسبی برای پردازش موازی استفاده می‌شوند. تولیدکنندگان این پردازنده‌ها همواره در ساخت این پردازنده‌ها نسبت مستقیمی بین توان پردازشی پردازنده‌ی گرافیکی و پهنای باند آن‌ها رعایت نموده‌اند. بررسی‌های ما نشان داد برای تعداد زیادی از بارهای کاری موجود (CILMA) پهنای باند پردازنده‌ی گرافیکی بهره‌وری پایینی دارد. در این پژوهش با ارائه‌ی نتایج حاصل از شبیه‌سازی ثابت شد که نیاز موجود برای ارائه‌ی یک پردازنده‌ی گرافیکی با توان پردازشی بالا و پهنای باند پایین قابل چشم‌پوشی نیست. کاهش پهنای باند در پردازنده‌ی گرافیکی قدرتمندی مانند GTX 480 پیامدهایی از جمله کاهش توان و بدست آمدن یک فضای خالی را در سطح تراشه در پی دارد. فضای خالی بدست آمده را می‌توان برای افزایش ظرفیت‌های پردازشی و منابع ذخیره‌سازی بر روی تراشه، استفاده کرد. در این پژوهش، به معرفی روش‌هایی برای افزایش کارایی بارهای کاری CILMA با در نظر گرفتن بودجه‌ی توان و مساحت پرداختیم.

۹- مراجع

[1] NVIDIA Corporation. CUDA Programming Guide, V4.0.
[2] NVIDIA Corporation. CUDA Toolkit, 2012. Version 4.2 as of Sep. 2012, http://developer.nvidia.com/cuda/cuda-downloads.

[3] T. D. Han and T. S. Abdelrahman, "hiCUDA: High-level GPGPU programming," I*EEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 1, pp. 78-90, 2011.
[4] A. Sethia and S. Mahlke, "Equalizer: Dynamic tuning of gpu resources for efficient execution," *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, IEEE Computer Society, pp. 647-658, 2014.
[5] D. B. Kirk and W. Hwu, "Programming massively parallel processors: a hands-on approach," Morgan Kaufmann, 2012.
[6] H. Kim, R. Vuduc, S. Baghsorkhi, J. Choi, W. Hwu, "Performance analysis and tuning for general purpose graphics processing units (GPGPU)," *Synthesis Lectures on Computer Architecture*, vol. 7, no. 2, pp. 1-96, 2012.
[7] C.M. Wittenbrink, E. Kilgariff, and A. Prabhu, "Fermi GF100 GPU architecture," *IEEE Micro*, vol. 31, no. 2, pp. 50-59, 2011.
[8] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "GPUWattch: enabling energy optimizations in GPGPUs," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 487-498, 2013.
[9] J. Lim, N.B. Lakshminarayana, H. Kim, W. Song, S. Yalamanchili and W. Sung, "Power modeling for GPU architectures using McPAT," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 19, no. 3, pp. 1-24, 2014.
[10] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," I*EEE International Symposium on Workload Characterization (IISWC)*, IEEE, pp. 44-54, 2009.
[11] J. A. Stratton, C. Rodrigues, I. Sung, N. Obeid, L. Chang, N. Anssari, G. D. Liu, and W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *Center for Reliable and High-Performance Computing*, vol. 127, 2012.
[12] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, pp. 163-174, 2009.
[13] M.H. Samavatian, M. Arjomand, R. Bashizade and H. Sarbazi-Azad, "Architecting the last-level cache for GPUs using STT-RAM technology," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 4, pp. 1-24, 2015.
[14] A. Jog, O. Kayiran, A.K. Mishra, M.T. Kandemir, O. Mutlu, R. Iyer and C.R. Das, "Orchestrated scheduling and prefetching for GPGPUs," *In Proceedings of the 40th Annual International Symposium on Computer Architecture*, pp. 332-343, 2013.
[15] A. Jog, O. Kayiran, N. Chidambaram Nachiappan, A.K. Mishra, M.T. Kandemir, O. Mutlu, R. Iyer and C.R. Das, "OWL: cooperative thread array aware scheduling techniques for improving GPGPU performance." *ACM SIGPLAN Notices*, vol. 48, no. 4, pp. 395-406, 2013.
[16] T.G. Rogers, M. OConnor, and T.M. Aamodt, "Cache-conscious wavefront scheduling," *In 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 72-83, IEEE, 2012.
[17] M. Gebhart, S.W. Keckler, B. Khailany, R. Krashinsky, and W.J. Dally, "Unifying primary cache, scratch, and register file memories in a throughput processor," *In 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 96-106, IEEE, 2012.
[18] X. Liu, M. Mao, X. Bi, H. Li, and Y. Chen, "An efficient STT-RAM-based register file in GPU architectures," *In The 20th Asia and South Pacific Design Automation Conference*, pp. 490-495, IEEE, 2015.
[19] H. Jeon, G.S. Ravi, N.S. Kim, and M. Annavaram, "GPU register file virtualization," *In Proceedings of the 48th International Symposium on Microarchitecture*, pp. 420-432, 2015.

هژیر باخویشی مدرک کارشناسی و کارشناسی‌ارشد خود را در رشته مهندسی کامپیوتر از دانشگاه‌های صنعتی امیرکبیر و صنعتی شریف در سال‌های ۱۳۹۲ و ۱۳۹۵ کسب کرد. از حوزه های تحقیقاتی مورد علاقه وی می‌توان به معماری سیستم های چند هسته ای و پردازنده های گرافیکی اشاره کرد. آدرس پست الکترونیکی ایشان عبارت است:
bakhishi@ce.sharif.edu

نگار اکبرزاده مدرک کارشناسی و کارشناسی‌ارشد خود را در رشته مهندسی برق گرایش الکترونیک از دانشگاه شهید بهشتی در سال‌های ۱۳۹۳ و ۱۳۹۵ کسب کرد. وی در حال حاضر دانشجوی دکتری رشته مهندسی کامپیوتر دانشگاه صنعتی شریف است. از حوزه‌های تحقیقاتی مورد علاقه وی می‌توان به معماری کامپیوتر، حافظه های مقاومتی و پردازنده‌های گرافیکی اشاره کرد. آدرس پست الکترونیکی ایشان عبارت است:
akbarzadeh@ce.sharif.edu

سید محمد صدرالساداتی در حال حاضر پژوهشگر پسادکترا در پژوهشگاه دانش‌های بنیادی است. او مدرک کارشناسی، کارشناسی ارشد و دکترای خود را در رشته‌ی مهندسی کامپیوتر به ترتیب در سال‌های ۱۳۹۱، ۱۳۹۳ و ۱۳۹۸ از دانشگاه صنعتی‌شریف کسب کرد. از حوزه‌های تحقیقاتی مورد علاقه وی می‌توان به معماری سیستم‌های چندهسته‌ای و زیادهسته‌ای، سیستم‌های حافظه و پردازش داخل حافظه اشاره کرد. وی در حوزه‌های تحقیقاتی خود مقالات متنوعی در همایش‌ها و ژورنال‌های معتبر بین‌المللی به چاپ رسانده است. در نتیجه تحقیقات عمیق وی در بهبود کارآمدی مصرف انرژی در پردازنده‌های گرافیکی، او در سال ۱۳۹۹ به عنوان برگزیده‌ی اول جشنواره جوان خوارزمی در بخش پژوهش‌های بنیادی انتخاب شد. آدرس پست الکترونیکی ایشان عبارت است:
sadrosadati@ipm.ir

حمید سربازی‌آزاد مدرک کارشناسی خود را در رشته مهندسی کامپیوتر از دانشگاه شهید بهشتی در سال ۱۹۹۲ اخذ نمود. وی مدرک کارشناسی‌ارشد و دکتری خود را در رشته مهندسی کامپیوتر به ترتیب در سال‌های ۱۹۹۴ و ۲۰۰۲ از دانشگاه های صنعتی شریف و گلاسگو انگلستان کسب کرد. او هم اکنون استاد تمام دانشگاه صنعتی شریف و استاد پژوهشگر در پژوهشگاه دانش های بنیادی است. از علایق پژوهشی وی می‌توان به معماری کامپیوتر، معماری حافظه، شبکه و سیستم های روی تراشه، سیستم های موازی و توزیع شده، مدلسازی و ارزیابی کارایی و سیستم های ذخیره سازی اشاره کرد. وی دارای بیش از ۳۰۰ مقاله منتشر شده در کنفرانس و مجلات معتبر بین المللی است. او همچنین، موفق به کسب جایزه بین المللی خوارزمی در سال ۲۰۰۶ و جایزه دانشمند جوان TWAS در سال ۲۰۰۷ شده و در سال‌های ۲۰۰۴، ۲۰۰۷، ۲۰۰۸، ۲۰۱۰ و ۲۰۱۳ به عنوان پژوهشگر برتر دانشگاه صنعتی شریف انتخاب شده است. آدرس پست الکترونیکی ایشان عبارت است:
azad@sharif.edu , azad@ipm.ir

[1] Thread
[2] Warp
[3] Single Instruction Multiple Data (SIMD)
[4] Streaming Multiprocessor (SM)
[5] Compute-Intensive Low Memory Access (CILMA)
[6] Fermi
[7] Single Instruction Multiple Thread (SIMT)
[8] Cache
[9] Network-On-Chip (NOC)
[10] Memory Controller
[11] Off-Chip Memory
[12] Reduced Bandwidth (RBW)

The CSI Journal on Computing Science and Information Technology
Vol. 18, No.2, 2021

Improving Performance of GPUs through Proper Utilization of Memory Controllers

Hazhir Bakhishi[1], Negar Akbarzadeh[2], Mohammad Sadrosadati[3], Hamid Sarbazi-Azad[4]

[1,2,4] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[3,4] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

Abstract

With the advent of CUDA architecture, Graphic Processing Units (GPUs) have become a desirable platform for general purpose processing. In recent years, bandwidth of GPUs has been increased constantly with the growth of processing capability and on-chip resources. At first sight, the direct relation between the performance and bandwidth of the GPUs seems rational. Since workloads process the amount of data that they read from the memory. However, by looking deeper at different workloads, we can find out that a significant number of workloads underutilize bandwidth. This is while the utilization of processing cores and on-chip resources is still high. Therefore, it is not necessary to maintain the direct relation between the performance and bandwidth of the GPU for every workload. In this research, we first propose a new approach to design high-performance low-bandwidth GPUs based on the obtained observations and then investigate the opportunities of the freed-up space arising from the lower bandwidth and provide solutions to utilize this space.

Keywords: General Purpose Graphics Processing Unit (GPGPU); bandwidth; processing capability; improving performance.