تاثیر دانه‌بندی در بهبود عملکرد کرنل OpenCL در CPU چندهسته‌ای

عطیه جواهری[1]، مجتبی منصوری[2]، علی غلامی رودی [3]

[1]کارشناسی ارشد کامپیوتر، دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی نوشیروانی بابل، a.javaheri@stu.nit.ac.ir

[2] استادیار دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی نوشیروانی بابل، mansoori@nit.ac.ir

[3] استادیار دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی نوشیروانی بابل، gholamirudi@nit.ac.ir

چکیده

با توجه به هدف طراحی قابلیت حمل‌پذیری OpenCL، علاوه بر استفاده این پلت‌فرم در محیط GPU، در پردازنده چندهسته‌ای نیز قابل بهره‌برداری می‌باشد. یکی از مسائل مربوطه این پلت‌فرم در محیط‌های CPU قابلیت حمل‌پذیری عملکرد OpenCL است. در این مقاله تاثیر دانه‌بندی در بهبود عملکرد کرنل OpenCL مورد بررسی قرار می‌گیرد. چندین برنامه از بنچمارک Rodinia در پردازنده چندهسته‌ای با مجموعه داده‌های مختلف آزمایش می‌شود و اندازه بهینه دانه‌بندی برای هر یک تعیین می‌گردد. با هدف شناسایی دانه‌بندی مناسب، برای دسته‌های مختلف برنامه است که در چه اندازه work-group، عملکرد بهتری دارند. نتایج نشان می‌دهد این متغیر در عملکرد برنامه‌های مختلف به شکل متفاوتی اثر می‌گذارد. در این مقاله، اثر ویژگی‌های برنامه‌های بررسی شده در تاثیر دانه‌بندی بر عملکرد بررسی می‌شود. نتایج نشان می‌دهد که به طور کلی با کاهش اندازه work-group و یا به عبارت دیگر با افزایش تعداد work-group در فضای index، عملکرد برنامه‌ها بهبود می‌یابد.

واژه‌های کلیدی

دانه‌بندی، OpenCL، چندهسته‌ای، حمل‌پذیری، work-group.

۱ - مقدمه

سیستم‌های چندهسته‌ای، با قراردادن چندین هسته روی یک تراشه واحد طراحی می‌شوند. با ایجاد این سیستم‌ها، ابزارهای موازی‌سازی برای بهره‌مندی بهتر از قابلیت هسته‌های تعبیه‌شده در این تراشه‌ها مورد نیاز می‌باشد. از این رو زبان‌های موازی‌سازی مقیاس‌پذیری برای بهره‌مندی بهتر از این سیستم‌ها بوجود آمدند تا همگام با پیشرفت سخت‌افزار استفاده آن را نیز تسهیل نمایند. امروزه با تمرکز در محاسبات موازی، سیستم‌های جدیدی وارد بازار شدند که از ترکیب معماری‌هایی مانند پردازنده‌های چندهسته‌ای، پردازنده‌های سیگنال‌دیجیتالی، FPGAها و GPUها تشکیل شده‌اند. این سیستم‌های ناهمگن نیازمند محیط برنامه‌نویسی استاندارد و یکنواختی هستند تا بتوانند منابع مختلف را در یک چارچوب مشترک مدیریت کنند.

امروزه به دلیل چند سطحی شدن محاسبات و بهره‌برداری بیشتر از پردازنده‌های ناهمگن مسئله مهم در زبان‌های موازی‌سازی برخوردار بودن قابلیت حمل‌پذیری است تا قابلیت پورت شدن برنامه‌ها از یک معماری به معماری دیگر وجود داشته باشند. در بین زبان‌های موازی توسعه‌یافته برای این پلت‌فرم‌ها، تنها زبان OpenCL دارای این ویژگی است و زبان‌های دیگر از قبیل CUDA مختص پلت‌فرم سخت‌افزاری واحدی هستند. OpenCL یک مدل برنامه‌سازی حمل‌پذیری است که با به‌روزرسانی بسیار اندک بخش‌های پیکربندی در کد برنامه در معماری‌های مختلف نظیر GPU، DSP، پردازنده‌های چندهسته‌ای و تسریع‌کننده‌ها با حافظه اشتراکی قابل اجرا است[۱،۲].

اصولاً محاسبات با OpenCL در محیط GPU عملکرد بهتری دارند زیرا پلت‌فرم OpenCL با پلت‌فرم GPU مطابقت بیشتری دارد ولی نظر به اهمیت سیستم‌های پردازنده چندهسته‌ای و رایج بودن آنها در لپ‌تاپ‌ها و دسک‌تاپ‌ها قابلیت استفاده از این پلت‌فرم و پورت برنامه‌ها به این سیستم‌ها با حفظ عملکرد تسریع، امکانات محاسبات فوق سریع با هزینه اندکی در اختیار کاربر قرار می‌دهد. پورت برنامه‌های تولید شده برای GPU، صرفاً از طریق کامپایل مجدد آنها برای محیط پردازنده چندهسته‌ای، عملکرد محیط GPU را به همراه نخواهد داشت[۳]. پارامترهای متفاوتی در حفظ عملکرد کارآمد هستند این تحقیق به بررسی یکی از پارامترها که دانه‌بندی در work-group است، می‌پردازد.

در این گزارش، ابتدا در بخش ۲ به کارهای پیشین مرتبط اشاره می‌شود. دربخش ۳ مدل OpenCL بررسی می‌شود. بخش ۴ به کارهای انجام شده این تحقیق و نمودارها و نتایج تجربی و بخش ۵ به تحلیل آنها پرداخته می‌شود. درنهایت در بخش ۶ نتیجه‌گیری و پیشنهادات ارائه می‌گردد.

۲ - کارهای مرتبط

Membarth و همکاران [۴] پنج مدل برنامه نویسی موازی (OpenMP، Cilk++، TBB، RapidMind و OpenCL) را تنها توسط یک برنامه بر روی پردازنده‌های چندهسته‌ای آزمایش نمودند و از نظر قابلیت استفاده و عمکرد ارزیابی کردند. بر اساس نتایج آنها، OpenMP و TBB بهترین عملکرد را داشتند و OpenCL تنها برای مجموعه داده‌هایی با حجم بالا عملکرد بهتری داشت.

Ali و همکاران [۵] سه مدل برنامه نویسی موازی (OpenCL، TBB و openMP) را در زمینه بهینه‌سازی کامپایلر و مقیاس‌پذیری مقایسه کردند که نتایج آنها نشان داد openMP بهتر از مدل های دیگر است.

Shen و همکاران [۳] دو مدل OpenCL، OpenMP را با هم در پردازنده چندهسته‌ای مقایسه کردند تا به فاکتورهایی برای بهبود عملکرد OpenCL در CPU دست یابند. در بیشتر موارد OpenMP عمکرد بهتری از OpenCL داشت و دلایل این اختلاف عملکرد را پلت‌فرم و دانه‌بندی و کامپایلر OpenCL دانستند. آنها آزمایش‌هایشان را با پنج برنامه از بنچمارک Rodinia [۶] با مجموعه‌داده‌های مختلف بر روی سه پلت‌فرم چندهسته‌ای انجام دادند و با دو کامپایلر (Intel و AMD) اندازه work-group برنامه‌های PathFinder، HotSpot، CFD و K-means را تغییر دادند و این دو کامپایلر را با هم مقایسه کردند.

Hwan Lee و همکاران [۲] عملکرد برنامه‌های OpenCL را از جنبه معماری که شامل سربار زمان‌بندی، موازی‌سازی در سطح دستورالعمل، سربار API و فضای آدرس و ... است در پردازنده‌های چندهسته‌ای ارزیابی کردند. پلت‌فرم محاسباتی آنها از پردازنده چهارهسته‌ای و GPU تشکیل شده است. با تغییر اندازه work-group در سه برنامه از بنچمارک parboil [۷] مشاهده کردند که با افزایش اندازه work-group عملکرد در پردازنده افزایش می‌یابد. آنها همچنین دریافتند که تخصیص کار بیشتر به work-itemها و یا به عبارتی دیگر کاهش تعداد آن‌ها، عملکرد پردازنده را ارتقاء می‌بخشد.

Shen و همکاران [۸] آزمایش‌های خود را با سه برنامه از بنچمارک Rodinia با پیاده‌سازی OpenCL و OpenMP با مجموعه‌داده‌های مختلف بر روی سه پلت‌فرم چندهسته‌ای انجام دادند. عامل اصلی در عملکرد نامطلوب OpenCL نسبت به OpenMP، استفاده نادرست از cache در بین work-itemها دانستند. تمرکز این مقاله پیدا کردن علت های تفاوت عملکرد در دو مدل موازی‌سازی OpenMP و OpenCL در پردازنده‌های چندهسته‌ای است. یکی از راه‌حل‌های پیشنهادی آنها جهت بهبود عملکرد OpenCL، افزایش اندازه دانه‌های work-item در OpenCL است.

Shen و همکاران [۹] تله[1]هایی را شناسایی کردند که منجر به کاهش عملکرد OpenCL در پردازنده می‌شود. راه‌حل‌های پیشنهاد شده آنها از بین بردن کپی‌های غیرضروری و بهبود data-locality با استفاده از cache و استفاده یا بهبود از SIMD در پردازنده‌های چندهسته‌ای است که برای استفاده بهتر از cache افزایش دانه‌بندی را پیشنهاد دادند.

۳ - مدل OpenCL

محیط برنامه‌نویسی OpenCL دارای یک host و deviceهای محاسباتی است. سیستم host در استاندارد OpenCL، پردازنده است که با deviceهای OpenCL ارتباط دارد. deviceهای OpenCL می‌تواند GPU، DSP، پردازنده و شتاب‌دهنده‌ها باشد[۵].

شکل ۱ مدل پلت‌فرم OpenCL را نشان می‌دهد که در آن device محاسباتی شامل چندین واحد محاسباتی[2] است. هر یک از واحدهای محاسباتی نیز شامل چندین عناصر پردازشی است. عناصر پردازشی یک دستور العمل واحد با چندین داده[3] یا یک برنامه واحد با چندین داده[4] را اجرا می‌کنند. این دستورالعمل‌ها بخش کرنل OpenCL را تشکیل می‌دهند[۳].

محاسبات موازی در deviceهای محاسباتی در یک فضای مسئله n-بعدی تعریف می‌شود. زمانی که کرنل برای اجرا، در صف قرار می‌گیرد، یک فضای index تعریف می‌شود. در این فضای index هر عنصر مستقل که یک نمونه از کرنل مسئله است، work-item نامیده می‌شود. همه work-itemها در توابع کرنل کار یکسان اما با داده‌های مختلف انجام می‌دهند. زمانی که کرنل در صف‌اجرا قرارمی‌گیرد، فضای index باید برای دربرگرفتن تعداد کامل work-itemها برای اجرا تعریف شود. این فضای index، NDRANGE یا تعداد بعد مسئله نام دارد و n-بعدی است که n می‌تواند با توجه به ابعاد مسئله۱، ۲ و ۳ باشد.

work-itemها می‌توانند با هم ترکیب شوند و work-group را ایجاد نمایند. اندازه هر work-group توسط فضای index محلی تعریف می‌شود. همه‌ی work-itemها در یک work-group با همدیگر در یک واحد محاسباتی اجرا می‌شوند[۱،۱۰].

مدل حافظه دارای چهار بخش شامل حافظه سراسری[5]، حافظه ثابت، حافظه محلی و حافظه خصوصی است که هر چهار بخش از حافظه در host و deviceهای محاسباتی قابل دسترسی هستند. همه‌ی work-

---

[1] Trap

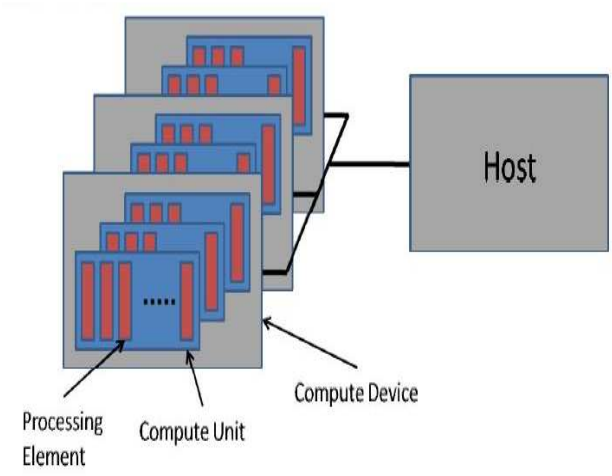
[2] Comput unit

[3] SIMD

[4] SPMD

[5] Global memory

itemها در هر work-group قابلیت خواندن و نوشتن در host و deviceهای محاسباتی را در حافظه سراسری دارند. این بخش از حافظه، تنها توسط host در زمان اجرا تخصیص می‌یابد.



شکل ۱: مدل پلت‌فرم OpenCL [۱۰].

حافظه ثابت، یک منطقه از حافظه سراسری است که در زمان اجرای کرنل ثابت می‌ماند. work-itemها تنها در آن بخش، قابلیت خواندن دارند. host می تواند هم دسترسی خواندن و نوشتن داشته باشد.

حافظه محلی، یک بخشی از حافظه استفاده شده برای داده‌های اشتراکی توسط work-item در work-group است. همه‌ی work-itemها در یک work-group حقوق دسترسی برای خواندن و نوشتن در این فضای حافظه را دارند.

حافظه خصوصی بخشی از حافظه است که تنها برای یک work-item قابل دسترس است[۵،۱۰].

٤ - نتایج تجربی

آزمایش‌ها در محیط OpenCL تحت سیستم عامل لینوکس، توزیع Ubuntu، نسخه‌ی 14.04 با کامپایلر AMD-APP-SDK نسخه‌ی v3.0.130.135، بر روی پلت‌فرم چهارهسته‌ای (Intel core i5-4460 3.20GHZ) انجام شده است. در این آزمایش‌ها هشت برنامه از بنچمارک Rodinia، شامل NW، PathFinder، CFD، Particle Filter، HotSpot، K-means، LUD و BFS است با مجموعه‌داده‌های مختلف بر روی این پلت‌فرم، به اجرا گذاشته شد.

اجرای یک برنامه OpenCL دارای چهار مرحله است. این مراحل عبارتند از:۱) پیکربندی و مقداردهی اولیه. ۲) انتقال داده‌های مسئله از host به device. ۳) اجرای کرنل. ۴) انتقال نتایج حل مسئله از device به host. هر یک از این مراحل در زمان اجرا سهمی در هزینه زمانی و یا عملکرد دارند. از زمان های انتقال host به device و device به host می‌توان صرف نظر کرد زیرا device و host در CPU چندهسته‌ای در یک تراشه واحد هستند. همچنین می‌توان زمان پیکربندی را نادیده گرفت زیرا مقداردهی اولیه و پیکربندی برای هر برنامه OpenCL تنها یکبار اتفاق می‌افتد. در نتیجه زمان اجرا برنامه برابر با زمان اجرا کرنل است[۳].

دانه‌بندی به دو شکل تعیین می‌گردد: ۱)تغییر workload ۲) تغییر اندازه work-group که با افزایش هر یک از این کمیت‌ها دانه‌بندی درشت‌تر و با کاهش آن، دانه‌بندی ریزتر می‌شود. در ادامه توضیح دانه‌بندی درشت‌تر را برای هر دو شکل به ترتیب ذکر می‌شود. با افزایش workload، بار محاسباتی در هر work-item افزایش می‌یابد. به این صورت که تعداد work-itemها در یک work-group و همچنین در فضای index کاهش می‌یابد. با افزایش اندازه work-group، تعداد work-itemها در یک work-group افزایش می‌یابد و تعداد work-group در فضای index کاهش می‌یابد.

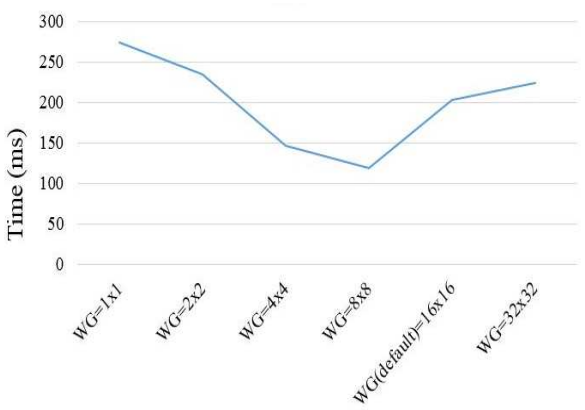
در این تحقیق با تغییر اندازه work-group برنامه‌های بررسی شده در بنچمارک، زمان اجرا آن اندازه‌گیری گردید. تعداد اجرای برنامه برای هریک از اندازه‌ها، ۱۰ در نظر گرفته شده و زمان اجرا، بر حسب میانگین تعداد اجراها محاسبه می‌شود. زمان اجرا هر کدام از برنامه‌ها، با زمان اجرا با اندازه work-group در حالت پیش‌فرض بنچمارک که برای محیط CPU آزمایش شده است مقایسه شده و تسریع آن در جدول ۱ گزارش شده است. مقدار تسریع در این جدول، نسبت زمان اجرا با اندازه work-group در حالت پیش‌فرض بنچمارک، به زمان اجرایی است که حداقل زمان را دارد. برای برنامه HotSpot در مجموعه داده 4K تسریع قابل محاسبه نیست زیرا عملکرد بهینه برای اندازه work-group در حالت پیش‌فرض بنچمارک است.

برای تمام شکل‌های نموداری که در زیر گزارش شدند، محور افقی اندازه work-group را نشان می‌دهد. به طور مثال WG=2x2 به این معنی است که اندازه work-group ۲ در ۲ می‌باشد و تعداد work-itemها برابر با ۴ است و همچنین کل فضای index به work-groupهای ۲ در ۲ تقسیم‌بندی می‌شود. محور عمودی نمودار زمان اجرای برنامه مورد نظر را بر حسب میلی ثانیه نشان می‌دهد.

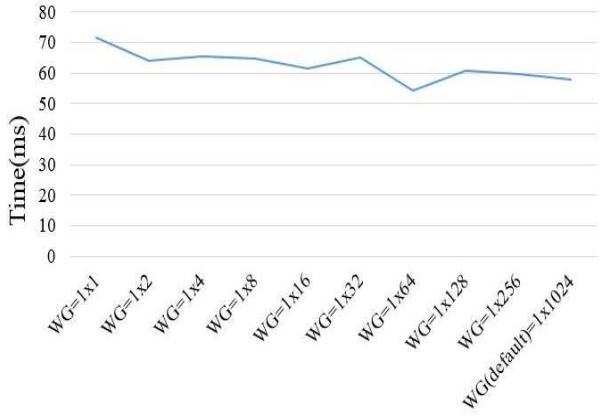
در شکل ۲ بهترین عملکرد برای NW با اندازه ۸ در ۸ است که از زمان کمتری برخوردار است زیرا در این اندازه، تعداد ارجاع به حافظه اصلی کمتر می‌شود. شکل ۳ عملکرد برنامه PathFinder با اندازه ورودی 100x100x20 که به ترتیب معادل ستون و سطر و ارتفاع هرم هستند را نشان می‌دهد. این مسئله با تغییر اندازه work-group، تاثیر چندانی در عملکرد نشان نمی‌دهد. در شکل۴ با افزایش مجموعه داده، تاثیر چندانی در عملکرد برنامه CFD حاصل نمی‌شود. در شکل ۵ برنامه Particle Filter با افزایش work-group به زمان بیشتری برای اجرا نیاز دارد. بنابراین هر چه اندازه work-group کمتر باشد عملکرد بهتری دارد. در شکل ۶ برنامه HotSpot با کاهش اندازه work-group عملکرد بهتری دارد و با افزایش اندازه مجموعه داده تسریع بیشتری از خود نشان می‌دهد. row نشان‌دهنده سطر و ستون گرید است که به عنوان ورودی به برنامه داده می‌شود. در یک مجموعه داده ثابت، هر چه مقدار row بیشتر

شود عملکرد بهتری را از خود نشان می‌دهد.

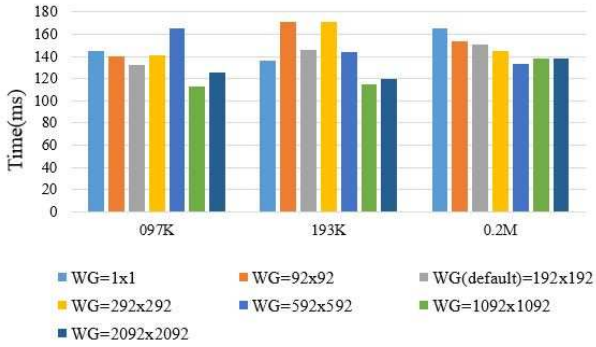
شکل ۷ برنامه K-means را نشان می دهد که تغییر اندازه work-group تاثیر زیادی در عملکرد آن ندارد.



شکل ۲: تاثیراندازه work-group در زمان اجرا برای برنامه NW.

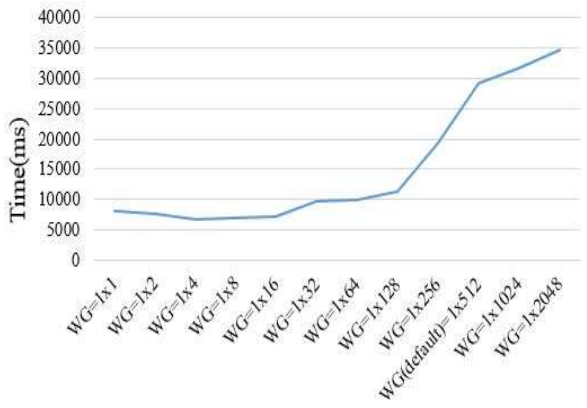


شکل ۳: تاثیراندازه work-group در زمان اجرا برای برنامه PathFinder.

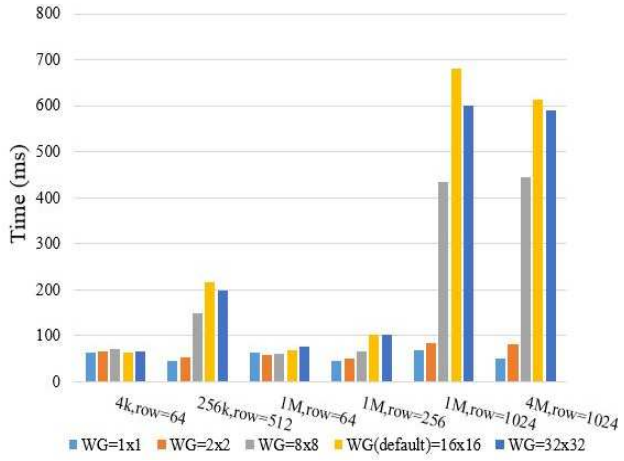


شکل ۴: تاثیراندازه work-group در زمان اجرا برای برنامه CFD.

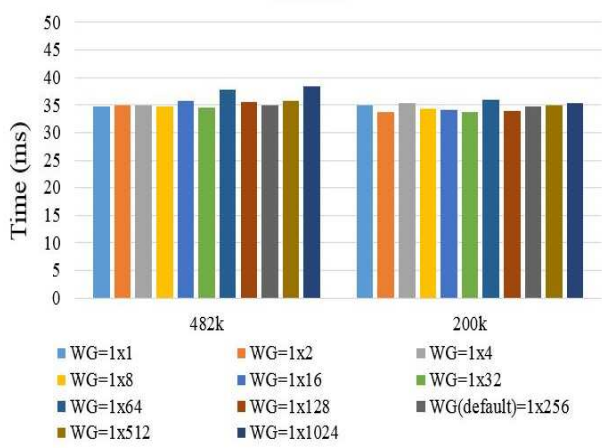
شکل ۸ برنامه LUD را نشان می‌دهد که برای همه مجموعه داده‌ها تاثیر یکسانی دارد. زمانی که اندازه work-group برابر با یک باشد به این معنی است که تعداد work-item در هر work-group برابر با یک است و یا به عبارت دیگر تعداد work-groupها برابر با تعداد work-itemها است، در این حالت عملکرد بهتری دارد. در شکل ۹ اندازه work-group تاثیری در عملکرد ندارد.



شکل ۵: تاثیراندازه work-group در زمان اجرا برای برنامه Particle Filter.

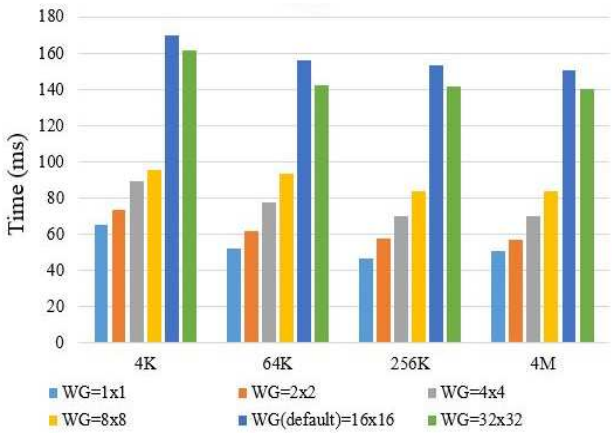


شکل ۶: تاثیراندازه work-group در زمان اجرا برای برنامه HotSpot.



شکل ۷: تاثیراندازه work-group در زمان اجرا برای برنامه K-means.

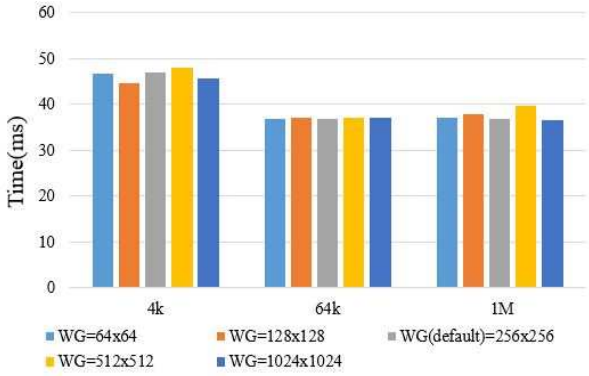
جدول ۱ تاثیر مثبت متغیر work-group در برنامه را نشان می‌دهد. در این جدول عملکرد بهینه تعداد بیشتری از برنامه‌ها با WG=1x1 گزارش شده است. یکی از دلایل بزرگتر بودن تاثیر این متغیر در برنامه، استفاده بهتر از cache است. این نتایج، نتایج گزارش شده توسط Shen و همکاران [۳] را برای برنامه‌های PathFinder، CFD و K-means تایید می‌کند که با تغییر اندازه work-group تاثیر چندانی در عملکرد این برنامه‌ها ندارند و برای برنامه HotSpot نتیجه گرفتند تغییر دانه‌بندی در آن اثر دارد و بهبود عملکرد آن با افزایش اندازه work-group است. درحالی که در این مقاله نتیجه بهبود عملکرد با دانه‌بندی ریزتر است. تفاوت این نتیجه را می‌توان تاثیر شرایط آزمایش مثل تفاوت در تعداد هسته‌ها و فرکانس و اندازه cache پردازنده در این دو مقاله دانست. همچنین تفاوت نتایج در این مقاله با نتایج گزارش شده توسط Hwan Lee و همکاران [۲] را می‌توان تفاوت پلتفرم و پردازنده آن دانست زیرا علاوه بر CPU دارای GPU نیز است و اندازه حافظه نهان در سطح اول آن بزرگتر است و همچنین نوع کامپایلر و برنامه‌های آزمایش شده آن متفاوت است.



شکل ۸: تاثیراندازه work-group در زمان اجرا برای برنامه LUD.

جدول ۱:خلاصه نتایج

برنامه	طبقه‌بندی(ساختار برنامه)	قلمرو	مجموعه داده	بهترین WG	تسریع
NW	Dynamic Programming	Bioinformatics	1M	8	1.70
PathFinder	Dynamic Programming	Grid Traversal	10K	64	1.07
CFD	Unstructured Grid	Fluid Dynamic	097K	1092	1.17
			193K	1092	1.28
			0.2M	592	1.12
Particle Filter	Structured Grid	Medical Imaging	160K	4	4.33
HotSpot	Structured Grid	Physics Simulation	4k,row=64	16	----
			256k,row=512	1	4.77
			1M,row=64	2	1.16
			1M,row=256	1	2.31
			1M,row=1024	1	9.90
			4M,row=1024	1	12.5
K-means	Dense Linear Algebra	Data Mining	482K	32	1.01
			200K	2	1.02
LUD	Dense Linear Algebra	Linear Algebra	4K	1	2.60
			64K	1	3.00
			256K	1	3.29
			4M	1	2.97
BFS	Graph Traversal	Graph Algorithms	4K	128	1.05
			64K	64	1
			1M	1024	1



شکل ۹: تاثیراندازه work-group در زمان اجرا برای برنامه BFS.

۵ - تحلیل و نتایج آزمایش‌ها

perf [۱۱] ابزار عمومی برای سیستم لینوکس و دارای مجموعه‌ای از دستورات است که داده‌ها را ردیابی و عملکرد را تجزیه و تحلیل می‌کند. با نصب این ابزار علت تاثیر تغییر اندازه work-group در سطح پایین‌تر که مربوط به عملکرد CPU است، بررسی شده است و دلایل اصلی و تاثیرگذار بر روی برنامه‌ها یافته شد که با توجه به این مشاهدات، برنامه هایی که مقدار missهای بارگذاری در cache داده سطح l1 [۶] در آنها کمتر است با دانه‌بندی کمتر و برنامه هایی که مقدار cache-miss در آنها کمتر است با دانه‌بندی بیشتر یا به عبارت دیگر اندازه work-group بیشتر، عملکرد بهتری دارند. این ویژگی‌ها به خصوص در مجموعه داده‌های کوچک‌تر دیده می‌شود که در آن تعداد missها کمتر می‌شود. علاوه برآن به نظر می‌رسد به صورت کلی در برنامه‌هایی که ساختار گرید دارند، در اندازه work-group کمتر میزان missهای بارگذاری در cache داده سطح l1 کمتر است. این عامل عملکرد بهتر دانه‌بندی ریزتر را در آزمایش‌های شکل ۵ و ۶ توجیه می‌کند.

۶ - نتیجه گیری و پیشنهادات

در این مقاله چندین برنامه از بنچمارک Rodinia و اثرات برنامه‌ها با تغییر دانه‌بندی و با افزایش مجموعه داده‌ها بررسی گردید. دسته‌های مختلفی از برنامه‌ها شناسایی شدند که تغییر work-group می‌تواند اثر مثبت و یا بدون تاثیر باشد. می‌توان نتیجه گرفت دانه‌بندی می‌تواند عملکرد OpenCL در CPU را بهبود دهد. برای برنامه‌هایی که ساختار گرید دارند در عملکرد تاثیرگذار است و برای برنامه HotSpot با مجموعه داده با حجم بالاتر، تسریع بیشتر می‌شود. برای کارهای آینده علت تاثیرگذاری تغییرات در work-group برای این برنامه‌ها مورد بررسی قرار گیرد و با توجه به ویژگی برنامه‌ها استدلال گردد.

مراجع

[1] A. Ali, "Comparative study of parallel programming models for multicore computing," MasterThesis, Linköping University, Sweden, 2013, pp. 1–87.

[2] J. Hwan Lee, N. Nimit, K. Hyesoon, P. Kaushik, and K. Hyojong, "OpenCL Performance Evaluation on Modern Multicore CPUs," Hindawi Publishing Corporation, Scientific Programming, 2015.

[3] J. Shen, J. Fang, H. Sips, AL. Varbanescu , "An application-centric evaluation of OpenCL on multi-core CPUs," Parallel Computing, Elsevier, 2013, vol. 39, pp. 834–850.

[4] R. Membarth, F. Hannig, J. Teich, M. Körner, W. Eckert, "Frameworks for multi-core architectures: a comprehensive evaluation using 2D/3D image registration," Architecture of Computing Systems – ARCS, 2011, vol. 6566, pp. 62–73.

[5] A. Ali, U. Dastgeer, C. Kessler, "OpenCL for programming shared memory multicore CPUs," in MULTIPROG, in conjunction with HiPEAC 2012, paris, 2012.

[6] Rodinia3.1, http://lava.cs.virginia.edu/Rodinia/download_links.html (accessed january 2, 2016).

[7] The IMPACT Research Group and UIUC, "Parboil benchmark suite, " http://impact.crhc.illinois.edu/Parboil/parboil.aspx.

[8] J. Shen, J. Fang, A.L. Varbanescu, H. Sips, "Performance gaps between OpenMP and OpenCL for multi-core CPUs," in: ICPP, Workshop 2012, 2012, pp. 116–125.

[9] J. Shen, J. Fang, H. Sips, AL Varbanescu , "Performance Traps in OpenCL for CPUs- Parallel," Distributed and Systems Group, IEEE, 2013, pp. 38–45.

[10] D. Black-schaffer, "introduction to opencl programming," 2012, pp. 1–14.

[11] Perf, https://perf.wiki.kernel.org/index.php/Tutorial (accessed may 25, 2016).

[6] L1-dcache-load-miss