یک الگوریتم مرتب‌سازی موازی با کاهش تعداد پردازنده‌ها در کامپیوترهای SIMD

صادق وهابزاده زرگری[×]، سعید صدیقیان کاشی [†]، عادل ترکمان رحمانی [‡]، سعید پارسا [§]

چکیده

تئوری الگوریتم‌های موازی در راستای تسریع پردازش داده‌ها ارائه شده است. همانطور که می‌دانیم مرتب‌سازی یکی از عملیات مهم بر روی داده‌ها می‌باشد. الگوریتم موازی CRCW_Sort با سیاست خواندن همزمان مقادیر و نوشتن همزمان مجموع مقادیر در حافظه مشترک، می‌تواند عمل مرتب‌سازی n عنصر را در زمان O(1) بر روی کامپیوترهای SIMD با تعداد n^2 پردازنده انجام دهد.
در این مقاله با ارائه یک راهکار، تعداد پردازنده‌ها در الگوریتم موازی CRCW از n^2 به $n^2/2-n/2$ یعنی کمتر از نصف تقلیل یافته است. با استناد به راهکار ارائه شده، می‌توان مرتب سازی 2n عنصر را در زمان O(1) بر روی n^2 پردازنده انجام داد.

کلمات کلیدی

الگوریتم‌های موازی، مرتب‌سازی، خواندن و نوشتن همزمان(CRCW)، کامپیوترهای SIMD

A Parallel Sorting Algorithm that reduces the number of processors on SIMD computers

Sadegh Vahabzadeh Zargari[××], Saeed Sedighian Kashi[††], Adel Torkaman Rahmani[‡‡], Saeed Parsa[§§]

Abstract

Over the past forty years dramatic increases in computing speed by parallel algorithms theory. We know that, sorting is an important operation on data elements. The CRCW_Sort parallel algorithm is designed for the CRCW model of computation. The CRCW model is Concurrent Read and Concurrent Write policy that resolve write conflict problem by storing the sum of numbers into the same address. In this model, we can sort n data elements over n^2 processors on a constant time equal with O(1).
In this paper, we propose an algorithm that reduces the number of processors to half. Hence we can sort 2n data elements on O(1) over n^2 processors.

Keywords

Parallel Algorithms, Sorting, Concurrent Read Concurrent Write (CRCW), SIMD computers

[*] دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، Vahabzadeh@comp.iust.ac.ir
[†] دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، Saeedsedighian@yahoo.com
[‡] عضو هیأت علمی دانشگاه، دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، Rahmani@iust.ac.ir
[§] عضو هیأت علمی دانشگاه، دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، Parsa@iust.ac.ir
[**] Iran University of Science and Technology, Computer Engineering Department, Vahabzadeh@comp.iust.ac.ir
[††] Iran University of Science and Technology, Computer Engineering Department, Saeedsedighian@yahoo.com
[‡‡] Academic Staff, Iran University of Science and Technology, Computer Engineering Department, Rahmani@iust.ac.ir
[§§] Academic Staff, Iran University of Science and Technology, Computer Engineering Department, Parsa@iust.ac.ir

۱- مقدمه

با توجه به گسترش روز افزون کاربردهایی که برای انجام سریع محاسبات بر روی داده‌ها به کامپیوترهای سریع نیاز دارند، موازی سازی پردازش داده‌ها می‌تواند راهکار مناسبی برای افزایش سرعت پردازش باشد. ماهواره‌ها نرخ ارسال داده در حد بسیار بالایی دارند و به طور متوسط در هر ثانیه بالغ بر 10^{10} بیت از هر کانال ماهواره‌ای در زمینه‌های مختلف جمعیت، آب و هوا و منابع طبیعی و جغرافیایی ارسال می‌گردد. در زمینه‌های مختلف علمی و کاربردی نیز مثالهایی مشابه وجود دارد. پردازش سه بعدی بدن و ارگانیزم بیمار، شبیه سازی صنعت هواپیمایی و پردازش داده‌های به دست آمده در تحقیقات سنگین اتمی، هسته‌ای، فیزیکی و شیمیایی از کاربردهای مهم قابل ذکر می‌باشند. برای پردازش چنین حجم بالایی از اطلاعات به حداقل سرعت پردازشی در حد 10^{15} عمل در ثانیه مورد نیاز است.

آنچه که مسلم است، کامپیوتری که بتواند پاسخگوی چنین نیازی باشد وجود ندارد. راه حل اساسی برای رفع مشکل فوق موازی‌سازی است. الگوریتم‌های موازی با تقسیم داده‌ها به بخش‌های مجزا، نقش مهمی در موازی‌سازی محاسبات و تقسیم بار محاسبه بر روی چندین کامپیوتر ایفا می‌کنند[۱].

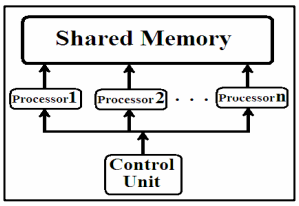
مرتب‌سازی یکی از مسائل مهم در بحث پردازش داده‌ها می‌باشد. با توجه به وابسته بودن داده‌های نهایی مرتب شده به یکدیگر، حتی در صورت تقسیم داده‌ها به ناحیه‌های مجزا برای پردازش موازی، نتیجه نهایی با ادغام نتایج پردازش شده با یکدیگر به دست می آید[۵]. تا کنون الگوریتم‌های موازی مختلفی برای مرتب‌سازی ارائه شده است[۲]. هر یک از این الگوریتم‌ها بر روی ساختار خاصی از کامپیوترهای موازی کارآیی دارند. به عنوان مثال هر یک از ساختارهای خطی، مش، مکعبی و شبکه‌ای در زمینه‌های خاص کاربرد دارند. بهینه‌سازی عمل مرتب‌سازی چالش مهمی در مبحث الگوریتم‌های موازی است[۳]. با ارائه الگوریتم‌های سریع می‌توان بهبود قابل ملاحظه‌ای در مرتب‌سازی ایجاد نمود[۴].

یکی از الگوریتم‌های قابل توجه الگوریتم موازی CRCW_Sort است که بر روی ساختار کامپیوتر SIMD ارائه شده است[۱]. در این روش استفاده از یک حافظه مشترک مد نظر قرار گرفته است. با ذکر این نکته که مسئله برخورد در استفاده از حافظه مشترک باید حل گردد[۶]، سیاست نوشتن مجموع مقادیر در نوشتن همزمان، اِعمال شده است. در روش مزبور تعداد پردازنده‌های مورد استفاده n^2 می‌باشد که بعدا خواهیم دید که این تعداد را می‌توان به نصف تقلیل داد.

ادامه این مقاله از بخش های زیر تشکیل شده است. در بخش دوم الگوریتم CRCW_Sort معرفی شده است. در بخش سوم الگوریتم پیشنهادی ارائه شده است. بخش چهارم به ارزیابی نتایج حاصله اختصاص یافته است و در بخش پنجم ضمن نتیجه گیری نهایی ، برنامه ها و کارهای آتی در زمینه مقاله ارائه شده، بیان شده است.

۲- روش CRCW_Sort

روش CRCW_Sort بر روی کامپیوترهای SIMD[1] مطرح می‌باشد. در این دسته از کامپیوترها با یک دستورالعمل چندین پردازنده به طور همزمان فعال می‌شوند و داده‌های مختلف را مورد پردازش قرار می‌دهند. در کامپیوتر SIMD یک واحد کنترل تعدادی از پردازنده‌ها را به طور همزمان کنترل می‌نماید. هر پردازنده ممکن است حافظه خاص خودش را داشته باشد و یا اینکه از حافظه مشترک استفاده کند. ارتباط بین پردازنده‌ها هم می‌تواند از یک ساختار خاص تبعیت کند. در روش CRCW_Sort پردازنده‌ها از حافظه مشترک استفاده می‌کنند . در شکل ۱ معماری SIMD قابل مشاهده می‌باشد.

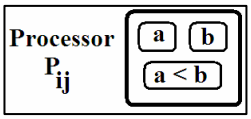


شکل (۱): معماری SIMD

نوع استفاده از حافظه مشترک، خواندن همزمان و نوشتن همزمان (CRCW[2]) می‌باشد. در خواندن همزمان پردازنده‌ها از حافظه مشترک مشکلی وجود ندارد. در مورد نوشتن همزمان سه سیاست مطرح است. اول این که نوشتن فقط در صورتی امکان‌پذیر باشد که مقادیر پردازنده‌ها یکسان باشند. دوم این که اولویت نوشتن به پردازنده با اولویت بیشتر یا کمتر داده شود. سیاست سوم که سیاست مورد استفاده در روش CRCW_Sort است، نوشتن جمع مقادیر در حافظه مشترک است.

اگر $S=\{s_1,s_2, \dots ,s_n\}$ رشته‌ای از اعداد در حافظه مشترک باشند، می‌توان با استفاده از n^2 پردازنده این اعداد را در زمان $O(1)$ مرتب نمود. روش کار به این صورت است که رتبه هر عدد در انتهای اجرای الگوریتم در یک رشته دیگر به نام $C=\{c_1,c_2, \dots ,c_n\}$ که هر عنصر آن مکانی در حافظه مشترک است، نوشته می‌شود.

با توجه به شکل ۲ هر پردازنده حداقل دارای دو رجیستر a, b است و می تواند مقادیری را از حافظه مشترک خوانده و به رجیسترهای خود منتقل نموده و مقایسه لازم را انجام دهد.

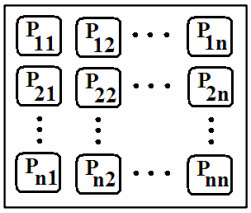


شکل (۲): ساختار پردازنده

تعداد پردازنده‌ها n^2 می‌باشد و با تاکید این نکته که ساختار قرار گرفتن پردازنده‌ها لزوما ماتریسی نیست، شماره‌گذاری پردازنده‌ها به صورت ماتریسی خواهد بود. در شکل ۳ شماره‌گذاری پردازنده‌ها قابل مشاهده می‌باشد.

---

[1] Single Instruction Multiple Data
[2] Concurrent Read Concurrent Write



شکل (۳): شماره‌گذاری پردازنده‌ها

الگوریتم CRCW_Sort در شکل ۴ قابل مشاهده است. در مرحله اول تمام پردازنده‌ها به طور موازی مقادیر دو رشته را خوانده و به رجیسترهای خود منتقل می‌کنند. تمام این اعمال با یک دستور کنترلی مشترک انجام می‌گیرند. هر پردازنده مقادیر دو رجیستر خود را مقایسه نموده و بر اساس آن مقایسه، نتیجه صفر یا یک را در یکی از مکان‌های C که متناظر با اندیس سطری در شماره‌گذاری پردازنده می‌باشد، می‌نویسد. با توجه به این که تمام پردازنده‌ها به طور همزمان و با یک دستور کنترلی در مکان‌های متناظر خود در C می‌نویسند، با اِعمال سیاست نوشتن مجموع مقادیر در زمان نوشتن همزمان، رتبه هر عنصر در مکان مرتبط به دست می‌آید.

```
Procedure CRCW_Sort( S , n )
Begin
//   Step 0:
     For  i:= 1  to  n  do  in  Parallel
          Pi1:  Write  0  in  Ci ;
//   Step 1:
     For  i:= 1  to  n  do  in  Parallel
       For  j:= 1  to  n  do  in  Parallel
        begin
          Pij:  Read Si;   Move  a,Si
          Pij:  Read Sj;   Move  b,Sj
          Pij:  If ( (a>b) or (a=b and i>j) )
                                 Pij  Writes 1 into  Ci
                Else
                                 Pij  Writes 0 into  Ci
        end;
//   Step 2:
     For  i:= 1  to  n  do  in  Parallel
       begin
          Pi1:  Read Si;   Move  a,Si
          Pi1:  Writes Si  into  Location Ci+1 of S
       end;
End.
```

شکل (۴): الگوریتم CRCW_Sort

مثلا اگر n=10 باشد و بعد از اجرای مرحله اول الگوریتم $c_5=6$ باشد، نشان دهنده این است که ۶ پردازنده عدد یک و ۴ پردازنده هم عدد صفر را به طور همزمان در مکان c_5 نوشته‌اند که نشان دهنده رتبه c_5 می‌باشد و با توجه به اینکه تعداد پردازنده‌ها ۱۰۰=$۱۰^۲$ می باشد، ۹۰ پردازنده دیگر بر روی خانه‌های دیگر C به طور همزمان فعال هستند.

در مرحله دوم نیز تعدادی از پردازنده‌ها مقادیر ثبت شده در S را به طور همزمان خوانده و به رجیسترهای خود منتقل می‌کنند. سپس هر پردازنده با نوشتن همزمان مقدار ثبت شده در رجیستر خود، در مکان‌های ثبت شده در C موجب مرتب شدن رشته مورد نظر خود می‌گردد. آنچه که مشخص است زمان الگوریتم فوق با توجه به موازی بودن پردازنده‌ها از مرتبه O(1) خواهد بود.

۳- روش پیشنهادی

با اندکی تغییر در الگوریتم روش CRCW_Sort خواهیم دید که تعداد پردازنده‌ها را می‌توان به نصف تقلیل داد. در شکل ۵ می‌توانید الگوریتم پیشنهادی را مشاهده نمایید.

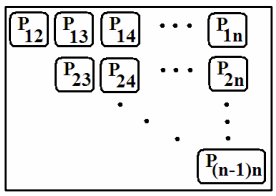
```
//روش پیشنهادی با n²/2-n/2 پردازنده//
Procedure new_CRCW_Sort( S , n )
Begin
//   Step 0:
     For  i:= 1  to  n  do  in  Parallel
          Pi1:  Write  0  in  Ci ;
//   Step 1:
     For  i:= 1  to  n-1  do  in  Parallel
       For  j:= i+1  to  n  do  in  Parallel
        begin
          Pij:  Read Si;   Move  a,Si
          Pij:  Read Sj;   Move  b,Sj
          Pij:  If ( a>b )
                                 Pij  Writes 1 into  Ci
                Else
                                 Pij  Writes 1 into  Cj
        end;
//   Step 2:
     For  i:= 1  to  n  do  in  Parallel
       begin
          Pi1:  Read Si;   Move  a,Si
          Pi1:  Writes Si  into  Location Ci+1 of S
       end;
End.
```

شکل (۵): الگوریتم پیشنهادی

اگر بخواهیم الگوریتم پیشنهادی را بررسی کنیم، مثلا اگر n=10 باشد و بعد از اجرای مرحله اول الگوریتم $c_5=6$ باشد، نشان دهنده این است که ۶ پردازنده عدد یک را به طور همزمان در c_5 نوشته‌اند و پردازنده‌های دیگری که قصد نوشتن مقدار خود را در c_5 داشته‌اند، مقدار صفر را نوشته‌اند.

در روش پیشنهادی پردازنده P_{ij} مقدار خود را پس از مقایسه یا در c_i می‌نویسد و یا در c_j . اگر $S_i > S_j$ باشد مقدار یک در c_i نوشته می‌شود که نشان دهنده رتبه بالای S_i نسبت به S_j می‌باشد و در غیر این صورت مقدار یک در c_j نوشته می‌شود که نشان دهنده رتبه بالای S_j نسبت به S_i می‌باشد.

در شکل ۶ نحوه شماره‌گذاری و آرایش پردازنده‌ها در روش پیشنهادی قابل مشاهده می‌باشد.



شکل (۶): پردازنده‌ها در روش پیشنهادی

تعداد پردازنده‌ها از رابطه ۱ محاسبه می‌شود.

$$1+2+\ldots+(n-1)=\frac{n(n-1)}{2}=n^2/2-n/2 \qquad (۱)$$

با توجه به رابطه ذکر شده می‌توان نتیجه گرفت که تعداد پردازنده‌ها از نصف هم کمتر خواهد بود.

مثلا برای n=10 به جای ۱۰۰=۱۰^۲ پردازنده، می‌توان تعداد ۴۵ پردازنده استفاده نمود که از نصف ۱۰^۲ نیز کمتر خواهد بود.

زمان الگوریتم فوق نیز با توجه به موازی بودن پردازنده‌ها از مرتبه O(1) خواهد بود.

۴- ارزیابی نتایج حاصل

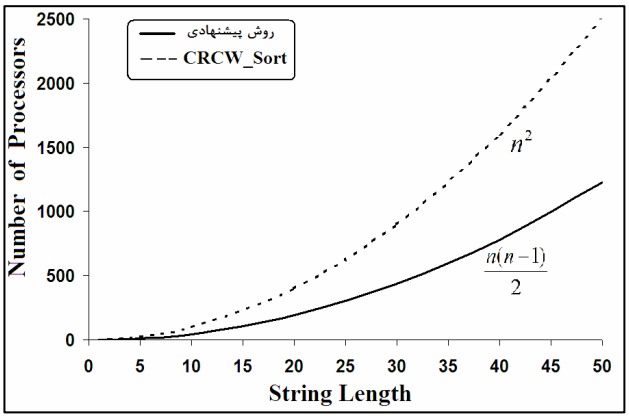
در شکل ۷ نمودار حاصل از مقایسه تعداد پردازنده‌ها در دو روش قابل مشاهده می‌باشد. با افزایش طول رشته S تعداد پردازنده‌ها هم متناسب با مربع آن افزایش می‌یابد. این تعداد در روش پیشنهادی از نصف روش CRCW_Sort کمتر خواهد بود که نشان دهنده این است که روش پیشنهادی بهتر از روش قبلی عمل می‌کند. در ضمن می‌توان نتیجه گرفت که هزینه اجرای روش پیشنهادی نصف روش قبلی می‌باشد.

در ضمن با توجه به نمودار شکل ۸، با استفاده از کمتر از n^2 پردازنده و الگوریتم پیشنهادی می‌توان رشته‌های با طول 2n را در زمانی برابر با O(1) مرتب نمود که این عدد در روش قبلی برابر n می‌باشد. این نمودار نشان می دهد که کارآیی استفاده از پردازنده‌ها در روش پیشنهادی دو برابر روش قبلی شده است.

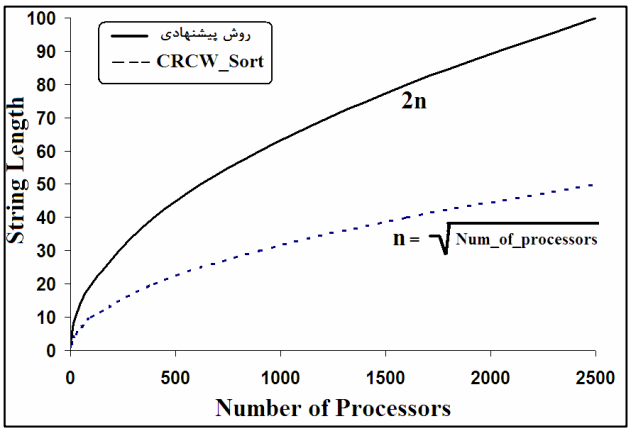
۵- نتیجه‌گیری و کارهای آتی

نمودار شکل ۷ نشان دهنده این است که روش پیشنهادی بهتر از روش CRCW_Sort عمل می‌کند و هزینه پیاده‌سازی آن پایین‌تر است. در ضمن نتایج به دست آمده از نمودار شکل ۸ حاکی از این است که روش پیشنهادی توانسته است با دو برابر نمودن کارآیی پردازنده‌ها در روش CRCW_Sort، آن را توسعه دهد و بهبود بخشد.

پیشنهاد می‌گردد در آینده بر روی الگوریتم‌های موازی دیگر تفکر و کار بیشتری صورت پذیرد. در ضمن با توجه به اینکه تعداد پردازنده‌ها از n^2 به $n^2/2-n/2$ که از نصف کمتر است تقلیل می‌یابد می‌توان به دنبال راهکاری بود که از n/2 پردازنده بیکار نیز حداکثر استفاده صورت پذیرد.



شکل (۷): نمودار مقایسه تعداد پردازنده‌ها



شکل (۸): نمودار مقایسه طول رشته قابل مرتب سازی

سپاسگزاری

در این قسمت لازم می‌دانم از استاد عزیزم جناب آقای دکتر پارسا که زمینه اصلی رشد اینجانب در زمینه الگوریتم‌های موازی هستند، استاد بزرگوار آقای دکتر رحمانی که همیشه راهنمای بنده بوده‌اند و دوست عزیزم مهندس صدیقیان که ایده اصلی از ذهن ایشان تراوش نموده است، کمال تشکر و قدردانی را داشته باشم.

مراجع

[1] Selim G. Akl., Kingstone, Ontario, *The design and analysis of parallel algotithms*, Prentice-Hall, Inc., 1989.
[2] Akl, S.G., *Parallel Sorting Algotithms*, Academic, Orlando, Fl., 1985.
[3] Baudet, G.M., and Stevenson, D.J., *Optimal sorting algorithms for parallel computers*, *IEEE Transactions on Computers*, VOl. C-27, No.1, January 1978, pp 84-87.
[4] Hirschberg, D.S., *Fast parallel sorting algorithms*, *Communications of the ACM*, Vol.21, No.8, August 1978, pp. 657-661.
[5] Kruskal, C.P., *Searching, merging and sorting in parallel computations, IEEE Transactions on Computers*, VOl. C-23, No.10, October 1983, pp 942-946.
[6] Kućera, L., *Parallel computation and conflicts in memoy access, Information Processing Letters*, Vol.14, April 1982, pp.93-96.