مکانیک سازه‌ها و شاره‌ها/ سال ۱۳۹۷/ دوره ۸/ شماره ۳/ صفحه ۳۰۳-۳۱۴

مجله علمی پژوهشی مکانیک سازه‌ها و شاره‌ها

DOI: 10.22044/jsfm.2018.5553.2361

ارتقاء عملکرد روش توماس شطرنجی با بهره‌گیری از حافظه اشتراکی برای حل مسائل انتقال حرارت روی پردازنده گرافیکی

سید علیرضا ذوالفقاری[۱,*] و علی فؤادالدینی[۲]

[۱] عضو هیأت علمی گروه مهندسی مکانیک و مدیر گروه پژوهشی انرژی در ساختمان و آسایش حرارتی، دانشگاه بیرجند، بیرجند

[۲] دانشجوی دکتری، مهندسی مکانیک، دانشگاه بیرجند، بیرجند

مقاله مستقل، تاریخ دریافت: ۱۳۹۶/۰۱/۰۹؛ تاریخ بازنگری: ۱۳۹۷/۰۳/۰۶؛ تاریخ پذیرش: ۱۳۹۷/۰۶/۱۰

چکیده

پردازنده گرافیکی همه منظوره کاربر را قادر می‌سازد تا از پردازنده گرافیکی برای مقاصد محاسباتی عمومی بهره بگیرد. استفاده از این نوع پردازنده‌ها، موجب افزایش سرعت محاسبات در حل مسائل دینامیک سیالات محاسباتی می‌شود. تحقیقات متعددی جهت بررسی مزیت استفاده از پردازنده گرافیکی در محاسبات از جمله بکارگیری آن برای حل دستگاه معادلات سه‌قطری صورت گرفته است. در سال ۲۰۱۶ روش توماس شطرنجی برای حل دستگاه معادلات سه‌قطری در حلگر ADI ارائه شد. در این روش هر دستگاه معادلات به چندین دستگاه معادلات مستقل و با ابعاد کوچکتر تقسیم می‌شود. سپس هر یک از این دستگاه‌ها، به وسیله الگوریتم توماس به صورت شطرنجی حل می‌شوند؛ همچنین، در این روش علاوه بر مشارکت تعداد زیادی نخ محاسباتی در حل امکان ذخیره اطلاعات مربوط به هر دستگاه معادلات در حافظه اشتراکی میسر می‌شود. در تحقیق حاضر با توجه ملاحظات مرتبط با استفاده از حافظه اشتراکی، راهبردی جهت بکارگیری این حافظه در روش توماس شطرنجی ارائه شده است. نتایج نشان می‌دهد که بکارگیری حافظه اشتراکی، موجب افزایش سرعتی بین ۱/۲ تا ۱/۶ برابر نسبت به وضعیت بکارگیری حافظه سراسری شده است؛ همچنین مشخص شد که تداخل دسترسی به حافظه اشتراکی، موجب کاهش سرعتی بین ۱۰/۹ تا ۱۸/۸ درصد در روش توماس شطرنجی می‌شود.

کلمات کلیدی: روش توماس شطرنجی؛ پردازنده گرافیکی همه منظوره؛ حافظه اشتراکی؛ دستگاه معادلات سه‌قطری.

Enhancing the Performance of Checkerboard Thomas Method by Using Shared Memory for Solving the Heat Transfer Problems on GPU

S. A. Zolfaghari [1,*], A. Foadaddini [2]

[1] Department of Mechanical Engineering, University of Birjand, Birjand, Iran.

[2] Ph.D. Student, Mechanical Engeering, University of Birjand, Birjand, Iran.

Abstract

General Purpose Graphics Processing Unite (GPGPU) allows the user to utilize GPU for general computing purposes. Using these processors can cause a significant speedup in numerical calculation for solving CFD problems. Several studies have been performed to investigate the advantages of using the GPGPU in numerical calculations including solving tridiagonal set of equations. In 2016, Checkerboard method introduced for solving tridiagonal set of equations in ADI solvers. In this method each set of equations is divided in to several smaller independent set of equations. Then each one of them will be solved by Thomas algorithm in checkerboard style. In addition to the participation of many threads, in this method it is possible to store the information of each set of equation in shared memory. In the present research, according to consideration around using shared memory, a strategy for using this memory in checkerboard Thomas method has been offered. Results shows that utilizing shared memory has been caused to computing speedup between 1.2x to 1.6x, compared with utilizing global memory. Also, it was found that bank conflict causes to decrease the speed from 10.9% to 18.8% in checkerboard Thomas method.

Keywords: Checkerboard Thomas Method; GPGPU; Shared Memory; Tridiagonal Set of Equations.

* نویسنده مسئول؛ تلفن: ۰۵۶۳۲۲۰۲۰۵۰؛ فکس: ۰۵۶۳۲۲۰۲۳۴۸

آدرس پست الکترونیک: zolfaghari@birjand.ac.ir

۱- مقدمه

استفاده از هسته‌های پردازنده متعدد و بهره‌گیری از معماری چندنخی[1]، پردازنده گرافیکی[2] را به یک ابزار قدرتمند پردازشی تبدیل کرده است. پردازنده گرافیکی همه منظوره[3] استفاده از امکانات پردازنده گرافیکی را برای مقاصد محاسباتی عمومی فراهم کرده است. استفاده از پردازنده گرافیکی همه منظوره، برای انجام محاسبات عددی از سال ۲۰۰۱ آغاز شد و برای اولین بار مزیت بکارگیری آن در انجام عملیات فاکتورگیری ال-یو[4] در سال ۲۰۰۷ واضح گردید [۱]. اصلاح روش‌های عددی برای ایجاد سازگاری با ساختمان پردازنده گرافیکی، لازمه بهره‌گیری موثر از این ابزار برای انجام محاسبات است. از جمله روش‌های عددی پرکاربرد در محاسبات عددی الگوریتم‌های حل دستگاه معادلات سه‌قطری هستند. در سال‌های اخیر، تحقیقات زیادی برای توسعه الگوریتم حل دستگاه معادلات سه‌قطری برای ایجاد سازگاری با معماری پردازنده گرافیکی انجام گرفته است. سنگوپتا و همکاران [۲] در سال ۲۰۰۷ برای اولین بار الگوریتم کاهش متناوب[5] را روی پردازنده گرافیکی جهت مدل‌سازی جریان بکار گرفتند. ساخارنی [۳] در سال ۲۰۰۹، الگوریتم توماس موازی[6] را جهت مدل‌سازی جریان سیالات بکار برد. وی حداکثر افزایش سرعت ۱۱ برابری را در مقایسه با حل پردازنده مرکزی گزارش کرد؛ همچنین ژانگ و همکاران [۴]، روشی ترکیبی در کنار الگوریتم‌های توماس، کاهش متناوب، کاهش متناوب موازی[7] و دوبرابر سازی معکوس[8] پیشنهاد کردند. تحقیقات ایشان بر نقش عملیات زمانبر انتقال اطلاعات بین حافظه سراسری و حافظه میزبان در کاهش سرعت محاسبات صحه گذاشت؛ همچنین در تحقیقات ایشان وجود تداخل در دسترسی به حافظه اشتراکی[9] در الگوریتم کاهش متناوب و مزیت بهره‌گیری از الگوریتم‌های ترکیبی مورد بررسی قرار گرفت. گودک و همکاران [۵] در سال ۲۰۱۱، الگوریتم کاهش متناوب ارائه کردند که در آن تداخل دسترسی به حافظه اشتراکی[10] اتفاق نمی‌افتد. الگوریتم کاهش متناوب در سال ۲۰۱۱ توسط دیویدسون و همکاران [۶]، به روش فشرده‌سازی حافظه ثبات[11] بهینه‌سازی شد. در همان سال ساخارنیخ [۷]، ترکیبی از الگوریتم کاهش متناوب موازی و توماس را ارائه کرد. روش مذکور توسط ژانگ و همکاران [۸] مورد بررسی قرار گرفت و افزایش سرعت حدودا ۲ برابری نسبت به الگوریتم کاهش متناوب گزارش شد. اگلوف [۹] نمونه‌ای از الگوریتم کاهش متناوب موازی را برای حلگرهای تفاضل محدود جهت حل دستگاه معادلات سه قطری در ابعاد بزرگ ارائه نمود. وی و همکاران [۱۰] با ارائه روشی جدید جهت سازگاری الگوریتم کاهش متناوب موازی با دستگاه معادلات بزرگ و ایجاد امکان دسترسی هم مکان[12] به حافظه سراسری[13] عملکرد حلگر ضمنی جهت متغیر[14] را برای حل معادلات هدایت حرارتی گذرا، بهبود بخشیدند. کیم و همکاران [۱۱] با ارائه روشی مرکب از الگوریتم کاهش متناوب موازی موزاییکی[15] و توماس موازی جهت حل دستگاه معادلات بزرگ به افزایش سرعت ۸ و ۴۹ برابری نسبت به روش چند نخی و سری با استفاده از حلگر ام کا ال[16] دست یافت. در سال ۲۰۱۲ اصفهانیان و همکاران [۱۲]، پیاده‌سازی طرح‌های ونو[17] را روی پردازنده گرافیکی مورد مطالعه قرار دادند؛ همچنین در سال ۲۰۱۳ اصفهانیان و همکاران [۱۳]، تاثیر نحوه دسترسی به حافظه سراسری را روی الگوریتم کاهش متناوب، مورد مطالعه قرار دادند. ایشان به افزایش سرعت حل ۱/۹ تا ۱۵/۲ برابر در دو بعد و ۶/۴ تا ۲۰/۳ برابر در سه بعد دست یافتند. همچنین داریان و اصفهانیان [۱۴] در سال ۲۰۱۴، افزایش سرعت ۱۰۵ برابر را در حل معادلات اویلر به کمک پردازنده گرافیکی گزارش کردند. در سال ۲۰۱۶، ذوالفقاری و فوادالدینی [۱۵] با توجه به برتری الگوریتم توماس نسبت به الگوریتم کاهش متناوب موازی از نظیر پیچیدگی محاسباتی

[1] Multithreaded
[2] GPU (Graphics Processing Unit)
[3] GPGPU (General Purpose Graphics Processing Unit)
[4] LU Factorization
[5] Cyclic Reduction (CR)
[6] Parallel Thomas
[7] Parallel Cyclic Reduction (PCR)
[8] Recursive Doubling (RD)
[9] Shared Memory
[10] Bank Conflict
[11] Register Packing
[12] Coalesced Access
[13] Global Memory
[14] Alternating Direction Implicit solver (ADI)
[15] Tiled PCR
[16] MKL (Math Kernel Library)
[17] WENO

و نداشتن محدودیت پیرامون ابعاد دستگاه معادلات، روشی نوین تحت عنوان توماس شطرنجی را ارائه نمودند. به‌کارگیری این روش در حلگر ADI، موجب افزایش سرعتی بین بین ۵/۷ تا ۲۲/۲ نسبت به حل پردازنده مرکزی و افزایش سرعت حدودا ۲ برابری نسبت به الگوریتم کاهش متناوب موازی شد.

روش ADI برای حل معادله هدایت پایا در ۲ بعد شامل، دو مرحله است. در هر مرحله مقادیر شبکه ساختار یافته در یک جهت به صورت ضمنی و در جهت دیگر به صورت صریح، مورد استفاده قرار می‌گیرند. حل ضمنی در هر جهت نیازمند تشکیل یک دستگاه معادله سه‌قطری به ازای هر ردیف یا ستون از شبکه است. در روش توماس شطرنجی به جای تشکیل یک دستگاه معادله، امکان تشکیل چندین دستگاه معادله به ازای هر ردیف یا ستون از شبکه ایجاد می‌شود. به این طریق امکان مشارکت تعداد زیادی نخ محاسباتی در حل هر دستگاه معادله فراهم می‌شود. با ایجاد دستگاه معادلات مستقل و کوچکتر، می‌توان اطلاعات مربوط به آنها را در حافظه اشتراکی ذخیره نمود و سرعت دسترسی به آن را افزایش داد. در تحقیق حاضر با توجه به ملاحظات مرتبط با استفاده از حافظه اشتراکی نظیر، حجم کم این حافظه و دسترسی بدون تداخل، راهبردی جهت بهره‌گیری از آن در روش توماس شطرنجی ارائه شده است؛ همچنین عملکر روش توماس شطرنجی هنگام بهره‌گیری از حافظه اشتراکی و تاثیر تداخل دسترسی با توجه به معماری پردازنده گرافیکی، مورد تحلیل قرار گرفته است.

۲- پردازش بهینه روی پردازنده گرافیکی

پردازنده گرافیکی با استفاده از تعداد زیادی هسته پردازنده و هزاران نخ[1] محاسباتی، امکانات گسترده‌ای را برای سرعت بخشیدن به محاسبات ایجاد کرده است. با این حال، بهره‌گیری از مزایا یک پردازنده گرافیکی مستلزم رعایت اصولی است که استفاده از ساختار چند نخی را ثمربخش می‌سازد. اولین گام ضروری در این زمینه، تغییر روش‌های عددی است. روش‌های عددی مناسب برای ساختار پردازنده گرافیکی، باید بتوانند تعداد زیادی نخ پردازنده را برای پنهان سازی تاخیرات حافظه فراهم کند. به همین دلیل در این روش‌ها محاسبات به بخش‌های مستقل و جدای از هم تقسیم می‌شوند که امکان انجام آن به صورت موازی وجود داشته باشد.

گام اساسی دیگر در استفاده از مزایای پردازنده گرافیکی، به‌کارگیری حافظه مناسب و دسترسی بهینه به حافظه است. همان‌گونه که بیان شد، در ساختار پردازنده گرافیکی انواع متفاوتی از حافظه شامل، سراسری، اشتراکی، ثابت[2]، ثبات، بافتی[3] و ... وجود دارد. شکل ۱ نمایی کلی از ارتباط میان هسته‌های پردازنده و حافظه‌ها در ساختار پردازنده گرافیکی را نشان می‌دهد. فلش یک طرفه به این معناست که پردازنده تنها قادر به خواندن حافظه است و فلش‌های دوطرفه به این معناست که پردازنده قادر است، هم حافظه را خوانده و هم بنویسد. هر یک از این حافظه‌ها دارای ویژگی خاصی هستند که آنها را برای کاربرد ویژه‌ای مناسب می‌سازد. از جمله هر یک از آنها دارای حجم، سرعت و دامنه دسترسی[4] خاصی هستند که هنگام به‌کارگیری باید مورد توجه قرار گیرد. همچنین نحوه نوشتن و خواندن اطلاعات در این حافظه‌ها از اهمیت بالایی برخوردار است و می‌تواند تاثیر بسزایی بر سرعت انجام عملیات محاسباتی داشته باشد. در تحقیق حاضر تمرکز اصلی روی بهره‌گیری از حافظه‌های سریعتر برای تسریع انجام محاسبات است. به همین دلیل استفاده از حافظه اشتراکی به جای حافظه سراسری در روش توماس شطرنجی، مورد بررسی و تحلیل قرار گرفته است.

۳- حافظه اشتراکی

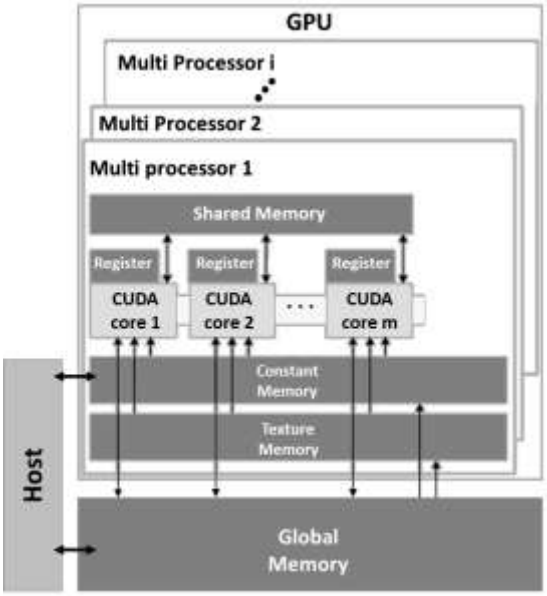
حافظه اشتراکی یک حافظه پرسرعت در ساختار پردازنده گرافیکی است (سریعتر از حافظه سراسری). در صورت نیاز، اطلاعات ورودی معمولا در شروع محاسبات از حافظه سراسری به این حافظه منتقل می‌شود. هر مجتمع پردازشی[5] دارای مقدار حافظه اشتراکی محدودی است. این مقدار نسبت به حجم حافظه سراسری بسیار اندک است؛

[1] Thread

[2] Constant
[3] Texture
[4] Scope
[5] Multiprocessor

بنابراین زمانی از این حافظه بهره‌گیری می‌شود که اطلاعات ذخیره شده روی آن به دفعات زیاد مورد خواندن و نوشتن قرار بگیرند.

هنگام اجرای برنامه پس از قرارگیری بلاک‌ها[1] در مجتمع‌های پردازشی در صورتی که استفاده از حافظه اشتراکی مورد نیاز باشد، به همه بلاک‌ها مقدار برابری از آن اختصاص می‌یابد. باید توجه داشت که دامنه دسترسی به این حافظه، نخ‌های یک بلاک است؛ به این معنی که نخ‌های یک بلاک به اطلاعات ذخیره شده در حافظه اشتراکی مربوط به بلاک دیگر به منظور خواندن یا نوشتن دسترسی ندارند.



شکل ۱- ارتباط حافظه‌ها و هسته‌های کودا در ساختار پردازنده گرافیکی

۳-۱- حافظه اشتراکی و مشغولیت پردازنده گرافیکی

پردازنده گرافیکی در مواقعی که اجرای دستورالعمل‌های محول شده به نخ‌های یک وارپ[2] زمانبر باشد، دست به زمانبندی دستورالعمل‌های مربوط به وارپ دیگر می‌زند. به این ترتیب تاخیرات زمانی مربوط به عملیات‌های زمانبر نظیر دسترسی به حافظه سراسری پنهان می‌شود. پنهان ساختن تاخیرات حافظه در صورتی به طور کامل انجام پذیر می‌باشد، که نخ‌های پردازشی فعال کافی در مجتمع پردازشی وجود داشته باشد. حداکثر تعداد وارپ‌های محاسباتی فعال، از ویژگی‌های پردازنده گرافیکی است که به وسیله قابلیت محاسباتی[3] (نسخه) آن مشخص می‌شود. در صورتی که نخ‌های محاسباتی فعال با این مقدار حداکثر برابر باشد، بیشترین امکان جهت پنهان ساختن تاخیرات حافظه فراهم می‌شود. نسبت میزان وارپ‌های فعال به حداکثر وارپ‌های فعال ممکن، مشغولیت[4] نامیده می‌شود. یکی از عوامل موثر در مشغولیت، نحوه بهره‌گیری از حافظه اشتراکی است.

بسته به قابلیت محاسباتی (نسخه) پردازنده گرافیکی، هر مجتمع پردازشی از مقدار مشخصی حافظه اشتراکی برخوردار است. این مقدار حافظه اشتراکی باید بین بلاک‌های موجود در آن تقسیم گردد، بنابراین هرچه مقدار حافظه اشتراکی اختصاص یافته به هر بلاک بیشتر شود، تعداد بلاک‌های موجود در مجتمع پردازشی با توجه به محدودیت منابع کاهش می‌یابد. کاهش تعداد بلاک‌ها به معنی کاهش وارپ‌ها و کاهش مشغولیت پردازنده گرافیکی خواهد بود. این موضوع تاثیری منفی بر توان پردازنده گرافیکی برای پنهان‌سازی تاخیرات حافظه خواهد داشت و سرعت انجام محاسبات را کاهش خواهد داد.

۳-۲- تداخل دسترسی به حافظه اشتراکی

نحوه دسترسی به حافظه اشتراکی نیز، از عوامل مهمی است که باید مورد توجه قرار گیرد. یکی از عواملی که موجب کاهش سرعت دسترسی به حافظه اشتراکی می‌گردد، تداخل دسترسی به حافظه است. نخ‌های موجود در یک وارپ از طریق ۳۲ بنک[5] به اطلاعات حافظه اشتراکی دسترسی پیدا می‌کنند. در صورتی که اطلاعات مورد نیاز این ۳۲ نخ از بنک‌های جداگانه مورد دسترسی قرار بگیرد، دسترسی به حافظه سریع خواهد بود. در غیر این صورت تداخل دسترسی

[1] Block
[2] Warp
[3] Computational Capability
[4] Occupancy
[6] Bank

به حافظه رخ می‌دهد. تداخل دسترسی به حافظه اشتراکی، موجب سری شدن دسترسی به حافظه شده و موجب کاهش سرعت عملیات می‌شود.

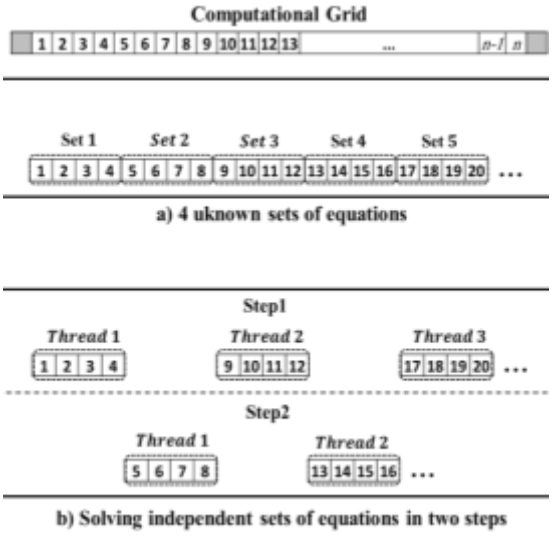
۴- الگوریتم توماس شطرنجی

تا کنون روش‌های متعددی برای حل دستگاه معادلات سه‌قطری ارائه شده است. یکی از مشهورترین این روش‌ها، الگوریتم توماس است. استفاده از این روش هنگام انجام محاسبات به صورت سری و با بهره‌گیری از پردازنده مرکزی بسیار رایج است، با این حال در این روش امکان انجام محاسبات به صورت موازی و مستقل از هم وجود ندارد. در حل دستگاه معادلات سه‌قطری به صورت موازی و به وسیله پردازنده گرافیکی، دو روش کاهش متناوب (CR) و کاهش متناوب موازی (PCR) نسبت به الگوریتم توماس مزیت دارند. در این دو روش، امکان انجام محاسبات مستقل از هم فراهم شده است که می‌توان آنها را به صورتی موازی و روی نخ‌های پردازشی متعدد پیگیری کرد. دو الگوریتم CR و PCR، در مقایسه با الگوریتم توماس دارای تعداد مراحل سری کمتری هستند. این ویژگی، روش‌های مذکور را جهت انجام محاسبات روی پردازنده گرافیکی مناسب می‌سازد، با این حال تعداد عملیات محاسباتی و دفعات دسترسی به حافظه در این روش‌ها برای بدست آوردن مجهولات، بسیار بیشتر از الگوریتم توماس است. بعلاوه، الگوریتم CR و PCR تنها قادر به حل دستگاه معادلاتی با 2^n مجهول است؛ حال آنکه الگوریتم توماس محدودیتی از جهت اندازه دستگاه معادلات مد نظر ندارد. با توجه به مزیت‌های ذکر شده برای الگوریتم توماس نسبت به دو الگوریتم دیگر، ذوالفقاری و فوادالدینی [۱۵] روشی جدید تحت عنوان الگوریتم توماس شطرنجی را ارائه نمودند که در آن امکان بهره‌گیری از الگوریتم توماس در حلگر ADI، به‌گونه‌ای سازگار با معماری پردازنده گرافیکی فراهم شده است.

سری بودن الگوریتم توماس، مهمترین مانع بهره‌گیری از این الگوریتم روی پردازنده گرافیکی است که استفاده از مزایای آن از جمله سادگی محاسباتی و دفعات پایین دسترسی به حافظه را محدود می‌سازد. جهت رفع این مانع می‌توان با تقسیم دستگاه معادله اولیه به چندین دستگاه کوچکتر و حل تکراری آن، امکان مشارکت پردازنده‌های متعدد در روند حل را فراهم کرد. در این روش هر دستگاه معادله اولیه به nop دستگاه معادله با اندازه dop تقسیم می‌شود و حل این دستگاه معادلات به صورت تکراری و توسط nop نخ محاسباتی صورت می‌گیرد. برای مثال مطابق شکل ۲ شبکه محاسباتی یک بعدی، متشکل از n مجهول مدنظر است (n مضرب صحیحی از 4 می‌باشد). به جای حل مجهولات از طریق حل یک دستگاه معادله n مجهولی، می‌توان از طریق حل تکراری $nop = n/4$ دستگاه معادله چهار مجهولی به پاسخ رسید. در این حل تکراری، برای شروع، مقداری اولیه به همه مجهولات داده شده و در هر تکرار دستگاه معادلات همواره بر اساس مقادیر معلوم شبکه و معادله انفصال حاکم تشکیل خواهند شد. برای مثال، دستگاه معادله اول شامل، ۴ مجهول مربوط به نقاط ۱، ۲، ۳ و ۴ از شبکه است. این دستگاه معادله بر اساس معادله انفصال حاکم و بر اساس مقادیر معلوم مربوط به نقطه مرزی و نقطه ۵ شبکه محاسباتی به دست آمده است. مقدار معلوم نقطه ۵ در حقیقت مقداری است که در تکرار پیشین حل حاصل شده است. به همین ترتیب، مقادیر معلوم مورد استفاده در تشکیل دستگاه معادلات دوم نیز، مقادیر مربوط به نقاط ۴ و ۹ شبکه هستند که در تکرار قبل به دست آمده‌اند. حل تکراری به دو صورت قابل انجام است. می‌توان در هر تکرار هر nop دستگاه معادله را به صورت همزمان و به وسیله nop نخ محاسباتی حل کرد. به همین شکل در این روش تمام مقادیر جدید بر اساس تکرار قبل به دست می‌آیند؛ همچنین می‌توان مطابق آنچه در شکل ۲ نشان داده شده است، حل را به صورت دو مرحله انجام داد؛ به طوری که در هر مرحله دستگاه معادلات به صورت شطرنجی یا یکی در میان حل می‌شوند. در این روش، نخ‌های پردازشی در مرحله دوم از نتایج مرحله اول همان تکرار به عنوان مقادیر معلوم جهت تشکیل دستگاه معادلات بهره می‌برند. با توجه به این خاصیت، سرعت همگرایی روش شطرنجی بیشتر خواهد بود؛ چراکه این مقادیر نسبت به مقادیر تکرار قبل اصلاح شده‌تر می‌باشند و این باعث می‌شود که پاسخ دستگاه معادله به مقادیر حل نهایی شبکه نزدیکتر شود. به این ترتیب می‌توان ویژگی‌های زیر را برای روش توماس شطرنجی ذکر نمود:

۱- با این روش می‌توان هر دستگاه معادله سه‌قطری را به صورت تکراری حل نمود.
۲- این روش برخلاف الگوریتم CR و PCR هیچ محدودیتی پیرامون اندازه دستگاه معادلات ندارد.
۳- در این روش می‌توان دستگاه معادلات مستقل از هم را با توجه به مقدار n به تعداد دلخواه (nop) تعیین نمود. تنها کافی است که n بر dop بخش‌پذیر باشد. حال آنکه در روش CR و PCR تعداد بخش‌های مستقل حل برابر اندازه دستگاه معادلات است.
۴- پیاده‌سازی این روش نسبت به الگوریتم CR و PCR ساده‌تر است.

البته در این روش n نیز باید به قدر کافی بزرگ باشد تا تعداد دستگاه معادله کافی برای مشغول نگهداشتن تمام هسته‌های پردازنده و پنهان کردن تاخیرات حافظه فراهم شود. تعداد تقسیمات دستگاه معادلات (nop) از جمله پارامترهای است که عملکرد الگوریتم توماس شطرنجی را تحت تاثیر قرار می‌دهد.



شکل ۲- حل دستگاه معادلات به‌صورت شطرنجی

افزایش nop موجب می‌شود که نخ‌های محاسباتی بیشتری جهت حل دستگاه معادلات فعال شوند و از طرفی تعداد تکرارها جهت رسیدن به همگرایی را افزایش می‌دهد. هر چه تعداد این نخ‌ها بیشتر باشد، هسته‌های پردازشی بیشتری بکارگرفته شده و تاخیرات حافظه نیز بهتر پنهان می‌شود. با این حال، افزایش بیش از حد تعداد تکرارها، ضمن افزایش nop می‌تواند از سرعت همگرایی الگوریتم بکاهد. فواید و کاربرد الگوریتم توماس شطرنجی در حل تکراری دستگاه معادلات سه قطری زمانی نمود بیشتری خواهد داشت که از آن در یک حلگر ADI بهره گرفته شود.

۵- حل مسئله انتقال حرارت هدایت پایا با روش توماس شطرنجی

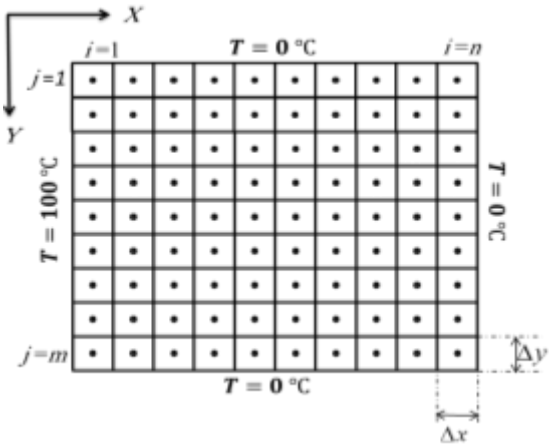
۵-۱- تعریف مسئله نمونه با حلگر ADI

حلگر ADI یک روش تحلیل عددی است که برای حل معادلات مشتق جزئی بیضوی، سهموی و هذلولی در فضای چند بعدی بکار می‌رود. این موضوع موجب شده است که این روش به طور گسترده در علوم پایه و مهندسی به کار گرفته شود. در روش ADI متناسب با ابعاد مسئله هر مرحله از محاسبات به چند زیر- مرحله تقسیم می‌شود که در آن، معادله دیفرانسیل مربوطه در یک راستا به صورت ضمنی حل می‌شود؛ در حالی که اطلاعات مسئله در سایر جهت‌ها به صورت صریح، مورد استفاده قرار می‌گیرد؛ در نتیجه، این روش بدون پیش شرط پایدار است. یک خاصیت جالب توجه این روش این است که در آن برای حل معادلات در هر زیر- مرحله می‌توان از الگوریتم حل ماتریس سه قطری بهره برد.
حل معادله هدایت حرارتی پایا به عنوان مسئله‌ای ساده، مثال خوبی برای بهره‌گیری از حلگر ADI است. معادله هدایت حرارت پایا در دو بعد به صورت معادله ۱ است:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)=0 \qquad (۱)$$

استخراج معادله انفصال به روش حجم محدود و روی شبکه نشان داده شده در شکل ۳ صورت گرفته و نتیجه به این صورت رابطه (۲) است. در صورتی که معادله انفصال مطابق رابطه (۲) جهت حل مقادیر مجهول در یک شبکه دو بعدی مورد استفاده قرار گیرد، یک دستگاه معادله ۵ قطری تشکیل می‌شود. با حل این دستگاه معادله پاسخ مورد نظر به صورت مستقیم به دست خواهد آمد؛ اما در حلگر ADI حل معادله انفصالی (۲) به صورت تکراری انجام می‌شود. در مرحله اول معادله انفصال به صورت ضمنی در جهت x حل می‌شود. در این مرحله اطلاعات شبکه در جهت y به صورت صریح مورد استفاده قرار می‌گیرند. رابطه (۳) شکل معادله

انفصالی را در این وضعیت نشان می‌دهد. در مرحله دوم معادله انفصال به صورت ضمنی در جهت y حل می‌شود. در این مرحله اطلاعات شبکه در جهت x به صورت صریح مورد استفاده قرار می‌گیرند. رابطه (۴) شکل معادله انفصالی را در این وضعیت نشان می‌دهد.



شکل ۳- شبکه و شرایط مرزی برای مسئله هدایت دو بعدی

اعمال روابط (۳) و (۴) موجب شکل‌گیری دستگاه معادلات سه قطری می‌شود که برای حل آنها می‌توان از الگوریتم‌های حل دستگاه معادلات سه‌قطری استفاده کرد. در این روابط، بالا نویس K و $K\text{-}1$ به ترتیب اشاره به تکرار جدید و تکرار قبل دارند. شرایط مرزی درنظر گرفته شده برای مسئله حاضر مطابق شکل ۴ است. ضمن اینکه حل تحلیلی مسئله حاضر در مرجع [۱۶] ارائه شده است.

$$\left(2\frac{k\Delta y}{\Delta x}+2\frac{k\Delta y}{\Delta x}\right)T(i,j) = \frac{k\Delta y}{\Delta x}T(i-1,j)+\frac{k\Delta y}{\Delta x}T(i+1,j) + \frac{k\Delta x}{\Delta y}T(i,j+1)+\frac{k\Delta x}{\Delta y}T(i,j-1) \quad (۲)$$

$$\left(2\frac{k\Delta y}{\Delta x}+2\frac{k\Delta y}{\Delta x}\right)T^{K}(i,j) = \frac{k\Delta y}{\Delta x}T^{K}(i-1,j)+\frac{k\Delta y}{\Delta x}T^{K}(i+1,j) + \frac{k\Delta x}{\Delta y}T^{K-1}(i,j+1)+\frac{k\Delta x}{\Delta y}T^{K-1}(i,j-1) \quad (۳)$$

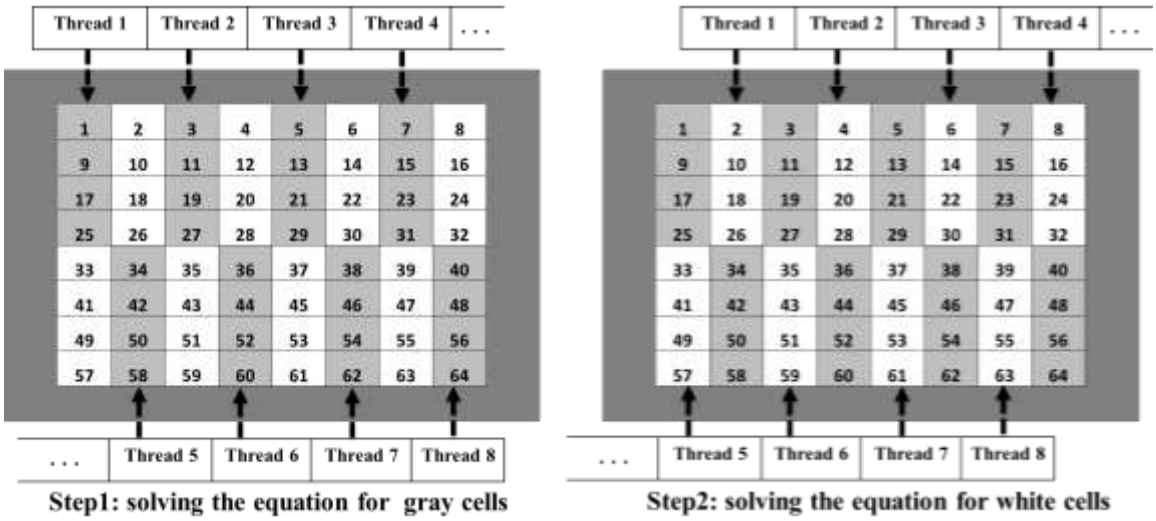
$$\left(2\frac{k\Delta y}{\Delta x}+2\frac{k\Delta y}{\Delta x}\right)T^{K}(i,j) = \frac{k\Delta y}{\Delta x}T^{K-1}(i-1,j)+\frac{k\Delta y}{\Delta x}T^{K-1}(i+1,j) + \frac{k\Delta x}{\Delta y}T^{K}(i,j+1)+\frac{k\Delta x}{\Delta y}T^{K}(i,j-1) \quad (۴)$$

۵-۲- پیاده سازی حلگر با بکارگیری الگوریتم توماس شطرنجی و با بهره‌گیری از حافظه اشتراکی

در مرجع [۱۵] شیوه پیاده‌سازی حلگر ADI برای حل معادله انتقال حرارت هدایت پایای در ۲ بعد با استفاده از الگوریتم توماس شطرنجی ارائه شده است. در این روش برای حل ضمنی معادله دیفرانسیل در هر جهت، میدان حل در آن راستا به nop قسمت تقسیم می‌شود. این nop قسمت به صورت یکی در میان به وسیله $nop/2$ نخ محاسباتی در دو مرحله حل می‌شوند. به عنوان مثال در شکل ۴، مراحل حل معادله دیفرانسیل در جهت y برای شبکه‌ای به اندازه ۸×۸ و $nop = 2$ نشان داده شده است. همانطور که در شکل مذکور ارائه شده است، حل در هر جهت به صورت دو مرحله‌ای صورت می‌گیرد. در یک مرحله تمام سلول‌های خاکستری و در مرحله بعد تمام سلول‌های سفید به وسیله نخ‌های محاسباتی اختصاصی حل می‌شوند. به دلایلی که در مرجع [۱۵] به طور کامل توضیح داده شده است، جهت حل ضمنی معادله در جهت x به‌جای تقسیم جداسازی معادله دیفرانسیل در این جهت، کل ماتریس مربوط به مقادیر شبکه ترانهاده شده و معادله انفصال مجددا در جهت y استخراج و حل می‌گردد. لازم به ذکر است که این شیوه به‌کارگیری الگوریتم توماس شطرنجی در حلگر ADI مختص حل مسئله هدایت پایا است و برای مسائل گذرا شیوه پیاده‌سازی از جهاتی متفاوت است.

برای اعتبارسنجی این روش، حلگر ADI برای مسئله‌ای با هندسه و شرایط مرزی شکل ۳ پیاده‌سازی شده و نتایج حل عددی با حل تحلیلی مرجع [۱۶] مقایسه شده است. در حل عددی از شرط توقف ارائه شده در رابطه (۵) استفاده شده است:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}(T^{K}(i,j)-T^{K-1}(i,j))^2 < 10^{-6} \quad (۵)$$



شکل ۴- حل معادله دیفرانسیل به صورت صریح در جهت y به وسیله حلگر ADI

در جدول ۱ دمای مرکز دامنه حل ($x=0.5, y=0.5$) برای دو روش توماس شطرنجی ($dop=8$) و توماس معمولی و اندازه شبکه‌های مختلف ارائه شده و خطای حل عددی بر اساس رابطه (۶) محاسبه شده است:

$$Error = |T_A - T_N| \qquad (۶)$$

که در آن T_A مقدار پاسخ حل تحلیلی و T_N مقدار پاسخ حل عددی است.

جدول ۱- دمای مرکز دامنه حل ($x=0.5, y=0.5$) برای دو روش توماس شطرنجی ($dop=8$) و توماس معمولی

		اندازه شبکه			
		۱۲۸×۱۲۸	۲۵۶×۲۵۶	۵۱۲×۵۱۲	۱۰۲۴×۱۰۲۴
CH-Thomas	T	۲۴/۳۴	۲۴/۶۴	۲۴/۷۸	۲۴/۸۰
	Error	۰/۶۶	۰/۳۶	۰/۲۲	۰/۲
Thomas	T	۲۴/۳۴	۲۴/۶۵	۲۴/۷۹	۲۴/۸۱
	Error	۰/۶۶	۰/۳۵	۰/۲۱	۰/۱۹

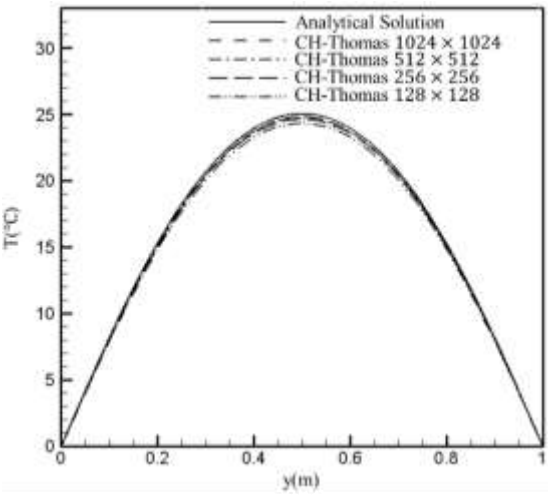
همانطور که در جدول ۱ ارائه شده است، خطای روش توماس شطرنجی بین ۰/۶۶ تا ۰/۲ است و با افزایش تقسیمات شبکه کاهش می‌یابد. همچنین اختلاف نتایج حل عددی برای دو روش توماس شطرنجی و توماس معمولی حدود ۰/۰۱ است.

در شکل ۵ توزیع دما در خط میانی دامنه حل (x=0.5) برای روش توماس شطرنجی (با $dop=8$) و حل تحلیلی نمایش داده شده است. همانگونه که مشاهده می‌شود، توزیع دما در روش توماس شطرنجی تطابق خوبی با حل تحلیلی داشته و با افزایش اندازه شبکه بهبود می‌یابد.

همانگونه که قبلا گفته شد در روش توماس شطرنجی دستگاه معادلات بزرگ به دستگاه معادلات کوچکتر تقسیم می‌شوند تا امکان حل موازی به‌وسیله الگوریتم توماس به صورت تکراری فراهم گردد. کوچک کردن دستگاه معادلات همچنین (با توجه به محدودیت حجم حافظه اشتراکی) این امکان را فراهم می‌کند تا جهت حل این دستگاه معادلات بتوان از حافظه اشتراکی استفاده نمود؛ چراکه امکان ذخیره اطلاعات مربوط به چندین دستگاه معادله در حافظه اشتراکی اختصاص یافته به یک بلاک فراهم می‌گردد.

همانگونه که پیش از این در بخش ۳ بیان شد، با توجه به محدودیت اندازه حافظه اشتراکی تنها اطلاعاتی در این حافظه قرار می‌گیرند که به صورت متناوب خوانده و نوشته

شوند. در روش به‌کار رفته در تحقیق حاضر ضرایب c و d به دلیل اینکه در هر حلقه از الگوریتم به صورت متناوب فراخوانده می‌شوند، در حافظه اشتراکی ذخیره شده و ضرایب a و b در حافظه سراسری ذخیره می‌شوند. همچنین با توجه به محدودیت حافظه اشتراکی اختصاص یافته به هر بلاک باید اندازه هر بلاک و اندازه دستگاه معادله‌ای که به وسیله هر نخ محاسباتی حل می‌شود، باید کوچک باشد. در تحقیق حاضر اندازه هر بلاک ۳۲ نخ و اندازه هر دستگاه معادله نیز، کوچکتر از ۳۲ انتخاب شده است (dop<32). مسئله حائز اهمیت هنگام بهره‌گیری از حافظه اشتراکی پیشگیری از تداخل دسترسی به حافظه اشتراکی است. در روش حاضر، می‌توان با ذخیره مناسب اطلاعات در حافظه از تداخل دسترسی به حافظه جلوگیری به عمل آورد. در بخش نتایج تاثیر تداخل دسترسی به حافظه بر سرعت حلگر، مورد بررسی قرار گرفته است.



شکل ۵- توزیع دما در خط میانی دامنه حل (x=0.5) برای روش توماس شطرنجی (با *dop=8*) و حل تحلیلی

۶- معرفی سیستم محاسباتی

در تحقیق حاضر از پردازنده گرافیکی GTX750M استفاده شده است که دارای قابلیت محاسباتی (نسخه) ۳.۰ است. مشخصات این پردازنده در جدول ۲ ارائه شده است.

همچنین، نسخه کودای به کار رفته ۶.۵ است؛ همچنین پردازنده مرکزی به کار رفته اینتل Core i7 4500U است که در زمره پردازنده‌های هم‌رده با پردازنده گرافیکی مذکور قرار می‌گیرد.

جدول ۲- مشخصات پردازنده به کار رفته در تحقیق حاضر

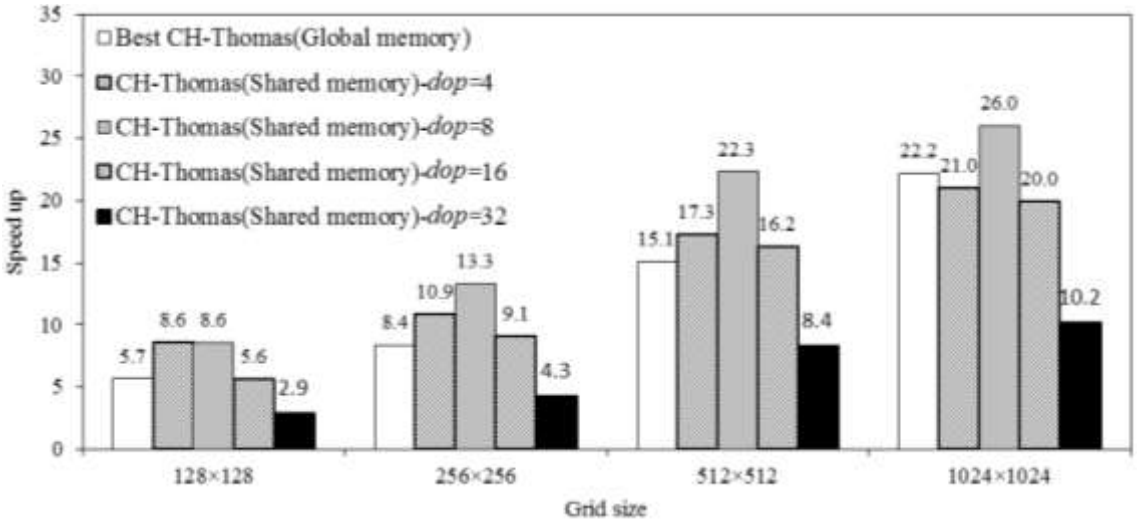
مشخصه	مقدار
تعداد مجتمع پردازشی	۲
تعداد هسته کودا در هر مجتمع پردازشی	۱۹۲
بیشترین میزان حافظه اشتراکی هر مجتمع پردازشی	۴۸ kB
بیشترین تعداد نخ‌های محاسباتی در هر بلاک	۱۰۲۴
بیشترین تعداد نخ‌های محاسباتی هر مجتمع پردازشی	۲۰۴۸
بیشترین تعداد وارپ ساکن در هر مجتمع پردازشی	۶۴
بیشترین تعداد بلاک ساکن در هر مجتمع پردازشی	۱۶

۷- نتایج

در این بخش تاثیر به‌کارگیری حافظه اشتراکی در عملکرد روش توماس شطرنجی، مورد بررسی قرار گرفته است. همانگونه که گفته شد، جهت به کارگیری حافظه اشتراکی در روش توماس شطرنجی dop باید به اندازه کافی کوچک باشد تا بتوان اطلاعات مربوط به سمت راست معادلات (d) و ضریب c را در حافظه اشتراکی ذخیره نمود. در صورتی که اندازه هر بلاک از شبکه محاسباتی ۳۲ باشد، مقدار dop مناسب جهت به کارگیری حافظه اشتراکی مقادیر ۳۲ و کمتر از آن را شامل می‌شود.

شکل ۶ میزان افزایش سرعت روش توماس شطرنجی (نسبت به توماس) در وضعیت به کارگیری حافظه اشتراکی را نشان می‌دهد؛ همچنین میزان افزایش سرعت برای بهترین عملکرد روش توماس شطرنجی در وضعیت به‌کارگیری حافظه سراسری جهت مقایسه نمایش داده شده است. افزایش سرعت معیاری است، برای سنجش میزان بهبود روش‌های مذکور به لحاظ زمانی نسبت به یک روش مبنا (که در اینجا الگوریتم توماس(پردازنده مرکزی) می‌باشد) و مقدار آن از رابطه (۶) به دست می‌آید:

$$Speed\ up = \frac{t_{CPU}}{t} \qquad (۶)$$

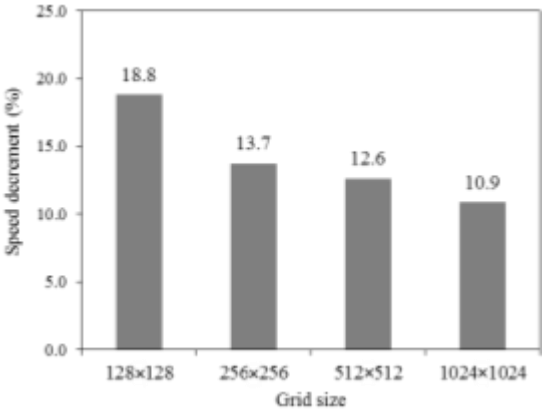


شکل ۶- افزایش سرعت برای روش توماس شطرنجی (نسبت به الگوریتم توماس)

که در آن t_{CPU} مدت زمان حل مسئله توسط الگوریتم توماس (پردازنده مرکزی) و t مدت زمان حل مسئله به وسیله روش مدنظر می‌باشد. لازم به ذکر است، مقایسه در شرایطی انجام شده که هیچگونه تداخلی در دسترسی به حافظه اشتراکی وجود ندارد. نتایج نشان می‌دهد که بیشترین افزایش سرعت روش توماس شطرنجی در وضعیت به کارگیری حافظه اشتراکی در dop=8 واقع شده است. همانگونه که مشاهده می‌شود، به‌کارگیری حافظه اشتراکی موجب بهبود نسبی عملکر روش توماس شطرنجی شده است. در بهترین وضعیت برای شبکه‌ای با اندازه ۲۵۶×۲۵۶ استفاده از حافظه اشتراکی موجب افزایش سرعتی در حدود ۱/۶ برابر نسبت به وضعیت به‌کارگیری حافظه سراسری شده است. در بدترین وضعیت برای شبکه‌ای با اندازه ۱۰۲۴×۱۰۲۴ استفاده از حافظه اشتراکی موجب افزایش سرعتی در حدود ۱/۲ برابر نسبت به وضعیت به‌کارگیری حافظه سراسری شده است.

همانگونه که مشاهده می‌شود، در dop=32 و dop=16 اندازه دستگاه معادلات بیش از حد بزرگ است و استفاده زیاد از حافظه اشتراکی، موجب کاهش ضریب مشغولیت و افت عملکرد روش توماس شطرنجی شده است. در dop=4 نیز تعداد زیاد تقسیمات دستگاه معادلات موجب افزایش بیش از حد تعداد تکرارها و حجم محاسبات شده و موجب افت عملکرد می‌گردد.

شکل ۷ میزان کاهش سرعت ناشی از تداخل دسترسی به حافظه اشتراکی را برای اندازه شبکه‌های مختلف در dop=8 نشان می‌دهد. همانگونه که مشاهده می‌شود، تداخل دسترسی به حافظه اشتراکی، موجب کاهش سرعتی بین ۱۰/۹ تا ۱۸/۸ درصد شده است.



شکل ۷- کاهش سرعت ناشی از دسترسی غیرهم‌مکان به حافظه سراسری (dop=8)

۸- نتیجه گیری

توماس شطرنجی روشی است که بهره‌گیری موثر از الگوریتم توماس درحلگر ADI جهت اجرا روی پردازنده گرافیکی را ممکن می‌سازد. حافظه اشتراکی به نسبت حافظه سراسری سریعتر است. با این حال حجم آن نسبت به حافظه سراسری

بسیار اندک است. علاوه بر این مشکل تداخل دسترسی عملکرد حافظه اشتراکی را به شدت تحت تاثیر قرار داده و سرعت آن را کاهش می‌دهد؛ بنابراین به‌کارگیری این حافظه نیازمند رعایت ملاحظاتی است.

در تحقیق حاضر الگوریتم توماس شطرنجی با استفاده از حافظه اشتراکی پیاده‌سازی شده و عملکرد آن تحلیل شده است. مهمترین نتایج تحقیق حاضر به شرح زیر می‌باشد:

۱- به‌کارگیری حافظه اشتراکی، موجب بهبود عملکرد روش توماس شطرنجی شده و سرعت آن را تا ۱/۶ برابر افزایش داده است.

۲- میزان بهبود عملکر روش توماس شطرنجی با بهره‌گیری از حافظه اشتراکی با مقدار پارامتر *dop* در ارتباط است. به نحوی که در *dop* بالاتر کاهش مشغولیت و در *dop* پایین‌تر، افزایش حجم محاسبات موجب افت عملکر این روش شده است. بهترین عملکرد الگوریتم ارتقاء یافته توماس شطرنجی در dop=8 است.

۳- بیشترین افزایش سرعت مربوط به اندازه شبکه ۲۵۶×۲۵۶ و کمترین افزایش سرعت، مربوط به اندازه شبکه ۱۰۲۴×۱۰۲۴ است

۴- تداخل دسترسی به حافظه اشتراکی، موجب کاهش سرعتی بین ۱۰/۹ تا ۱۸/۸ درصد در روش توماس شطرنجی می‌شود.

در نهایت می‌توان نتیجه گرفت که استفاده از حافظه اشتراکی در صورت تنظیم مناسب پارامتر *dop* و جلوگیری از تداخل دسترسی می‌توان موجب افزایش سرعت محاسبات در روش توماس شطرنجی شود.

۹- فهرست علائم

a	قطر پایین در ماتریس سه قطری
b	قطر اصلی در ماتریس سه قطری
c	قطر بالا در ماتریس سه قطری
d	بردار مقادیر معلوم
dop	اندازه دستگاه معادلات پس از انجام تقسیمات در روش توماس شطرنجی
dx	طول سلول‌های شبکه m
dy	عرض سلول‌های شبکه m
i	شمارنده سلول شبکه در راستای محور طول
j	شمارنده سلول شبکه در راستای محور عرض
k	ضریب هدایت حرارت $\mathrm{Wm°C^{-1}}$
n	تعداد تقسیمات شبکه در راستای محور طول
nop	تعداد تقسیمات دستگاه معادله در روش توماس شطرنجی
m	تعداد تقسیمات شبکه در راستای محور عرض
T	دما °C
t	مدت زمان انجام حل مسئله
t_{CPU}	مدت زمان انجام حل مسئله به وسیله پردازنده مرکزی
x	محور طول
y	محور عرض
Δx	طول سلول‌های شبکه m
Δy	عرض سلول‌های شبکه m

۱۰- مراجع

[1] Du P, Weber R, Luszczek P, Tomov S, Peterson G, Dongarra J (2012) From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming. Parallel Comput 38(8): 391-407.

[2] Sengupta S, Harris M, Zhang Y, Owens JD (2007) Scan primitives for GPU computing. Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware, San Diego, California.

[3] Sakharnykh N (2009) Tridiagonal solvers on the GPU and applications to fluid simulation. NVIDIA GPU Technology Conference, San Jose, California, USA.

[4] Zhang Y, Cohen J, Owens JD (2010) Fast tridiagonal solvers on the GPU. ACM Sigplan Notices 45(5): 127-136.

[5] Göddeke D, Strzodka R (2011) Cyclic reduction tridiagonal solvers on GPUs applied to mixed-precision multigrid. IEEE T Parall Distr 22(1): 22-32.

[6] Davidson A, Owens JD (2011) Register packing for cyclic reduction: a case study, Proceedings of the fourth workshop on general purpose processing on graphics processing units, Newport Beach, California, USA.

[7] Sakharnykh N (2010) Efficient tridiagonal solvers for ADI methods and fluid simulation. NVIDIA GPU Technology Conference, San Jose, California, USA.

[8] Egloff D (2011) Pricing financial derivatives with high performance finite difference solvers on GPUs, Wen-mei W. Hwu (Eds.), GPU Computing Gems Jade Edition, pp. 23.309-23.322, Burlington: Morgan Kaufmann.

[9] Zhang Y, Cohen J, Davidson AA, Owens JD (2011) A Hybrid Method for Solving Tridiagonal Systems on the GPU, Wen-mei W. Hwu (Eds.), GPU Computing Gems Jade Edition, pp. 11.117-11.132, Burlington: Morgan Kaufmann.

[10] Wei Z, Jang B, Zhang Y, Jia Y (2013) Parallelizing alternating direction implicit solver on GPUs, Procedia Computer Science. 18: 389-398.

[11] Kim HS, Wu S, Chang LW, Hwu WMW (2011) A scalable tridiagonal solver for gpus. International Conference on Parallel Processing (ICPP), Taipei, Taiwan.

[12] Esfahanian V, Darian HM, Gohari SI (2013) Assessment of WENO schemes for numerical simulation of some hyperbolic equations using GPU. Comput Fluids 80: 260-268.

[13] Esfahanian V, Baghapour B, Torabzadeh M, Chizari H (2014) An efficient GPU implementation of cyclic reduction solver for high-order compressible viscous flow simulations. Comput Fluids 92: 160-171.

[14] Darian HM, Esfahanian V (2014) Assessment of WENO schemes for multi-dimensional Euler equations using GPU. Int J Numer Meth Fl 76(12): 961-981.

[15] Zolfaghari A, Foadaddini A (2016) Developing new Checkerboard Thomas algorithm for solving tridiagonal set of equations on GPU. Modares Mechanical Engineering 16(2): 309-318. (in Persian)

[16] Beck JV, Wright N, Haji-Sheikh A, Cole KD, Amos D (2008) Conduction in rectangular plates with boundary temperatures specified. Heat Mass Transfer 51: 4676-4690.