سیزدهمین کنفرانس ملی انجمن کامپیوتر ایران
جزیره کیش، خلیج فارس، ایران ۱۹ الی ۲۱ اسفند ۱۳۸۶
انجمن کامپیوتر ایران
Computer Society of Iran
دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر
پوستر

ایجاد تصویر MRI با استفاده از ارزشیابی موازی درخت BSP بر روی مدل محاسباتی BSP

حمزه وحیدی فر
مربی گروه مهندسی کامپیوتر، دانشگاه پی‌ام نور
hdeldari@yahoo.com

حسین دلداری
دانشیار گروه مهندسی کامپیوتر، دانشگاه فردوسی مشهد
Hvf_83@yahoo.com

چکیده: یکی از محدودیت ها و معضلات استفاده از کامپیوتر در کارهای گرافیکی، داده های حجیم و بالا بودن زمان پردازش داده ها در بسیاری از پردازش ها و کاربردهای گرافیکی می‌باشد. بنابراین نیازمند روش هایی هستیم که داده های گرافیکی بسیار زیادی را در زمانی معقول پردازش نماید. پیشرفت تکنولوژی کامپیوتر در عصر حاضر، استفاده از سیستمهای کامپیوتری موازی را در بسیاری از زمینه ها مقدور نموده است. بطوری که می‌توان نرم افزارهای موازی بر مبنای ساختمان داده مجرد مستقل از معماری ماشین طراحی و پیاده سازی نمود که توسعه پذیر بوده و کارآیی آن قابل پیش بینی و عملی باشد با توجه به کاربرد زیاد اطلاعات گرافیکی، پردازش و رندرسازی داده های گرافیکی بصورت موازی از اهمیت خاصی برخوردار شده است.

در این مقاله، سعی شده است ایجاد تصویر MRI[1] با استفاده از درخت BSP[2]، بر روی مدل محاسباتی موازی و عمومی مرسوم به BSP[3]، طراحی و پیاده سازی گردد. جهت بررسی کارآیی ایجاد تصویر MRI، درخت مزبور بر روی شبکه ای از کامپیوترها پیاده‌سازی گردید. زمان ایجاد تصویر MRI از لحظه ارسال اطلاعات تا زمان دریافت نتیجه مورد نظر، اندازه گیری شد. نتایج بدست آمده حاکی از کارآیی مناسب این روش هنگام افزایش تعداد پردازنده ها در شبکه نسبت به حالت تک پردازنده ای می‌باشد این امر دلالت بر کارآیی توسعه پذیری الگوریتم دارد.

کلمات کلیدی: محاسبات موازی، محاسبات توزیع شده، مدل محاسباتی BSP، درخت BSP، ابرگام[4]، رندر سازی موازیSL[5]، MRI.

۱- مقدمه

یکی از محدودیتها و مشکلات استفاده از کامپیوتر در کارهای گرافیکی، بالا بودن زمان پردازش برنامه‌ها می‌باشد. که ناشی از کمبود منابع زیر می‌باشد محدودیت محاسباتی[1]، محدودیت گرافیکی[2]، محدودیت رابط[3] و محدودیت نمایش[4][۱].

یکی از بهترین راه‌حل‌ها برای کاهش زمان پردازشهای گرافیکی، تقسیم مساله به مسائل کوچکتر و حل آنها می‌باشد برای این منظور از ساختمان داده درخت BSP استفاده می‌کنیم. حال اگر این درخت بصورت موازی پیاده سازی و ارزشیابی شود باعث می‌شود کارآیی نرم افزارهایی که از این ساختمان داده استفاده می‌کنند به نحو بسیار مطلوبی بوده و ایجاد تصویر MRI در مدت زمان کمتری امکان پذیر می‌شود.

در این مقاله، هدف طراحی و پیاده سازی روشی برای ایجاد تصویر MRI با استفاده از ارزشیابی موازی درخت BSP، تحت مدل محاسباتی BSP می‌باشد؛ در ابتدا درخت BSP، مدل BSP و MRI مورد بررسی قرار گرفته است. سپس الگوریتم پیشنهادی در بستر مدل محاسباتی BSP ارائه شده، آنگاه با استفاده از امکانات ارتباطی و همزمانی کتابخانه Oxford BSPlib[۲]، الگوریتم مذکور بر روی شبکه ای از کامپیوترها طراحی و پیاده سازی گردیده است. پس از بی‌ان نتایج عملی حاصله و رهنمودهایی برای تحقیق بی‌شتر، از آنجایی که بالا بردن سرعت پردازش اطلاعات یکی از مهمترین انگیزه های استفاده از محیط های موازی برنامه نویسی است. در این مقاله، زمان اجرای الگوریتم بعنوان پارامتر کارآیی در نظر گرفته شده است.

۲-۱- درخت BSP

درخت BSP، یک تقسیم بندی بازگشتی و سلسله مراتبی فضای n بعدی، به زیرفضاهای محدب می‌باشد که طی آن یک فضای n بعدی توسط یک سطح n-1 بعدی تقسیم شده و نتیجه آن، ایجاد دو زیرفضای جدید n بعدی می‌باشد در هر یک از زیرفضاهای جدید، می‌توان تقسیم

[1] Magnetic Resonance Imaging
[2] Binary Space Partition tree
[3] Bulk Synchronous Parallel Model
[4] SuperStep
[5] sort-last Parallel Rendering

بندی بازگشتی فضا را ادامه داد. عمل تقسیم می‌تواند تا رسیدن به عمق ماکزیممی از درخت BSP، یا قرار گرفتن میزان داده خاصی در برگ های درخت BSP، ادامه یابد[۳.۴.۱۴].

پس از ارائه تعریف درخت BSP، تحقیق بر روی خواص و کاربردهای درخت BSP گسترش یافت. از مهمترین زمینه های کاربرد درخت BSP، به حذف سطوح مخفی[۵] [۵]، اثر اشعه‌ای[۶] [۶]، ساخت اجسام هندسی[۷] [۳]، ایجاد سایه[۸] [۷]، یافتن موقعیت نقطه[۹] [۸]، تشخیص برخورد[۱۰] [۹] حرکت روباتها[۱۱] [۱۰]و ... می‌توان اشاره کرد.

در سالهای اخیر، استفاده از روشهایی همچون درخت BSP موازی رایج شده است. با این روش براحتی می‌توان داده ها را بر روی شبکه ای از کامپیوترها توزیع نمود. که از جمله کاربردهای مهم درخت BSP موازی می‌توان به رندرسازی حجم به صورت موازی که Silva و Ramakrishnan اشاره نمود[۱۱،۱۲].

۲-۲- مدل BSP

در دنیای محاسبات موازی، مدلهای محاسباتی مختلفی وجود دارد. یکی از این مدلهای مطرح، مدل همه منظوره BSP است. که در سال ۱۹۹۰ توسط Valiant از دانشگاه هاروارد پیشنهاد گردید[۱۳]. مدل مذکور، برنامه نویسی موازی را توجیه پذیر نموده است[۱۵]. معماری مدل BSP دارای اجزاء زیر می‌باشد:

- تعدادی پردازنده به انضمام حافظه های مربوطه
- یک شبکه ارتباطی که امکان اتصال نقطه به نقطه را بین پردازنده ها فراهم می‌نماید.
- یک مکانیزم هماهنگ سازی که باعث همگام سازی کلیه یا بخشی از پردازنده ها می‌شود.

مهمترین خواص مدل BSP عبارتند از:

۱- برنامه نویسی در آن، شبیه برنامه نویسی ترتیبی بوده و فقط تعداد کمی دستور ساده به آن اضافه شده، لذا برنامه نویسی با این مدل ساده و آسان می‌باشد.

۲- مدلBSP به معماری ماشین وابسته نمی‌باشد.

۳- زمان اجرای برنامه با استفاده از متن برنامه و تعدادی پارامتر ساده از معماری ماشین، به راحتی قابل محاسبه می‌باشد.

مدل BSP می‌تواند در سیستم ها و زبانهای برنامه نویسی مختلفی از قبیل PVM و Cary's SHMEM جا داده شود. همچنین می‌توان با افزودن کتابخانه Oxford BSPlib به زبان‌های C/C++ و فرترن، می‌توان از برنامه نویسی BSP استفاده نمود.

مدل BSP می تواند بوسیله خواص چهارگانه ذیل مشخص شود: P(تعداد پردازنده ها)، S(سرعت پردازنده ها (گامهای زمانی در هر ثانیه))، l(تأخیر در پردازش (کمترین تعداد گام های زمانی در هماهنگ سازی عملگرها))، g(تعداد عملیاتی محلی انجام شده توسط پردازنده ها، تقسیم بر تعداد کلمات جاری که بوسیله شبکه ارتباطی در یک ثانیه تحویل داده شده است).

هزینه یک ابرگام در این مدل برابر است با تعداد عملیات محلی انجام شده بوسیله هر پردازنده در خلال زمان s، توسط h تعداد گره های موجود در شبکه.

$$\text{هزینه یک ابرگام} = \underbrace{\text{MAX } w_i}_{\text{محاسبات}} + \underbrace{\text{MAX } h_i * g + l}_{\text{ارتباطات}}$$

(i شماره پردازشگر)

۲-۳- MRI و کاربرد های آن

MRI به معنی تصویر برداری تشدید مغناطیسی می‌باشد با MRI می‌توان از یک عضو تصویرسازی کرد. این شیوه تصویربرداری برای بررسی بیماریهای کبدی، کلیوی، ریوی، نارسایی‌های قلبی، بررسی جریان خون، بیماری وبا، بررسی ماهیچه، عروق و رباطها کاربرد دارد[۱۶].

۳- طراحی درخت BSP موازی با محاسباتی BSP

جهت ایجاد تصویر MRI بصورت موازی، درخت BSP بر روی تعدادی از کامپیوترهای موجود در شبکه توزیع می‌گردد. در هنگام رندرسازی، کاربر(ماشین صفر) ایجاد تصویر MRI از نقطه دید خاصی را درخواست می‌کند؛ این درخواست به کلیه کامپیوترهای موجود در شبکه ارسال می شود.. با دریافت این پیام بطور همزمان توسط کلیه کامپیوترها، ابرگام آغاز شده و رندرسازی داده ها با استفاده از درخت BSP بر روی هر کامپیوتر صورت می‌پذیرد. در نهایت با ترکیب نتایج بدست آمده در نود ریشه، تصویر نهایی بوجود آمده و در اختیار پزشکان قرار می‌گیرد.

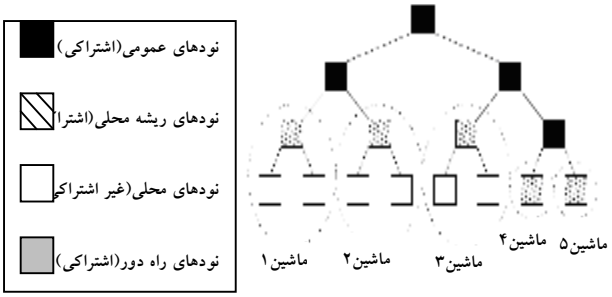
جهت تامین ارتباطات نقطه به نقطه بین پردازنده ها، از مدل شبکه‌ای ستاره امکانات کتابخانه Oxford BSPlib استفاده شده است.

۳-۱- تشریح الگوریتم پیشنهادی

در این الگوریتم، پس از دریافت مجموعه داده از دستگاه MRI، مجموعه داده را به تعداد پردازنده ها، به چند قسمت مساوی تقسیم می‌نماییم. و هر کدام از نواحی فوق را برای یک پردازشگر ارسال می-کنیم؛ سپس پردازشگرها عملیات رندرسازی را به صورت موازی انجام داده و نتایج حاصل را با هم ترکیب نموده و تصویر MRI نهایی بوجود می‌آید. با توجه به مطالب ذکر شده، الگوریتم یاد شده شامل سه مرحله می باشد:

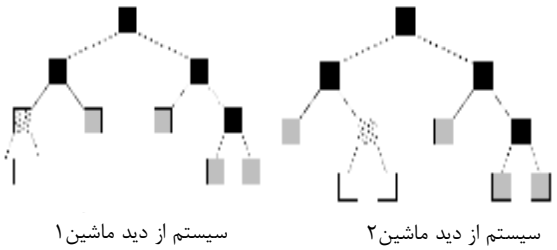
۱. دریافت مجموعه داده و توزیع نمودن داده ها بر روی پردازنده ها(تشکیل درخت BSP).
۲. انجام عمل رندر سازی توسط هر پردازشگر بر روی داده اختصاصی اش به صورت موازی.
۳. ترکیب نتایج رندر شده در داخل فایلی واحد و نمایش تصویر.

در مرحله اول الگوریتم، پس از دریافت مجموعه داده ها و تعداد پردازشگرها(n و توانی از ۲) بصورت پارامتری، پردازشگر شماره صفر با توجه به اینکه مشخصات ظاهری همه پردازشگرها یکسان می‌باشد. برای ایجاد تعادل بار، مجموعه داده ها را به n قسمت مساوی تقسیم نموده و به ترتیب برای پردازشگرهای P_0 تا P_{n-1} توزیع می‌کند. سپس هر پردازشگری قسمتی از مجموعه داده را در اختیار داشته و می‌تواند داده ها را تا زمان رسیدن به یک پیکسل توسط الگوریتم برش بر اساس نقطه میانی، تقسیم نموده به این ترتیب درخت BSP بوجود می آید. دید کلی سیستم به صورت شکل(۲) خواهد بود:



شکل(۲): دید کلی سیستم در روش پیشنهادی

همچنین در شکل(۳) سیستم از دید ماشین های ۱ و ۲ نشان داده شده است.



شکل(۳): از دید ماشین ۱ و ماشین ۲

در مرحله دوم(فاز رندرسازی)، با توجه به نقطه دیدی که کاربر دارد هر پردازشگری با استفاده از یک سطح جداکننده، که پردازشگر اصلی مشخص می‌کند. مجموعه داده مربوط به خود را برش زده و نقاطی که در تصویر نهایی قابل مشاهده نخواهند بود را بوسیله مقدار دهی با یک پرچم مشخص می‌کند. بعبارت دیگر، نودهای رندرساز، حلقه زیر را اجرا می‌نمایند:

```
For each(view point v)
  ComputeImage(v);
  P = WaitForToken();
  SendImage();
```

آخرین مرحله الگوریتم(فاز ترکیب)، ادغام نتایج بدست آمده می‌باشد از آنجایی که شماره هر پردازشگر، موید جایگاه آن قسمت در داخل درخت می‌باشد لذا در ابتدا هر پردازشگر نتایج مربوط به خود را به پردازشگر قبلی خود با مسافت یک، خواهد داد(پردازشگر شماره یک به پردازشگر شماره صفر و ...).

پردازشگری که فایلی را دریافت نموده به انتهای فایل خود اضافه می‌نماید حال مسافت دو برابر شده و مجدداً روال فوق با تعداد نصف پردازنده ها ادامه می‌یابد مثلاً پردازشگر شماره دو فایل جدید خود را به پردازشگر شماره صفر می دهد و ... روال فوق زمانی متوقف می‌شود که مسافت، کوچکتر مساوی تعداد پردازشگرها باشد بعبارت دیگر فایل نهایی در اختیار پردازشگر شماره صفر، قرار بگیرد پردازشگر صفر نیز با استفاده از یک نرم افزار DICOM Viewer تصویر مورد نظر را نمایش خواهد داد. بعبارت دیگر، نودهای ترکیبی، حلقه زیر را انجام می‌دهند:

```
For each(view point v)
  CompositeImage(v);
  P = WaitForToken();
  SendImage();
```

۴- برنامه پیاده سازی شده

جهت پیاده سازی سیستم از یک برنامه شبیه ساز تحت سیستم عامل Windows XP استفاده گردید. وظیفه این برنامه، دریافت داده ها از دستگاه MRI و ایجاد تصویر نهایی همچنین اندازه گیری زمان تلف شده بین دریافت داده ها تا ایجاد نهایی نتیجه می‌باشد.

در این برنامه، از زبان برنامه نویسی Visual C++ 6 و کتابخانه Oxford BSPlib که در واقع یک زبان برنامه نویسی موازی می‌باشد استفاده گردیده است. کتابخانه Oxford BSPlib می تواند بستره مدل محاسباتی BSP را به راحتی تامین کند همچنین می تواند در انواع شبکه های کامپیوتری می‌تواند مورد استفاده واقع شود.

در نهایت پردازشگر شماره صفر، فایل فوق را بررسی نموده و پیکسلهایی که مقدار پرچم در آنها صفر می‌باشد را در قالب یک فایل DICOM ذخیره نموده و سپس توسط نرم افزار DICOM Viewer در تصویر سه بعدی، نمایش خواهد داد.

۴-۱- ارزیابی کارآیی تئوری الگوریتم

در ادامه مطالب، کارآیی الگوریتم را به صورت تئوری ارزیابی می‌کنیم. با فرض اینکه تعداد پردازشگرها N بوده و برنامه از ۳ ابرگام تشکیل شده است. برای محاسبه افزایش سرعت نسبی و ارزیابی کارآیی برنامه، ابتدا برنامه را با یک پردازشگر و سپس با N پردازشگر اجرا می‌کنیم.

در ابرگام اول، در حالت موازی، پردازشگر شماره صفر تمام داده ها را در حافظه بقیه پردازشگرها کپی می‌نماید. بنابراین اگر n داده داشته باشیم، هزینه ارتباطات n.g خواهد شد. از آنجایی که در این ابرگام محاسبه محلی نداریم، W برابر صفر بوده و داریم:

هزینه ابر گام اول با یک پردازشگر = ۰

هزینه ابر گام اول با N پردازشگر = n.g + L

اگر زمان برش مجموعه داده توسط یک پنجره w باشد و نیز بخواهیم درختی به عمق m بوجود آوریم، در این حالت تک پردازشگر نیاز به زمانی برابر با $2^m.w$ داریم. فرض کنید 2^m برابر N باشد، اگر همین درخت را با N پردازشگر موازی بخواهیم بوجود آوریم، تنها به زمان w برای برش زدن نیاز داریم. در نتیجه در ابرگام دوم با توجه به اینکه h_2 مساوی صفر است داریم:

N.w = هزینه ابر گام دوم با یک پردازشگر

W + L = هزینه ابر گام دوم با N پردازشگر

در ابر گام سوم اگر هزینه انتقال اطلاعات یک پردازشگر داخل فایل، s در نظر گرفته شود؛ داریم:

N.s = هزینه ابر گام سوم با یک پردازشگر

s + L = هزینه ابر گام سوم با N پردازشگر

بنابراین هزینه کلی برنامه به قرار زیر است:

N.w + N.s = هزینه کل با یک پردازشگر

n.g + w +s + 3L = هزینه کل با N پردازشگر

افزایش سرعت نسبی برابر است با زمان اجرای الگوریتم موازی با یک پردازشگر تقسیم بر زمان اجرای الگوریتم موازی با N پردازشگر. اگر افزایش سرعت نسبی را با S_{up} و کارآیی را با E نمایش می دهیم؛ داریم:

$$S_{up} = \frac{N.w + N.s}{n.g + w + s + 3L}$$

$$E = \frac{S_{up}}{N} = \frac{w + s}{n.g + w + s + 3L}$$

چون L مقدار کوچکی در مقابل w بوده و نیز با توجه به پهنای باند بالای شبکه ارتباطی ماشین های موازی (بیش از 100 MB/sec)، مقدار n.g (هزینه ارتباطات رابطه بالا) مقدار کوچکی می باشد. لذا ملاحظه می شود که کارآیی الگوریتم تقریباً بالا بوده و نزدیک به عدد یک می‌باشد. نتایج حاصل از اجرای الگوریتم نیز آنرا تایید می‌نماید.

۴-۲ ارزیابی کارآیی عملی الگوریتم و مقایسه آنها

در این سیستم از ۱۶ کامپیوتر با مشخصات جدول(۱) استفاده گردیده که از طریق یک سوئیچ ۱۰۰Mb به هم متصل شده اند. همه سیستم ها دارای سیستم عامل Windows XP بوده و برای برنامه نویسی از زبان Visual C++ 6 و کتابخانه BSP با نسخه Oxford BSPlib استفاده شده است.

جدول(۱): مشخصات سیستم

Component	Type
CPU	Dual Intel Xeon 1.70GHZ
Memory	2x512M
Network card	Intel pro100
Graphics board	NVIDIA Geforce3(128MB RAM)
Hard drive	18GB seagate Cheetah U160

در این برنامه جهت بررسی کارآیی الگوریتم، از پارامترهای زمان اجرا، اندازه مساله و تعداد فریم بر ثانیه، استفاده می‌نماییم. در این اندازه-گیری‌ها، در واقع اندازه‌گیری هزینه مدل(تاخیر شبکه در مدل BSP)، هزینه‌های ارتباطی، میزان تبادل داده، دسترسی به اطلاعات درخت BSP، رندر کردن و نمایش تصویر نهایی لحاظ شده است.

در این سیستم از فریم بافر ۳۲ بیتی RGBA استفاده شده است که ۸ بیت برای هر رنگ R,G,B و ۸ بیت نیز برای کانال آلفا بکار می‌رود برای رسم پیکسل ها، از تابع OpenGl به نام glDrawPixel() استفاده می‌کنیم در ادامه مطالب، چندین سناریو در سیستم تعریف نموده و نتایج حاصل از اجرای آنها نشان داده شده است.

به عنوان مثال برای ایجاد تصویر برای مجموعه داده ای ۲۰۹×۲۵۶×۲۵۶ بایتی با وضوح تصویر ۲۵۶×۲۵۶، زمان بدست آمده از اجرای برنامه توسط یک کامپیوتر، ۳۰/۱۵ ثانیه بود. در حالیکه همین عمل با دو کامپیوتر زمانی برابر ۱۷/۱۲ ثانیه، با چهار کامپیوتر زمانی برابر ۱۴/۶۸ ثانیه، با هشت کامپیوتر زمانی برابر ۱۳/۲۴ ثانیه و با ۱۶ کامپیوتر زمانی برابر ۱۶/۳۰ ثانیه داشت. افزایش سرعت نسبی و کارآیی اجرای برنامه با دو پردازشگر را می‌توان به صورت زیر محاسبه نمود:

S_{up} = ۳۰/۱۵ ÷ ۱۷/۱۲ = ۱/۷۶

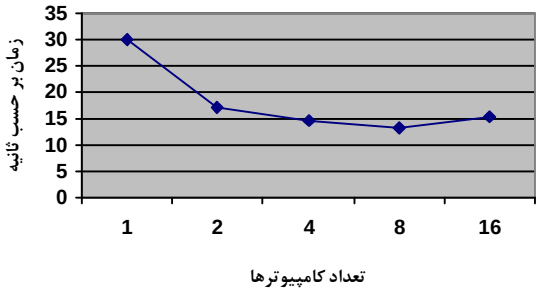
E = ۱/۷۶ ÷ ۲ = ۰/۸۸

افزایش سرعت نسبی و کارآیی اجرای برنامه با چهار کامپیوتر را می‌توان به صورت زیر محاسبه نمود:

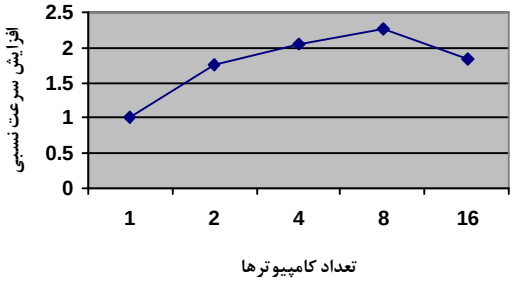
S_{up} = ۳۰/۱۵ ÷ ۱۴/۶۸ = ۲/۰۵

E = ۲/۰۵ ÷ ۴ = ۰/۵۱

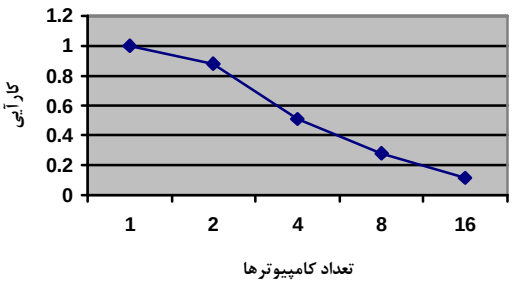
اگر محاسبات فوق را برای سایر حالات نیز تکرار کنیم. آنگاه نمودارهای مربوط به زمان اجرای برنامه، افزایش سرعت نسبی و کارآیی نتایج فوق به ترتیب بصورت شکل(۴)، شکل(۵) و شکل(۶) خواهد بود:



شکل(۴): زمان برنامه برای وضوح تصویر ۲۵۶×۲۵۶



شکل(۵): سرعت نسبی برنامه برای وضوح تصویر ۲۵۶×۲۵۶



شکل(۶): زمان برنامه برای وضوح تصویر ۲۵۶×۲۵۶

از بررسی نمودارهای فوق مشخص می‌گردد که افزایش درجه وضوح و اندازه مجموعه داده باعث زیاد شدن زمان اجرای برنامه می‌شود و اضافه شدن تعداد پردازنده ها، رابطه ای تصاعدی با ازدیاد سرعت رندر کردن اطلاعات و ایجاد تصویر MRI دارد. بدین معنا که برای رندر کردن و ایجاد تصویر MRI، بهتر است از روش موازی استفاده کنیم.

با توجه به نمودارهای بالا، ملاحظه می‌شود که کارآیی تئوری و عملی الگوریتم با هم مطابقت دارند و E در برخی موارد با زیاد شدن تعداد پردازشگرها کاهش می‌یابد که علت اصلی آن بالا رفتن هزینه همزمانی بین پردازشگرها(L) به علت زیاد شدن پردازشگرها و عدم موازنه درخت می‌باشد(به علت عدم موازنه درخت، اختلاف زمان اتمام کار پردازشگرها، زیاد می‌گردد). در نتیجه کارآیی الگوریتم توجیه پذیر و قابل قبول می‌باشد.

۵- نتیجه گیری

در این بخش بطور خلاصه به بیان اهداف و برخی از دستاوردها و نتایج این می‌پردازیم.

برای کارهایی از قبیل رندر کردن می‌توان مجموعه داده های بزرگ را با استفاده از ساختار درخت BSP به قسمت های کوچکتر تقسیم نمود و درخت مزبور را بر روی شبکه ای از کامپیوترها توزیع کرد. بصورت موازی می‌توان این داده ها را رندر نموده و برای بدست آودن جواب نهایی، نتایج حاصله را ادغام کرد. در این تحقیق، عمل موازی سازی طبق مدل محاسباتی BSP انجام گردیده است.

در این الگوریتم برای عملیات رندرسازی از SL استفاده کردیم چون SL ها پیاده‌سازی ساده‌ای داشته و در طیف وسیعی از الگوریتم‌ها قابل استفاده می‌باشند. از مهمترین مزایای استفاده از رندر کننده های SL می‌توان به موازنه بار خیلی خوب و قیاس پذیر بودن اشاره نمود. یعنی تعداد رندر سازها را می‌توان افزایش داد بطوریکه داده ها بصورت تقریباً مساوی بین پردازشگرها تقسیم می شوند.

در این الگوریتم از موازنه بار استاتیک استفاده گردید. بدین صورت که در ابتدای اجرای الگوریتم با مشخص شدن تعداد پردازنده ها، داده ها را به نسبت مساوی بین پردازنده ها تقسیم می‌شود بطوریکه در الگوریتم پیشنهادی، با افزایش تعداد پردازشگرها می‌توان زمان رندر کردن و ایجاد تصویر نهایی MRI را به مقدار زیادی کاهش داد که این دستاورد خوبی برای توجیه کارآ بودن الگوریتم می‌باشد.

از زمینه های باز تحقیقاتی مرتبط با این مقاله، استفاده از الگـوریتم پیشنهادی در یک بازی تحت شبکه با تعداد زیادی بازیکن می‌باشد که انجام عمل رندر سازی، طبق الگوریتم خاصی که در اینجا مطرح گردید و مقایسه نتایج این روش، با سایر روشها می‌باشد. همچنین استفاده از الگوریتم پیشنهادی در پیاده سازی شبیه ساز پرواز اشاره کرد که دارای تعداد زیادی نمایشگر بوده و بصورت کاشی وار در کنار هم قرار گرفته-اند که هر نمایشگر توسط یک پردازنده کنترل می‌شود و مقایسه نتایج عملی این روش با حالتی که از یک نمایشگر بزرگ استفاده مـی‌شـود؛ مقایسه نمود.

سپاسگزاری

بدینوسیله از زحمات پرسنل بخش MRI بیمارستان قائم مشهد کـه از ارائه اطلاعات مناسب، دریغ ننمودندکمال تشکر و سپاسگزاری را داریم.

منابع و ماخذ

[1]. Charles Smith, “Distributed Rendering of Particle Systems” , B.A. Computer Science, April 22, 2003
[2]. http://www.bsp-worldwide.org/

زیرنویس‌ها

[1] Compute limited
[2] Graphics limited
[3] Interface limited
[4] Display limited
[5] Hidden-surface removal
[6] ray tracing
[7] Constructive solid geometry
[8] Shadow generation
[9] Point location
[10] Collision detection
[11] Motion robotics

[3]. S. F. Buchele, "Three-Dimensional Binary Space Partitioning Tree and Constructive Solid Geometry Tree Construction from Algebraic Boundary representations", Ph.D Dissertation, The University of Texas at Austin,1999.
[4]. Cbasa D,"Binary Space Partitions", September 26, 2004
[5]. T. M. Murali, Efficient hidden-surface removal in theoryand in practice, Ph.D. Thesis, Department of Computer Science, Brown University, Providence, 1998
[6]. B. Naylor and W. Thibault, Application of BSP trees to ray-tracing and CSG Evaluation, Technicl Report GIS-ICS, Georgia Institude of Twch, School of information and Computer Science, 1996.
[7]. H. C. Batagelo and I. Costa Jr., Real-time shadow generation using BSP trees and stencil buffers, in proc. 12th Brazilian sympos. Comp. Graph. And Image Processing, 1999, pp. 93-102.
[8]. S Arya, M. Malamatos, and D. M. Mount, Nearly Optimal expected-case planar point location, 41th sympos. Foundation of Comp. Sci., IEEE press,2000, pp. 208-21.
[9]. S. Ar, G Montage, and A. Tal Deferred, self-organizating BSP Trees, Comput. Graph. Forum 21 (3) (2002), 269-278.
[10]. C. Ballieux, Motion planning using binary bspace partitions, Technical Report Inf/src/93-25,Utrecht University, 1993.
[11]. C. R Ramakrishnan, Claudio T. Silva, "Optimal Processor Allocation for Sort-Last Compositing under BSP-Tree Ordering", University Of Computer Science State University of New Yorkat Stony Brook, 1999.
[12]. Claudio Silva, Drik Bartz, Bengt-Olaf Schneider, "Rendering and Visualization in Parallel Environments", University of Tubingen, IBM T.J Watson Research Center, AT&T Labs-Research (SIGGRAPH2000 Course on"Rendering and Visualization in Parallel Environments", 2000.
[13]. L. Valiant, "A Bridge Model for Parallel Computing", Communication of the ACM, 1990
[14]. Steven Cento, "BSP Tuterial by Steven Cento", Department of Computer Science, University of Stellenbosch, Private Bag X1,7602 Matieland,South Africa, 2002.
[15]. D.B. Skillicorn, Jonathan M.D. Hill and W.F. McColl, "Question and Answer About BSP", Department of Computer and Information Science Queen's University, Kingston, Canada, Computing Laboratory University of Oxford, Novamber 1996.
[16]. http://en.wikipedia.org/wiki/MRI