فصلنامه علمی- پژوهشی
فناوری اطلاعات و ارتباطات ایران

سال نهم، شماره‌های ۳۱ و ۳۲، بهار و تابستان ۱۳۹۶
صص: ۷۸- ۶۷

بررسی تاثیر تنظیمات پارامترهای سخت‌افزاری بر انرژی مصرفی در الگوریتم ضرب برداری ماتریس‌های تنک بر روی پردازنده‌های گرافیکی

[*] مینا عاشوری [**] فرشاد خون‌جوش
* کارشناسی ارشد، مهندسی معماری کامپیوتر، دانشگاه شیراز، شیراز
** استادیار، دانشکده مهندسی برق و کامپیوتر، دانشگاه شیراز، شیراز
تاریخ دریافت: ۱۳۹۷/۰۴/۰۷ تاریخ پذیرش: ۱۳۹۷/۰۹/۲۰

چکیده

ضرب برداری ماتریس‌های تنک الگوریتمی ساده اما بخش بسیار مهمی از برنامه‌های جبر خطی و علمی در حوزه‌ی ریاضی و فیزیک است و به دلیل طبیعت قابل موازی سازی آن، پردازنده‌های گرافیکی یکی از گزینه‌های بسیار مناسب و مهم برای انتخاب بستر اجرایی آن است. در طی سال‌های اخیر با توجه به تاکید محققان برای در نظر گرفتن انرژی مصرفی به عنوان یکی از اهداف اصلی طراحی در کنار کارآیی، تلاش‌های بسیار کمی جهت بهبود انرژی مصرفی این الگوریتم بر روی پردازنده‌ی گرافیکی انجام شده است. در این مقاله از منظر بهینگی مصرف انرژی در کارآیی به دست آمده، به این مسئله پرداخته شده است.

با بهره وری از قابلیت تنظیم پیکربندی که در پردازنده‌های گرافیکی مدرن معرفی شده است، با بررسی آماری رفتار این الگوریتم هنگام استفاده از قالب‌های مختلف ذخیره سازی ماتریس تنک و تنظیمات مختلف سخت افزاری برای بیش از ۲۰۰ ماتریس نمونه‌ی تنک، بهترین تنظیمات پیکربندی برای الگوریتم ضرب برداری ماتریس تنک با قالب‌های مختلف ذخیره سازی بر روی پردازنده‌ی گرافیکی به دست آمده است. این پیکربندی برای هر قالب ذخیره سازی، به گونه‌ای انتخاب شده است که در تمام نمونه‌های بررسی شده به عنوان بهترین پیکربندی نتیجه داده باشد.

واژه‌های کلیدی: ضرب برداری ماتریس‌های تنک، انرژی مصرفی، کارآیی، قالب‌های ذخیره سازی تنک، پردازنده‌ی گرافیکی.

مقدمه

امروزه مسائل علمی و مهندسی مدرن، در طیف گسترده‌ای از بـرنامه‌های کاربـردی روزانـه و علمـی با دقت بالا استفاده می‌شوند که شامل مقیاس بسیار بزرگی از برنامه‌های شبیه‌سازی مثل فیزیک انرژی بالا[1] شبیه‌سازی‌های علمی،

[1] High Energy physics

نویسندهٔ عهده‌دار مکاتبات: فرشاد خون‌جوش khunjush@shirazu.ac.ir

داده کاوی[۲] [۱]، جبرخطی [۲]، تجزیه و تحلیل گراف‌ها [۳]، پیش بینی آب و هوا و زلزله و ... می‌شوند که در همه‌ی آن‌ها، ماتریس‌های تنک کاربرد دارند.

از دهه‌ی گذشته تا به امروز از روش محاسبات موازی استفاده می‌شود تا بتوان مسائل پیچیده را به روشی مقرون به صرفه حل کرد به طوری‌که بار کار[۳] بتواند بین پردازنده‌های مختلف تقسیم شود.

در سال‌های اخیر، پردازنده‌های گرافیکی[۴] به دلیل هزینه‌ی پایین کارآیی در واحد[۵] و میزان بالای کارآیی در واحد توان[۶] به یکی از دستگاه‌های اصلی برای محاسبات با کارآیی[۷] بالا تبدیل شده‌اند [۴]. قدرت محاسباتی بالا و پهنای باند این پردازنده‌ها موجب شده تا بتوان به عنوان جای‌گزینی برای سیستم‌های توزیع‌شده سنتی و یا پردازنده‌های چند هسته ای، در طیف وسیعی از شاخه‌های علوم مختلف از آن‌ها بهره جست. پردازنده‌های گرافیکی با استفاده از هزاران هسته[۸] و نخ[۹] حجم بسیاری از موازات را فراهم می آورند که سبب افزایش کارایی خواهند شد و برای برنامه‌های علمی موازی، سود کارآیی با اجرا روی پردازنده‌های گرافیکی، به صورت مستقیم قابل دستیابی است. به همین دلیل، اخیرا محاسبات علمی، توسط جامعه‌ی پژوهش، برای اجرا به وسیله‌ی پردازنده‌های گرافیکی انطباق داده شده‌اند.

معماری یکپارچه‌ی دستگاه محاسبه[۱۰] (CUDA) به وسیله‌ی NVIDIA در سال ۲۰۰۶ به منظور بهره‌برداری از قدرت محاسباتی پردازنده‌های گرافیکی مطرح شد که یک زبان برنامه‌نویسی سطح بالا بر اساس پردازنده‌ی گرافیکی همه منظوره[۱۱] است و از آن زمان تاکنون برای برنامه نویسی پردازنده‌های گرافیکی مورد استفاده‌ی همگانی قرار می‌گیرد.

ضرب برداری ماتریس‌های تنک بخشی کوچک اما با اهمیت بسیار زیاد در زیربرنامه‌های اساسی تنک جبر خطی[۱۲] می‌باشد. بیشترین هزینه محاسباتی در بسیاری از روش‌های تکراری[۱۳] برای حل کردن سیستم‌های خطی در مقیاس بالا و مسائل با پاسخ خاص، مربوط به ضرب برداری ماتریس‌های تنک می‌باشد که در بسیاری از برنامه‌های کاربردی با اهمیت بالا مثل سیستم‌های شبیه سازی، تصویربرداری پزشکی، بازیابی اطلاعات، مدل‌سازی تغییرات اقلیمی و اقتصادی و بسیاری دیگر از طیف‌های برنامه‌های کاربردی وجود دارد.

ماتریس‌های تنک متفاوتی که از برنامه‌های کاربردی متفاوت استخراج شده‌اند هرکدام دارای الگوی تنک بودن خاصی هستند و الگوی یکسانی ندارند. پس نمی‌توان از یک قالب یکسان برای ذخیره کردن همه استفاده کرد.

ماتریس‌های متفاوت برای داشتن بهترین کارآیی در عملیات ضرب برداری نیاز به قالب نمایش مناسب برای الگوی تنک بودن خود دارند. با توجه به جنبه‌ی بی قاعده‌ی این عملیات، هم در دستیابی به حافظه و هم در جریان کنترل[۱۴] کارآیی ضرب برداری ماتریس‌های تنک به هردو پارامتر الگوی تنک بودن ماتریس ورودی و ویژگی‌های سخت‌افزاری بستگی دارد. برای حل این مشکل تکنیک‌های بسیاری جهت بهینه سازی کارآیی ضرب برداری ماتریس‌های تنک انجام گرفته است که شامل استفاده از قالب‌های متفاوت [۵][۶] و روش‌های متفاوت تنظیم خودکار[۱۶] با تنظیم کردن پارامترهای مختلف مطابق ساختار ماتریس است [۷].

البته اکثر تلاش‌ها برای بهبود ضرب برداری ماتریس‌های تنک در زمینه‌ی کارآیی آن‌ها بوده و در حالی‌که این عملیات برروی پردازنده‌های گرافیکی، با استفاده از قالب‌های

---

[2] Data mining
[3] Work-load
[4] Graphic processing units
[5] Performance per unit
[6] Performance per watt
[7] High-performance computing
[8] Core
[9] Thread
[10] Compute unified device architecture

[1] General-purpose GPU
[2] Basic sparse linear algebra(BLAS)
[3] Iterative methods
[4] Control flow
[5] Format
[6] Auto-tuning

ذخیره‌سازی مختلف، کارآیی نسبتا مناسبی دارد اما تلاش چندانی برای بهبود مصرف انرژی آن انجام نشده است. برای مثال در [8] برای تخمین کارآیی و انتخاب بهترین قالب از مدل کردن کارآیی استفاده شده است.

بهینگی در مصرف انرژی، در حالی‌که صنعت نیمه هادی از پردازنده‌های چند هسته ای[17]به سمت پردازنده‌های با هسته‌های زیاد[18]سوق پیدا می‌کند، به یک نیاز حیاتی در طراحی تبدیل شده است. طراحان در حال ترکیب ویژگی‌های سخت‌افزاری و نرم‌افزاری در این شتاب‌دهنده‌های بازده‌گرا[19]هستند تا بتوان برنامه‌های کاربردی بدون ساختار با الگوهای متفاوت و بی‌قاعده‌ی دستیابی به حافظه را بر روی این پردازنده‌ها به خوبی اجرا کرد.

در حالی‌که مطالعات اخیر پیشنهاد می‌کند که بهینگی در مصرف انرژی به عنوان یک هدف اصلی در کنار کارآیی در طراحی سخت‌افزار و نرم‌افزار قرار بگیرد، اغلب کارهایی که برای بهبود ضرب برداری ماتریس‌های تنک انجام گرفته در جهت بهبود کارآیی به تنهایی بوده است و مصرف انرژی را شامل نشده است. مطالعات بسیار کمی جهت بررسی مصرف انرژی اجرای این هسته[20]وی پردازنده گرافیکی انجام شده است [9][10]. هیچکدام از این مطالعات، ارتباط بهینگی مصرف انرژی در این هسته را با الگوی تنک بودن ماتریس ورودی و ویژگی‌های سخت افزاری بررسی مطالعه نکرده‌اند. در [۱۱] به بررسی ارتباط ساختار ماتریس ورودی با انتخاب قالب‌های ذخیره‌سازی و تاثیر آن برروی توان مصرفی پرداخته شده است ولی در نهایت تابه‌حال هیچ تلاشی انجام نشده است که دو هدف کارآیی وبهینگی انرژی مصرفی را در طراحی یک سیستم بهینه برای ضرب برداری ماتریس‌های تنک بر روی پردازنده‌های گرافیکی دنبال کند و بین این دو هدف متضاد مصالحه[21]ی ایجاد کند. در حالی‌که امروزه محققان تاکید می‌کنند بهینگی انرژی باید به عنوان یکی از اهداف اصلی در طراحی، در کنار کارآیی قرار بگیرد، تلاش‌های بسیار کمی در جهت مطالعه‌ی مصرف انرژی در عملیات ضرب برداری ماتریس‌های تنک و بهبود آن انجام گرفته است.

هدف از انجام این تحقیق، بررسی تاثیر تنظیمات سخت افزاری برروی انرژی مصرفی در واحد کارآیی برای مسئله ی انتخاب قالب ذخیره سازی برای ضرب برداری ماتریس‌های تنک برروی پردازنده‌های گرافیکی کپلر[22]است. معیار بهینگی در این تحقیق مقدار کارآیی در واحد توان مصرفی[23] است که عبارت است از نسبت کارآیی عملیات[24]ضرب برداری ماتریس تنک به توان مصرف شده در این عملیات. به عبارتی دیگر علملیاتی با کارآیی بالاتر و توان مصرفی کمتر مطلوب تر است. در این تحقیق مقدار کارآیی معادل زمان خالص اجرای هسته‌ی ضرب برداری ماتریس تنک برروی پردازنده‌ی گرافیکی تعریف شده و توان مصرفی هم بر حسب میلی وات[25]توسط حسگرهای[26] پردازنده‌ی گرافیکی کپلر اندازه گیری شده و بر حسب وات[27]گزارش شده است.

در فصل ۲ به صورت خلاصه به پژوهش‌های پیشینی که در این زمینه انجام شده پرداخته می‌شود و در فصل ۳ پیش‌زمینه‌ای درباره‌ی پردازنده‌های گرافیکی، قالب‌های ذخیره‌سازی ماتریس تنک و زبان برنامه نویسی CUDA ارائه می‌شود. در فصل ۴ روش انجام این پژوهش توضیح داده شده و در فصل ۵ نتایج آزمایشات این پژوهش ذکر شده است. در فصل ۶ نیز نتیجه گیری و کارهایی که درآینده می‌توان با الهام از این پژوهش انجام شود بیان شده است.

1 Multi-core processors
1 Many-core processors
1 Throughput-oriented accelerators
2 Kernel
2 Trade-off
2 Kepler
2 Performance per watt
2 Operation
2 Milli-watt
2 Sensor
2 Watt

۲-پژوهش‌های پیشین

یکی از اولین پیاده‌سازی‌های عملیات ضرب برداری ماتریس‌های تنک برروی پردازنده‌های گرافیکی توسط بلتز و همکاران [۸] انجام گرفت. اما کار منحصر بفردی که در رابطه با سرعت بخشیدن به این هسته برای اجرا برروی پردازنده‌های گرافیکی با قابلیت اجرای CUDA انجام گرفت توسط بل و گارلند در [۵][۱۲] انجام گرفت که در این پژوهش، آن‌ها یک مطالعه‌ی دقیق از قالب‌های ماتریس‌های تنک، الگوی دسترسی[۲۸]آن‌ها در پردازنده‌های گرافیکی و هسته‌ی عملیات‌های با قالب‌های کلاسیک و پیاده سازی شده با CUDA برای اجرا برروی پردازنده‌های گرافیکی ارائه دادند. این قالب‌های کلاسیک عبارت بودند از COO,DIA,ELL,CSR و HYB .

تعدادی از پژوهش‌هایی که اخیرا انجام شده است با استفاده از قالب‌های زیادی که برای ماتریس تنک تعریف شده است سعی در بهبود کارآیی عملیات ضرب برداری ماتریس‌های تنک بر روی پردازنده‌ی گرافیکی داشته‌اند. اغلب آن‌ها به دنبال بهبود کارآیی بوسیله‌ی انتخاب قالب مناسب در زمان اجرای عملیات بوده‌اند. اما تلاش‌های بسیار کمی برای بهینگی مصرف انرژی هسته‌ی ضرب برداری ماتریس‌های تنک برروی پردازنده‌های گرافیکی انجام شده است.

در [۱۳] نویسندگان میزان کارآیی در واحد وات را برای سه هسته از جمله هسته‌ی ضرب برداری ماتریس‌های تنک برروی دو سکوی پردازنده‌ی[۲۹] Intel Sandy Bridge و پردازنده‌ی گرافیکی فرمی NVIDIA مطالعه کرده‌اند. آن‌ها نشان داده‌اند که با معیار کارآیی در واحد وات GFLOPs/watt این هسته برروی پردازنده‌ی Intel Sandy Bridge بهتر عمل می‌کند ولی آن‌ها فقط از یک نوع ماتریس R-MAT استفاده کرده و برای ذخیره سازی آن‌ها فقط از قالب CSR استفاده کرده‌اند. در [۱۴] انزت و همکاران مرزهای جدیدی را برای بهینگی انرژی و کارآیی برای محاسبات تنک بر روی ابرکامپیوترهای مبتنی بر پردازنده‌ی گرافیکی ارائه داده‌اند. LOBPCG به عنوان محک[۳۰]انتخاب شده است چون شامل عملیات ضرب برداری ماتریس‌های تنک نیز بوده است.

در باب اندازه‌گیری توان مصرفی در پردازنده‌های گرافیکی، نویسندگان [۱۵] مطالعه‌ای درباره‌ی چگونگی اندازه گیری توان لحظه‌ای و انرژی مصرفی با استفاده از حسگرهای روی بورد پردازنده‌های گرافیکی جدید ارائه داده‌اند. برای پردازنده‌های گرافیکی بدون سنسور تعبیه شده می‌توان از مدلسازی‌هایی که برای توان صورت گرفته بهره برد یا از دستگاه‌های خارجی جهت اتصال به پردازنده‌ی گرافیکی و اندازه گیری توان مصرفی آن استفاده کرد [۱۶].

۳-مبانی نظری پژوهش

در این فصل به معرفی معماری پردازنده‌های گرافیکی و نحوه‌ی اجرای یک مدل برنامه‌نویسی رایج که همان CUDA است و همچنین عملیات ضرب برداری ماتریس-های تنک یا به اختصار[۳۱]SpMV و قالب‌های آن خواهیم پرداخت.

۳-۱- پردازنده‌های گرافیکی همه منظوره یا GPGPUS و مدل برنامه نویسی CUDA

مدل معماری GPGPU ارائه شده توسط شرکت NVIDIA مبتنی بر یک آرایه‌ای مقیاس پذیر از چندپردازنده‌هی جریانی[۳۲]و چند نخی[۳۳]که هرکدام شامل تعدادی ثابت از پردازنده‌های عددی، یک یا چند واحد واکشی، سخت افزار ویژه‌ی توابع خاص و حافظه‌ی سریع روی تراشه است که در نسل‌های کپلر و نسل‌های قبلی فرمی به حافظه‌ی نهان سطح یک[۳۴]و حافظه‌ی مشترک تقسیم می‌شود و در

---

[0] Benchmark
[1] Sparse matrix-vector multiplication
[2] Streamed multi-processor
[3] Multi-threaded
[4] L1 cache
[8] Access pattern
[9] CPU

نسل‌های جدیدتر ماکسول و پاسکال به صورت فیزیکی جدا شده‌اند.

یک برنامه‌ی CUDA شامل یک برنامه‌ی میزبان و هسته است که برنامه‌ی میزبان روی CPU میزبان اجرا می‌شود و برنامه هسته روی دستگاه GPU اجرا می‌شود.

محاسبات توسط نخ‌ها انجام می‌شود که در بلوک‌ها گروه‌بندی می‌شوند. بیشتر از یک بلوک می‌تواند برروی یک چند پردازنده‌ی یکسان اجرا شوند و بلوک‌های نخی به صورت هم‌روند اجرا می‌شوند.

طی فراخوانی هسته (شبکه[3]نیز نامیده می‌شود)، برنامه‌ی میزبان تنظیمات اجرایی را که شامل موارد زیر است تعریف می‌کند:

- تعداد بلوک‌های نخی برای اجرا
- تعداد نخ‌ها در هر بلوک

۲-۳-قالب‌های ذخیره‌سازی تنک

در ضرب برداری ماتریس‌های تنک، هر المان ماتریس فقط یک بار مورد دسترسی قرار می‌گیرد. قالب‌های عمومی برای نگهداری و دخیره کردن اطلاعات المان‌های غیر صفر ماتریس تنک محتمل سرباری زیادی می‌شوند. یکی از دلایل استفاده از قالب‌های تنک کاهش این سربار و کاهش دسترسی‌های غیر منظم، غیر تلفقی و کاهش واگرایی در نخ‌های یک ریسمان است. در پیاده‌سازی هایی که در این پژوهش انجام شده است برای ذخیره‌ی بردار ضرب شونده‌ی x از حافظه‌ی Texture بهره وری شده است.

قالب CSR: قالب CSR محبوب ترین قالب برای ذخیره کردن ماتریس تنک است. این قالب مقادر غیر صفر و شاخص ستونی آن‌ها را به صورت غیر ضمنی در دو آرایه ذخیره می‌کند و از یک آرایه‌ی سوم برای ذخیره‌ی مرزهای هر سطر استفاده می‌کند. پیاده سازی این قالب را به دو صورت عددی و برداری می‌توان انجام داد که در نوع برداری مشهور به VCSR از روش parallel reduction استفاده می‌شود.

قالب ELL : ایده‌ی قالب ELL دسته بندی کردن همه‌ی اعداد غیر صفر ماتریس از سمت چپ آن و دخیره‌ی ماتریس متراکم نتیجه است. این قالب از دو آرایه برای ذخیره‌ی المان‌های ماتریس متراکم و ذخیره‌ی شاخص ستونی این المان‌ها استفاده می‌کند.

قالب SELL: ایده‌ی آن بریدن ماتریس در تکه‌های هم طول ولی با عرض متفاوت است. این قالب از ۳ آرایه استفاده می‌کند. آرایه‌ی اول اشاره‌گرهای شروع هر Slice را ذخیره می‌کند. آرایه‌ی دیگر نیز مشابه قالب ELL است.

قالب HYB: برای ماتریسی که طول سطرهایشان متغیر است می‌توان از ترکیب دو قالب ELL و CSR برای ذخیره‌ی ماتریس استفاده کرد. زیرا برای سطرهایی با طول کمتر از طول ریسمان استفاده از قالب ELL باعث افزایش کارآیی و برای سطرهایی با طول بزرگتر از ریسمان استفاده از قالب CSR باعث بهبود کارآیی می‌شود. الگوریتم استفاده شده در این قالب تقسیم ماتریس به دو قسمت و ذخیره‌ی هر قسمت در هرکدام از این قالب‌ها است.

قالب BCSR: این قالب از سه آرایه تشکیل شده است. آرایه‌ی اول حاوی اشاره‌گرهایی به ابتدای هر ابر سطر است. با تقسیم ماتریس به بلوک‌های ۲×۲ هر ۲ سطر در یک ابر سطر قرار می‌گیرند. آرایه‌ی دوم نیز مشابه CSR است اما در این قالب به جای ذخیره‌ی پشت سرهم المان‌ها در این آرایه، المان‌های هر بلوک به ترتیب بلوک‌ها بصورت پشت سرهم ذخیره می‌شوند. آرایه‌ی سوم نیز شاخص هر بلوک را در هر ابر سطر ذخیره می‌کند.

قالب BELL: این قالب یکی از نسخه‌های ELL است و در آن به جای ذخیره‌ی المان‌ها به صورت پشت سرهم، ماتریس به بلوک‌های هم اندازه تقسیم شده وبلوک‌های حاوی اعداد غیر صفر به صورت پشت سرهم مشابه قالب ELL ذخیره می‌شوند. هر بلوک و المان‌های درونش فقط به یک شاخص نیاز دارند پس دسترسی به حافظه کاهش می‌یابد. دسترسی به بردار ورودی برای همه‌ی المان‌های ابر سطرکاهش یافته و در بهترین حالت فقط یک بار اتفاق می‌افتد. این قالب از دو آرایه تشکیل شده است. آرایه‌ی اول شاخص ستونی هر بلوک را در ماتریس تقسیم شده به بلوک‌ها ذخیره می‌کند.

[3] Grid [5]

آرایه‌ی دوم المان‌های بلوک‌ها را پشت سرهم و مشابه قالب ELL ذخیره می‌کند.

قالب DIA: همان طور که از نام قالب DIA که مخفف قطری است مشخص است، این قالب مناسب ماتریس‌هایی با ساختار قطری است زیرا قطرهای ماتریس را به صورت ستونی در یک ماتریس فشرده ذخیره می‌کند.

قالب CDS: این قالب جهت بهره‌وری از ساختار بلوکی در ماتریس‌های محاسبات شبکه‌ای طراحی گردیده است. با تقسیم ماتریس به بلوک‌های هم اندازه، در این قالب به جای ذخیره‌ی هر قطر غیر صفر ماتریس، هر ابر قطر ذخیره می‌شود.

۴-روش‌شناسی

در [۱۷] نشان داده شده است که انتخاب قالب مناسب برای ذخیره‌سازی ماتریس تنک تاثیر بسیار زیادی برروی کارآیی خواهد داشت. از طرفی مصرف انرژی در پردازنده‌ی گرافیکی در رابطه‌ی مستقیم با کارآیی است پس می‌توان گفت که انتخاب قالب متناسب با ساختار ماتریس ورودی، می‌تواند با بهبود دسترسی به حافظه، کاهش سربار محاسباتی و استفاده از پهنای باند حافظه، استفاده‌ی بهینه مناسب از حافظه‌ی نهان و ... باعث بهبود در مصرف انرژی شود. علاوه بر انتخاب قالب مناسب، پارامترهای قابل تنظیمی در پردازنده‌ی گرافیکی وجود دارند که نتایج این پژوهش و پژوهش‌های دیگر نشان داده‌اند که با تنظیم صحیح آن‌ها می‌توان تاثیر مثبتی در بهبود کارآیی و توان مصرفی ایجاد کرد. این پارامترهای سخت افزاری عبارتند از :

۱.اندازه‌ی بلوک نخی که تعداد نخ‌های هر بلوک را مشخص می‌کند

۲.تعداد ثبات‌های اختصاص داده شده به هر نخ

۳.سلسله مراتب حافظه

اندازه‌ی بلوک نخی پارامتری است که به شدت برروی کارآیی و توان مصرفی پردازنده‌ی گرافیکی تاثیر می‌گذارد.

تعریف دقیق تعداد نخ‌ها در یک بلوک نقش اساسی در رسیدن به معیار کارآیی در وات و اشغال[۳۶]پردازنده‌ی گرافیکی دارد. درصد اشغال که نرخ بهره‌وری از هر SM[۳۷] را نشان می‌دهد بسیار وابسته به تعداد نخ‌ها در بلوک نخی است زیرا برای تعداد بلوک فعال در هر SM محدودیتی وجود دارد و با تعریف مقدار صحیح اندازه‌ی این بلوک‌ها میتوان تعداد نخ‌های فعال در هر SM را به مقدار حداکثر آن نزدیک کرده و درصد اشغال SM و در نتیجه بهره‌وری از آن را افزایش داد. البته درصد اشغال به محدودیت منابعی دیگر نظیر تعداد ثبات‌ها برای هر نخ و حافظه‌ی مشترک موجود نیز بستگی دارد.

تعداد بلوک‌های فعال در هر SM به محدودیت منابعی مثل تعداد ثبات‌ها نیز بستگی دارد. در پردازنده‌های گرافیکی با قابلیت محاسباتی[۳۸] ۳ به بالا می‌توان به وسیله‌ی کامپایلر تعداد ثبات‌ها را از حداقل ۱۶ تا حداقل ۲۵۵ تعریف کرد. کاهش تعداد ثبات‌ها برای هر نخ تا تعداد مورد نیازشان می‌تواند باعث شود که محدودیت ثبات‌های موجود کمتر شود و نخ‌های فعال بیشتری برای اجرا روی پردازنده‌ی گرافیکی وجود داشته باشد و موازی بودن در سطح نخ[۳۹] بیشتری ایجاد کرد. مقدار ثبات‌ها برای هر نخ را می‌توان با پرچم maxrregcount[۴۰] در زمان کامپایل تعریف کرد.

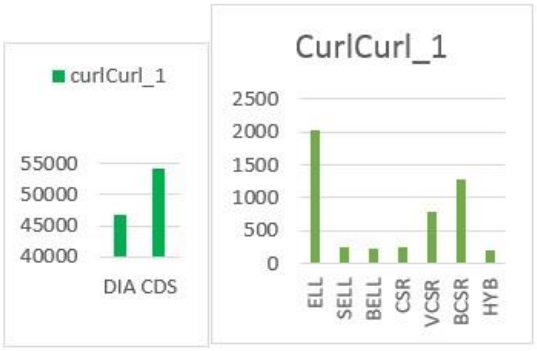
قابلیت تنظیم سلسله مراتب حافظه در پردازنده‌های گرافیکی باعث شده است که برنامه‌های کاربردی محدود به حافظه بتوانند کارآیی و توان مصرفی بهتری بر روی پردازنده‌ی گرافیکی داشته باشند. پردازنده‌های گرافیکی نسل‌های قبل فقط دارای حافظه‌های ثابت[۴۱] بافته بودند اما پردازنده‌های گرافیکی نسل‌های جدید دارای قابلیت‌های نهان سازی سطح اول و دوم یا همان L1 cache و L2 chache نیز هستند. اندازه‌ی حافظه‌ی نهان سطح اول یا

---

3 Occupancy
6
3 Streaming multiprocessor
7
3 Compute capability
8
3 Thread-level parallelism
9
4 Flag
0
4 Constant
1

L1 cache و حافظه‌ی اشتراکی توسط برنامه نویس قابل تنظیم است. عملیات ضرب برداری ماتریس تنک نیز یک برنامه‌ی محدود به حافظه است و معمولا به دلیل دسترسی-هایس به حافظه‌ی سراسری، تاخیر اجرایی و متوسط توان مصرفی نیز به تبع آن افزایش پیدا می‌کند. با معرفی پردازنده‌های گرافیکی مبتنی بر حافظه‌ی نهان و قابلیت تنظیم پیکربندی حافظه‌ی اشتراکی و نهان، می‌توان این نوع دسترسی را مدیریت کرد. برای برنامه‌های با الگوی دسترسی منظم به حافظه بهتر است اندازه‌ی حافظه‌ی اشتراکی را افزایش داد و برای برنامه‌های با الگوی دسترسی نامظم بهتر است اندازه‌ی حافظه‌ی نهان را بیشتر کرد تا بهره‌وری از این حافظه با توجه به الگوی دسترسی نامظم افزایش یابد [۱۸].

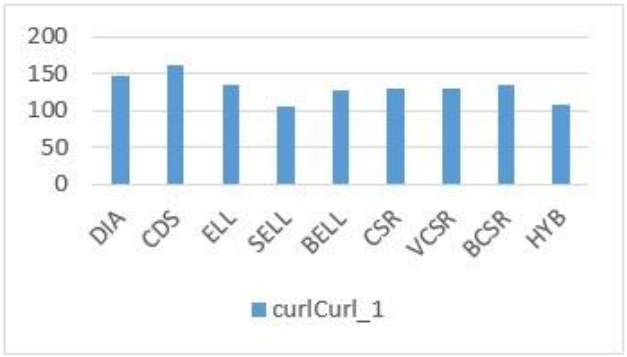
هدف این پژوهش یافتن تنظیمات سخت افزاری مناسب برای قالب‌های ذخیره سازی تنک در الگوریتم ضرب برداری ماتریس تنک، با انرژی مصرفی بهینه در واحد کارآیی است. معیار بهینگی در این سیستم برای یک عملیات ضرب برداری معیار کارآیی در واحد توان یا کارآیی در وات است.



شکل ۱. زمان اجرای الگوریتم ضرب برداری ماتریس تنک CurlCurl_1 برحسب میکروثانیه در قالب‌های DIA, CDS, ELL, SELL, BELL, CSR, VCSR, BCSR, HYB

به این معنی که تنظیماتی مطلوب‌تر است که کمترین میزان مصرف انرژی را به ازای کارآیی بیشتر نتیجه دهد. برای دستیابی به بالاترین میزان این معیار برای هر ماتریس تنک ورودی، می‌توان متناسب با الگوی ساختاری ماتریس و قالب منطبق بر این ساختار برای فشرده سازی تنظیمات مناسب سخت افزاری مناسب را برای پردازنده‌ی گرافیکی انتخاب کرد. این ایده بر اساس آزمایشاتی شکل گرفته است که نشان داده‌اند برای هر ماتریس تنک ورودی با هر الگوی ساختاری تنظیمات سخت افزاری و قالب خاصی برای ذخیره‌سازی وجود دارد که بهترین میزان کارآیی در وات را نتیجه می‌دهد.

پس در ابتدا نیاز به انجام آزمایشاتی است که نحوه‌ی تاثیر تنظیم این پارامترها پس از انتخاب قالب مناسب را بر کارآیی و توان مصرفی نشان دهد. برای انجام این آزمایشات تعدادی ماتریس تنک از مجموعه‌ی ۲۰۰ ماتریس تنک ورودی که از کتابخانه‌ی تنک دانشگاه فلوریدا جمع آوری شده و به صورت تصادفی انتخاب شده به طوری‌که اندازه‌ی ماتریس متراکم شده‌ی آن‌ها در قالب‌های این پژوهش، در حافظه‌ی RAM سیستم قابل تعریف باشد. محور افقی نمودارها نشان‌دهنده‌ی مقدار کارآیی در وات با تنظیمات سخت افزاری است و t/p و perf/pow مخفف Performance(time)/power می‌باشد و اعداد بعد به ترتیب اندازه‌ی بلوک، تعداد ثبات در نخ و اولویت سلسله مراتب حافظه را نشان می‌دهد.



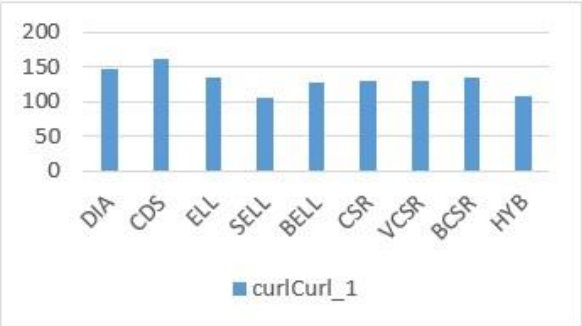
شکل ۲. توان مصرفی الگوریتم ضرب برداری ماتریس تنک CurlCurl_1 برحسب وات در قالب‌های DIA, CDS, ELL, SELL, BELL, CSR, VCSR, BCSR, HYB

۱-۴-معیار کارآیی در واحد توان مصرفی(وات)

در این آزمایشات ۹ قالب ماتریس تنک مورد بررسی قرار گرفته‌اند. نتیجه‌ی آزمایشات در نمودارهای زیر نشان می‌دهند که متناسب با ساختار ماتریس ورودی، انتخاب یک

قالب صحیح برای فشرده سازی و ذخیره‌ی ماتریس، تاثیر بسزایی بر روی کارآیی و توان مصرفی و متناظر با آن معیار کارآیی در واحد توان یا کارآیی در وات خواهد داشت. با انتخاب قالب مناسب می‌توان با کاهش صفرهای افزوده متناظر با الگوی ساختاری ماتریس و کاهش فضای مورد نیاز در حافظه برای ذخیره‌ی ماتریس از پهنای باند حافظه بهره‌وری بهتری داشت. با انتخاب صحیح روش فشرده سازی همچنین می‌توان از محلیت حافظه‌ی نهان بهتر استفاده کرد و با ایجاد تعادل بار در نخ‌ها، از قدرت محاسباتی پردازنده‌ی گرافیکی استفاده‌ی بهینه‌تری داشت.

شکل ۱ نشان دهنده‌ی زمان اجرای هسته‌ی این الگوریتم با ماتریس تنک CurlCurl_1 بر حسب میکرو ثانیه در این ۹ قالب است. شکل ۲ میزان توان مصرفی را برای این ماتریس در قالب‌های مختلف نشان می‌دهد. بهترین میزان کارآیی برای این ماتریس در قالب‌های BELL، SELL و HYB به دست می‌آید ولی بهترین میزان توان مصرفی برای این ماتریس در قالب SELL است. اما شکل ۳ نشان می‌دهد که بهترین میزان کارآیی در واحد توان مصرفی برای این ماتریس در قالب HYB است. لذا معیار کارآیی یا انرژی مصرفی به تنهایی نمی‌تواند مناسب بودن یک قالب را نشان دهد و معیار مناسب میزان انرژی مصرف شده برای رسیدن به کارآیی به دست آمده است.



شکل ۳. مقدار کارآیی در واحد توان مصرفی الگوریتم ضرب برداری برای ماتریس تنک CurlCurl_1 در قالب‌های DIA, CDS, ELL, SELL, BELL, CSR, VCSR, BCSR

برای اندازه‌گیری کارآیی از تعریف معکوس زمان اجرای الگوریتم بر روی پردازنده‌ی گرافیکی استفاده شده است و برای اندازه گیری زمان اجرای الگوریتم از روش چند نخی استفاده شده است. در نخ والد و اصلی، یک حلقه از شروع ارسال هسته برروی پردازنده‌ی گرافیکی تا اتمام انجام عملیات هسته مشغول نمونه گیری از توان مصرفی لحظه‌ای پردازنده‌ی گرافیکی و محاسبه‌ی مدت زمان اجرا است. برای نمونه گیری از حسگرهای پردازنده‌ی گرافیکی برای اندازه گیری توان مصرفی لحظه‌ای، از رابط کتابخانه‌ی NVML[۴۲] استفاده شده است. به دلیل محدودیت حسگرهای اندازه گیری توان مصرفی که روی پردازنده‌ی گرافیکی قرار دارند هر الگوریتم ۵۰۰ بار اجرا شده است و زمان اجرا و توان مصرفی در این پنجره اندازه گیری شده است. در این پنجره هرگاه توان مصرفی پایدار و به صورت دوره‌ای تکرار شد، متوسط توان لحظه‌ای هایی که در این دوره وجود دارند به عنوان متوسط توان مصرفی لحظه‌ای الگوریتم در نظر گرفته می‌شود.

۲-۴-تنظیمات سیستم انجام آزمایشات

این پژوهش بر روی سیستمی انجام شده است که دو پردازنده‌ی K40m داشته است. تعداد نخ‌ها در هر SMX ۲۰۴۸ عدد بوده و دارای ۶۴ کیلوبایت حافظه‌ی روی تراشه است که به به ۳ صورت ۱۶ و۴۸ ، ۳۲ و ۳۲ یا ۴۸ و ۱۶ کیلوبایت برای حافظه‌ی نهان و اشتراکی قابل تعریف است. پهنای باند این پردازنده‌ی گرافیکی به حافظه‌ی سراسری ۲۸۸ گیگابایت در ثانیه می‌باشد.

۵-آزمایشات و نتایج

سه پارامتر سخت افزاری قابل تنظیم منتخب در این پژوهش، اندازه‌ی بلوک نخی، تعداد ثبات‌های اختصاص داده شده به هر نخ و سلسله مراتب حافظه است. تنظیمات قابل اعمال برای اندازه‌ی بلوک نخی از ۱۲۸ تا ۱۰۲۴ است و برای تعداد ثبات‌های اختصاص داده شده برای هر نخ مقادیر ۱۶، ۳۲، ۶۴ و ۲۵۶ در نظر گرفته شده و سلسله مراتب حافظه مطابق قابلیت تنظیم پیکربندی حافظه‌ی روی تراشه اعمال شده است. اندازه‌های ۱۶-۴۸ یا ۳۲-۳۲ کیلوبایت

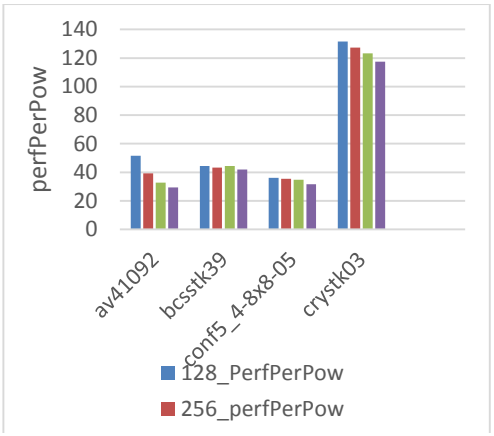
[4] Nvidia management library [2]

برای حافظه‌ی اشتراکی و حافظه‌ی نهان قابل تخصیص است.

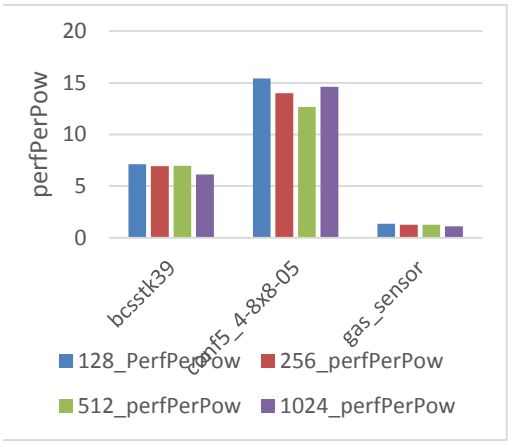
شکل‌های ۴ تا ۶ نشان‌دهنده‌ی تاثیر تنظیمات مختلف اندازه‌ی بلوک نخی برروی کارآیی در واحد توان برای چند ماتریس نمونه‌ی تنک در قالب‌های VCSR، DIA و BCSR است.

شکل‌های ۷ تا ۹ نشان‌دهنده‌ی تاثیر تنظیمات مختلف تعداد ثبات در نخ بر روی کارآیی در واحد توان برای قالب‌های BELL، CDS و ELL می‌باشد. در محور افقی عـدد اول، انـدازه‌ی بلوک نخـی و عدد دوم، تعداد ثبات‌های اختصاص داده شده به هر نخ را نشان می‌دهد.

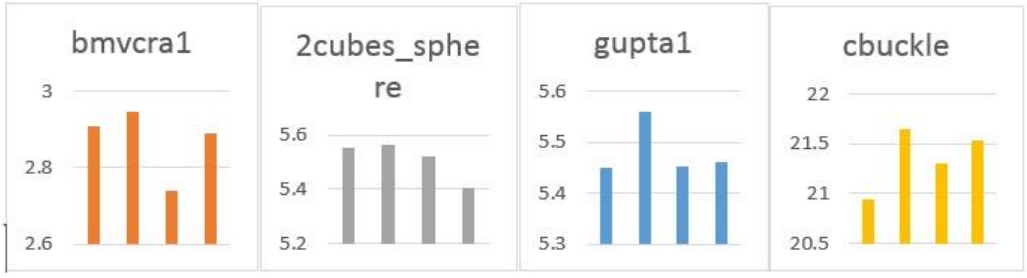
شکل ۱۰ نیز نشان‌دهنده‌ی تاثیر تنظیمات مختلف سلسله مراتب حافظه بر روی کارآیی در واحد توان برای قالب VCSR است. نموداری که با حروف l1 پایان میابد مقدار کارآیی در وات را برای تنظیماتی با اندازه‌ی حافظه نهان ۴۸ کیلوبایت نشان می‌دهد و نمودارهای دارای حروف eq و smem نیز نشان‌دهنده‌ی کارآیی در وات برای تنظیمات اندازه‌ی حافظه‌ی نهان ۳۲ و ۱۶ است.



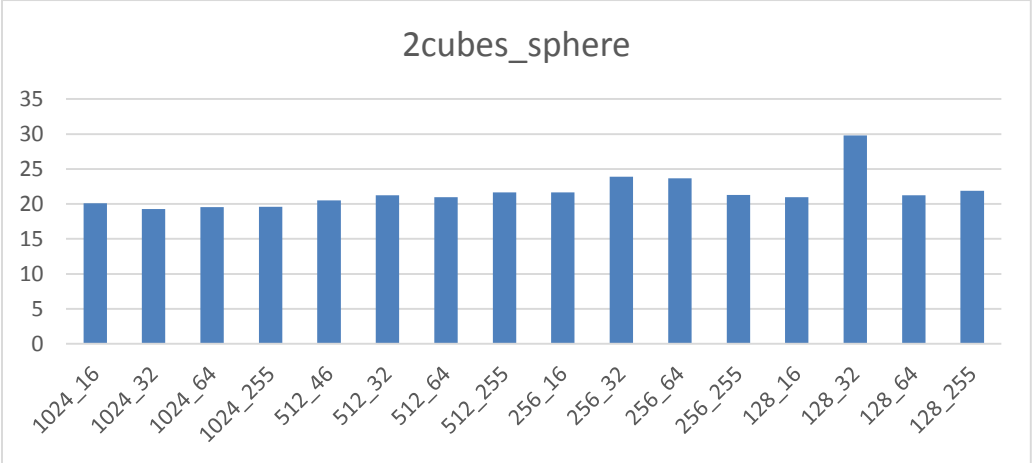
شکل ۴. مقدار کارآیی در واحد توان مصرفی الگوریتم ضرب برداری برای ماتریس تنک نمونه در قالب VCSR در اندازه‌های بلوک نخی ۱۲۸، ۲۵۶، ۵۱۲ و ۱۰۲۴



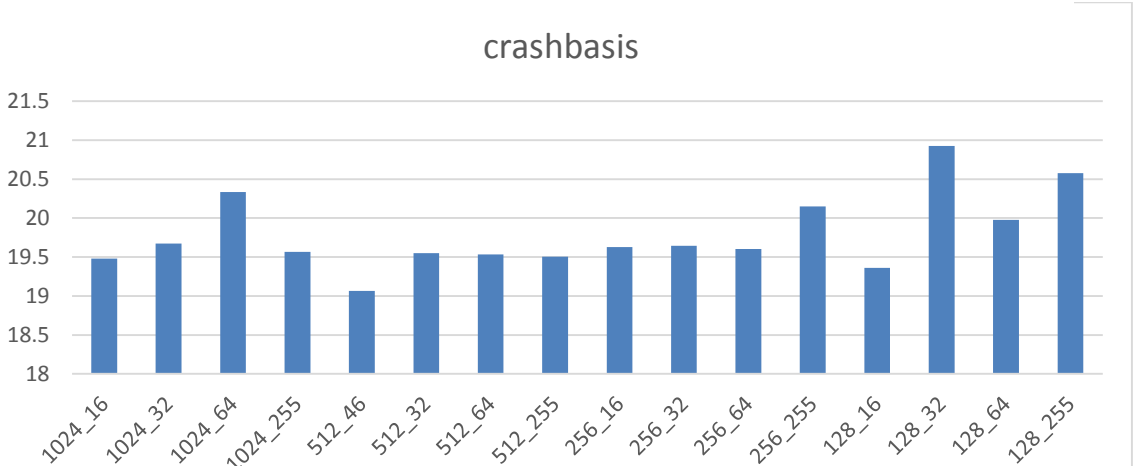
شکل ۵. مقدار کارآیی در واحد توان مصرفی الگوریتم ضرب برداری برای ماتریس تنک نمونه در قالب DIA در اندازه‌های بلوک نخی ۱۲۸، ۲۵۶، ۵۱۲ و ۱۰۲۴



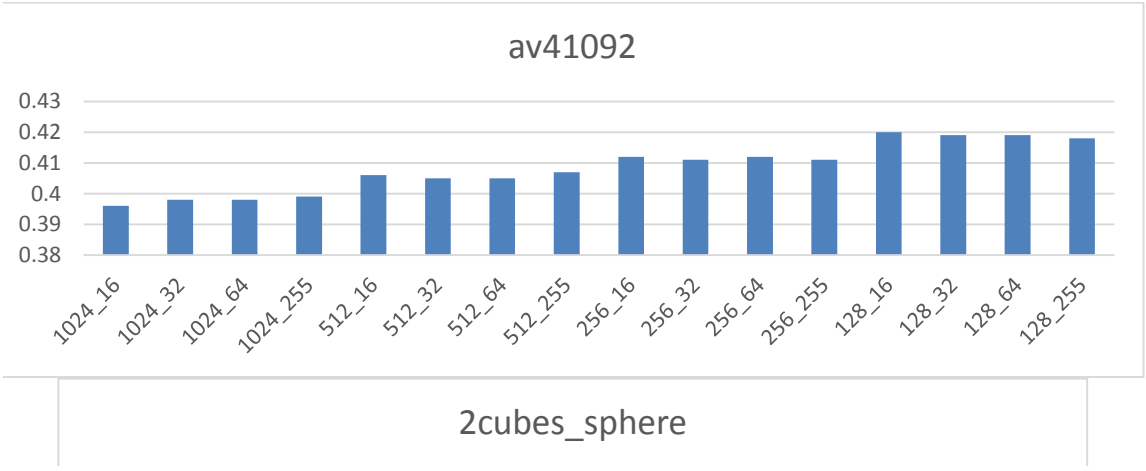
شکل ۶. مقدار کارآیی در واحد توان مصرفی الگوریتم ضرب برداری برای ماتریس تنک نمونه در قالب BCSR در اندازه‌های بلوک نخی ۱۲۸، ۲۵۶، ۵۱۲ و ۱۰۲۴



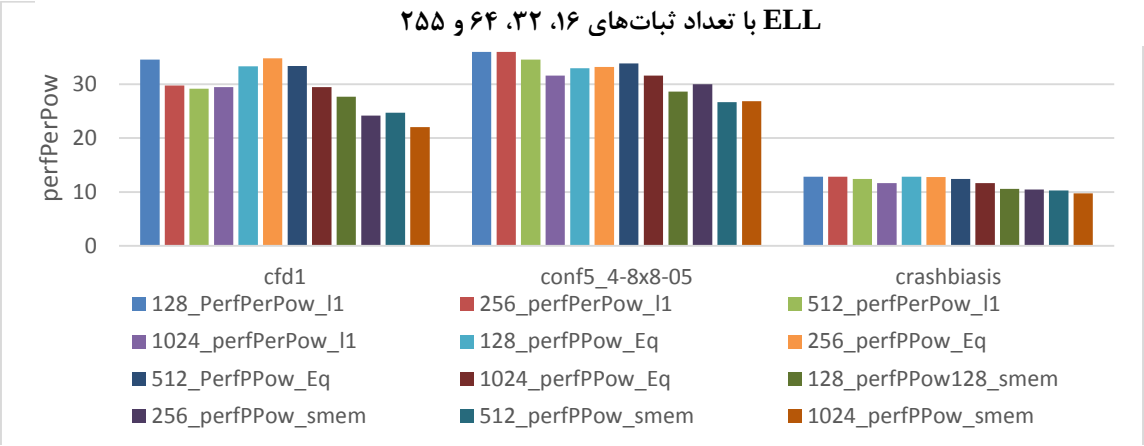
شکل ۷. مقدار کارآیی در واحد توان مصرفی الگوریتم ضرب برداری برای ماتریس تنک نمونه 2cubes_sphere در قالب BELL با تعداد ثبات‌های ۱۶، ۳۲، ۶۴ و ۲۵۵



شکل ۸. مقدار کارآیی در واحد توان مصرفی الگوریتم ضرب برداری برای ماتریس تنک نمونه crashbasis در قالب CDS با تعداد ثبات‌های ۱۶، ۳۲، ۶۴ و ۲۵۵



شکل ۹. مقدار کارآیی در واحد توان مصرفی الگوریتم ضرب برداری برای ماتریس تنک نمونه av41092 در قالب ELL با تعداد ثبات‌های ۱۶، ۳۲، ۶۴ و ۲۵۵



شکل ۱۰. مقدار کارآیی در واحد توان مصرفی الگوریتم ضرب برداری برای ماتریس‌های تنک نمونه در قالب VCSR با تنظیمات مختلف سلسله مراتب حافظه

نتایج آزمایشات با تنظیمات مختلف ۳ پارامتر ذکر شده برای ۲۰۰ ماتریس نمونه نشان می‌دهند که رفتار الگوریتم با استفاده از هر قالب در تمام آزمایشات یکسان است چرا که در همه‌ی موارد همیشه یکی از ترکیبات این تنظیمات بهترین کارآیی در وات را نتیجه می‌دهد. جدول ۱ تنظیمات مناسب برای هر قالب را که از آزمایشات برروی ۲۰۰ ماتریس نمونه‌ی تنک به دست آمده، نشان می‌دهد.

جدول ۱. تنظیمات به دست آمده‌ی متناسب با رفتار هر قالب

نام قالب	اندازه‌ی بلوک نخی	تعداد ثبات در نخ	اندازه‌ی حافظه‌ی نهان(KB)
BCSR	256	16	48
CSR	256	16	48
SELL	256	32	48
ELL	128	16	48
BELL	128	32	48
VCSR	128	32	48
HYB	128	16	48
CDS	128	16	48
DIA	128	32	48

با این‌که قالب VCSR از حافظه‌ی اشتراکی برای ذخیره‌ی داده‌های متناظر با هر بلوک نخی استفاده می‌کند اما در شکل ۳-۲۸ مشاهده می‌شود که برای ماتریس‌های مختلف در این قالب کارآیی در وات با پیکربندی حافظه با بیشترین اندازه‌ی حافظه‌ی نهان به حداکثر خود رسیده است. در این نمودارها مشخص است که با کاهش اندازه‌ی حافظه‌ی نهان، نمودار کارآیی در وات افت پیدا می‌کند.

۶-نتیجه گیری

همانگونه که پیشتر ذکر شد، مطالعات برروی محاسبات تنک در غالب موارد بر روی کارآیی هسته‌ی ضرب برداری ماتریس‌هاس تنک بر روی پردازنده‌ی گرافیکی تمرکز کرده‌اند و مطالعات انگشت شماری بر روی انرژی مصرفی این هسته صورت گرفته است. همچنین هیچ پژوهشی جهت طراحی سیستمی بهینه در مصرف انرژی با در نظر گرفتن کارآیی این هسته بر روی پردازنده‌ی گرافیکی انجام نشده است و امروزه که محققان بر روی لزوم در نظر گرفتن انرژی مصرفی در کنار کارآیی تاکید زیادی می‌کنند، برای عملیات ضرب برداری ماتریس‌های تنک که در اکثر برنامه‌های علمی ریاضی و فیزیک در مقیاس گسترده کاربرد دارند، نیاز به چنین مطالعه‌ای حس می‌شود. در این پژوهش مشاهده شد که رفتار الگوریتم ضرب برداری با توجه به قالبی که برای ذخیره سازی ماتریس تنک استفاده می‌کند متفاوت است و نمی‌توان یک پیکربندی را برای الگوریتم‌هایی با رفتارهای متفاوت در نظر گرفت. در این پژوهش تنظیمات مناسب برای سه پارامتر اندازه‌ی بلوک نخی، تعداد ثبات در نخ و سلسله مراتب حافظه برای مشهورترین و بهینه ترین قالب‌های ذخیره سازی تنک به دست آمده است.

منابع

1. J. Im and K. Yelick, "Optimization of Sparse Matrix Kernels for Data Mining," in *Proc. of the Workshop on Text Mining*, 2001.

2. L. N. Trefethen and D. Bau, III, Numerical Linear Algebra. Society for Industrial and Applied Mathematics,1997.

3. R. Gilbert, S. Reinhardt, and V. B. Shah, "Highperformance Graph Algorithms from Parallel Sparse Matrices," in *Proc. of the Int'l Workshop on Applied Parallel Computing*, 2006.

4. Owens JD, Luebke D, Govindaraju N, Harris M, Krüger J, Lefohn AE, Purcell TJ (2007) "A survey of general-purpose computation on graphics hardware," In: *Computer graphics forum*, vol 26. Wiley Online Library, pp 80–113.

5.Bell and M. Garland, "Efficient sparse matrix-vector multiplication on CUDA," *Nvidia Technical Report* NVR-2008-004, Nvidia Corporation2008.

6. S. Yan, C. Li, Y. Zhang, and H. Zhou, "yaspmv: Yet another spmv framework on gpus," in *ACM SIGPLAN Notices*, 2014, pp. 107-118.

7. J. W. Choi, A. Singh, and R. W. Vuduc, "Model-driven autotuning of sparse matrix-vector multiply on GPUs," in *ACM Sigplan Notices*, 2010, pp. 115-126.

8. Bolz, Ian Farmer, Eitan Grinspun, and Peter Schrooder, "Sparse matrix solvers on the GPU: Conjugate gradients and multigrid," *ACM Trans. Graph.*, 22(3):917-924, 2003.

9. R. Gilbert, S. Reinhardt, and V. B. Shah, "Highperformance Graph Algorithms from Parallel Sparse Matrices," in *Proc. of the Int'l Workshop on Applied Parallel Computing*, 2006.

10. NVIDIA, "N**VI**DIA Corporation. *CUDA Toolkit Reference Manual*, 8.0 edition. Available on line at: http://developer.nvidia.com/cuda-toolkit-80.

11. Benatia, W Ji, Y Wang, F Shi, "Energy evaluation of Sparse Matrix-Vector Multiplication on GPU " in Green and Sustainable Computing Conference ,2016, p. 1-6. N.

12. Bell and M. Garland, "Implementing sparse matrix-vector multiplication throughput-oriented processors," In *Proc. of Int'l Conf. on High Performance Computing Networking, Storage and Analysis, SC '09*, pages 18:1-18:11. ACM, 2009.

13. S. Mullen, M. M. Wolf, and A. Klein, "Pakck: Performance and power analysis of key computational kernels on cpus and gpus," in *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*, 2013, pp. 1-6.

14. H. Anzt, S. Tomov, and J. Dongarra, "Energy efficiency and performance frontiers for sparse computations on GPU supercomputers," in *Proceedings of the sixth international workshop on programming models and applications for multicores and manycores*, 2015, pp. 1-10.

15. Burtscher, I. Zecena, and Z. Zong, "Measuring GPU power with the K20 built-in sensor," in *Proceedings of Workshop on General Purpose Processing Using GPUs*, 2014, p. 28.

16. S. Song, C. Su, B. Rountree, and K. W. Cameron, "A simplified and accurate model of power-performance efficiency on emergent gpu architectures," in *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, 2013, pp. 673-686.

17. Zardoshti, P., Khunjush, F. & Sarbazi-Azad, H, "Adaptive sparse matrix representation for efficient matrix–vector multiplication," *Journal of Supercomputing (2016)* 72: 3366. https://doi.org/10.1007/s11227-015-1571-0.

18. NVIDIA, "NVIDIA Corporation (2014) Tuning CUDA applications for Kepler," *Technical report*, August 2014. http://docs.nvidia.com/cuda/pdf/Kepler_Tuning_Guide.pdf