موازی سازی الگوریتم غذایابی باکتری

با استفاده از معماری CUDA

نویسنده : علی رفیعی

سرشناسه	:	رفیعی، علی، ۱۳۵۸ -
عنوان و نام پدیدآور	:	موازی‌سازی الگوریتم غذایابی باکتری با استفاده از معماری CUDA/ نویسنده علی رفیعی.
مشخصات نشر	:	تهران: آرنا،۱۳۹۵.
مشخصات ظاهری	:	۹۰ص.
شابک	:	978-600-356-566-1
وضعیت فهرست نویسی	:	فیپا
یادداشت	:	کتابنامه.
موضوع	:	کودا (ساختار کامپیوتر)
موضوع	:	CUDA (Computer architecture)
موضوع	:	الگوریتم‌های ژنتیک
موضوع	:	Genetic algorithms
موضوع	:	الگوریتم‌های فراابتکاری
موضوع	:	Metaheuristic algorithms
موضوع	:	الگوریتم‌های موازی
موضوع	:	Parallel algorithms
موضوع	:	پردازش موازی
موضوع	:	Parallel processing (Electronic computers)
رده بندی کنگره	:	QA۷۶/۶۴۲ ۱۳۹۵م۸ /ر۷
رده بندی دیویی	:	۶/۰۰۶
شماره کتابشناسی ملی	:	۴۴۹۹۷۷۱

عنوان	:	موازی سازی الگوریتم غذایابی باکتری با استفاده از معماری CUDA
نویسنده	:	علی رفیعی
ناشر	:	آرنا
چاپ	:	اول ۱۳۹۵
شمارگان	:	۱۰۰۰ نسخه
قیمت	:	۱۱۰۰۰ تومان
شابک	:	۹۷۸-۶۰۰-۳۵۶-۵۶۶-۱

نشر آرنا

www.arnapub.com

فهرست مطالب

مقدمه :

دنیای پردازنده‌های گرافیکی[1] دنیای غریبی است. این تراشه‌ها کار خود را با نمایش کاراکترهای ساده آغاز کردند و با نمایش اولین رنگ‌ها همه را شگفت‌زده کردند. کار پردازنده‌های گرافیکی به جایی رسیده است که برای پردازنده‌های مرکزی[2] سیستم شاخ و شانه می‌کشند و حتی شعار منسوخ کردن آن‌ها را سر می‌دهند[42].

امروزه پردازنده‌های گرافیکی که بر روی کارت‌گرافیک‌های گران‌قیمت نصب می‌شوند توان پردازشی خارق‌العاده‌ای را نسبت به پردازنده‌های مرکزی ارائه می‌دهند، این موضوع موجب گسترش کاربردهای این پردازنده‌ها در حوزه‌هایی فراتر از بازی‌های کامپیوتری گشته است، پردازنده‌های گرافیکی مدرن با معماری موازی خود پردازنده‌های بسیار سریعی به شمار می‌روند، در عین حال با قیمت و توان مصرفی کمتری عرضه می‌شوند و جهت پیاده‌سازی الگوریتم‌ها ، یک راه‌کار اقتصادی و کارآمد به شمار می‌رود، طوری که برنامه نویسان بدون نیاز به فراگیری واسطهای برنامه نویسی گرافیکی به کمک کتابخانه‌های موجود برای این‌کار می‌توانند بار پردازشی برنامه خود را به سادگی از پردازنده مرکزی به پردازنده گرافیکی منتقل کنند.

بکار گیری پردازنده‌گرافیکی در محاسبات عمومی جایگاه جدیدی برای کارت‌گرافیک‌های قدرتمند ایجاد کرده است، جایگاهی که از پردازنده‌های گرافیکی دیگر فقط برای پردازش محاسبات گرافیکی بازی‌های‌کامپیوتری استفاده نمی‌شود، در عوض در نقش یک پردازنده کمکی، بخش یا تمامی بار محاسباتی پردازنده مرکزی را تقبل کرده و به عملیات برنامه‌ها سرعت می‌بخشد.

برای ارتباط و برنامه نویسی در پردازنده های گرافیکی روش ها و معماری های مختلفی ارائه شده است.شرکت Nvidia برای کارتهای گرافیکی خود پلتفرم CUDA[3] را طراحی کرده است که کار با سخت افزار پردازنده گرافیکی را بسیار آسان می کند.عدم درگیری مستقیم با سخت افزار، امکان برنامه نویسی با زبان C ، سرعت بالای اجرا و آسانی استفاده از مزایای این

---

[1] Graphic Processing Unit (GPU)
[2] Central Processing Unit (CPU)
[3] Compute Unified Device Architecture

این پلتفرم است. مزایای برنامه نویسی روی پردازنده‌گرافیکی بسیار زیاد است. این که می توان با استفاده از پردازنده های گرافیکی سرعت اجرا را تا بیش از ۱۰ برابر افزایش داد موضوعی فوق‌العاده است. با این حال تا مدتها این مزیت بوسیله یک مانع بزرگ محدود شده بود وآن شیوه برنامه نویسی در پردازنده گرافیکی بود و برنامه نویسانی که با زبانهای معمول مانند C و C++ کار می کردند به سختی می توانستند با زبان جدید پردازنده‌گرافیکی کار کنند.این همان چیزی است که CUDA را محبوب کرده است. در حقیقت با استفاده از این معماری می توان با زبان C برنامه نوشت و روی پردازنده گرافیکی اجرا کرد. در حقیقت CUDA یک معماری است که به جان محدود کردن شما توسط کتابخانه ها اجازه می دهد با زبان C کارهای مورد نظر خود را انجام دهید.

یکی از حوزه هایی که پردازنده های گرافیکی در آن به خوبی ظاهر شده اند الگوریتم های فرا مکاشفه ای و مبتنی بر هوش جمعی برای حل مسائل بهینه سازی است. مسائل بهینه سازی دسته مهمی از مسائل هستند که در بیشتر شاخه های علوم کاربرد دارند. در اینگونه مسائل هدف پیدا کردن مقدار کمینه یا بیشینه (بهینه) از یک تابع یا فرایند است. برای حل اینگونه مسائل الگوریتم های مختلفی ارائه شده است اما در چند دهه اخیر طبیعت الهام بخش ابداع تعداد نسباتاً زیادی از الگوریتم های توانمند بهینه سازی بوده است. از جمله قابلیت های این نوع الگوریتم ها می توان به توانایی جستجوی موثر فضاهای بسیار بزرگ در زمان کم، عدم نیاز به مشتق تابع هدف، توانایی گریز از نقاط بهینه محلی، هزینه محاسباتی بسیار کم و ریاضیات آسان اشاره کرد. به دلیل وجود ذرات مصنوعی در این الگوریتم ها ، ارائه نسخه موازی آنها که قابلیت پردازش توسط پردازنده های گرافیکی را داشته باشد ، بسیار مورد توجه قرار گرفته است. تا کنون برای بسیاری از این الگوریتم ها مثل الگوریتم ژنتیک[24]، الگوریتم بهینه سازی مورچه[29] ، الگوریتم بهینه سازی زنبور عسل [27]، الگوریتم بهینه سازی تراکم ذرات [31]و بسیاری دیگر، نسخه های موازی قابل اجرا در پردازنده گرافیکی طراحی شده که سرعت اجرا و نتیجه را تا ده ها برابر افزایش داده اند. استراتژی ها و روش های مختلفی برای موازی سازی اینگونه الگوریتم ها ارائه شده است . برای مثال در [11] یک نسخه الگوریتم بهینه سازی ژنتیک به صورت موازی در معماری CUDA پیاده سازی شده است که در این الگوریتم از روش Island استفاده می شود و با استفاده از حافظه اشتراکی در پردازنده‌گرافیک

نیاز به استفاده ازSystem Bus را از میان برده است و سرعت اجرا به واسطه بالا رفتن سرعت ارتباط، افزایش یافته است.در این الگوریتم برای هر رشته از الگوریتم ، یک نخ در پردازنده‌گرافیکی در نظر گرفته شده است. مهاجرت بین Island ها توسط حافظه اصلی در پردازنده‌گرافیکی انجام می شود و برای مرتب سازی مقادیر تابع تناسب از الگوریتم Bitonic merge Sort استفاده شده است. همچنین در [10] نویسندگان با استفاده از الگوریتم بهینه‌سازی زنبور و تکنولوژی CUDA یک الگوریتم موازی زنبور به نام CUBA طراحی کرده اند که قابلیت اجرا بر روی سخت افزار SIMD موجود در پردازنده‌گرافیکی را دارد. در این الگوریتم در هر بلوک از سخت افزار کارت گرافیکی ، چند کلونی زنبور پیاده‌سازی شده است. ارتباط بین کلونی ها از طریق Shared Memory انجام می شود که باعث بالا رفتن سرعت ارتباط خواهد شد. همچنین طراحان برای بالا بردن سرعت اجرا از تکنیک Odd Even Sort برای مرتب سازی مقادیر تابع هدف زنبور ها، به صورت موازی استفاده کرده اند که این امر کارائی الگوریتم را بالا برده است. سرعت دسترسی به جواب برای چندین تابع معیار بین ۴ تا ۱۶ برابر سریعتر از الگوریتم ترتیبی زنبور است. در[12] نویسندگان یک الگوریتم تکامل دیفرانسیل موازی طراحی کرده اند که با استفاده از معماری CUDA بر روی پردازنده های گرافیکی شرکت Nvidia قابل اجرا می باشد. در این الگوریتم سازمان داده ها در داخل پردازنده گرافیکی به صورت یک آرایه یک بعدی طراحی شده است. همچنین نکته جالب توجه که سرعت این الگوریتم را افزایش داده است ، تولید اعداد تصادفی مورد نیاز اجرای الگوریتم در ابتدای عملیات و قبل از شروع اجرای الگوریتم می باشد. اجرای این الگوریتم برای چند تابع معیار معروف ، افزایش سرعت ۱۱ برابری را نشان می دهد. محققان در مقاله [9] یک الگوریتم کلونی مورچه موازی قابل اجرا در سخت افزار پردازنده‌گرافیکی ارائه کرده‌اند.این الگوریتم بر اساس الگوریتم استاندارد MMAS طراحی شده است. از این الگوریتم برای حل مسئله فروشنده دوره گرد استفاده شده است که توانسته در مقایسه با الگوریتم ترتیبی استاندارد سرعت را به اندازه ۲۳,۶ برابر افزایش دهد.در این الگوریتم از دو روش کلونی مورچه متعدد و مورچه های موازی استفاده شده است. در روش کلونی مورچه متعدد، چندین کلونی مورچه بین واحد های پردازشی پخش می شوند و به هر کلونی یک Block اختصاص می‌یابد و برای هر مورچه یک Thread از همان Block

در نظر گرفته می شود. در روش مورچه های موازی از یک کلونی مورچه استفادهٔ می شود و مورچه ها بین عناصر پردازشی توزیع می شوند و برای هر مورچه یک Thread در نظر گرفته شده که عملیات محاسباتی مربوط به آن را انجام می دهد. در همه روش های ارائه شده برای موازی سازی و اجرای الگوریتم های مبتنی بر جمعیت در پردازنده گرافیکی ، کارائی و سرعت اجرای الگوریتم ها تا ده ها برابر افزایش یافته است که این مطلب اهمیت و توانایی پردازنده های گرافیکی را در اجرای الگوریتم های بهینه سازی نشان می دهد. یکی دیگر از این الگوریتم‌ها الگوریتم بهینه سازی غذایابی باکتری[1] می باشد[16]. این الگوریتم نیز یک الگوریتم مبتنی بر هوش جمعی است که از روش غذا یابی باکتری ها در طبیعت الهام گرفته است. با توجه به وجود ذرات مصنوعی یعنی همان باکتری ها در الگوریتم بهینه سازی غذایابی باکتری ، امکان موازی سازی آن و طراحی برای اجرا در پردازنده گرافیکی وجود دارد.

در این کتاب یک نسخه موازی از الگوریتم بهینه سازی غذایابی باکتری طراحی و پیاده سازی شده است که قابلیت اجرا بر روی پردازنده گرافیکی را دارد. این طراحی توسط پلتفرم CUDA انجام شده که قابلیت اجرا در پردازنده های گرافیکی شرکت Nvidia را دارا می‌باشد. سپس الگوریتم ارائه شده با نام CB-BFO[2] بر روی چندین مسئله معروف بهینه‌سازی پیوسته با الگوریتم بهینه سازی غذایابی باکتری استاندارد مقایسه شده است و نتایج نشان می‌دهد که الگوریتم موازی ارائه شده سرعت همگرایی بیشتری نسبت به الگوریتم غذایابی باکتری دارد.

ساختار کتاب :

فصل اول : در این فصل به بررسی سخت افزار پردازنده گرافیکی پرداخته می شود. عناصر تشکیل دهنده پردازنده، نحوه ارتباط، انواع حافظه های موجود و عناصر پردازشی در پردازنده گرافیکی مورد بررسی قرار می گیرد. همچنین نحوه ارتباط توسعه دهندگان با سخت افزار

[1] Bacterial Foraging Optimization (BFO)
[2]

پردازنده گرافیکی توسط پلتفرم CUDA بررسی می شود. در این فصل چگونگی ارتباط بین پردازنده اصلی و پردازنده گرافیکی با استفاده از CUDA معرفی و بررسی می شود. قابلیت‌های نرم افزاری CUDA و چگونگی استفاده از آن در زبان C در این فصل ارائه می‌شود.

فصل دوم : در این فصل به معرفی و آشنایی با الگوریتم غذایابی باکتری پرداخته شده است نحوه غذایابی باکتری در طبیعت ، الگوریتم استاندارد غذایابی باکتری ، معرفی پارامتر های مهم باکتری و تحلیل آنها از موارد مورد بحث در این فصل می باشد. همچنین نقاط ضعف الگوریتم غذایابی باکتری و کارهای انجام شده برای پوشش این نقاط ضعف نیز مورد بررسی و تحلیل قرار می گیرد.

فصل سوم : در این فصل الگوریتم موازی غذایابی باکتری CB-BFO ارائه شده و مورد بررسی قرار می گیرد. تغییرات انجام شده بر روی الگوریتم ، نحوه موازی سازی و بهره گیری از امکانات سخت افزاری و نرم افزاری ، مورد بررسی قرار می گیرد.

فصل پنجم : در این فصل نحوه مقایسه الگوریتم موازی غذایابی باکتری و الگوریتم استاندارد غذایابی باکتری تشریح می شود. توابع محک مورد استفاده معرفی می شوند. پارامترهای مورد استفاده برای اجرای الگوریتم ها مشخص می شوند. همچنین نتیجه مقایسه شبیه سازی دو الگوریتم ارائه شده و جداول و نمودار های مربوط به نتایج اجرا مشخص شده و مورد بحث قرار می گیرد.